

POLITECNICO DI TORINO

SEDE DI TORINO

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale

Paradigmi di comunicazione e interazione collaborativa in AR

Relatore
prof. Andrea Giuseppe Bottino

Candidati
Enrico Ameglio
matricola: 233042
Simone Turello
matricola: 233142

Supervisore aziendale
Dead Pixels S.r.l.s.
dott. ing. Maurizio Marseguerra

Anno accademico 2018-2019

Sommario

È possibile suddividere il lavoro svolto per la tesi in due macro-sezioni: la prima sezione, di stampo prettamente teorico, riguarda la ricerca e lo studio dei principali paradigmi di interazione e di awareness multi-utente all'interno di un ambiente in realtà aumentata; la seconda parte, di stampo puramente pratico ed implementativo, riguarda lo sviluppo di una applicazione in AR che implementi i suddetti paradigmi.

In particolare la prima sezione ha comportato un quanto più completo possibile studio dello stato dell'arte relativo l'interazione e l'*awareness* multi-utente in AR e VR e un'analisi di quali paradigmi e ausili graficirisultassero più efficaci e quali fossero migliorabili; la seconda sezione ha richiesto quindi lo sviluppo di due framework al fine di garantire la totale indipendenza dell'applicazione dalle varie librerie di AR e di *networking* presenti sul mercato e lo sviluppo di una applicazione di test che potesse essere usata per verificare l'efficacia degli elementi identificati precedentemente.

Ringraziamenti

Desidero ringraziare tutti coloro che mi hanno sostenuto e supportato in questi anni di studio e durante la stesura di questa tesi con suggerimenti ed osservazioni. Ringrazio anzitutto il professore e relatore Andrea Bottino e il dott.ing. Francesco Strada per la loro preziosa guida e il costante supporto.

Ringrazio Jacopo, il mio unico e vero migliore amico, per aver creduto in me ogni giorno, avermi sempre sostenuto in tutto e avermi reso una persona migliore con le sue critiche sempre costruttive. Ringrazio il mio amico e collega Simone, prezioso e fondamentale elemento in diversi progetti universitari, senza il quale non avrei mai potuto portare a termine questa tesi.

Un ringraziamento particolare va ai miei più cari amici, i quali mi hanno regalato innumerevoli momenti felici ed indimenticabili: ringrazio Alice e Francesca per la loro amicizia vera e indissolubile che perdura da ormai vent'anni; ringrazio Gabriele per aver reso migliori i giorni trascorsi insieme in università e per esserci sempre in caso di bisogno e ringrazio Giulia e Francesca con cui sono tornato finalmente a condividere il mio tempo libero. Ringrazio tutte le persone che hanno partecipato con entusiasmo alla sperimentazione, ringrazio Fabio che ha saputo gestire con competenza e serietà la mia assenza dagli impegni lavorativi di quest'ultimo periodo. Infine un ringraziamento va anche a tutti gli amici che ormai per un motivo o per un altro non fanno più parte della mia vita.

Un ringraziamento speciale ad Arianna, il mio primo vero amore, per essermi stata vicina ed avermi supportato e sopportato in questo duro periodo. A lei dedico questo lavoro.

Vorrei ringraziare la mia famiglia, mia mamma Rita, mio papà Enzo, mia sorella Elisa, mio cognato Luca e mia nonna Gina per essere la famiglia migliore che si possa mai desiderare.

Infine vorrei lasciare l'ultimo ringraziamento a chi purtroppo non potrà mai leggerlo, ma avrebbe desiderato tanto farlo; ringrazio mia nonna Tina per essersi sempre presa cura di me e per avermi cresciuto con pazienza e amore.

Indice

Elenco delle tabelle 8

Elenco delle figure 9

I Introduzione **15**

1 Finalità del lavoro svolto 17

II Studi preliminari **21**

2 Stato dell'arte 23
2.1 Distribuzione degli articoli e parametri di ricerca 24
2.2 Progettazione di un ambiente AR collaborativo 25
2.2.1 Comunicazione naturale (*seamless*) multi-utente, monitoraggio e rappresentazione dell'utente per potenziare interazione e *awareness* 25
2.2.2 Attirare l'attenzione dell'utente 30
2.2.3 Design dell'interfaccia e dell'interazione in AR 32
2.2.4 Potenziare la collaborazione attraverso tecniche di manipolazione e co-manipolazione di oggetti virtuali . . . 34
2.2.5 Multiplayer Augmented Reality 35
2.3 Considerazioni finali sullo stato dell'arte 36

3 Studio e definizione dei paradigmi di interazione e di *awareness* multi-utente 39
3.1 Considerazioni iniziali 39
3.2 Paradigmi di *awareness* 40
3.2.1 Informazioni visive relative alla *self-awareness* 40

3.2.2 Informazioni visive relative agli altri utenti 42
3.3 Attirare l'attenzione degli utenti 47
3.4 Interazione multi-utente 49
3.4.1 Interazione basata su sguardo 50
3.4.2 Manipolazione collaborativa 51
3.4.3 Interazione basata su posizione 53
3.4.4 Comunicazione verbale 55

III Applicazione di test **57**

4 Design dell'applicazione 59
4.1 Requisiti e problematiche 59
4.2 Soluzioni adottate 61
4.2.1 Multi-dispositivo: dispositivi utilizzati 62
4.2.2 Multi-piattaforma: le librerie AR 62
4.2.3 Multi-utente 69
4.2.4 Collaborazione in locale o remota 72
4.2.5 Possibili implementazioni di interfacce tangibili 73
4.3 Panoramica dell'applicazione scelta 73
4.3.1 Pre-scansione 77
4.3.2 Interazione con oggetti virtuali 77
4.3.3 Flusso di gioco 79

5 Sviluppo dell'applicazione 83
5.1 Strumenti e linguaggi utilizzati 85
5.1.1 Unity, VisualStudio e Sourcetree 85
5.1.2 Blender e Substance Painter 87
5.2 Modellazione, animazione e *texturing* 88
5.3 Framework di realtà aumentata 90
5.3.1 Elementi chiave 91
5.3.2 Dettagli tecnici su `ARLibrary` e `ARManager` 98
5.3.3 Implementazione dell'interfaccia con ARCore 108
5.4 Framework di networking 109
5.4.1 Piattaforme di networking considerate e implementazione con Photon Unity Networking 110
5.4.2 Sincronizzazione dell'ambiente condiviso 111
5.5 Scenari sperimentali 114
5.5.1 Interazione collaborativa basata sullo sguardo 115

5.5.2 Interazione collaborativa basata sullo manipolazione . . 118
5.5.3 Interazione collaborativa basata sulla posizione 121
5.5.4 Scenari di tipo escape room 125

IV Valutazioni **133**

6 Sperimentazione 135
6.1 Definizione del protocollo sperimentale 136
6.1.1 Obiettivi della sperimentazione e considerazioni iniziali 136
6.2 Questionari 140
6.2.1 Protocollo sperimentale completo 142
6.2.2 Semplificazioni e protocollo sperimentale finale 144
6.2.3 Somministrazione del protocollo 146
6.3 Valutazione dei risultati 147

7 Conclusioni 169
7.1 Contributi significativi 170
7.2 Sviluppi futuri 171

Appendici 173

A Questionario conoscitivo 173
A.1 Informazioni personali 173
A.2 Esperienze pregresse 173

B Questionari interazione collaborativa basata sullo sguardo 175
B.1 Visualizzazione base 175
B.1.1 Visualizzazione base: comprensibilità 175
B.1.2 Visualizzazione base: utilità delle informazioni visuali . 176
B.2 Visualizzazione completa 176
B.2.1 Visualizzazione completa: comprensibilità 176
B.2.2 Visualizzazione completa: utilità delle informazioni visuali 176
B.3 Considerazioni generali su questo tipo di interazione 177

C Questionari interazione collaborativa basata sulla manipolazione 179
C.1 Visualizzazione base 179
C.1.1 Visualizzazione base: comprensibilità 179

C.1.2 Visualizzazione base: utilità delle informazioni visuali . 180
C.2 Visualizzazione completa 180
C.2.1 Visualizzazione completa: comprensibilità 180
C.2.2 Visualizzazione completa: utilità delle informazioni visuali 180
C.3 Considerazioni generali su questo tipo di interazione 181

D Questionari interazione collaborativa basata sulla posizione 183
D.1 Visualizzazione base 183
D.1.1 Visualizzazione base: comprensibilità 183
D.1.2 Visualizzazione base: utilità delle informazioni visuali . 184
D.2 Visualizzazione completa 184
D.2.1 Visualizzazione completa: comprensibilità 184
D.2.2 Visualizzazione completa: utilità delle informazioni visuali 184
D.3 Considerazioni generali su questo tipo di interazione 185

E Questionario sulla comprensibilità in generale 187

F Questionario sulla manipolabilità in generale 189

G System usability scale 191

H Questionario sulla escape room virtuale 193

Elenco delle tabelle

2.1 Ricerche sulla collaborazione multi-utente in AR/VR. 28
3.1 Rappresentazione schematica delle interazioni multi-utente esaminate e delle possibili visual cues associabili 49
4.1 Rappresentazione schematica dei requisiti con le problematiche annesse e relative soluzioni 61
4.2 Panoramica giochi esaminati nella fase di pianificazione dell'applicazione 75
6.1 Punteggi SUS 151
6.2 Punteggi HARUS 157
6.3 Tempi medi task 165

Elenco delle figure

2.1 Grafico con la distribuzione di articoli pertinenti nel corso degli ultimi anni, pubblicati su *ScienceDirect*. 24
2.2 Da sinistra a destra: Grandi J.G. et al. in uno studio di manipolazione collaborativa di oggetti virtuali in AR, i frustum visuali in CoVAR, mano virtuale e cursore in CoVAR 26
3.1 Da queste catture schermo è possibile individuare come si è deciso di rappresentare graficamente il colore dell'utente sfruttando alcuni elementi d'interfaccia 41
3.2 Cattura schermo effettuata durante un *task* di manipolazione . 42
3.3 Rappresentazione schematica dei paradigmi di *awareness* con le *visual cues* 43
3.4 Rappresentazione dell'avatar nella parte sperimentale dell'applicazione, avatar nella *escape room* 43
3.5 Rappresentazione del frustum nell'applicazione 44
3.6 Rappresentazione della linea visiva nell'applicazione 45
3.7 Rappresentazione del cursore nell'applicazione 45
3.8 Mano virtuale nell'applicazione 46
3.9 Esempio di evidenziazione visiva nell'applicazione sperimentale e nell'*escape room* 47
3.10 Punto di interesse condiviso: a sinistra quando esso è nel campo visivo, a destra quando esso è al di fuori del campo visivo. 48
3.11 Esempio nell'applicazione di interazione con lo sguardo 51
3.12 Esempio nell'applicazione di manipolazione collaborativa . . . 52
3.13 Esempio nell'applicazione di interazione basata su posizione . 54
4.1 Esemplificazione del funzionamento delle *cloud anchors* 64
4.2 In senso orario da in alto a sinistra: Tsuro, Space Alert, Burgle Bros, Escape: The Curse of the Temple, Legends of Andor e Mansion of Madness 74
4.3 UML raffigurante l'intera struttura di gioco 81

5.1 UML raffigurante l'architettura generale dell'applicazione . . . 84
5.2 A sinistra Unity Game Engine mentre a destra Visual Studio . 86
5.3 A sinistra Blender mentre a destra Substance Painter 87
5.4 Una serie di oggetti di scena modellati in Blender e texturizzati con Substance Painter 90
5.5 Catture schermo dell'applicazione sviluppata per testare `ARLibrary` 93
5.6 A sinstra l'applicazione sviluppata per testare `ARLibrary` che presenta occlusione sui piani rilevati, mentre a destra una immagine di repertorio del *plane detection* `ARLibrary` 94
5.7 A sinistra le riflessioni dinamiche implementate da un modulo custom usando una versione di ARCore precedente alla 1.10, a destra un utilizzo di tali riflessioni in ARKit 96
5.8 Tre catture schermata dall'app di test di `ARLibrary` con *image tracking* di una immagine riprodotta sullo schermo di un PC . 98
5.9 UML semplificato 99
5.10 Catture schermo raffiguranti l'app di test, a destra il dettaglio dei pannelli con le varie impostazioni 109
5.11 Le principali piattaforme di networking prese in considerazione per l'implementazione dell'interfaccia multi-utente 110
5.12 Catture schermo raffiguranti l'app durante la fase di sincronizzazione con il marker 112
5.13 Catture schermo raffiguranti l'app durante l'esecuzione del primo task dello sguardo 116
5.14 Catture schermo raffiguranti l'app durante l'esecuzione del secondo task dello sguardo 117
5.15 Catture schermo raffiguranti l'app durante l'esecuzione del terzo task dello sguardo, a sinistra l'utente che ha il comando . 117
5.16 Catture schermo raffiguranti l'app durante l'esecuzione del primo task della manipolazione 120
5.17 Catture schermo raffiguranti l'app durante l'esecuzione del secondo task della manipolazione, a sinistra l'utente che ha il comando. Come è possibile notare, solo l'utente nell'immagine di sinistra può vedere i fori virtuali all'interno della griglia . . 121
5.18 Catture schermo raffiguranti l'app durante l'esecuzione del primo task della posizione 122
5.19 Catture schermo raffiguranti l'app durante l'esecuzione del secondo task della posizione 123
5.20 Catture schermo raffiguranti l'app durante l'esecuzione del terzo task della posizione 124

5.21 UML riassuntivo delle principali funzioni, strutture dati ed eventi forniti dalla libreria . . . 131
6.1 Dettagli di alcune catture schermo raffiguranti il cursore e il frustum . . . 137
6.2 Dettagli di alcune catture schermo raffiguranti il cursore, la linea visuale, l'avatar e la mano virtuale . . . 138
6.3 Dettagli di alcune catture schermo raffiguranti il cursore, la linea visuale, l'avatar, la mano virtuale e il punto di interesse condiviso . . . 138
6.4 Alcune catture schermo raffingurante le tre interazioni collaborative nel *setting* della sperimentazione . . . 142
6.5 Grafici a torta su informazioni riguardo a sesso e età. . . . 148
6.6 Grafici a torta riguardo alle esperienze pregresse. . . . 149
6.7 Boxplot delle domande relative al questionario SUS. . . . 150
6.8 Boxplot delle prime tre domande dell'HARUS per quanto riguarda i task relativi allo sguardo, comparando la visualizzazione base (V2) e completa (V3). . . . 152
6.9 Boxplot delle prime tre domande dell'HARUS per quanto riguarda i task relativi alla manipolazione, comparando la visualizzazione base (V2) e completa (V3). . . . 153
6.10 Boxplot delle prime tre domande dell'HARUS per quanto riguarda i task relativi alla posizione, comparando la visualizzazione base (V2) e completa (V3). . . . 154
6.11 Boxplot delle cinque domande generica sulla comprensibilità dell'HARUS. . . . 155
6.12 Boxplot delle cinque domande generiche sulla manipolabilità dell'HARUS. . . . 156
6.13 Boxplot delle domande relative all'utilità delle *visual cues*. . . . 158
6.14 Boxplot delle domande relative all'utilità dei punti di interesse condivisi. . . . 159
6.15 Grafici a torta sulla preferenza di visualizzazione nei task relativi a sguardo, manipolazione e posizione. . . . 160
6.16 Boxplot delle domande relative alla collaborazione divise per tipo di interazione. . . . 161
6.17 Boxplot dei tempi di completamento relativi ai task dello sguardo, divisi per visualizzazione e sequenza di visualizzazione. . . . 162
6.18 Boxplot dei tempi di completamento relativi ai task dello sguardo, divisi per visualizzazione e sequenza di visualizzazione. . . . 163

6.19 Boxplot dei tempi di completamento relativi ai task dello sguardo, divisi per visualizzazione e sequenza di visualizzazione. . . 164
6.20 Grafici a torta delle esperienze pregresse degli utenti per quanto riguarda le escape room e delle preferenze sugli scenari. . . 166
6.21 Boxplot delle domande relative alla usabilità della escape room. 167

Elenco dei codici

5.1 Implementazione del *singleton* di `ARManager` 100
5.2 Codice sorgente di `ARLibrary` 102

Parte I

Introduzione

Capitolo 1

Finalità del lavoro svolto

Lo scopo principale di questa tesi è lo studio delle interazioni collaborative e dei paradigmi di *awareness* multi-utente in una simulazione in realtà aumentata co-locata o remota tramite l'utilizzo di una applicazione multi-dispositivo da noi sviluppata. Al fine di portare a termine con successo questi studi, si è per prima cosa effettuato un lavoro preliminare di ricerca e analisi dello stato dell'arte per quanto riguarda la collaborazione in realtà aumentata e virtuale, per poi definire con precisione le componenti di interazione e di *awareness* multi-utente da implementare. In seguito si è proceduto con la definizione e progettazione dell'applicazione di verifica di tali paradigmi collaborativi. Poiché tra i vari requisiti che ci è posti nello sviluppo di questa applicazione figuravano anche l'indipendenza dalle maggiori librerie in commercio di realtà aumentata e di *networking*, si è proceduto con lo sviluppo di due framework (uno di AR e uno di *networking* appunto) che ci permettessero di ottenere questo risultato. Infine si è proseguito sviluppando l'applicazione finale e le applicazioni di test finalizzate alla valutazione eseguita a conclusione di questo lavoro con un gruppo di volontari. Di seguito un breve riassunto suddiviso in sezioni di tutto ciò che è stato affrontato al fine di portare a termine questa tesi.

Ricerca sullo stato dell'arte Sono state eseguite numerose ricerche relative alla collaborazione multi-utente in realtà aumentata o in realtà virtuale e anche diverse ulteriori ricerche su argomenti utili ai fini dello sviluppo dell'applicazione, ma non del tutto pertinenti con l'argomento principale della tesi. Sono stati trovati numerosi articoli scientifici e accademici che sono stati utili per individuare e selezionare i principali paradigmi di interazione multi-utente e di *awareness* in ambiente AR. Alla luce di questi studi è stato

possibile anche valutare quali argomenti relativi al tema principale della tesi non fossero stati affrontati approfonditamente: l'analisi di una interazione ibrida co-locale e remota in un ambiente puramente di realtà aumentata, oppure lo studio sull'utilizzo di metodi di attirare l'attenzione (come i *visual distractor*) all'interno di una simulazione cooperativa in AR.

Design dell'applicazione e definizione dei paradigmi d'interazione e awareness multi-utente A seguito dello studio dello stato dell'arte e all'individuazione conseguente dei principali e più efficaci paradigmi di collaborazione e *awareness* multi-utente, si è proceduto alla definizione della tipologia di applicazione da sviluppare e al suo design iniziale, con una particolare attenzione alla progettazione di scenari che sfruttassero nel modo più completo possibile i suddetti paradigmi.

Dopo una breve ricerca sui più rilevanti ambiti in cui la collaborazione tra gli utenti risultasse come componente principale dell'esperienza, si è selezionato quella che maggiormente avrebbe soddisfatto i requisiti richiesti e i nostri gusti personali. Tra i vari scenari ipotizzati figurano: scenari di progettazione condivisa, scenari di manutenzione remota o co-locata, scenari di training industriale e infine scenari di gioco. Si è quindi optato per quest'ultimo scenario ed in particolare si è deciso di impostare l'applicazione sullo stile di una *escape room*.

Dopo tali considerazioni è stato anche stabilito il numero di giocatori che la nostra applicazione avrebbe dovuto supportare e inizialmente si è pensato che tre giocatori sarebbe stato un numero accettabile per quanto riguarda un'efficace implementazione dei *task* collaborativi. Si è quindi proceduto al design di quattro casi d'uso specifici all'interno della *escape room* che permettessero di esplorare quanto più approfonditamente possibile i vari paradigmi scelti nella precedente fase di studio.

Sviluppo di framework di realtà aumentata e networking e implementazione dell'applicazione Dopo la fase di design dell'applicazione si è proseguito nell'individuare il motore di gioco da utilizzare per realizzarla, Unity, e quali delle funzioni base delle principali librerie di realtà aumentata e di *networking* sarebbero serviti per implementare l'applicazione in modo da renderla indipendente dalla piattaforma AR e dall'ambiente su cui sarebbe stata eseguita. Per prima cosa si è affrontato e risolto il problema relativo alla sincronizzazione dell'ambiente AR condiviso, poiché non sarebbe stato possibile sfruttare le varie soluzioni proprietarie di recente pubblicazione per problemi di interoperabilità tra sistemi e dispositivi diversi. Si è proceduto

quindi con la definizione di una libreria di realtà aumentata che astraesse l'uso effettivo di una specifica libreria commerciale e si è proceduto analogamente per la libreria di *networking*. Si è proceduto poi con l'implementazione dell'interfaccia relativa alla libreria ARCore di Google, avendo a disposizione dispositivi compatibili, lasciando per il futuro eventuali implementazioni di altre librerie quali ARKit di Apple e Windows Mixed Reality di Microsoft. Per quanto riguarda il *networking*, invece, si è deciso di utilizzare Photon Unity Networking, un framework per applicazioni multiplayer compatibile con Unity, e una struttura *peer-to-peer*.

Dopo aver completato lo sviluppo dei precedenti framework si è proceduto con l'implementazione effettiva della nostra applicazione sfruttando quanto più possibile tutte le peculiarità di una simulazione in realtà aumentata. Si è inizialmente sviluppato applicazioni di test per valutare la corretta implementazione delle nostre librerie e valutare l'efficacia di alcune scelte implementative effettuate e poi si è proseguito sviluppando effettivamente l'applicazione oggetto di questa tesi.

Sperimentazione e valutazione dei risultati Dopo aver terminato lo sviluppo dell'applicazione si è proseguito con la pianificazione di un protocollo sperimentale che permettesse di validare le funzioni sviluppate. Al fine di questa sperimentazione sono stati creati ulteriori scenari appositi da sottoporre ai volontari in modo che si potesse effettuare una valutazione puntuale e precisa dei vari paradigmi di interazione collaborativa e *awareness* multi-utente.

Sono stati innanzi tutto ideati alcuni scenari con il solo scopo di *training* in modo tale da permettere a ciascun partecipante di prendere familiarità con l'interfaccia e con i paradigmi di interazione base di uno scenario in realtà aumentata. In seguito si sono sviluppati i vari scenari di test, divisi per tipologia di interazione collaborativa, con alcune varianti. Si sono quindi individuati tre gruppi di informazioni visive e strumenti da fornire all'utente, chiamati visualizzazione "minimale", "base" e "completa", al fine di verificarne l'efficacia. I test sono stati infine eseguiti su 20 gruppi formati da 2 persone, che hanno ripetuto gli scenari con i vari tipi visualizzazione.

Come parametri di validazione di tale sperimentazione si è deciso di sfruttare appositi questionari per la valutazione qualitativa e opportuni dati di *log* dell'applicazione, come i tempi di esecuzione e il numero di volte che si sono utilizzate determinate funzioni, per la valutazione quantitativa. Una volta ottenuti i dati, si è proseguito con il loro processamento; si rimanda al capitolo finale per i dati definitivi e ciò che è possibile desumere da essi.

Parte II

Studi preliminari

Capitolo 2

Stato dell'arte

In un contesto collaborativo, l'uso della realtà aumentata come uno strumento di interazione e comunicazione è stato comprovato più volte essere particolarmente efficace, poiché permette di mantenere, e talvolta di potenziare, le tipiche componenti sociali e comunicative umane (per quanto riguarda principalmente un ambiente co-locato) [8]. La dimensione spaziale e le capacità multimodali di ambienti virtuali e aumentati multi-utente offrono nuove possibilità nel progettare, implementare e attuare attività collaborative.

Inoltre, metodi di interazione tangibile possono essere combinati con tecniche di visualizzazione AR per sviluppare interfacce con le quale gli oggetti fisici e le interazioni con essi risultano importanti tanto quando le immagini virtuali. Tali interfacce supportano infatti in modo naturale la collaborazione.

Attualmente esistono diversi strumenti e librerie per la realtà aumentata tra cui ARKit, ARCore, Vuforia ecc.e molti dispositivi tra cui *hand-held display* (HHD) e *hand-mounted display* (HMD). Sfruttare appieno il potenziale di questi strumenti, permettendo in particolare di utilizzarli nello stessa sessione condivisa, comporta il dover affrontare diverse sfide e criticità. Si vogliono quindi esplorare possibili soluzioni a questi problemi (empirici e tecnici) con un particolare focus su device AR di tipo HHD e HMD come Microsoft HoloLens e ambienti condivisi locali o distribuiti.

Di seguito una breve introduzione di ciò che è stato trovato a riguardo, i vari articoli e ricerche che trattano di interazioni collaborative in ambienti AR/MR e paradigmi di interazione multimodali (multi-device). Nonostante nelle varie *review* esaminate siano stati menzionati diversi lavori rilevanti (per esempio [8][9][14]), non è presente un articolo rigoroso che si focalizzi sullo specifico argomento che è alla base di questa tesi; di conseguenza l'obiettivo

di questo primo capitolo è quello di colmare questa mancanza fornendo un riassunto dello stato dell'arte dei principali studi sull'AR collaborativa.

2.1 Distribuzione degli articoli e parametri di ricerca

Per raggiungere lo scopo di questo capitolo, sono state esaminati numerosi database elettronici alla ricerca di articoli pubblicati tra il 2000 e il 2018. La le seguenti parole chiave sono state utilizzate per effettuare questa ricerca: *augmented and mixed reality*, *collaborative learning and education*. Sono stati inclusi, al fine di identificare pubblicazioni rilevanti, anche conference paper dai database di IEEE e Google Scholar. Successivamente sono state selezionate le pubblicazioni relative solamente al *collaborative AR learning*. In conclusione, sono state effettuati dibattiti, letture approfondite e son stati presi appunti sui punti chiave ricavati da queste pubblicazioni al fine di rispondere ai quesiti di questa ricerca.

Di seguito, in figura 2.1, il grafico riassuntivo relativo al sito *ScienceDirect* con la distribuzione delle pubblicazioni nel tempo, mostrando un crescente interesse nella realtà aumentata collaborativa.

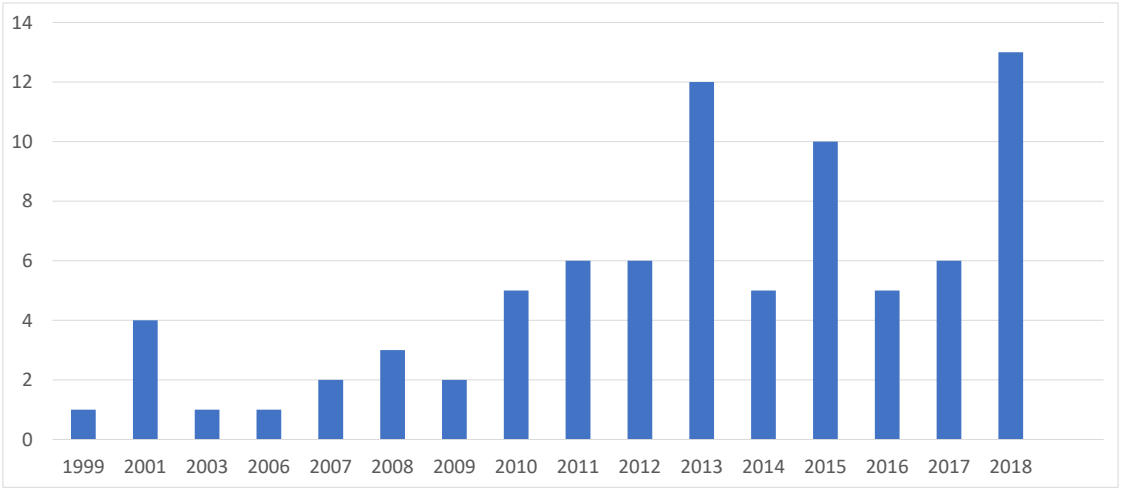


Figura 2.1. Grafico con la distribuzione di articoli pertinenti nel corso degli ultimi anni, pubblicati su *ScienceDirect*.

2.2 Progettazione di un ambiente AR collaborativo

Di seguito si sono organizzati in sotto-argomenti i vari aspetti da prendere in considerazione riguardanti i nuovi paradigmi di interazione e le varie problematiche che possono sorgere nell'implementazione di un ambiente AR multi-utente collaborativo. Per ciascuno di questi aspetti sono state analizzate quali, delle varie soluzioni trovate, risultano le più efficaci relativamente ai vari dispositivi e gruppi di utenti co-locati o remoti, considerando dapprima i metodi per veicolare la comunicazione, attirare l'attenzione degli utenti, le modalità di interazione utilizzabili in contesto AR e infine le problematiche relative alla sincronizzazione multi-utente e multipiattaforma.

2.2.1 Comunicazione naturale (*seamless*) multi-utente, monitoraggio e rappresentazione dell'utente per potenziare interazione e *awareness*

La base della collaborazione è la comunicazione, che, al fine di risultare efficace, ha bisogno di una serie di *visual cues* (sguardo, gesti delle mani, sincronizzazione labiale, espressioni facciali, posizione del corpo) a *verbal cues* (voce, intonazione, prosodia). Un contesto collaborativo in realtà aumentata deve garantire quindi che gli utenti siano in grado di comunicare nel modo più naturale possibile, assicurando il maggior numero possibile di indicazioni e limitando al minimo la presenza di discontinuità di tipo funzionale e cognitivo. Un vantaggio infatti della realtà aumentata è quello di permettere la coincidenza di *task space* e *communication space*, contrariamente a quanto accade di solito in una collaborazione *computer supported*, in cui è spesso difficile per gli utenti scambiarsi non *verbal cues*, anche in ambiente co-locato [7].

Infatti, l'efficacia nel compiere un task collaborativo in AR rispetto a uno scenario standard è stata ben comprovata da studi passati e recenti, sia in ambiente locale sia in remoto [9][62][11][3][10][12][20]. In queste ricerche si è visto come la collaborazione con tecnologie AR genera comportamenti che sono più simili alla naturale collaborazione *face-to-face* a differenza di come avverrebbe utilizzando una interfaccia *screen-based*; infatti, la comunicazione (di tipo visuale e verbale) è praticamente analoga tra ambiente AR colocato e *face-to-face* con l'unica differenza che i dispositivi AR utilizzati negli studi più datati erano limitati (in termini di potenza di calcolo e qualità visiva) e

causavano quindi una riduzione significativa di alcune tipologie di *cues*, compromettendo in parte la qualità percepita della collaborazione. È comunque importante sottolineare che, nonostante i progressi tecnologici, anche con dispositivi recenti si possono notare riduzione di *cues* per questioni analoghe (potenza di calcolo, *field-of-view*, risoluzione e dimensione dello schermo), sia per HMD e a maggior ragione per HHD.

Innanzi tutto, la voce e le *verbal cues* sono il metodo più immediato e naturale per comunicare un concetto e risultano di conseguenza alla base della comunicazione. Fortunatamente esse sono anche facilmente convogliabili in un ambiente AR sia locale sia remoto senza perdite di incisività.

Le informazioni aggiuntive convogliate dalle comuni *visual cues* sono comunque molto importanti per le interazioni sociali. In contesto AR, al fine di migliorare la comunicazione, è possibile quidni rendere più evidenti alcune di esse (per esempio quelle più significative, come il *Field of view* o lo sguardo) e introdurne di nuove, per fornire informazioni aggiuntive relative agli altri utenti, monitorandone visivamente lo stato e le attività. Ciò è utile al fine di rendere l'interazione il più naturale possibile e migliorare la comunicazione rispetto a un semplice contesto *face-to-face*.

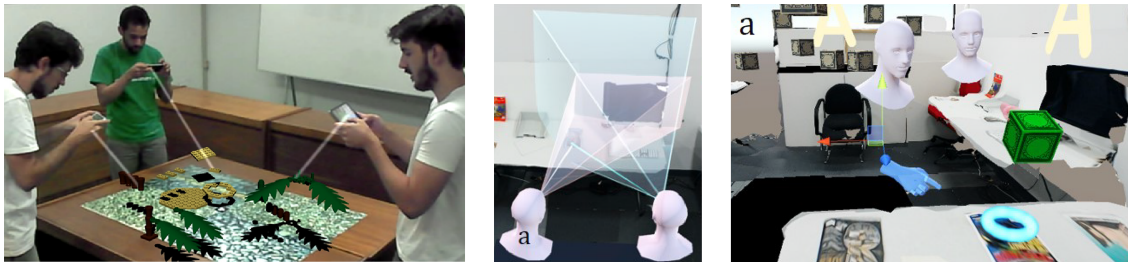


Figura 2.2. Da sinistra a destra: Grandi J.G. et al. in uno studio di manipolazione collaborativa di oggetti virtuali in AR, i frustum visuali in CoVAR, mano virtuale e cursore in CoVAR

In un contesto AR collaborativo co-locato le *visual cues*, per ovvi motivi, non risultano particolarmente problematiche, soprattutto con HMD, se non nel caso di limitazioni relative all'hardware o, nel caso in cui il contesto virtuale risulti troppo invasivo, andando ad occludere informazioni importanti del mondo reale. È bene notare che, nonostante l'HMD sia spesso visto come un dispositivo generalmente intrusivo e limitante, alcuni studi [21] hanno dimostrato come in un contesto *task oriented* e professionale il

decremento del contatto visivo dovuto a questi dispositivi non ha un effetto diretto significativo sulla collaborazione. Per quanto riguarda dispositivi *hand-held*, invece, le *visual cues* sono trasmesse in modo naturale al di fuori dello schermo e quindi non presentano punti critici, se non il dover dividere la concentrazione tra la finestra sul mondo aumentato, di dimensione molto ristretta, e il mondo reale, creando così una lieve separazione tra il *task space* e il *communication space*. Nonostante ciò, grazie all'introduzione di elementi aumentati, è possibile creare un sistema efficace per aumentare la consapevolezza nei confronti degli altri utenti, come mostrato nell'esperimento di J. G. Grandi et al. [39][38][43], in cui ciascun utente è in grado di visualizzare gli oggetti manipolati dagli altri utenti per mezzo di raggi virtuali tracciati dai dispositivi *hand-held* agli oggetti stessi.

Per quanto riguarda invece scenari remoti in cui uno o più utenti risultino fisicamente separati dagli altri è essenziale avere una rappresentazione quando più fedele possibile, o quantomeno significativa, dell'utente remoto. In molti casi, però, [31][44][20], data la complessità di una rappresentazione di questo tipo, si è scelto di non rappresentare gli utenti remoti, limitandosi ad un feedback visivo ottenuto tramite videocamera [22][29].

Uno dei vantaggi di questi scenari rispetto a quelli locali è la possibilità di scalare le dimensioni degli avatar, rendendoli più o meno ingombranti nella scena a seconda delle necessità. La possibilità di avere collaborazione AR remota ha senso qualora ci siano utenti che vogliano partecipare remotamente a una sessione co-locata; questi utenti però, essendo remoti, hanno la possibilità di prenderne parte in AR o in VR. Una configurazione totalmente AR non è stata ancora esplorata e quindi la maggior parte degli esempi considerati di seguito prevedono una configurazione ibrida AR/VR o esclusiva VR [27][36].

CoVAR [41][42] rappresenta gli utenti remoti come mezzibusti fluttuanti monocromatici provvisti di mani virtuali impostate su quattro *gesture* standard (*neutral*, *pointing*, *grasping*, *thumbs-up*), ciascuna associata ad un colore diverso, che consentono quindi il controllo delle attività degli altri utenti. Per il monitoraggio dell'utenza è anche previsto un sistema di visualizzazione della direzione dello sguardo e del campo di vista.

Autori	Sistema	Caratteristiche	Dispositivi
Billinghurst M. et al. [14], 1998	Shared Space	Studio sulla collaborazione AR face-to-face, multi-utente	HMD (video see-through)
Poupyrev I. et al. [62], 2001	Tiles		
Hedley N. R. et al. [11], 2002	AR Prism		
Billinghurst M. et al. [10], 2002	Urban Design Experiment		
Regenbrecht H.T. et al. [12], 2002	MagicMeeting		HMD (video see-through), PDA per l'interazione
Grandi J.G. et al. [39][38], 2017	Collaborative manipulation of 3D virtual objects	Studio sulla collaborazione AR face-to-face, multi-utente, monitoraggio utente con raggi virtuali	HHD (smartphones)
Piumsomboon T. et al. [41][42], 2017	CoVAR	Studio sulla collaborazione AR/VR remota, 1 utente AR + 1 utente VR, visualizzazione del FOV e dello sguardo utente, rappresentazione semplice dell'utente remoto, ricostruzione vitruale dell'ambiente reale dell'utente AR	HoloLens per la ricorstruzione 3D, HMD (HoloLens e HTC Vive) per la visualizzazione, eye-tracking (Pupil Labs)
Piumsomboon T. et al. [46], 2018	Mini-Me	Studio sulla collaborazione AR/VR remota, 1 utente AR + 1 utente VR, rappresentazione realistica dell'utente remoto, gesti e sguardo reindirizzati su una rappresentazione miniaturizzata dell'utente remoto, ricostruzione virtuale dell'ambiente reale dell'utente AR	
AltspaceVR inc. [60], 2016	AltspaceVR	Applicazione collaborativa in VR remota, multi-utente, rappresentazione personalizzabile dell'utente, sincronizzazione labiale tramite processamento del parlato	HMD (HTC Vive, Oclulus Rift)

Tabella 2.1. Ricerche sulla collaborazione multi-utente in AR/VR.

Autori	Sistema	Caratteristiche	Dispositivi
Facebook [51], 2017	Facebook Spaces	Applicazione collaborativa in VR remota, multi-utente, rappresentazione personalizzabile dell'utente, retargeting della cinematica inversa dai movimenti delle mani, raccolta di espressioni facciali scelte tramite hand gestures o controller, aumento della presenza introducendo artifici come perdita casuale del contatto visivo o simulazione dello sbattere le palpebre, sincronizzazione labiale tramite processamento del parlato con IA	HMD (HTC Vive, Oclulus Rift)
Mavridou I. et al. [40][55], 2017	FACETEQ	Studio sul riconoscimento delle emozioni	Piattaforma indossabile FACETEQ + HMD
Orts-Escolano S. et al. [36], 2016	Holoportation	Studio sulla collaborazione AR/VR remota, due utenti, ricostruzione 3D real-time di alta qualità di un intero spazio di lavoro includendo persone, arredamento e oggettistica	Camere di profondità ed RGB multiple per la risoctruzione 3D, HMD (HoloLens) per la visualizzazione

Mini-Me [46] implementa una rappresentazione più raffinata dell'avatar virtuale dell'utente remoto. Al fine di aumentare il senso di presenza e di ovviare al problema del ristretto campo di vista dei dispositivi AR *head-mounted*, una copia in miniatura dell'avatar viene mantenuta costantemente in una parte del campo visivo; su questa copia vengono reindirizzati sguardo e gesti dell'utente.

Dai due precedenti progetti è stato possibile trarre risultati promettenti per quanto riguarda l'uso di rappresentazioni virtuali e di *awareness cues*. Le *awareness cues* si sono rivelate molto utili in addizione alla visualizzazione dell'avatar, poiché permettono agli utenti di trovare più velocemente la *gaze area* dei collaboratori grazie alla *FOV cue* e di individuare l'esatto bersaglio

dello sguardo tramite il *gaze-ray cue*. Relativamente all'uso del Mini-me si è notato come questo sia in grado di migliorare la consapevolezza di ciascun utente del proprio partner, dando la possibilità di controllare costantemente lo stato del collaboratore e di migliorare la sua *social presence* convogliando diverse *cue* non verbali.

In ambiente prettamente social network, anche soluzioni come **Facebook Spaces** o **AltspaceVR** permettono rappresentazioni naturali di utenti remoti, con avatar virtuali impostati e personalizzati dagli utenti. In particolare, Facebook Spaces, data l'incapacità degli attuali headset AR/VR commerciali di rilevare emozioni, sfrutta una libreria di espressioni facciali standard da richiamare tramite gesti o controller per far trasparire le proprie emozioni sulla rappresentazione virtuale di sé stessi [60][51].

Per migliorare ulteriormente il senso di presenza di utenti remoti sono stati anche effettuati alcuni studi relativi all'emotion sensing, in particolare **Affective Wear** e **FACETEQ by Emteq** [55][40], che, tramite l'uso di particolari sensori, si avvale di una gamma di diverse tecniche di rilevamento facciale, che includono il rilevamento dell'attività elettrica muscolare o della prossimità muscolare, rilevamento del movimento oculare, monitoraggio della frequenza cardiaca e delle variazioni della frequenza cardiaca, monitoraggio dello stress e della posizione della testa.

Una soluzione che permette una rappresentazione praticamente perfetta di utenti e oggetti remoti, in contesto esclusivamente di realtà aumentata, è l'**Holoportation** di Microsoft Research che permette una telepresenza in tempo reale estremamente raffinata e fedele con l'utilizzo di multiple camere di profondità per l'acquisizione di modelli tridimensionali e HoloLens per la loro visualizzazione [36].

2.2.2 Attirare l'attenzione dell'utente

Attirare l'attenzione di uno o più utenti su un elemento specifico dell'ambiente risulta particolarmente problematico nei casi in cui siano presenti utenti remoti e/o dotati di dispositivi *hand-held*, in quanto richiedono accorgimenti particolari. È opportuno distinguere due aspetti: **come individuare l'oggetto (reale o virtuale) selezionato da un utente** e **come segnalare l'oggetto selezionato agli altri utenti (locali o remoti)** in modo da attirare la loro attenzione.

Per quanto riguarda il primo aspetto le soluzioni proposte sono molteplici e comprovate sia per dispositivi *hand-held* che dispositivi *head-mounted*. Metodi di tipo *ray casting* sono utilizzati per identificare l'oggetto selezionato (o parte specifica di esso) sia che l'input corrisponda ad un semplice *tap* sullo schermo, al tracciamento della testa o sguardo, ad una *hand gesture* [41][42] o ad una interazione con controller. Bisogna però sottolineare le problematiche che sorgono quando l'oggetto puntato non è virtuale, ma reale; in questo caso è necessaria una ricostruzione 3D dell'ambiente reale come per esempio avviene tramite la camera di profondità delle HoloLens: in CovAR e Mini-Me l'ambiente ricostruito viene trasmesso in streaming all'utente remoto in realtà virtuale; Skype fornisce la possibilità di interagire da remoto con l'ambiente virtuale ricostruito dall'interlocutore con finalità di tele-assistenza in tempo reale [29][61]. È quindi possibile, avendo il modello della scena, identificare porzioni dello spazio in modo semplice tramite i precedenti algoritmi di *ray casting*. Un'altra tecnica interessante in tale ambito è il *pixel-point-volume segmentation* [34], che permette, sempre tramite una opportuna scansione 3D dell'ambiente, a utenti posti a diversi punti di vista, di interpretare riferimenti ad oggetti, specificati tramite *gesture* circolari su uno schermo tattile.

Esistono diversi metodi per la segnalazione ad altri utenti di oggetti selezionati che sono completamente *out-of-sight* e che quindi guidino l'attenzione (sguardo) dell'utente. Il più comune, utilizzato anche in contesto videoludico o VR, è la tecnica del *distractor* [37] che consiste nell'inserimento all'interno del campo di vista, tipicamente ai margini, di un *visual distractor* (una freccia, una icona ecc.) che informi l'utente su dove dirigere il proprio sguardo per inquadrare completamente l'oggetto segnalato. In aggiunta al *visual distractor* è possibile anche introdurre un *distractor* di tipo sonoro [18] che, sfruttando la spazializzazione dell'audio, fornisca *hint* sulla posizione approssimativa dell'oggetto considerato. Ciò diventa problematico però in contesto collaborativo, soprattutto co-locato, in quanto, dato che l'uso di auricolari risulta scomodo per ovvi motivi, la segnalazione acustica potrebbe sovrastare la voce degli altri e causare confusione. Non è comunque da escludere una sperimentazione con auricolari che permettano di non isolare completamente gli utenti in modo che sia comunque possibile la comunicazione nell'ambiente co-locato. Un'altra tecnica proposta è il cosiddetto *ominidirectional attention funnel*[15]: esso si configura come una guida virtuale consistente in una serie di figure rettangolari che procede seguendo una curva definita tra la direzione di vista dell'utente e l'oggetto di interesse.

Infine, una volta che l'utente ha diretto lo sguardo verso il punto desiderato, è consigliabile che l'elemento segnalato venga opportunamente posto in risalto tramite alcune tecniche visive che prevedono l'uso di un cursore o di *visual highlighting*.

Si è notato però che l'applicazione di queste tecniche potrebbe portare ad alcune problematiche relative soprattutto all'ingombro degli elementi visivi che, oltre ad andare potenzialmente a limitare la visibilità (già ridotta in caso di visori AR dal ristretto FOV come HoloLens), potrebbe essere visivamente troppo pesante da gestire per l'utente.

2.2.3 Design dell'interfaccia e dell'interazione in AR

É opportuno premettere che in una applicazione in realtà aumentata risulta essenziale definire tecniche di interazione, a prescindere dal fatto che l'applicazione sia collaborativa.

L'interagire con il mondo aumentato può risultare difficile se non si implementano correttamente tecniche di interazione efficaci. Nella realtà aumentata esiste un forte legame tra gli oggetti virtuali e quelli fisici; ciò suggerisce che una direzione promettente per una buon design dell'interfaccia e dell'interazione potrebbe essere quella di trarre vantaggio dell'immediatezza e familiarità degli oggetti fisici quotidiani [14]. Quindi, per ottenete un tipo di interazione *seamless* tra mondo reale e virtuale è possibile ed auspicabile utilizzare delle *tangible user interface* (TUI). Queste Interfacce sono molto potenti in contesto collaborativo, in quanto gli oggetti fisici sono parti integranti del processo di comunicazione *face-to-face*. Mostrare informazioni su una TUI però risulta difficoltoso ed è altresì difficile cambiare le proprietà fisiche dell'oggetto.

L'AR permette di risolvere alcune limitazioni delle TUI con la metafora di interazione chiamata *Tangible Augmented Reality*, in cui si lega un oggetto reale a uno virtuale; in questo modo vengono fornite vere registrazioni spaziali e presentazioni di oggetti virtuali ovunque nell'ambiente fisico, mentre, allo stesso tempo, consente agli utenti di interagire con questi contenuti virtuali utilizzando le stesse tecniche utilizzate con un reale oggetto fisico [10]. L'AR, infatti, sfruttando algoritmi di computer vision, permette di identificare posizione e orientamento di marker o oggetti reali (con particolari feature e precedentemente scansionati), per aumentare direttamente questi elementi (ancorando ad essi in modo semplice oggetti virtuali) e utilizzarli quindi come strumenti di interazione e visualizzazione.

Dunque, una ideale interfaccia tangibile AR faciliterebbe una interazione ed una visualizzazione *seamless*. Son stati effettuati diversi studi [10][9][16][17] per analizzare l'efficacia di interfacce tangibili per la visualizzazione e manipolazione di oggetti virtuali durante un task collaborativo. In questi esperimenti sono stati utilizzati marker stampati come *input devices* e son state utilizzate tecniche di interazione 3D spaziali (per esempio la *object proximity*) e interazioni sia *time* che *space-multiplexed*. In particolare, con una interfaccia *space-multiplexed*, ciascuna funzione che influenza l'ambiente virtuale è assegnata ad un singolo dispositivo reale che occupa un proprio spazio fisico. Al contrario, in un contesto *time-multiplexed* un singolo dispositivo reale controlla differenti funzionalità in differenti intervalli di tempo [14].

Da questi studi, nonostante le limitazioni tecnologiche dell'epoca, si è evinto che la metafora di interazione di tipo tangibile è valida ed efficace e che gli utenti sentivano di poter raccogliere e spostare oggetti in un ambiente AR con la stessa facilità di una condizione *face-to-face*.

Alcuni usi interessanti di TUI in contesto AR sono **Magic Paddle** [5], in cui uno strumento a forma di pagaia viene utilizzato come *time-multiplexed device* per la manipolazione di oggetti virtuali, e **Checkmate** [44], in cui tramite l'uso di una superficie tattile e interagendo con particolari modelli di scacchi stampati 3D è possibile giocare a scacchi con un utente remoto.

I progetti **Art of Defense** e il **lavoro di Ulbricht et al.** [19][13] sono invece esempi emblematici di AR tangibile collaborativa in un contesto *board game*. Entrambi sfruttano marker fisici per l'ancoraggio e la manipolazione dei modelli (Art of Defense come tile di costruzione in un *tower defense game*, mentre il lavoro di Ulbricht et al. come strumento per lo spostamento di mulini e catapulte virtuali).

Analizzando le soluzioni trovate, nonostante gli eccellenti risultati in scenari collaborativi locali, si possono riscontrare problematiche relative a configurazioni remote, in quanto, nonostante il paradigma di interazione tangibile rimanga una buona soluzione come metodo di input prettamente non collaborativo, è difficile, se non impossibile, sincronizzare e gestire un ambiente di AR tangibile in remoto dotato di oggetti fisici. Un'ulteriore limitazione riguarda la necessità di utilizzare oggetti fisici specifici, che devono essere registrati o creati ad hoc. Si è proposto quindi di utilizzare metodi di interazione tangibile alternativi, come quello basato sull'uso diretto e naturale delle mani [23][30][32]: gli utenti sono in grado di interagire con contenuto virtuale con le loro mani, afferrare oggetti virtuali e passarli ad altri utenti

come farebbero con oggetti reali. Questo setup richiede però un sistema di tracking delle mani, che può essere ottenuto tramite l'utilizzo di marker (colorati e non), metodi di segmentazioni delle immagini basate sul colore della pelle o di sistemi di scansione 3D (Kinect, Leap Motion); questi ultimi sono quelli che forniscono i risultati più accurati ed efficaci.

Per quanto riguarda i dispositivi AR *hand-held*, l'interazione tangibile può presentare problematiche ed è stata valutata in alcune ricerche [30][32][33][26][45]. Si è notato come gli utenti siano limitati da due fattori: essi devono tenere il dispositivo almeno con una mano, limitando quindi all'altra mano le possibili interazioni con oggetti fisici; gli utenti, inoltre, tendono a osservare spesso la scena attraverso la camera del proprio dispositivo e non direttamente, rendendo problematica, data la limitata dimensione dello schermo, una interazione fisica secondaria. Per questi motivi l'interazione con lo schermo tattile del dispositivo *hand-held* può rimanere una valida alternativa, data la familiarità degli utenti con i gesti *touch* [24]. Un metodo interessante di interazione 3D per dispositivi *hand-held* [39][38] consiste nell'utilizzare, oltre alle tipiche *touchscreen gestures*, la posizione e l'orientamento del dispositivo per la manipolazione di parametri fisici dell'oggetto: è possibile, quindi, traslare e ruotare gli oggetti virtuali aumentati collegandoli alla posa attuale del dispositivo.

2.2.4 Potenziare la collaborazione attraverso tecniche di manipolazione e co-manipolazione di oggetti virtuali

Si procede con l'analisi di alcune tecniche significative di manipolazione collaborativa di oggetti virtuali (rotazione, traslazione, scalamento): l'**esperimento di Grandi J.G. et al.** [39][38] prevede la manipolazione simultanea dello stesso oggetto virtuale agendo su DOF differenti; in **ColCo** [35] gli oggetti virtuali presentano 8 *handles* e gli utenti collaborano dapprima per concordare sull'oggetto un *constraint* definendo un asse di trasformazione, per poi applicare un'operazione relativa a questo asse tramite conferma unanime.

Esistono altre soluzioni che permettono di potenziare il grado di collaborazione tra gli utenti, in particolar modo introducendo speciali oggetti virtuali o tecniche con il solo scopo di incentivare la coordinazione. **CoVAR** [41][42], per esempio, tramite una tecnica chiamata *collaborative gaze*, offre

la possibilità di selezionare oggetti virtuali e innescare (*trigger*) azioni specifiche coordinando gli sguardi dei due utenti. Essendo questo ambito poco esplorato, esso potrebbe essere un valido argomento di ricerca.

2.2.5 Multiplayer Augmented Reality

La possibilità di condividere esperienze AR con uno o più utenti in un ambiente locale collaborativo o competitivo è comunemente nota come *multiplayer AR* e può essere eseguita in modo asincrono o sincrono (in tempo reale). Ai fini di questo capitolo sarà considerata solo la collaborazione AR multiplayer in tempo reale.

Per far funzionare efficacemente l'AR multigiocatore, è necessario esaminare alcune considerazioni iniziali. I dispositivi utilizzati durante la sessione AR devono conoscere la posizione l'uno rispetto all'altro; questi dati di localizzazione devono essere coerenti durante le trasformazioni dei dispositivi e possono essere sincronizzati tramite un servizio cloud o una connessione di rete peer-to-peer, ad esempio. È anche preferibile che qualsiasi ricostruzione del mondo e informazioni semantiche che ogni dispositivo possiede sia condivisa con altri dispositivi.

Queste considerazioni portano alla conclusione che i principali problemi che devono essere risolti riguardano le sfide relative alla *computer vision*, in particolare il problema della **relocalizzazione** e la necessità di una **infrastrutture cloud** in grado di gestire i dati di localizzazione.

Relocalizzazione Consiste generalmente nella ricerca attraverso una mappa SLAM, che rappresenta un'area fisica, di dove il dispositivo attuale si trova relativamente a un altro dispositivo. Più è lontana la posizione relativa dei dispositivi, più è complesso questo processo. Attualmente i sistemi di *deep learning* stanno dando risultati importanti per migliorare questo processo riducendo le finestre di ricerca per la rilocalizzazione in ampie aree distanti dall'utente iniziale. Esistono tre modi per eseguire una rilocalizzazione per i sistemi di tracciamento *inside-out*:

- Utilizzare solo il sistema di coordinate GPS di latitudine e longitudine (molto impreciso a causa dell'elevato errore nella definizione di una posizione GPS che può essere di molti metri);

- Utilizzare una comune immagine reale di tracciamento (marker) che rappresenti l'origine comune del sistema di coordinate tra tutti i dispositivi (preciso, ma privo di flessibilità a causa dell'oggetto fisico che rappresenta il marker);
- Condividendo le mappe SLAM tra i dispositivi, gli utenti devono tenere i propri dispositivi nella stessa posizione per finalizzare correttamente la rilocalizzazione (accurata, ma l'esperienza utente non è ottimale).

In un ambiente AR multiplayer, l'esperienza utente ha una rilevanza capitale. Ci sono diversi requisiti che l'esperienza utente deve soddisfare per definire un AR multiplayer coerente ed efficace:

- Il sistema di sincronizzazione SLAM deve funzionare correttamente indipendentemente dalla distanza o dagli angoli relativi tra i giocatori;
- Ridurre al minimo o eliminare la fase di pre-scansione che può annoiare l'utente se non gestita correttamente;
- Avere un allineamento accurato delle mesh virtuali tra gli utenti;
- Nessun tempo di caricamento, l'utente non deve aspettare;
- L'esperienza dovrebbe funzionare su più piattaforme;
- La privacy degli utenti deve essere salvaguardata.

L'implementazione di una infrastruttura cloud coerente dovrebbe comprendere tutti questi requisiti al fine di fornire un AR multiplayer ottimale [57]. Attualmente ci sono diverse *startup* che propongono le loro soluzioni e anche le più grandi piattaforme di realtà aumentata come ARCore di Google e ARkit di Apple iniziano ad adottare questo paradigma.

2.3 Considerazioni finali sullo stato dell'arte

Questa primo capitolo contribuisce a definire le linee guida per la creazione di applicazioni AR collaborative. Queste linee guida possono essere quindi utilizzate per aiutare ricercatori e sviluppatori a comprendere le problematiche e sviluppare possibili soluzioni per *task* collaborativi in contesti remoti o co-locati, con particolare attenzione a dispositivi *hand-held* e HMD.

In particolare, l'efficacia dell'*augmented reality* in contesto locale cooperativo è stata appurata da molti studi, mentre gli scenari remoti in AR, presentando numerose problematiche legate alla sincronizzazione di ambienti reali diversi, sono stati poco esplorati, essendo complessi da realizzare e

quindi preferendo soluzioni ibride AR/VR o esclusivamente VR. Per quanto riguarda i dispositivi utilizzati, invece, la crescente diffusione di smartphone e tablet sempre più potenti ha dato la possibilità di fruire della realtà aumentata con costi relativamente contenuti tramite HHD. Essi presentano però, come già esaminato, diverse limitazioni nei campi dell'interazione e della visualizzazione, che degradano inevitabilmente l'efficacia della cooperazione; tali problematiche sono in parte risolte da dispositivi HMD, più costosi e non ancora privi di difetti.

Per garantire un'efficace collaborazione tra diversi utenti sarebbe consigliato anche assicurare un certo grado di adattabilità, permettendo l'uso contemporaneo di dispositivi AR di diverso tipo all'interno dello stesso *task* o permettendo un'esperienza *seamless* tra una combinazione di utenti remoti e co-locati. In mancanza ancora di un framework AR universale indipendente dalla piattaforma (sono presenti tuttavia soluzioni a riguardo in fase di pubblicazione), risulta necessario adottare un sistema in grado di effettuare un *mapping* tra la rappresentazione del mondo generata da un dispositivo e quella generata dagli altri basata su altre piattaforme di realtà aumentata. In campo strettamente AR non sono stati effettuati esperimenti a riguardo, mentre in campo MR si possono trovare alcuni esempi [41][42][46] che combinano visori di VR e di AR.

Capitolo 3

Studio e definizione dei paradigmi di interazione e di *awareness* multi-utente

3.1 Considerazioni iniziali

Lo scopo della tesi, come già esposto, consiste nello sviluppo di una applicazione in realtà aumentata multiutente collaborativa, multipiattaforma e co-locata compatibile con più dispositivi. Prima di procedere con la descrizione dettagliata dell'applicazione e dello sviluppo delle sue varie componenti, è opportuno illustrare con questo capitolo, alla luce dell'analisi dello stato dell'arte del capitolo precedente, lo studio che è stato condotto e le scelte che sono state compiute al fine di implementare una esperienza AR collaborativa quanto più efficace e consistente possibile.

Anche se tali argomenti verranno approfonditi in capitoli successivi è opportuno anticipare brevemente alcuni concetti che potranno risultare utili al fine di comprendere appieno gli argomenti trattati in questo capitolo. Prima di sviluppare effettivamente l'applicazione oggetto di questa tesi, seguendo una filosofia di interoperabilità e indipendenza dalle maggiori piattaforme, sono stati scritti due *framework* al fine di astrarre l'utilizzo di una specifica libreria di realtà aumentata o di *networking*. Ciò ha permesso di sviluppare l'applicazione utilizzando direttamente le interfacce da noi scritte. Poiché l'implementazione di tutte le interfacce relative alle varie librerie AR in commercio avrebbe richiesto una mole di lavoro eccessiva ai fini di questo lavoro,

si è optato per implementare solamente l'interfaccia della libreria ARCore di Google. Quindi, quando di seguito si farà riferimento a sessione AR e piani rilevati, per quanto ciò che è scritto abbia piena validità anche se si considerassero altre librerie, si farà riferimento a sessione ARCore e rilevamento piani di ARCore.

In questo capitolo quindi si andranno ad elencare le informazioni visive rilevanti relative all'*awareness* degli utenti, le modalità di attirare l'attenzione e le tipologie di interazioni che sono state implementate nell'applicazione.

3.2 Paradigmi di *awareness*

La consapevolezza di sè e degli altri utenti in un ambiente AR collaborativo è di fondamentale importanza al fine di rendere l'esperienza quanto più gratificante, immersiva e consistente possibile. É opportuno quindi progettare una interfaccia che consenta di aumentare l'*awareness* di ciascun utente nei confronti degli altri in modo tale da rendere la collaborazione il più semplice possibile ed incrementarne di conseguenza l'efficacia rispetto ad un classico contesto *face-to-face*. É possibile distinguere la *self-awareness* dalla *awareness* relativa agli altri utenti e per quest'ultima è possibile effettuare una ulteriore distinzione in *awareness* di posizione e orientamento e *awareness* di attività.

Per quanto riguarda la *self-awareness* è imperativo riuscire a segnalare correttamente all'utente, con opportuni feedback, che operazioni sta eseguendo, quali operazioni può eseguire e in che punto della simulazione sta indirizzando il proprio dispositivo. A tal proposito è stata studiata una interfaccia utente con l'obiettivo di risolvere nel modo più efficace possibile queste criticità, sfruttando le *visual cues* e quanto appreso dal precedente lavoro di ricerca.

Inoltre, per la distinzione tra i vari partecipanti alla simulazione si è scelta una soluzione tipica che consiste nell'adottare un opportuno codice colori associando a ciascun utente un colore univoco, come si può notare in figura 3.1. Tale colore viene poi utilizzato anche per renderizzare le varie *visual cues* associate.

3.2.1 Informazioni visive relative alla *self-awareness*

Di seguito verranno elencati brevemente gli accorgimenti studiati al fine di garantire la *self-awareness*. Per prima cosa, un cursore semi-trasparente e

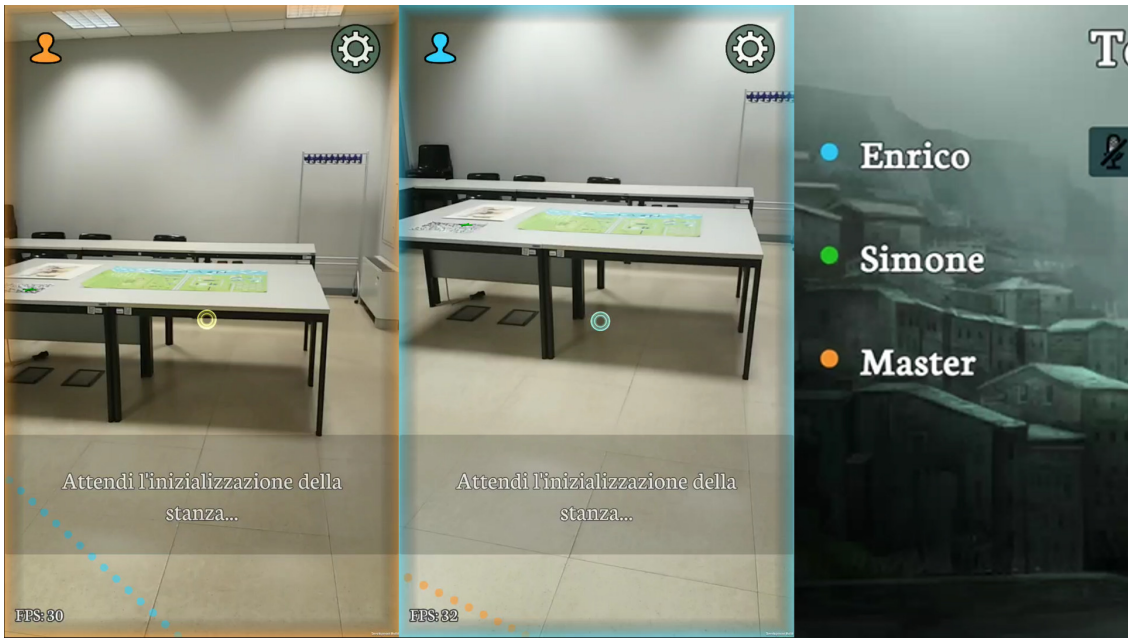


Figura 3.1. Da queste catture schermo è possibile individuare come si è deciso di rappresentare graficamente il colore dell'utente sfruttando alcuni elementi d'interfaccia

poco invasivo dovrebbe essere proiettato sulle superfici rilevate e sugli oggetti virtuali nel punto in cui il vettore direzione del dispositivo li interseca. Questo permetterebbe all'utente di sapere sempre con esattezza cosa sta puntando con il proprio dispositivo. Quando questo cursore viene proiettato su un oggetto virtuale che può essere manipolato e quindi definito "interattivo", tale oggetto dovrebbe essere evidenziato tramite una tecnica di *visual highlighting* al fine di trasmettere all'utente che può essere intrapresa una azione su quell'oggetto.

Nella nostra applicazione, volendo dare la possibilità di manipolare alcuni oggetti virtuali, si è scelto per semplicità di impedire la manipolazione contemporanea di questi oggetti; si è quindi introdotto il concetto di proprietà di un oggetto: se si vuole manipolare un oggetto è necessario prima prenderne possesso e finché non si riappoggia su una superficie, esso può essere manipolato solo dal suo proprietario.

Quando un oggetto virtuale interattivo è in fase di manipolazione, è opportuno indicare in qualche modo all'utente, con un elemento dell'interfaccia, che tale oggetto è attualmente di sua proprietà. Tra le varie soluzioni esaminate, si è scelto di considerare un metodo di *visual highlighting* ovvero



Figura 3.2. Cattura schermo effettuata durante un *task* di manipolazione

creare sull'oggetto un contorno del colore dell'utente e l'utilizzo di una mano virtuale semi-trasparente posta al di sotto dell'oggetto interattivo. Queste tecniche risultano fondamentali per "ritrovare" all'interno della simulazione un oggetto virtuale che si sta manipolando, qualora esso sia stato ancorato.

Per indicare all'utente di capire che sta avvicinando l'oggetto manipolato ad una superficie su cui è possibile rilasciarlo, invece, si è deciso di renderizzare sulla superficie una particolare trama esagonale e, quando l'oggetto risulterà ad una distanza tale da poter essere effettivamente rilasciato e appoggiato sulla superficie, fornire all'utente un ulteriore ausilio grafico, nella forma di una circonferenza bianca (figura 3.2).

3.2.2 Informazioni visive relative agli altri utenti

Avatar

Per aumentare la consapevolezza relativa alla posizione e all'orientamento degli utenti partecipanti alla simulazione si è deciso di dare la possibilità di mostrare a schermo una loro semplice rappresentazione coincidente con la posizione dei dispositivi utilizzati. Tale rappresentazione corrisponde all'**avatar** dell'utente che sarà specifico per ciascun utente in quanto

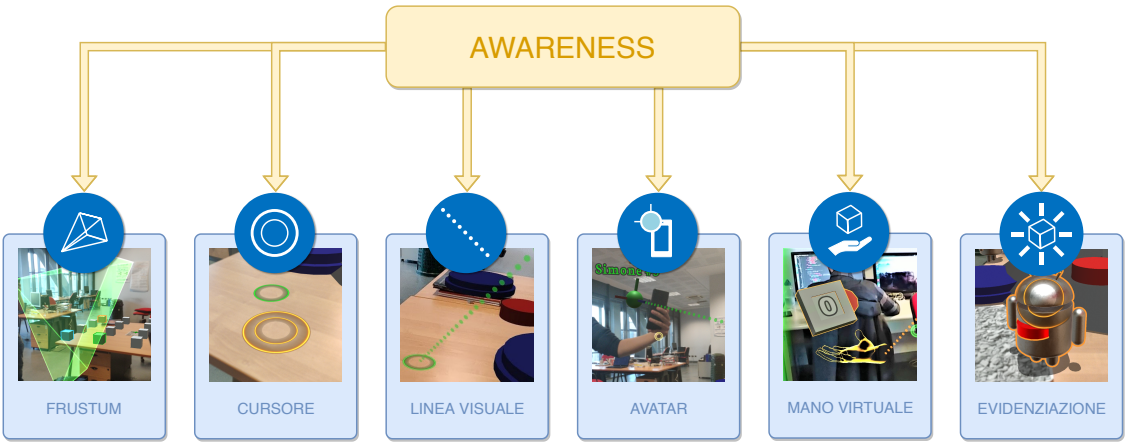


Figura 3.3. Rappresentazione schematica dei paradigmi di *awareness* con le *visual cues*

provvisto di caratteristiche come colore ed **etichetta** personalizzabili (figura 3.4).

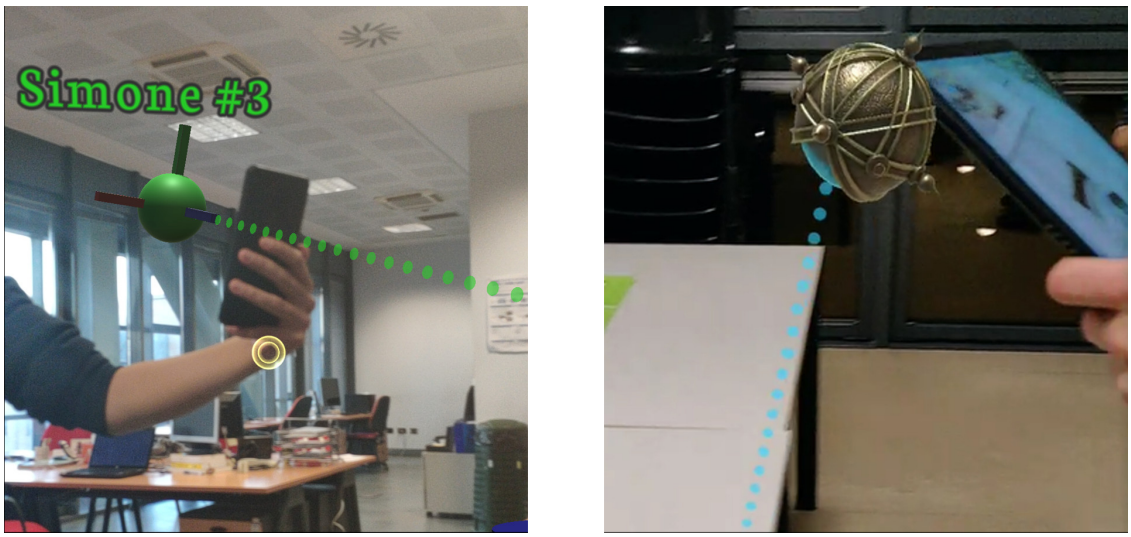


Figura 3.4. Rappresentazione dell'avatar nella parte sperimentale dell'applicazione, avatar nella *escape room*

È fondamentale anche disporre di elementi visivi che permettano di aiutare a capire quale attività stanno svolgendo e dove stanno concentrando la propria attenzione. Si è deciso quindi di adottare la maggior parte delle soluzioni e delle *visual cues* individuate nello stato dell'arte, al fine di testare e ottenere la combinazione di esse più efficace in scenari collaborativi. Si

è quindi stabilito l'utilizzo delle seguenti *visual cues* descritte nel dettaglio nei prossimi paragrafi: frustum visuale, linea visuale, cursore proiettato, mano virtuale ed evidenziazione visiva. Nella figura 3.3 una rappresentazione schematica di questi paradigmi di *awareness*.

Frustum visuale

Un frustum visuale che si origina dal dispositivo dell'utente permette di rappresentare la porzione di spazio che l'utente è in grado di inquadrare davanti a sé. Non fornisce informazioni puntuali su cosa effettivamente l'utente sta osservando, ma è possibile avere un'idea approssimativa di cosa in quel momento sta inquadrando. Questo elemento permette anche in parte di dare informazioni relative alla posizione e all'orientamento dell'utente. Nella nostra applicazione, si è deciso di rappresentare il frustum nella sua forma tipica di tronco di piramide e con una marcata trasparenza al fine di rendere la sua presenza quanto meno invasiva possibile ed essere facilmente associabile all'utente a cui appartiene.

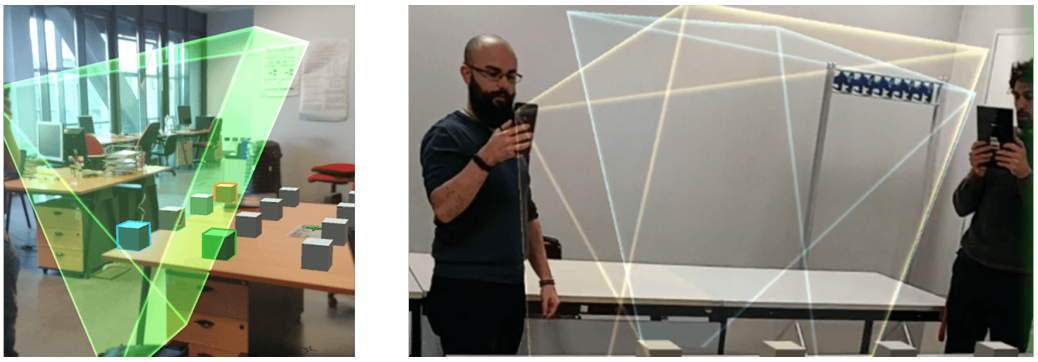


Figura 3.5. Rappresentazione del frustum nell'applicazione

Linea visuale

Una linea visuale che si origina dal dispositivo dell'utente e che termina sull'oggetto virtuale o sulla superficie rilevata che si sta puntando, oppure ad una determinata distanza dal dispositivo, se non si interseca alcuno degli oggetti precedenti. Tale linea risulta molto utile per riuscire a "risalire" alla posizione dell'utente, se si sta osservando il punto di intersezione con l'ambiente virtuale, oppure alla posizione del cursore, se si sta osservando l'utente,

creando una connessione semantica efficace e immediata tra la posa dei partecipanti alla simulazione e gli oggetti o piani virtuali puntati. Tale linea può essere rappresentata in diversi modi, ovvero come una linea continua o anche una serie di trattini o punti, come si è scelto nel nostro caso; la cosa importante è che, come per il frustum, presenti una leggera trasparenza e sia dello stesso colore di quello associato all'utente.

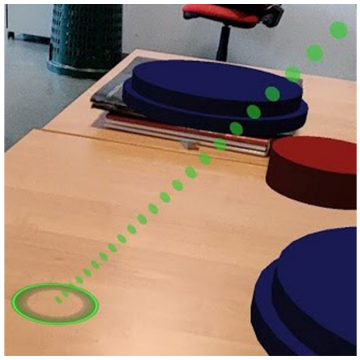


Figura 3.6. Rappresentazione della linea visiva nell'applicazione

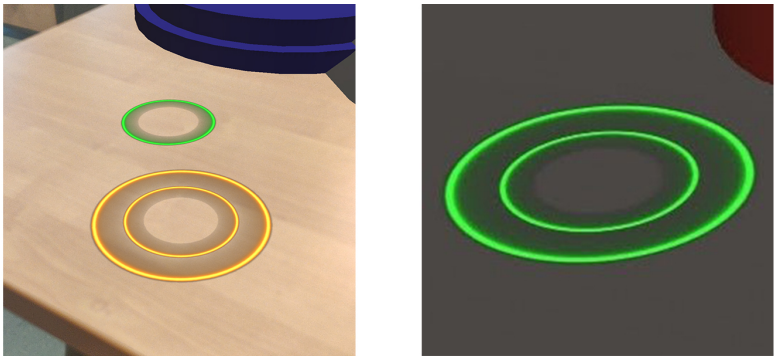


Figura 3.7. Rappresentazione del cursore nell'applicazione

Cursore proiettato

Un **cursore proiettato** viene generato all'intersezione della linea visuale con gli oggetti virtuali o i piani rilevati. Tale focus deve permette di identificare con precisione il punto dello spazio virtuale che in quel momento un utente sta puntando con il proprio dispositivo ed deve permettere anche di individuare esattamente la porzione di oggetto virtuale puntata. Oltre ad

essere un possibile strumento di interazione, si ritiene questo tipo di *visual cue* tra le più importanti ai fini dell'*awareness*; in una simulazione dove la collaborazione è fondamentale, poter sapere con molta precisione il luogo in cui gli altri utenti stanno puntando permette di agevolare notevolmente tutto il processo cooperativo. Anche per il cursore si è cercato di adottare una rappresentazione quanto meno invasiva possibile, quindi potrebbe configurarsi come un piccolo cerchio in parte trasparente del colore dell'utente a cui è associato.

Mano virtuale

Una mano virtuale che compare nel momento in cui un utente prende possesso di un oggetto virtuale. Tale mano deve essere visibile sia all'utente che sta effettuando le operazioni su tale oggetto, come precedentemente visto nel paragrafo sulla *self-awareness*, che a tutti gli altri utenti partecipanti. Tramite questa rappresentazione è possibile capire con un colpo d'occhio quando un utente sta effettuando manipolazioni su determinati oggetti. Anche questa *visual cue* deve risultare il meno invasiva possibile e in quanto elemento dell'interfaccia, non occupare troppo spazio. Una mano di dimensioni ragionevoli (che occupi magari meno del 20% dell'interfaccia), di media trasparenza e del colore dell'utente a cui è associata potrebbe essere una valida soluzione.

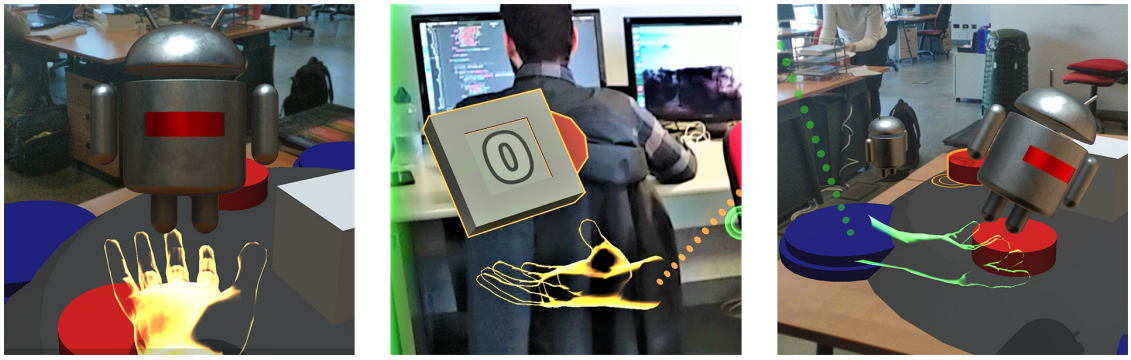


Figura 3.8. Mano virtuale nell'applicazione

Evidenziazione visiva

Tecniche di evidenziazione visiva o, più propriamente, *visual highlighting* possono essere adottate molto diffusamente in quanto generalmente molto utili ai fini dell'*awareness*. Possono essere utilizzate, per esempio, per segnalare

quali oggetti virtuali l'utente sta in quel momento osservando/manipolando. In aggiunta alla mano virtuale, ciascun oggetto in fase di manipolazione può essere evidenziato con un contorno colorato del colore dell'utente che lo possiede in quel determinato istante di tempo. Tale contorno è ottenuto tramite l'applicazione sugli oggetti virtuali di uno *shader* apposito che ne effettua l'*outline*, evidenziandone i bordi e colorandoli in modo molto acceso e luminoso.

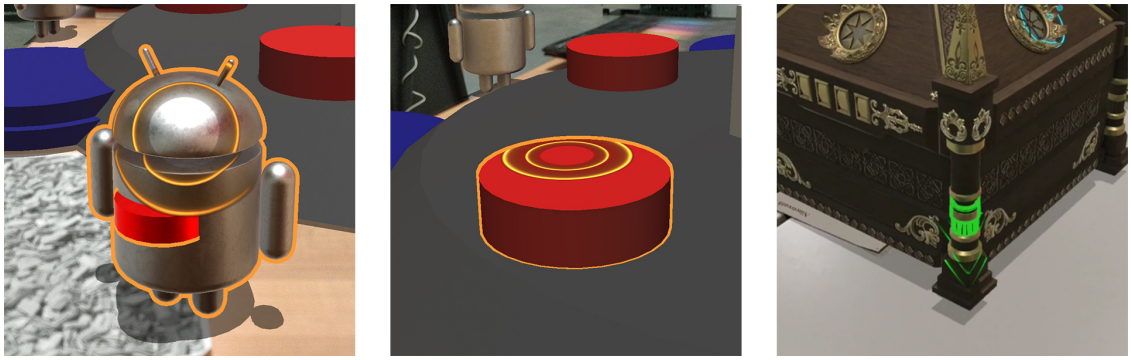


Figura 3.9. Esempio di evidenziazione visiva nell'applicazione sperimentale e nell'*escape room*

3.3 Attirare l'attenzione degli utenti

É fondamentale in contesto collaborativo essere in grado di attirare l'attenzione dei vari utenti su punti specifici dell'ambiente. In una simulazione di realtà aumentata multi-utente la comunicazione effettiva tra i vari partecipanti è soprattutto di tipo verbale per ovvi motivi; nonostante quindi sia possa attirare l'attenzione delle altre persone semplicemente parlando, è possibile, sfruttando le potenzialità della realtà aumentata e opportune tecniche, rendere ancora più immediata ed efficace tale attività.

Una delle tecniche più semplici che si è individuato prevede di indicare tramite l'uso del proprio cursore: se gli utenti sono in grado di visualizzare la posizione del cursore degli altri utenti, esso può essere posato sul punto specifico che si vuole evidenziare. Questa tecnica però presenta delle problematiche: è necessario continuare a inquadrare il punto su cui attirare l'attenzione degli altri utenti; l'efficacia di questo metodo dipende molto dalla velocità in cui si riesce ad individuare nell'ambiente il cursore degli altri utenti (a questo scopo può essere utile la visualizzazione del raggio visuale).

Al fine di colmare questi problemi si è creato un strumento all'interno dell'applicazione che combina alcune tecniche individuate nelle ricerche preliminari, che chiameremo **punto di interesse condiviso**.

Punto di interesse condiviso

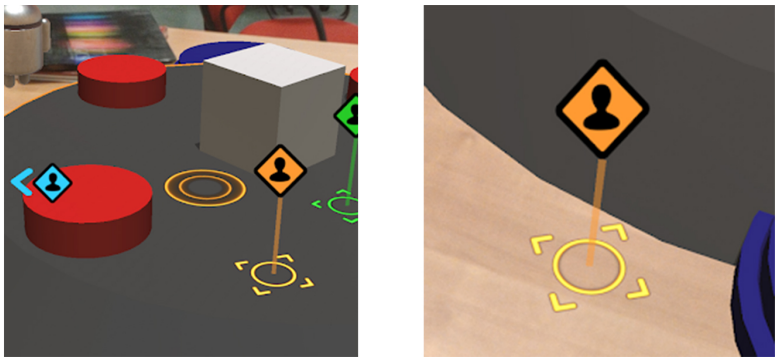


Figura 3.10. Punto di interesse condiviso: a sinistra quando esso è nel campo visivo, a destra quando esso è al di fuori del campo visivo.

Il punto di interesse condiviso è un punto identificato nell'ambiente, in particolare su una superficie rilevata o su un oggetto virtuale, che viene sincronizzato e mostrato a tutti gli utenti.

In particolare, per evidenziare questo punto, qualora sia nel proprio campo visivo, si è deciso di proiettare sulla superficie un cursore specifico che aiuti anche ad identificare meglio la sua posizione all'interno dell'ambiente e di mostrare una icona al di sopra di esso. Qualora il punto di interesse ricada su un oggetto virtuale è possibile attirare ulteriormente l'attenzione su di esso tramite un bordo colorato.

Quando invece il punto di interesse condiviso non è inquadrato direttamente, l'icona relativa va a posarsi e a scorrere sui bordi del dispositivo di visualizzazione insieme a una piccola freccia, creando un *distractor* con lo scopo di indicare la direzione verso la quale l'utente deve spostare il dispositivo al fine di inquadrare l'oggetto o il punto del piano indicato.

Tali punti di interesse possono essere generati dalla logica della applicazione, qualora sia richiesto attirare l'attenzione degli utenti in un determinato punto durante la simulazione, oppure generati dagli utenti stessi, effettuando un doppio tocco sullo schermo (da cui partirà un raycast per individuare il punto nell'ambiente). I punti generati dagli utenti hanno una icona specifica

colorata del colore dell'utente che li ha creati, sono massimo uno per utente e ai fini dell'applicazione permangono nell'ambiente finché non vengono distrutti (tramite tap).

3.4 Interazione multi-utente

Si è estrapolato, a seguito delle varie ricerche e studi effettuati in questo ambito, quattro tipologie base di interazione collaborativa di seguito brevemente elencate:

- interazione collaborativa basata sullo sguardo (guardare contemporaneamente uno stesso punto all'interno della simulazione, oppure guardare contemporaneamente punti specifici diversi);
- interazione collaborativa basata sulla manipolazione di oggetti virtuali (traslazione, rotazione e scalamento di un oggetto specifico o di più oggetti diversi non necessariamente nello stesso luogo);
- interazione collaborativa basata sulla posizione degli utenti (disporsi in uno stesso luogo o in luoghi specifici diversi all'interno della simulazione)
- interazione collaborativa basata sulla comunicazione verbale

Interazione collaborativa	Tipologia	Visual cues associate
Sguardo	Su un singolo punto o su più punti contemporaneamente	Frustum, linea visuale, cursore
Manipolazione di oggetti virtuali	Di un singolo oggetto o di più oggetti contemporaneamente	Mano virtuale, visual highlighting, linea visuale
Posizionamento	Su un singolo luogo o su più luoghi contemporaneamente	Avatar utente
Comunicazione vebale	Locale o remota tramite chat vocale	

Tabella 3.1. Rappresentazione schematica delle interazioni multi-utente esaminate e delle possibili visual cues associabili

Nelle sottosezioni successive tali tipologie verranno esaminate più nel dettaglio, indicando con esempi pratici casi d'uso in cui tali interazioni posso essere sfruttate appieno. Nella tabella 3.1 viene fornita una rappresentazione schematica di tali interazioni con l'anticipazione di alcuni concetti approfonditi nelle sezioni successive.

3.4.1 Interazione basata su sguardo

Questo tipo di interazione collaborativa prevede che lo sguardo degli utenti si combini contemporaneamente su un unico punto specifico o su più punti diversi della simulazione. Questa tipologia di interazione può essere utilizzata insieme a tecniche di *visual highlighting* e *distractor* per indicare efficacemente il punto da osservare.

Casi d'uso L'interazione basata su sguardo può essere adottata e sfruttata in molteplici situazioni collaborative in cui è necessario che gli utenti partecipanti concentrino la propria attenzione su un determinato punto dell'ambiente o oggetto in un preciso instante di tempo. Per esempio, in un contesto applicativo studiato a fini manutentivi o durante una sessione di progettazione condivisa, un partecipante alla simulazione con un determinato ruolo (oppure direttamente l'applicazione se lo scopo è quello di formare nuovi tecnici) può indicare, all'interno della scena, il punto in cui l'attenzione di tutti i partecipanti dovrà essere focalizzata. Solo dopo che tutti gli utenti avranno osservato il punto indicato la simulazione potrà proseguire. L'azione o evento che viene innescato a seguito della verifica di questa interazione può essere definita dall'applicazione.

Prendendo un esempio più concreto, potrebbe essere necessario che, a seguito di alcune operazioni su determinati strumenti o apparecchiature, l'attenzione di tutti i tecnici debba concentrarsi su un determinato macchinario al fine di verificarne il corretto funzionamento. Oppure è necessario che ciascun tecnico osservi determinati macchinari diversi simultaneamente. Oppure è possibile innescare l'animazione di un determinato apparecchio raffigurato come oggetto virtuale solo dopo che si è accertato che tutti gli utenti stiano osservando quell'oggetto.

Applicazione sviluppata Relativamente all'applicazione sviluppata per questa tesi, è stato pensato di sfruttare tale interazione collaborativa al fine di:

- indurre gli utenti ad osservare simultaneamente un determinato punto, per poi per esempio innescare una animazione e permettere a tutti di poter osservare tale animazione;
- indurre gli utenti ad osservare simultaneamente punti diversi all'interno della simulazione.

Questo è stato realizzato dando la possibilità via codice di definire delle porzioni sferica dello spazio, che, se inquadrate da tutti gli utenti oppure da un utente specifico, permettono di innescare un evento.

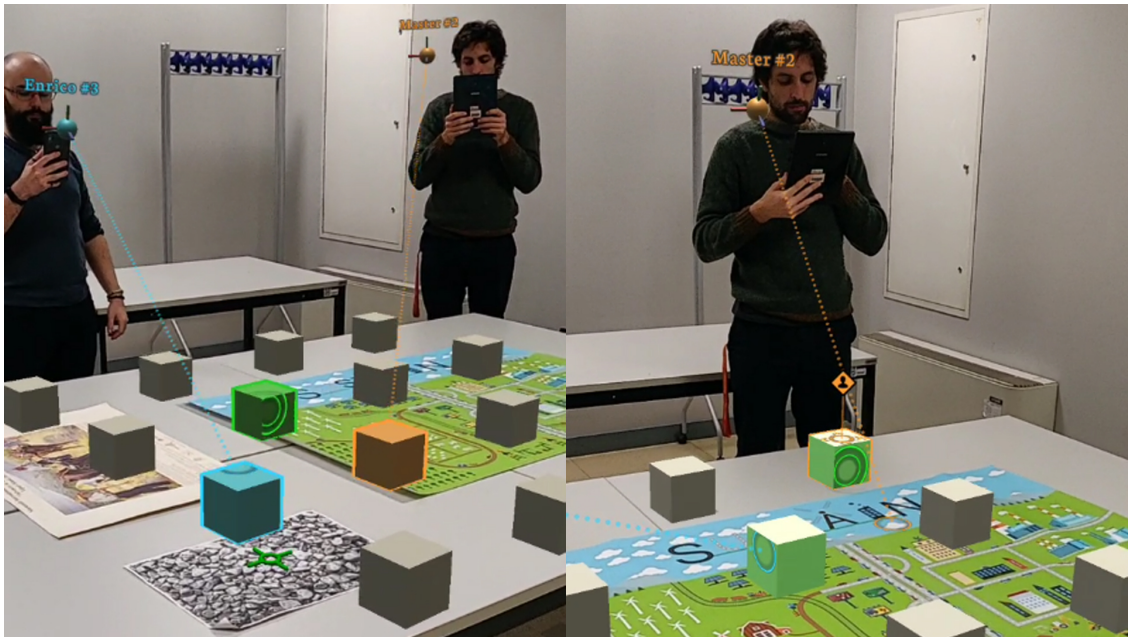


Figura 3.11. Esempio nell'applicazione di interazione con lo sguardo

3.4.2 Manipolazione collaborativa

Questa interazione collaborativa consiste nella manipolazione di oggetti virtuali, in particolare quando è richiesto da parte degli utenti di modificare certe proprietà degli oggetti virtuali (quali traslazioni, rotazioni o scalamenti) e tali modifiche vanno eseguite in contemporanea o in momenti diversi, ma comunque da più persone distinte.

Casi d'uso Questo tipo di interazione collaborativa si presta facilmente a diverse situazioni. Riprendendo l'esempio precedente di una applicazione in

realtà aumentata il cui scopo è eseguire determinate procedure di manutenzione oppure considerando applicazioni di training tecnico in cui è possibile addestrare il personale in un ambiente sicuro e controllato come quello di una simulazione AR, è chiaro come questo paradigma di interazione collaborativa risulti molto utile. Per esempio è possibile simulare il funzionamento di un grosso macchinario che richiede intervento tecnico simultaneo su alcune sue parti (valvole da aprire/chiudere, componenti da bloccare ecc.), oppure, al fine di portare a termine un compito collaborativo, è richiesto che diversi utenti con determinati ruoli agiscano sugli oggetti virtuali esclusivamente di loro competenza.



Figura 3.12. Esempio nell'applicazione di manipolazione collaborativa

Applicazione sviluppata Relativamente all'applicazione sviluppata per questa tesi, sono presenti sia interazioni che prevedono manipolazioni da parte degli utenti su oggetti distinti, sia operazioni simultanee di più utenti su più parti dello stesso oggetto. In particolare si hanno:

- oggetti virtuali interattivi che devono essere raccolti, esaminati e trasformati (alcuni presentano anche parti semoventi che devono essere manipolate per "attivare" l'oggetto) e poi utilizzati al fine di portare a termine un *task* collaborativo;

- oggetti virtuali provvisti di elementi interattivi che devono essere manipolati dagli utente per poter concludere un compito collaborativo (per esempio effettuare delle trasformazioni si rotazione o traslazione per allineare componenti di un determinato oggetto.

Poiché per la nostra applicazione si è deciso di non implementare ruoli diversi per ciascun utente (così da conferire completa libertà d'azione), tutti i partecipanti possono agire sui vari oggetti virtuali. É quindi possibile che, nel caso del primo punto, tutte le operazioni vengano eseguite da un solo utente; sta al buon senso dei partecipanti alla simulazione capire che agendo parallelamente sui vari oggetti virtuali è possibile eseguire le varie operazioni richieste in molto meno tempo.

3.4.3 Interazione basata su posizione

Questo paradigma d'interazione prevede che gli utenti si dispongano fisicamente in determinati punti o in una specifica posizione della simulazione AR. Tali punti nello spazio corrispondono a precise coordinate all'interno della sessione in realtà aumentata e potrebbero essere stabiliti dall'applicazione stessa o anche dagli utenti. Ad un primo approccio tale paradigma potrebbe risultare molto simile, quantomeno nella sua applicazione pratica, all'interazione collaborativa basata sullo sguardo; esistono però differenze marcate che conferiscono validità rilevante a tale tipologia di interazione collaborativa.

Casi d'uso Il posizionamento degli utenti all'interno della simulazione può essere sfruttato in diverse situazioni che prevedono il raggruppamento dei partecipanti in determinate zone dell'ambiente o che gli utenti raggiungano contemporaneamente determinati punti. Per esempio è possibile che, in uno scenario di progettazione condivisa, sia necessario che tutti gli utenti osservino la scena dallo stesso punto di vista (non necessariamente dalla stessa inquadratura) oppure in una situazione di manutenzione remota o training tecnico, tutti gli utenti debbano raccogliersi in una determinata *safe-zone*, determinata dall'attivazione di particolari operazioni potenzialmente pericolose.

A differenza del paradigma dello sguardo, dove appunto solo lo sguardo dell'utente deve concentrarsi su determinati punti, l'interazione basata sul posizionamento, come già visto, prevede lo spostamento fisico dei partecipanti. Questo permette, nel caso di situazioni di training o di manutenzione e controllo, di dare indicazioni precise agli utenti sulla posizione che devono

assumere in determinati istanti di tempo. Per esempio, nel caso di situazioni di training o manutentive, è richiesto agli utenti di disporsi a determinate coordinate del mondo al fine di raggiungere apparecchiature o macchinari specifici, per poi effettuare eventualmente operazioni su di essi. Questo paradigma può essere anche utilizzato in congiunzione con quello relativo allo sguardo per assicurarsi che gli utenti abbiano una determinato punto di vista, definendo il luogo in cui posizionarsi e il punto in cui guardare.

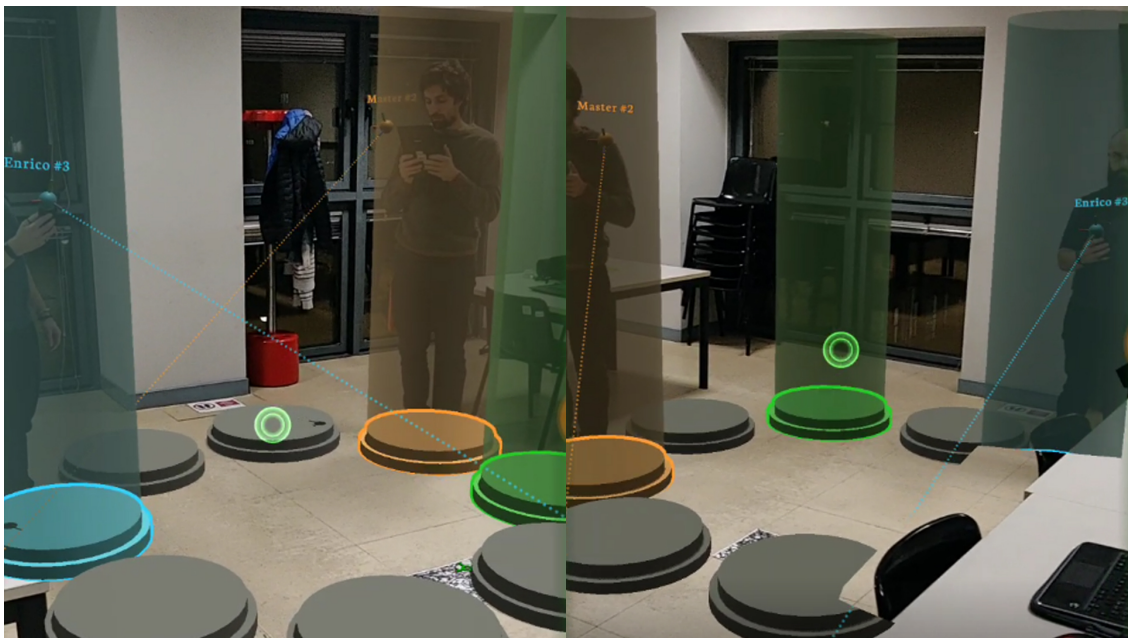


Figura 3.13. Esempio nell'applicazione di interazione basata su posizione

Applicazione sviluppata Riferendosi in particolare all'applicazione sviluppata per questa tesi, sono state implementate interazioni che richiedono sia il raggruppamento degli utenti in un luogo preciso, sia il posizionamento simultaneo degli utenti in luoghi diversi specifici. In particolare:

- gli utenti devono posizionarsi contemporaneamente su degli oggetti virtuali specifici posti sul pavimento, feedback visivi permetteranno di capire la conseguenza di questo posizionamento;
- ad un certo punto della simulazione, gli utenti devono raccogliersi nei pressi di un determinato oggetto virtuale e rimanere in quel punto per alcuni instanti di tempo.

Questo è stato realizzato dando la possibilità via codice di definire delle porzioni cilindriche dello spazio, che, se raggiunte da tutti gli utenti oppure da un utente specifico, permettono di innescare un evento.

3.4.4 Comunicazione verbale

La comunicazione verbale è la più semplice quanto immediata forma di interazione collaborativa. É essenziale mantenere un contatto vocale al fine di coordinamento delle operazioni. I casi d'uso relativi a questa forma di interazione sono innumerevoli e prevedono qualsiasi mansione che richieda un minimo di collaborazione tra i partecipanti. In particolare la comunicazione verbale è del tutto essenziale quando non tutti gli utenti posso vedere o avere accesso a tutte le informazioni; quando solo alcuni utenti dispongono di una determinata informazione o competenza, diventa imperativo comunicare verbalmente al fine di poter eseguire con successo specifiche operazioni.

Casi d'uso La comunicazione verbale può considerarsi l'interazione dominante all'interno di un contesto di manutenzione remota o co-locata dove solo uno o più partecipanti alla simulazione detengono le informazioni necessarie a portare a termine un particolare *task* (perché per esempio assumono determinati ruoli o dispongono di certe competenze). Il caso d'uso tipico dunque corrisponde a tutte le situazioni in cui si ha un utente che dirige le azioni di un altro utente, non potendo quest'ultimo agire autonomamente in modo corretto in quanto sprovvisto delle informazioni necessarie.

Applicazione sviluppata Per quanto riguarda l'applicazione sviluppata sono stati progettate alcune situazioni in cui solo determinati utenti possono accedere a specifiche informazioni che saranno la chiave per proseguire nell'esperienza. Di seguito una breve descrizione generica di ciò che è stato fatto per massimizzare la comunicazione verbale:

- solo un oggetto virtuale specifico della simulazione disporrà delle informazioni necessarie alla conclusione di un particolare *task* che prevede l'inserimento di tali informazioni all'interno di un oggetto virtuale diverso dal precedente. É possibile concludere questo task anche in autonomia, ma richiederebbe sicuramente molto più tempo e fatica dovendo l'utente continuamente spostarsi tra i due oggetti virtuali;

- l'utilizzo di particolari oggetti "equipaggiabili" permette di vedere a ciascun utente sono parti delle informazioni necessarie a proseguire nell'esperienza. Sarà compito degli utenti condividere queste informazioni tra di loro per ottenere il quadro completo.
- anche i potenziali utenti remoti possono interagire verbalmente con quelli in locale in quanto è stato predisposto un modulo di comunicazione vocale via network.

Parte III

Applicazione di test

Capitolo 4

Design dell'applicazione

Prima di procedere con la descrizione dell'implementazione dell'applicazione di test, che sarà oggetto del capito successivo, è opportuno dedicare un capitolo allo studio e alla progettazione dell'applicazione stessa. In questo capitolo verranno definiti e analizzati i requisiti che dovrà avere questa applicazione e le problematiche relative. Verranno descritte nel dettaglio le soluzioni che si è deciso di adottare, con particolare attenzione alla sincronizzazione, all'indipendenza dalle piattaforme AR e di *networking* disponibili e all'interazione col mondo virtuale. Si procede poi con una descrizione più dettagliata del tipo di applicazione che si è deciso di sviluppare, con una panoramica sulla storia e con la definizione delle sue meccaniche e caratteristiche peculiari.

4.1 Requisiti e problematiche

Progettare e sviluppare dalle basi un'applicazione in realtà aumentata, multiutente e multi-dispositivo al fine di analizzare i paradigmi di interazione e *awareness* collaborativi comporta una serie di sfide e problematiche da affrontare. Come prima cosa si è deciso di identificare con precisione tutti i requisiti che questa applicazione avrebbe dovuto soddisfare al fine di adempiere allo scopo prefissato, poi si sono analizzate attentamente tutte le problematiche che alcuni di questi requisiti avrebbero comportato e infine si sono studiate soluzioni opportune a tali criticità. Di seguito una breve panoramica sui requisiti richiesti e le problematiche annesse da affrontare, nelle successive sottosezioni una visione più approfondita di tali argomenti con le soluzioni adottate.

- L'applicazione deve essere predisposta a funzionare su più dispositivi diversi, con sistemi operativi diversi, quindi deve essere **multi-dispositivo** e **multi-piattaforma** e non deve appoggiarsi unicamente ad una specifica libreria di realtà aumentata disponibile in commercio. Va studiato di conseguenza un framework che permetta l'implementazione modulare delle varie declinazioni esistenti delle librerie AR.
- L'applicazione deve essere **multi-utente**, va trovato un modo per sincronizzare l'ambiente a tutti i partecipanti alla simulazione. Poiché per il punto precedente tali partecipanti possono essere su dispositivi o piattaforme diverse, occorre definire un sistema di sincronizzazione indipendente dalla piattaforma.
- L'applicazione deve essere **collaborativa**, quindi deve prevedere interazione tra gli utenti e interazione tra utenti e oggetti virtuali. Per prima cosa va predisposto un solido e consistente sistema di interazione tra i partecipanti e la simulazione e in seguito estendere tale interazione in contesto collaborativo. Vanno inoltre implementati i paradigmi di interazione collaborativa e *awareness* visti le capitolo precedente.
- L'applicazione deve prevedere un possibile scenario in cui gli utenti debbano collaborare in remoto e non nello stesso ambiente, quindi oltre ad essere **co-locata**, l'applicazione deve poter anche essere **remota**. Questo punto necessita lo sviluppo di una interfaccia di *networking* che preveda una simulazione remota o ibrida (locale e remota), l'implementazione obbligatoria di alcuni paradigmi di *awareness* che permettano di identificare più precisamente possibile la posizione degli utenti remoti all'interno della simulazione e di sincronizzare in qualche modo i due ambienti reali (locale e remoto)

Contrariamente a quanto suggerito dall'analisi dello stato dell'arte, si è deciso di non implementare interfacce tangibili all'interno di questa applicazione. L'utilizzo di tali interfacce, per quanto potesse risultare utile al fine di migliorare l'esperienza d'uso, avrebbe comportato una ulteriore mole di lavoro non trascurabile che avrebbe deviato dal proposito principale di questa tesi. Inoltre son già presenti studi a riguardo (anche se non propriamente coincidenti con lo scopo di questo lavoro). Per i precedenti motivi quindi si è stabilito di non implementare tali interfacce, ma si è comunque deciso di dedicarci un breve paragrafo elencando le possibili soluzioni adottabili. Come sviluppo futuro si potrebbe decidere di estendere questa applicazione aggiungendo il supporto per le interfacce tangibili.

Requisiti	Problematiche	Soluzioni
Multi-dispositivo	Fare in modo che l'app funzioni e sia facilmente fruibile in tutte le sue possibili declinazioni su dispositivi con form-factor diversi (smartphone, tablet e dispositivi indossabili)	Studiare una interfaccia utente che sia compatibile con questa tipologia di dispositivi
Multi-piattaforma	L'applicazione non deve dipendere da una libreria di realtà aumentata specifica	Sviluppo di un framework AR consistente in una libreria di realtà aumentata che astrae i concetti specifici delle librerie commerciali
Multi-utente	Gli utenti devono essere in grado scambiarsi informazioni all'interno della simulazione e devono potersi sincronizzare in qualche modo	Sviluppo di un framework di *networking* e di un sistema di sincronizzazione *marker-based*
Collaborazione	Gli utenti devono poter interagire tra di loro con tecniche e metodi specifici per la realtà aumentata	Sviluppo di un sistema di interazione multi-utente con particolare attenzione all'applicazione dei paradigmi di interazione e *awareness* visti nel capitolo precedente
In locale o in remoto	Gli utenti remoti devono essere facilmente identificabili e l'ambiente, seppur diverso, deve essere in qualche modo sincronizzato	Il framework di *networking* deve prevedere scenari remoti o ibridi, vanno applicati necessariamente specifici paradigmi di *awareness* e gli ambienti reali remoti e locali devono essere il più simili possibili

Tabella 4.1. Rappresentazione schematica dei requisiti con le problematiche annesse e relative soluzioni

4.2 Soluzioni adottate

Nelle successive sottosezioni verranno affrontati nello specifico i requisiti esaminati descrivendo dettagliatamente le soluzioni trovate. Nella tabella 4.1 si può vedere una rappresentazione schematica dei precedenti requisiti con un'anticipazione delle soluzioni adottate.

4.2.1 Multi-dispositivo: dispositivi utilizzati

Come anticipato nei paragrafi precedenti, l'applicazione deve essere fruibile da un ampio spettro di dispositivi che comprendono sia *hand-held display* che *head-mounted display*. Per ottenere tale risultato su HHD è stata sviluppata una interfaccia utente adattabile alla dimensione dello schermo del dispositivo su cui è in esecuzione; di conseguenza si avrà un'analoga esperienza sia che si stia utilizzando uno smartphone sia che si stia utilizzando un tablet. Per quanto riguarda invece la compatibilità con HMD, poiché non è stato possibile ottenere un dispositivo del genere, non sono state implementate GUI apposite per tali dispositivi; tuttavia durante lo sviluppo dell'applicazione, si è prestata particolare attenzione a definire paradigmi di interazione compatibili sia con le *gesture* tipiche degli schermi tattili che quelle tipiche degli HMD. Di conseguenza, nel caso si decidesse di proseguire nello sviluppo, risulterebbe agevolata l'implementazione di un supporto completo per *head-mounted display*.

Riassumendo i dispositivi che si sono presi in considerazione durante lo sviluppo della applicazione sono stati smartphone e tablet di ultima generazione. In particolare, disponendo solo di dispositivi Android, si è proceduto solamente all'implementazione ARCore della nostra libreria, ma il discorso è analogo anche per dispositivi iOS.

4.2.2 Multi-piattaforma: le librerie AR

Prima di tutto si sono cercate informazioni su eventuali framework che avrebbero semplificato o risolto il problema di compatibilità tra le varie librerie, unificandole. Dopo aver trovato alcune soluzioni promettenti che però, in questa fase iniziale di pianificazione, non risultavano ancora del tutto funzionanti o disponibili, si è deciso di creare ex-novo una interfaccia AR multipiattaforma e di conseguenza basare la nostra applicazione su di essa. Nel corso dello sviluppo, alcune delle soluzione esaminate sono state ufficialmente poi rese disponibili (per esempio AR Foundation di Unity), ma dato che si era già arrivati ad uno stadio avanzato di implementazione, si è deciso di proseguire con lo sviluppo utilizzando la nostra interfaccia.

Attualmente le principali librerie di realtà aumentata per dispositivi *handheld* disponibili in commercio sono ARCore di Google e ARKit di Apple, che però sono compatibili solo una cerchia abbastanza ristretta di *smartphone*, in genere di fascia alta. Per questo motivo si è inoltre discusso di estendere il supporto della nostra applicazione anche a dispositivi che non supportano

nativamente ARCore o ARKit, quindi appoggiandosi a librerie AR terze che fossero compatibili con il maggior numero di dispositivi possibili. Dopo aver impiegato diverso tempo a testare e valutare questi vari SDK disponibili, si è giunti alla conclusione che nessuno di essi soddisfacesse gli standard qualitativi che si erano stabiliti per la nostra applicazione al fine di garantire una esperienza fluida e appagante; standard qualitativi che invece ARCore o ARKit avrebbero senza dubbio soddisfatto. Questo non vuol dire però che non sia possibile integrare altre librerie all'infuori di queste nel framework sviluppato, semplicemente tali eventuali librerie devono fornire funzionalità base sufficientemente avanzate per poter essere compatibili (per esempio devono disporre di un modulo di *plane detection*). Di seguito una breve panoramica sulle librerie di AR considerate durante lo studio preliminare di pianificazione del framework di realtà aumentata, con una particolare attenzione alla descrizione di ARCore dato che l'implementazione della nostra libreria AR è stata eseguita su questa piattaforma. Si rimanda al capito successivo per i dettagli implementativi.

ARCore

É la principale piattaforma di realtà aumentata disponibile per Android ed è sviluppata da Google [52]. Tra tutti i vari framework AR Android disponibili è senza dubbio il più avanzato ed è quello che produce i risultati più soddisfacenti dal punto di vista dell'esperienza utente. ARCore non è disponibile su tutti i dispositivi Android, ma solo per una sottoinsieme di essi. Tale sottoinsieme è però in constante espansione in quanto il team di sviluppo si sta adoperando per rilasciare ARCore su quanti più dispositivi possibili; una operazione lunga e onerosa in quanto, al fine di rilasciare la piattaforma su un determinato dispositivo, è necessario una serie di test al fine di calibrarne i sensori e di garantire la miglior possibile esperienza utente.

Come già anticipato, alcune delle API di ARCore sono disponibili, oltre che per Android, anche per iOS al fine di abilitare esperienze AR condivise. ARCore presenta tre capacità chiave al fine di integrare il contenuto virtuale all'interno di all'interno del mondo reale così com'è catturato dalla fotocamera del dispositivo:

- il ***motion tracking*** che permette al dispositivo di capire e tenere traccia della propria posizione relativa all'interno del mondo;
- l'***environmental understanding*** che permette al dispositivo di rilevare la dimensione e la posizione di tutti i tipo di superficie, quindi

superfici orizzontali, verticali e superficie angolate come per esempio il terreno, la pareti e i tavoli;

- La ***light estimation*** che permette al dispositivo di stimare le attuali condizioni di illuminazione dell'ambiente.

Fondamentalmente, ARCore permette di eseguire due operazioni: monitorare la posizione del dispositivo mobile mentre si muove nello spazio e definire la propria ricostruzione virtuale del mondo reale. La tecnologia di tracciamento del movimento di ARCore utilizza la fotocamera del dispositivo per identificare punti di interesse, denominati ***features***, e tiene traccia di come questi punti si spostano nel tempo. Con una combinazione del movimento di questi punti e delle letture dai sensori inerziali del dispositivo, ARCore determina sia la posizione che l'orientamento del dispositivo all'interno della simulazione mentre si muove nello spazio reale. Oltre a identificare i punti chiave, ARCore può rilevare superfici piane, come un tavolo o il pavimento, e può anche stimare l'illuminazione media nell'area circostante.

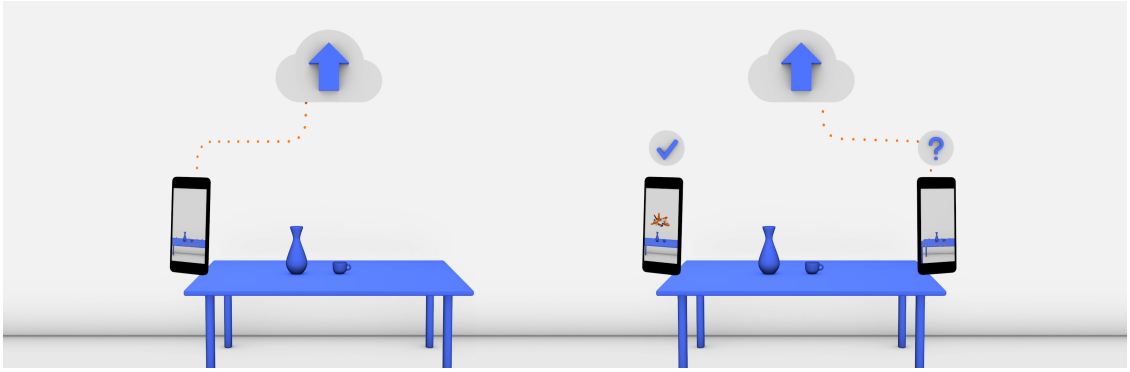


Figura 4.1. Esemplificazione del funzionamento delle *cloud anchors*

Una delle caratteristiche più interessanti di ARCore è senza dubbio il paradigma delle ***Cloud Anchors*** [53]. Tale tecnologia è stata adottata da Google per supportare il multiplayer AR. Il sistema delle Cloud Anchors prevede la generazione da parte della sessione ARCore di una *feature map* 3D dello spazio circondante l'ancora (il cosiddetto *center of interest*). Questi informazioni sono quindi costituite da diversi dati visuali caricati successivamente nel cloud di Google dal giocatore che desidera ospitare una sessione multi-utente. Una volta che i server Google ottengono la ricostruzione 3D dell'ambiente, viene generata una *cloud anchor* con un preciso identificativo

detto *Cloud Anchor ID*. La *cloud anchor* in sostanza non è altro che una mappa SLAM dell'area fisica scansionata memorizzata sui server Google. Altri giocatori possono partecipare alla sessione multi-giocatore scegliendo l'identificativo corretto, quindi caricando il proprio set di dati visuali per ricevere da Google una matrice di trasformazione appropriata, risultato del processo di rilocalizzazione che avviene quindi in remoto. In figura 4.1 è possibile osservare una rappresentazione grafica della sincronizzazione tramite *cloud anchors*. Questo sistema funziona esclusivamente se i dispositivi hanno accesso ad internet; inoltre i dati sulla *cloud anchor* vengono mantenuti sui server per 24 ore prima di essere eliminati e i dati visuali grezzi inviati agli utenti quando generano un'ancora sono invece mantenuti per 7 giorni.

ARKit

La soluzione Apple alle esperienze di realtà aumentata si concretizza in ARKit. É una piattaforma estremamente potente e nella sua ultimissima versione (ARKit 3) propone alcune *features* davvero notevoli (è bene notare a scopo puramente informativo che durante lo sviluppo della nostra applicazione era disponibile unicamente ARKit 2). Concretamente, molti dei paradigmi definiti per ARCore possono essere trasposti anche per ARKit: questa piattaforma combina il rilevamento del movimento del dispositivo, l'acquisizione della scena della telecamera e l'elaborazione avanzata delle scene per la generazione di una esperienza in realtà aumentata. È possibile utilizzare queste tecnologie per creare molti tipi di esperienze AR, utilizzando la fotocamera posteriore o frontale di un dispositivo iOS [47]. Quindi analogamente ad ARCore si hanno: ***motion tracking***, ***environmental understanding*** e una più precisa e performante ***light estimation***. Le *features* più interessanti, che differenziano sostanzialmente ARKit dalle altre piattaforme di realtà aumentata in commercio, in particolare ARCore, sono le seguenti [48]:

- a differenza di ARCore, ARKit supporta la persistenza della scena ricostruita e quindi delle ancore. Questo significa che se l'applicazione si chiude o va in background non perde i dati raccolti sull'ambiente reale e si può riprendere la simulazione da dove la si era lasciata (questo processo è chiamato *rilocalizzazione*);
- la rappresentazione nativa degli oggetti virtuali è molto realistica in quanto vengono applicate mappe di riflessione in tempo reale generate dal feed della telecamera oltre che a effetti di ombreggiatura sotto l'oggetto 3D;

- con ***People Occlusion*** ARKit sa come sono disposte le persone e gli oggetti virtuali all'interno della simulazione e occlude opportunamente la scena;
- con il ***Motion Capture*** ARKit è in grado di tenere traccia del movimento umano utilizzabile come input all'interno della simulazione;
- ARKit è in grado inoltre di tracciare fino a tre volti umani contemporaneamente.

Anche in ARKit è ovviamente disponibile un sistema di sincronizzazione degli scenari AR pressochè analogo a quello di ARCore, anche in questo caso viene generata una *point cloud* popolata dai *featur point* estrapolati dai dati visuali della fotocamera e di fatto l'algoritmo utilizzato è lo stesso, cioè ORB-SLAM2 *Monocular*.

Vuforia e Windows Mixed Reality

Vuforia Engine è la più usata e diffusa piattaforma di sviluppo per realtà aumentata disponibile, con ampio supporto per HHD e HMD, basata principalmente sul riconoscimento e il tracciamento di immagini piane [63]. É multipiattaforma e supporta applicazioni Android, iOS e UWP. Gran parte del successo di questa libreria è dovuto al fatto che è perfettamente integrabile all'interno dell'ambiente di sviluppo di Unity (oltre che le piattaforme native per Android, iOS e UWP), permettendo quindi la fruizione degli strumenti tipici per la generazione di uno scenario in realtà aumentata in maniera semplice e immediata.

Questa piattaforma supporta ARCore e ARKit ed è generalmente lo strumento principale quando si sviluppano applicazione in realtà aumentata in ambiente *Windows Mixed Reality*. Con il supporto per ARCore e ARKit si potrebbe pensare che questa libreria possa fungere da ponte e adempiere allo scopo che ci si era prefissati con lo sviluppo di un framework di realtà aumentata. In realtà tale supporto serve solo ad "attivare" le funzionalità specifiche di ARCore o ARKit nel caso il dispositivo di destinazione le supporti. Per esempio Vuforia ha un proprio sistema di rilevamento piani (limitato solo ad alcuni dispositivi dove è stato testato), ma se il dispositivo di destinazione supporta ARCore e tale libreria è abilitata all'interno del file di configurazione di Vuforia, allora l'algoritmo di rilevamento piano sfrutta anche le informazioni ricavate dalla sessione ARCore per ottenere risultati migliori. Non è però possibile attivare contemporaneamente la compatibilità

di Vuforia per ARCore o ARKit e accedere direttamente alle API disponibili per queste due piattaforme.

Windows Mixed Reality rappresenta il tentativo di Microsoft di portare la realtà virtuale e aumentata sul mercato di massa. La piattaforma è basata sull'ambiente di sviluppo UWP (*Universal Windows Platform*) e fornisce gli strumenti necessari per l'implementazione di esperienze AR/VR. Peculiarità della WMR, che la differenzia sostanzialmente dalle altre librerie esaminate, è che dispone di *tool* che permettono la ricostruzione 3D integrale dell'ambiente reale (sfruttando in genere le camere di profondità installate sugli *headset* compatibili, come le Hololes). Tali ricostruzioni, che permettono la definizione di *spatial anchors*, aumentano la stabilità e l'affidabilità della sessione AR e permettono di implementare l'occlusione degli oggetti reali.

AR Foundation e Multi-AR

AR Foundation è un ottimo candidato per assumere il ruolo di standard generico per la realtà aumentata hand-held all'interno dell'ambiente di sviluppo di Unity [66]. Questa libreria funge da *wrapper* per le altre piattaforme commerciali di AR (ARCore e ARKit), integrandole in una unica soluzione. Tale libreria corrisponde quasi esattamente ai propositi del nostro framework di realtà aumentata, ma nel momento di pianificazione dei lavori non era ancora stata resa pubblica e si è ritenuto non necessaria una migrazione del progetto a questa nuova piattaforma nel momento della pubblicazione. Anche l'asset *Multi-AR*, disponibile sull'asset store di Unity, si pone come piattaforma unificante di tutto il panorama della realtà aumentata, ma a differenza di AR Foundation presenta *features* un po' meno avanzate e un team di sviluppo molto meno numeroso [50]. Questa libreria fornisce un modo semplice per fruire delle varie *features* specifiche di diverse piattaforme AR, incluse quelle non hand-held, separando i dettagli specifici per piattaforma con l'effettivo sviluppo dello scenario AR. Si è avuto modo di analizzare il codice di Multi-AR e ci si è ispirati ad esso per sviluppare la nostra piattaforma, concentrando l'attenzione sulle le *features* utili per l'implementazione dell'applicazione.

6D.AI

6D.AI, una giovane *startup* emergente dall'Active Vision Lab della Oxford University [58], ha costruito, un'infrastruttura AR proprietaria molto interessante, con particolare attenzione alla privacy. Hanno basato il loro sistema su un algoritmo di rilocalizzazione di nuova generazione che implementa il paradigma *just-works* per l'esperienza utente. Tutta l'elaborazione

è su dispositivo e in tempo reale, le mappe vengono create ed estese in background mentre l'applicazione è in esecuzione. Le ancore prodotte non hanno informazioni personali e sono memorizzate permanentemente nel cloud, ogni utente può aggiornare le proprie mappe generate nello spazio fisico per migliorare la copertura e l'efficacia dell'ancora. La necessità di pre-scansione è ridotta al minimo o eventualmente eliminata. A differenza di Google questo sistema può occuparsi di occlusioni di oggetti reali, le mappe prodotte sono un modello 3D denso del mondo reale, possono funzionare offline e in un ambiente peer-to-peer o in un ambiente privato/sicuro. Le principali *features* di questa piattaforma sono:

- la ***Persistence***, cioè il contenuto virtuale rimane fissato nella posizione assegnata tra diverse sessioni AR ed è quindi semplice ritrovare un oggetto virtuale quando si inizia una nuova sessione nelle vicinanze;
- l'***Occlusion***, come per le altre librerie esaminate non è necessaria alcuna camera di profondità per la scansione del mondo reale in 3D. Questa soluzione è in grado di costruire mappe *three-dimensional semantic* e *crowd-sourced* del mondo fisico in tempo reale e in background, sfruttando unicamente la camera del dispositivo;
- il ***Multiplayer*** con la condivisione dell'esperienza in realtà aumentata quasi instantanea da ogni posizione;
- il ***Cross-platform***, qualsiasi dispositivo potrà usufruire della stessa esperienza allo stesso momento nel meglio delle proprie capacità hardware.

Putroppo questa libreria è in beta chiusa, per cui l'accesso ad essa è limitata a un gruppo ristretto di developer e per ora è esclusivamente compatibile con dispositivi iOS.

Altre librerie di AR

Infine si sono analizzate alcune soluzioni alternative alle più commerciali e diffuse. Tra le varie librerie che si sono testate con il dispositivo in dotazione (un Samsung Galaxy Tab S4 con supporto per ARCore), quelle degne di menzione sono le seguenti:

- **Wikitude**, si propone come una libreria AR *cross-platform* che può riconoscere e tracciare immagini, oggetti, scene e località geografiche. A

seguito dei test effettuati si è rilevato un *plane detection* poco performante e instabile (sia con ARCore disattivato che con ARCore attivato, dove le performance migliorano ma di poco [67];

- **Kudan** come il precedente è una libreria *cross-platform* di esperienze AR con marker e *markerless*, non richiede il supporto di ARCore e, a seguito dei test effettuati, presenta un *plane detection* con performance discrete. Questa libreria fornisce inoltre un algoritmo di SLAM per la definizione e la ricostruzione delle *features* di un ambiente reale al fine di permettere una esperienza AR priva di marker. Si sono effettuati alcuni test con le ancore e si è appurato che sono troppo facilmente spostabili [56];
- **MAXST**, anch'essa una libreria AR *cross-platform*, è presente un algoritmo di SLAM con nuvola di punti, ma i risultati, a seguito dei nostri test, erano alquanto scadenti. Il tracciamento marker invece è risultato buono mentre l'*instant tracking* (cioè un'esperienza AR *markerless*) è risultata discreta (si sono rilevati spostamenti delle ancore con il semplice movimento del tablet) [59].

In conclusione, come già anticipato, si sono ritenute tali librerie esaminate non all'altezza dell'esperienza che avremmo voluto offrire e si è di conseguenza optato per lo sviluppo di un nostro framework che implementasse le principale librerie AR (ARCore e ARKit) in grado di garantire risultati nettamente superiori.

4.2.3 Multi-utente

Framework di *networking*

Per prima cosa va affrontato il problema di come scambiare i dati tra gli utenti all'interno della simulazione e che tipo di architettura adottare, se peer-to-peer o client-server. Si è deciso, come per le piattaforme di AR, di sviluppare una interfaccia di *networking* che astraesse dalla particolare libreria utilizzata per lo scambio dati. Di conseguenza è stato speso del tempo ad esaminare le varie piattaforme di *networking* disponibili sul mercato, esaminandone accuratamente la documentazione al fine di identificare con precisione tutte le funzioni che ci sarebbe servito esporre all'interno della libreria. Non essendo però questo il focus di questa tesi, si rimanda al capitolo implementativo per alcuni dettagli in più a riguardo, con anche una

breve disamina delle principali librerie di *networking* e una breve descrizione di quale è stata scelta per l'implementazione.

Sincronizzazione dell'ambiente condiviso

Per poter sincronizzare l'ambiente condiviso son stati presi in considerazione due possibili approcci: una sincronizzazione ***markerless*** ed una sincronizzazione ***marker-based***.

Se i requisiti dell'applicazione si fossero limitati alla compatibilità tra *hand-held device* e considerando esclusivamente smartphone compatibili con ARCore o ARkit, allora una soluzione *markerless* sarebbe stata semplice e immediata: infatti, in tal caso, si sarebbero potute usare direttamente le API di ARCore relative alle *Cloud anchors*, compatibili anche con ARKit, per la sincronizzazione di ambienti condivisi. Malgrado la semplicità di questa soluzione e nonostante l'applicazione finale sia stata sviluppata esclusivamente per HHD, si è ritenuto opportuno però, come già sottolineato, progettare l'applicazione nell'ottica di una potenziale espansione del supporto a HMD, scartando quindi questa possibilità.

Il primo approccio che è stato considerato è quello relativo ad una esperienza *markerless*. Questo vuol dire riuscire a sincronizzare le esperienze tra i diversi dispositivi senza sfruttare elementi fisici specifici comuni, in genere delle immagini facilmente riconoscibili della computer vision (detti*marker*), e basandosi esclusivamente sui dati visuali acquisiti dalle diverse fotocamere. ARCore e ARKit, come già visto, permettono già questo tipo di sincronizzazione sfruttando determinati e complessi algoritmi di rilocalizzazione; al fine di estendere tale caratteristica agli HMD, risulta necessario ottenere prima di tutto i dati visuali catturati dalle specifiche sessioni AR dei vari dispositivi coinvolti ed in seguito elaborare questi dati con un opportuno algoritmo al fine di trasmettere a ciascun partecipante i dati necessari per effettuare la rilocalizzazione (in sostanza una matrice di trasformazione che permetta di traslare in modo opportuno la simulazione virtuale).

L'algoritmo di *mapping* utilizzato da ARCore e ARKit, come già accennato, è ORB-SLAM2 *Monocular* che permette di ottenere una ricostruzione del mondo fisico in tempo reale basata su *point-cloud* direttamente dal *feed* della fotocamera. HoloLens, o altri dispositivi HMD analoghi della piattaforma *Windows Mixed Reality*, utilizza invece un algoritmo diverso che produce una effettiva ricostruzione tridimensionale del mondo fisico (*mesh* 3D densa) attraverso la scansione effettuata dalle camere di profondità installate.

Per poter effettuare la rilocalizzazione servono, per prima cosa, le *point-cloud* degli ambienti scansionati; tali dati sono facilmente ottenibili da ARCore e ARKit tramite apposite chiamate alle API corrispondenti (ARCore fornisce una *point-cloud* solo per ciascun frame, mentre ARKit fornisce una *point-cloud* riferita all'intera sessione); per quanto riguarda invece WMR si dispone solo di una ricostruzione 3D, ottenuta tramite camere di profondità, del tutto diversa da una *point-cloud* che si basa principalmente sul rilevamento di *feature-points*.

Per poter ottenere una *point-cloud* anche dai dispositivi WMR si è ipotizzato di sfruttare il *feed* delle camere RGB di tali HMD ed utilizzarlo come input per un algoritmo di *mapping* e generazione di *point-cloud* come ORB-SLAM2. A tal fine son state reperite alcune implementazioni di algoritmi di *computer vision* di questo tipo: LSD-SLAM e ORB-SLAM [64][49].

Una volta ottenute le *point-cloud* da tutti i dispositivi, si può procedere alla rilocalizzazione; tale procedimento consiste nell'effettuare un *matching* tra le varie mappe ricostruite al fine di ottenere i parametri caratterizzanti di una opportuna matrice di trasformazione. Dopo aver ottenuto tale matrice, è possibile sincronizzare le esperienze semplicemente applicandola a tutte le trasformazioni eseguite all'interno della simulazione.

Conclusa quindi la fase di pianificazione di una possibile soluzione *markerless* si è proceduto con lo studio di fattibilità alla luce dei dispositivi disponibili, della potenza di calcolo necessaria e della complessità implementativa. Innanzi tutto va sottolineato che una problematica non indifferente potrebbe essere il fatto che le *point-cloud* generate dai sistemi diversi (ARCore o ARKit per esempio, o anche altri SDK o algoritmi) siano diverse per svariati motivi (calibrazioni diverse delle fotocamere, algoritmi diversi oppure applicazioni diverse di uno stesso algoritmo ecc.). Inoltre algoritmi di *mapping* come LSD-SLAM e ORB-SLAM, implementati con OpenCV, risultano computazionalmente molto pesanti per una applicazione *real-time* e la potenza di calcolo disponibile nei classici HMD difficilmente avrebbe permesso performance accettabili.

Alla luce di tali considerazioni, convenendo che l'implementazione ex-novo di un algoritmo di *mapping* e/o rilocalizzazione leggero e adatto agli scopi definiti per l'applicazione avrebbe richiesto una mole di lavoro eccessiva, deviando quindi considerevolmente dall'obiettivo di questo lavoro, si è deciso di scartare l'opzione *markerless* e di conseguenza adottare un approccio *marker-based*.

L'approccio *marker-based* risulta di gran lunga più semplice da definire e implementare. Innanzi tutto l'utilizzo di questo approccio esclude l'utilizzo di caratteristiche avanzate per la rilocalizzazione da parte delle librerie proprietarie (non è più necessario estrapolare i dati delle *point-cloud*); inoltre, il non dovere utilizzare tali funzionalità, permette potenzialmente di estendere il supporto dell'applicazione a tutti i dispositivi che supportano semplicemente l'*image tracking* e non necessariamente compatibili con ARCore o ARKit. Questa potenzialità non è stata comunque sfruttata perché si è ritenuta poco soddisfacente qualsiasi simulazione AR che non si basasse su ARCore o ARKit. L'unico punto critico dunque è che la libreria utilizzata sia provvista di un solido modulo per l'*image tracking* al fine di rilevare il *marker*. Tutte le librerie di realtà aumentata esaminate supportano tale funzionalità. In particolare ARCore non si è dimostrato particolarmente affidabile sotto questo punto di vista, ma con alcuni accorgimenti che verranno esaminati nel capitolo successivo è stato possibile ottenere risultati soddisfacenti.

Di seguito una breve disamina di come avviene la rilocalizzazione in contesto *marker-based*. Innanzi tutto il *marker* corrisponde, concettualmente, all'origine del sistema di riferimento del mondo aumentato e tutte le trasformazioni che avvengono all'interno di questo mondo sono eseguite in relazione a tale origine. Una volta che ciascun utente avrà rilevato correttamente il marker, l'ambiente condiviso risulterà consistente per tutti poiché ogni posa relativa del *marker* verrà registrata e scambiata ed utilizzata opportunamente per il calcolo di una matrice di trasformazione. Riassumendo, un utente, che disporrà di tutte le posizioni e orientamenti relativi del *marker* rilevate dagli altri utenti, potrà utilizzare queste informazioni per effettuare una trasformazione di tutti i dati di posizione e orientamento degli oggetti virtuali che richiedono sincronizzazione inviate dai suddetti utenti. Per una descrizione più dettagliata e chiara di tale processo si rimanda al successivo capitolo implementativo.

4.2.4 Collaborazione in locale o remota

Relativamente a questo punto le soluzioni adottate consistono nell'applicazione pratica delle interazioni collaborative e delle informazioni visive utili per l' *awareness* viste nel capitolo precedente. Vanno quindi studiati e progettati accuratamente scenari che agevolino l'implementazione di tali paradigmi, parallelamente alla definizione di un insieme di informazioni visive che faciliti la collaborazione. Per la definizione e la descrizione di tali scenari, come già

anticipato, si rimanda al capitolo successivo che esamina più nel dettaglio la realizzazione effettiva dell'applicazione.

4.2.5 Possibili implementazioni di interfacce tangibili

Non sono state adottate interfacce tangibili per la nostra applicazione, ma si è comunque effettuato un breve studio su quali soluzioni avremmo potuto adottare nel caso avessimo deciso di implementarle. A tal proposito, come già visto nello studio dello stato dell'arte al paragrafo 2.2.3, vanno distinti due tipi di interazione, una *time-multiplexed* e una *space-multiplexed*. Le interfacce tangibili sono applicabili sia per dispositivi HHD che dispositivi HMD e le soluzioni possibili individuate sono le seguenti:

- tessere che possono essere fisicamente spostate e girate (*space-multiplexed*) o semplicemente utilizzate come strumento di scorrimento di una lista di oggetti virtuali (*time-multiplexed*);
- palette o bacchette utilizzabili per la manipolazione di oggetti virtuali, per esempio agganciandoli e traslandoli (time-multiplexed);
- oggetti più complessi con particolari forme geometriche usati singolarmente o in combinazione per portare a termine determinati *task* (*space-multiplexed*)
- un oggetto a forma di lente con il quale andare ad interagire con il mondo virtuale al fine di visulizzare elementi aggiuntivi o ottenere nuove informazioni (*time-multiplexed*).

4.3 Panoramica dell'applicazione scelta

Prima di decidere che tipo di applicazione progettare come caso d'uso di tutte le informazioni raccolte, sono state effettuate alcune ricerche e studi preliminari al fine di individuare la miglior tipologia che avrebbe permesso di mettere in risalto l'implementazione dei paradigmi d'interazione collaborativa e *awareness* multi-utente.

Dopo una breve ricerca sui principali ambiti in cui la collaborazione tra gli utenti risultasse come componente di spicco dell'esperienza, si è individuato quella che maggiormente avrebbe soddisfatto i requisiti di questo lavoro e le nostre esigenze. Tra i vari ambiti che sono stati esaminati, figurano scenari di progettazione condivisa (per esempio in ambito architettonico), scenari di manutenzione remota o co-locata, scenari di training industriale e

infine scenari di gioco. Quest'ultimo è stato lo scenario che più ci ha convinto potesse risultare utile al fine dello studio alla base di questa tesi. In particolare era opportuno trovare uno scenario che oltre ad esaltare la componente collaborativa, sfruttasse anche le peculiarità di una simulazione in realtà aumentata.

Si sono quindi esaminate le diverse tipologie di videogioco collaborativo maggiormente diffusi, senza fare particolare distinzione tra realtà aumentata o virtuale; all'interno di questa ricerca si sono anche inclusi giochi collaborativi tipicamente non digitali, come giochi di ruolo da tavolo o di società trasportabili in contesto AR. Di seguito in tabella 4.2 un riassunto dei principali giochi esaminati e in figura 4.2 una loro rappresentazione.

Tra le diverse opzioni a disposizione quindi, si è inizialmente pensato che un gioco da tavolo collaborativo in realtà aumentata sarebbe stato una soluzione adeguata agli scopi preposti, ma dopo aver discusso a lungo in merito



Figura 4.2. In senso orario da in alto a sinistra: Tsuro, Space Alert, Burgle Bros, Escape: The Curse of the Temple, Legends of Andor e Mansion of Madness

si è deciso di estendere la ricerca a contesti che non fossero necessariamente legati ad un gioco da tavola o ad un videogioco, ma a tutte le attività ricreative che comportassero una forte interazione collaborativa tra i partecipanti al fine di portare a termine un *task* comune. Come prima opzione di tali attività ricreative, avendo diverse conoscenze a riguardo per numerose

Gioco o videogioco	Tipologia	Pro	Contro
Escape: The Curse of the Temple	Gioco da tavola	Tessere con cui interagire, limite di tempo	"Maledizioni" che limitano le interazioni
Legends of Andor		Ambientazione, plancia, ruoli e pedine con cui interagire	Complessità
Mansion of Madness		Ambientazione, plancia	Poca interazione con le tessere
Tsuro		Semplice, tessere e pedine con cui interagire	Poca interazione tra gli utenti
Space Alert		Ambientazione, plancia, pedine con cui interagire e limite di tempo	Complessità e poca interazione fisica
Burgle Bros		Ambientazione, tessere e pedine con cui interagire, ruoli	Complessità e dimensioni
Fable Legends	Videogioco	Ambientazione, ruoli, cooperazione contro un nemico comune interpretato da un giocatore	Complessità, applicabilità in AR, interazione fisica
Dungeon Defenders 2		Ambientazione, ruoli e cooperazione	Complessità e interazione fisica
Gun's Up		Ruoli, cooperazione	Complessità e interazione fisica
The Machine AR		Ambientazione, ruoli	Poche interazioni tra gli utenti

Tabella 4.2. Panoramica dei principali giochi esaminati nella fase di pianificazione dell'applicazione

esperienze personali, si è pensato ad una trasposizione in AR di un scenario da *escape room*, cioè un contesto dove i partecipanti devono collaborare per risolvere degli enigmi proposti dalla simulazione sparsi nell'ambiente reale in un tempo prefissato. É subito apparsa come una valida idea e si è proceduto a verificarne effettivamente la fattibilità in un contesto di realtà aumentata.

Tra le varie opzioni possibili per per portare una esperienza da *escape room* in un contesto di realtà aumentata si sono individuati due approcci predominanti: limitare gli elementi virtuali in una zona ben delimitata dell'ambiente reale dove concentrare l'interazione degli utenti in una sorta di virtualità aumentata oppure distribuire all'interno dell'ambiente reale gli elementi virtuali raffiguranti elementi tipici da *escape room* stimolando così l'esplorazione da parte dei giocatori. Si è considerato questo secondo approccio come il più adatto ai nostri scopi.

Infine, dopo aver definito che l'applicazione sarebbe stata una *escape room* in realtà aumentata e collaborativa, si è stabilito che il numero di giocatori previsto sarebbe stato tre, sia per non complicare eccessivamente il *design* delle interazioni collaborative e sia per non dover aumentare considerevolmente il numero di partecipanti alla sperimentazione. Si è poi proceduto con la definizione di quali enigmi e meccaniche proporre all'interno del gioco al fine di poter applicare nel miglior modo possibile i paradigmi di interazione collaborativa e di *awareness* multi-utente visti nel capitolo precedente. Sì è stabilito dunque che approccio utilizzare per la sincronizzazione iniziale, optando per una prescansione, e si sono definite le interazioni base con gli oggetti virtuali (interazioni non collaborative); nelle successive sottosezioni 4.3.1 e 4.3.2 tali argomenti verranno affrontati più nel dettaglio. Sì è proseguito quindi con un design completo dell'esperienza, individuando con precisione gli enigmi che sarebbero stati associati ad una determinata interazione collaborativa ed elaborando dettagliatamente tutte le connessioni logiche tra i vari enigmi/interazioni. Nella sottosezione 4.3.3 verrà esaminata più in dettaglio la storia dietro tale esperienza e tutta la struttura di gioco.

Poichè però le *escape room* sono di natura giochi difficili da risolvere a cause in particolare del limite di tempo imposto, dove i giocatori sono lasciati per lo più a sè stessi e soprattutto raramente è una esperienza che si riesce a portare a termine fino al fondo se non si ha esperienza a riguardo, abbiamo ritenuto opportuno estrapolare dal design completo del gioco quattro scenari principali che coprissero i paradigmi d'interazione studiati. Tali scenari sono proposti senza limitazioni di tempo e con la possibilità di ottenere alcuni aiuti. Tali semplificazioni inoltre son da considerarsi del tutto necessarie soprattutto ai fini della sperimentazione per la valutazione finale,

dato che proporre di testare una esperienza che per natura sarebbe stata difficile persino da completare non sarebbe stato appropriato al fine di trarre dati significativi sull'applicazione dei paradigmi di interazione collaborativa e *awareness*. Il *design* dei quattro scenari scelti sarà brevemente descritto nella sottosezione 4.3.3 e poi affrontato più nel dettaglio nel successivo capitolo implementativo.

4.3.1 Pre-scansione

Al fine di rendere consistente l'ambiente AR per ciascun utente partecipante alla simulazione in modo semplice e permettere una pianificazione flessibile dello scenario aumentato, si è optato per l'adozione di un sistema di pre-scansione in cui uno degli utenti partecipanti scansiona l'ambiente reale e posiziona tutti gli oggetti virtuali necessari per l'esperienza, condividendone opportunamente la posizione una volta ultimata questa fase. Inoltre la pre-scansione si è resa necessaria in quanto l'applicazione, al fine di poter essere compatibile con le varie librerie di AR in commercio, deve adattarsi alla piattaforma meno potente (con meno caratteristiche e funzioni) e ARCore non permette nativamente la persistenza della simulazione AR in quanto non consente di memorizzare la scena ricostruita.

In seguito varrà esposto come è stato deciso di affrontare l'interazione degli utenti con gli oggetti virtuali presenti nella scena.

4.3.2 Interazione con oggetti virtuali

A seguito di un'attenta pianificazione delle meccaniche di interazione generiche all'interno della simulazione è possibile distinguere gli oggetti virtuali presenti in due tipologie specifiche: oggetti virtuali statici (che possono presentare parti interattive) e oggetti virtuali dinamici (che saranno sempre oggetti interattivi e potranno essere equipaggiabili o non equipaggiabili). Ciascun utente che partecipa alla simulazione può vedere ogni oggetto virtuale presente nella scena correttamente posizionato e, grazie alla sincronizzazione, può monitorare in tempo reale qualsiasi cambiamento di proprietà dell'oggetto (per esempio rotazione e traslazione) anche se ad agire su di esso sono altri utenti.

Oggetti statici

Sono oggetti virtuali che non possono essere "afferrati" e mantengono costante nel tempo la propria posizione all'interno della simulazione. Per poter

essere ispezionati richiedono azioni di movimento da parte degli utenti che dovranno fisicamente girarci attorno e avvicinarsi per analizzarli. **Oggetti statici interattivi** prevedono l'esecuzione di attività particolari da parte degli utenti per portare a termine determinati compiti, in quando possono essere provvisti di parti interattive come pulsanti, cilindri rotanti ecc. attivabili tramite tocco su schermo o eventualmente gesti della mano. Sono in genere oggetti virtuali di grosse dimensioni.

Oggetti dinamici

Sono oggetti virtuali che possono essere "afferrati" dagli utenti tramite tocco o gesti della mano, manipolati e riposizionati. Gli oggetti dinamici inizialmente sono appoggiati su delle superfici rilevate e interagendoci è possibile "prenderli in mano" e ancorarli al proprio dispositivo; in questo stato gli altri utenti vedono l'oggetto muoversi insieme all'utente che lo ha afferrato. Una volta afferrati questi oggetti si rendono disponibili due diverse modalità per ispezionarli e manipolarli (disponibili esclusivamente all'utente che risulta attualmente il possessore di quell'oggetto virtuale).

Nella prima modalità l'oggetto virtuale è ancorato al dispositivo, di conseguenza è possibile effettuare traslazioni di questo oggetto, e quindi muoverlo nell'ambiente, semplicemente spostando il proprio dispositivo. Le altre manipolazioni dell'oggetto vengono effettuate sfruttando i comandi tattili tipici degli HHD: è possibile quindi ruotare l'oggetto semplicemente effettuando uno *swipe* sullo schermo o scalarlo eseguendo un *pinch*. In realtà non si effettua mai un reale scalamento dell'oggetto afferrato; l'integrazione dello scalamento all'interno della *escape room* avrebbe potuto infatti creare confusione tra gli utenti, in quanto ingrandire o rimpicciolire elementi virtuali che appaiono molto realistici sarebbe potuta essere considerata una operazione "innaturale" in un contesto che si pone come realistico. Di conseguenza quello che si effettua è una semplice traslazione in avanti rispetto alla direzione frontale del dispositivo se si vuole rimpicciolire l'oggetto o viceversa una traslazione all'indietro se si vuole ingrandire. Dato che tale approccio può essere facilmente male interpretato dai fruitori se non correttamente illustrato, sono state adottate tecniche come la raffigurazione della mano virtuale e la generazione delle ombre degli oggetti sulle superfici rilevate.

La seconda modalità prevede invece di ancorare gli oggetti virtuali ad una posizione specifica dell'ambiente e di muoversi fisicamente attorno ad

esso per poterlo ispezionare. Tale tecnica permette anche di estendere facilmente l'interattività con gli oggetti virtuali a dispositivi HMD; infatti la prima modalità andrebbe mappata su *hand gestures* non sempre disponibili per questi tipi di dispositivo, mentre la seconda modalità può essere implementata molto semplicemente così com'è. Questa modalità è molto simile ad una tecnica che è già stata oggetto di studio demominata Freeze Object [25].

L'utente, una volta che ha afferrato un oggetto virtuale, può decidere di riposizionarlo su uno qualsiasi dei piani rilevati della sessione AR. Per aumentare il livello di realismo ed interattività dell'intera esperienza si è fatto in modo che quando l'oggetto virtuale, a seguito di traslazioni effettuate dall'utente con il movimento del dispositivo o tramite *pinch gesture*, entra in contatto con un piano rilevato o altri oggetti virtuali esso collida correttamente come se effettivamente lo si stesse appoggiando sul piano o su di un altro oggetto.

Alcuni oggetti dinamici, quando afferrati, possono anche essere **equipaggiati** come strumenti; . In modalità "equipaggiato" gli oggetti vengono posizionati in modo da non occupare tutto il campo di vista dell'utente che li regge e rimangono ancorati ad esso senza però dare la possibilità di ruotarli e avvicinarli/allontanarli. Quando gli oggetti sono in questo stato è possibile utilizzarli per i loro scopi. Un esempio di oggetto equipaggiabile può essere una torcia elettrica: premere il pulsante di accensione sulla torcia elettrica permette di accenderla e di creare un fascio di luce al centro dello schermo, che può essere utilizzato per rivelare dettagli nascosti. Per disattivare lo stato equipaggiato è possibile premere un pulsante dell'interfaccia, riportando l'oggetto in modalità "afferrato".

4.3.3 Flusso di gioco

In genere le migliori *escape room* sono quelle che presentano una scenografia e una "storia" di sottofondo studiate accuratamente al fine di aumentare l'immersione e il livello di partecipazione dei giocatori. Di conseguenza si è rivelato importante scegliere con attenzione uno stile di rappresentazione degli oggetti virtuali e di narrazione che potesse mantenere alto l'interesse e l'attenzione del giocatore. Dopo aver aver effettuato una breve ricerca sulle migliori *escape room* presenti sul mercato e facendo affidamento sulle numerose esperienze personali si è infine definito un tema e uno stile narrativo per l'applicazione.

Il tema principale è di tipo horror, liberamente ispirato ai romanzi e i racconti del celebre autore statunitense H.P. Lovecraft. Di conseguenza tutti gli oggetti virtuali presenti nella simulazione saranno caratterizzati da forme e colori che ricorderanno attrezzatura vittoriana di fine '800, con particolare cura a renderli quanto più "inquietanti" possibili anche tramite complesse e intricate animazioni. La narrazione avverrà attraverso alcune lettere o annotazioni con cui si potrà interagire durante l'esperienza. Di seguito una breve descrizione della storia che fa da sfondo all'esperienza. É possibile anche consultare lo schema in figura 4.3 a fondo capitolo per una visione schematica del flusso di gioco completo.

Storia

Viene rinvenuta una antica e strana scatola (a cui successivamente si farà riferimento anche con il nome inglese *contraption*) che pare nascondere un qualche tipo di meccanismo al suo interno; dopo averla esaminata attentamente i giocatori riescono ad aprirla parzialmente, mettendo in moto un meccanismo che scandirà 60 minuti oltre i quali l'anima dei giocatori verrà intrappolata per sempre all'interno della scatola. È quindi compito dei giocatori riuscire a risolvere tutti gli enigmi proposti entro i 60 minuti; se l'*escape room* viene completata con successo, la scatola si aprirà, liberando i giocatori e l'anima di tutti i giocatori precedenti che hanno fallito, i cui nomi verrano mostrati.

La strana scatola, nella mitologia a sfondo del gioco, è sostanzialmente un artefatto del dio abissale Hastur studiata per catturare le anime dei mortali. Non è possibile risalire a quando questo artefatto è stato creato e non è possibile distruggerlo, è stato progettato per scaturire la curiosità dei mortali che saranno invogliati ad attivarlo, inconsapevoli del pericolo a cui vanno incontro. Hastur però ha deciso di concedere un'opportunità di salvezza a chiunque attivi l'artefatto: nel momento dell'attivazione infatti parte del mondo abissale interferirà con quello reale attraverso la comparsa di diversi oggetti, strumenti ed elementi ornamentali in punti diversi della scena. Tali oggetti fanno parte di un complesso enigma ideato da Hastur e solo chi è abbastanza intelligente e farà realmente gioco di squadra potrà sperare di arrivare alla fine dell'esperienza e salvare la propria anima e quella dei compagni di gioco.

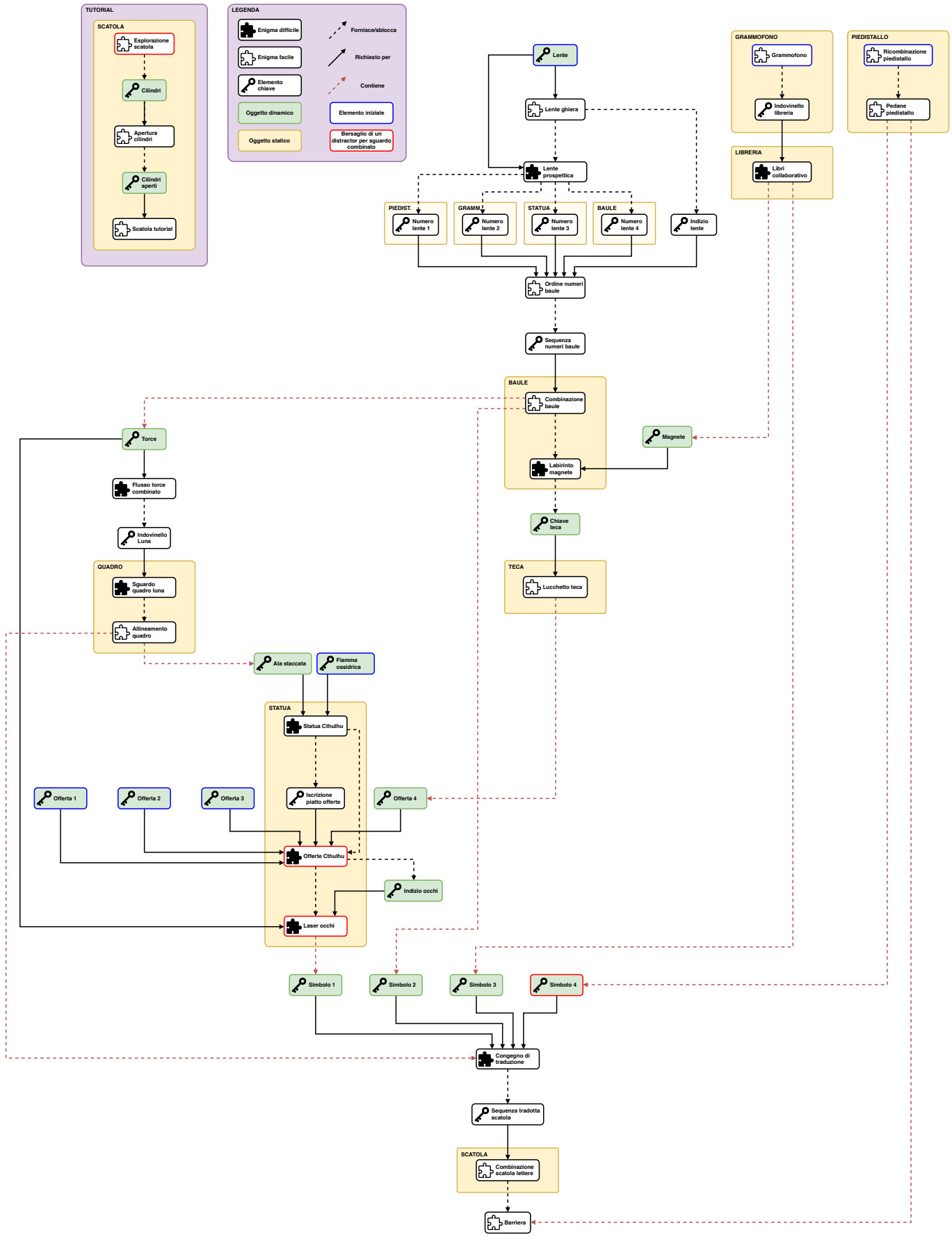


Figura 4.3. UML raffigurante l'intera struttura di gioco

Capitolo 5

Sviluppo dell'applicazione

In questo capitolo verrà affrontata principalmente l'implementazione effettiva dell'applicazione progettata nel capitolo precedente. Dopo una breve descrizione di tutti gli strumenti utilizzati per portare a termine lo sviluppo e delle operazioni di modellazione e texturing per la generazione degli *asset* necessari, si prosegue con una descrizione dettagliata dei vari framework che son stati programmati, con particolare attenzione ad evidenziare i vari punti salienti e le problematiche incontrate durante lo sviluppo. In particolare verranno esaminati: il framework di realtà aumentata, definito semplicemente come `ARLibrary`, con descrizione dettagliata delle più importanti funzioni fornite e come sono state implementate con la libreria di ARCore; il framework di *networking*, con dettagliata descrizione del sistema di sincronizzazione e di scambio dei dati; infine una panoramica sulle scene sviluppate al fine di poter effettuare una valutazione solida del lavoro svolto, con descrizione delle scene di *training*, delle scene sperimentali e infine delle scenari estratti dalla *escape room*.

L'architettura generale di ciò che è stato sviluppato consiste in un nucleo centrale corrispondente all'implementazione effettiva delle caratteristiche dell'applicazione chiamata `AR Client App`, che si appoggia su due interfacce da noi scritte e implementate, ovvero `ARLibrary` che permette di astrarre l'utilizzo di una particolare libreria di realtà aumentata e `INetworkManager` che invece permette l'indipendenza dalla specifica libreria di *networking* che si vuole utilizzare, dando la possibilità di cambiarla in futuro. La comunicazione e lo scambio dati tra le varie applicazioni avviene con un paradigma peer-to-peer dove però può esiste un particolare client chiamato `MasterClient` che può assumere ruoli particolari o effettuare proprie elaborazioni simulando un

contesto client-server. In figura 5.1 una rappresentazione di tale architettura; in questa figura si può notare una rappresentazione "generica" dell'applicazione con un numero arbitrario di client connessi. Come si può notare i moduli dell'interfaccia di `ARLibrary` che non sono stati stati implementati (ARKit, Windows Mixed Reality e il modulo per eventuali altre librerie) sono stati rappresentati in grigio. Non son stati rappresentati per semplicità i moduli relativi all'implementazione delle interfacce di `INetworkManager` delle varie piattaforme di *networking*.

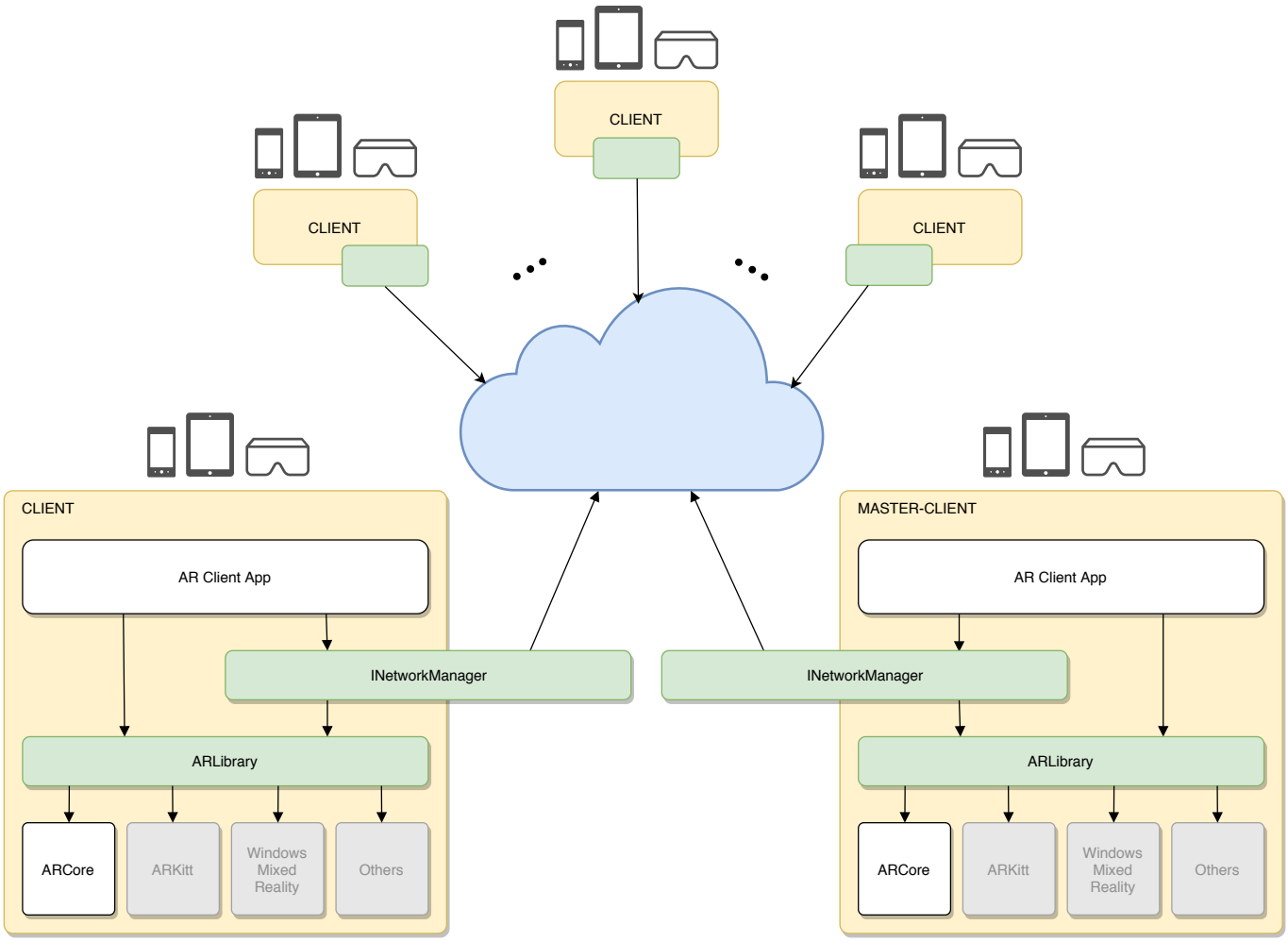


Figura 5.1. UML raffigurante l'architettura generale dell'applicazione

5.1 Strumenti e linguaggi utilizzati

Di seguito una breve panoramica degli strumenti che sono stati utilizzati per portare a termine questo lavoro. Il linguaggio principale in cui è stato scritta l'applicazione è il C#, il linguaggio di sviluppo di Unity che è stato il nostro principale ambiente di sviluppo assieme a VisualStudio; tutta la parte implementativa è stata effettuata con il supporto indispensabile del software di *versioning* Git, utilizzato tramite SourceTree. Per la parte puramente grafica dell'applicazione invece si è optato per Blender come strumento per la modellazione tridimensionale e Substance Painter in versione accademica per la parte di texturing e materiali. Per lo sviluppo delle caratteristiche peculiari della nostra applicazione abbiamo fatto ampio uso di plugin acquisiti dall'*Asset Store* di Unity di cui verrà fatto successivamente un breve elenco.

5.1.1 Unity, VisualStudio e Sourcetree

Unity Game Engine è il motore di gioco più semplice ed immediato da utilizzare, molto documentato e con una enorme *community* a supporto. Proprio grazie alla sua grande popolarità e semplicità d'uso, sono disponibili per questo motore moltissimi plugin ed estensioni forniti sia da appassionati che da grandi case produttrici a supporto dei propri prodotti (per esempio sono presenti i plugin relativi ad ARCore, ARKit e Windows Mixed Reality). Inoltre la grande versatilità di questo motore permette, in pochi passaggi, di pubblicare la propria applicazione per i più disparati dispositivi tra cui smartphone, tablet e dispositivi indossabili. Alla luce di tutto ciò e avendo dovuto già utilizzare questo strumento durante il percorso accademico, si è deciso di sfruttarlo come ambiente di sviluppo principale dell'applicazione e quindi adottare il C# come linguaggio di programmazione.

Per quanto riguarda invece l'IDE, si è deciso di non adottare quello presente integrato in Unity (MonoDevelop), ma di utilizzare Visual Studio Community di Microsoft. Le ragioni che ci hanno portato a questa scelta solo le seguenti: una maggior esperienza nell'utilizzo dell'IDE Microsoft maturata durante il percorso accademico e soprattutto l'ottima interfaccia utente, che consente un utilizzo piacevole e appagante del programma. Inoltre Visual Studio è generalmente considerato uno standard di settore come applicazione professionale d'eccellenza per lo sviluppo software. Di seguito in figura 5.2 alcune catture schermo di tali programmi in fase di sviluppo.

In aggiunta a Unity e Visual Studio si è deciso, per la parte implementativa, di fare affidamento su un ulteriore programma che si è rivelato di

cruciale importanza nel corso dello sviluppo. Si tratta di un programma di *version control*, Git, cioè un programma che permette di tenere traccia e di sincronizzare, tramite opportune operazioni di *push* e *pull*, tutte le operazioni di modifica che si sono eseguite sul codice sorgente. Questo ha permesso di lavorare in parallelo sullo stesso progetto ed eventualmente ripristinare precedenti versioni del codice in caso di problemi. Per comodità d'uso abbiamo utilizzato Git tramite il client desktop gratuito SourceTree.

Non si è ritenuto necessario rendere l'applicazione indipendente da qualsiasi plugin e di conseguenza dover implementare completamente tutto quanto, si è solo definita l'indipendenza da tutti i possibili plugin di realtà aumentata e tutte le estensioni relative al *networking*. Per il resto si sono utilizzate diverse estensioni presenti sull'*Asset Store* di Unity al fine di semplificare l'implementazione della logica dell'applicazione e e dell'interfaccia. Di seguito un breve elenco dei plugin che sono stati utilizzati.

Plugin di Unity installati

Tra i vari plugin che abbiamo ritenuto utili per l'applicazione figurano sia estensioni per semplificarne lo sviluppo effettivo, sia plugin per migliorare al massimo l'esperienza utente. Innanzi tutto è stato installato il plugin ufficiale di **ARCore** fornito da Google al fine di poter implementare correttamente l'interfaccia di `ARLibrary` riferita a questo modulo di realtà aumentata. **Fingers Lite** è stato utilizzato per implementare le *touch gesture* all'interno dell'applicazione. Per quanto riguarda il *networking* il discorso è analogo a plugin di realtà aumentata ed è quindi stato utilizzato il plugin ufficiale di **Photon Unity Networking** e **Photon Voice** per il modulo di chat vocale

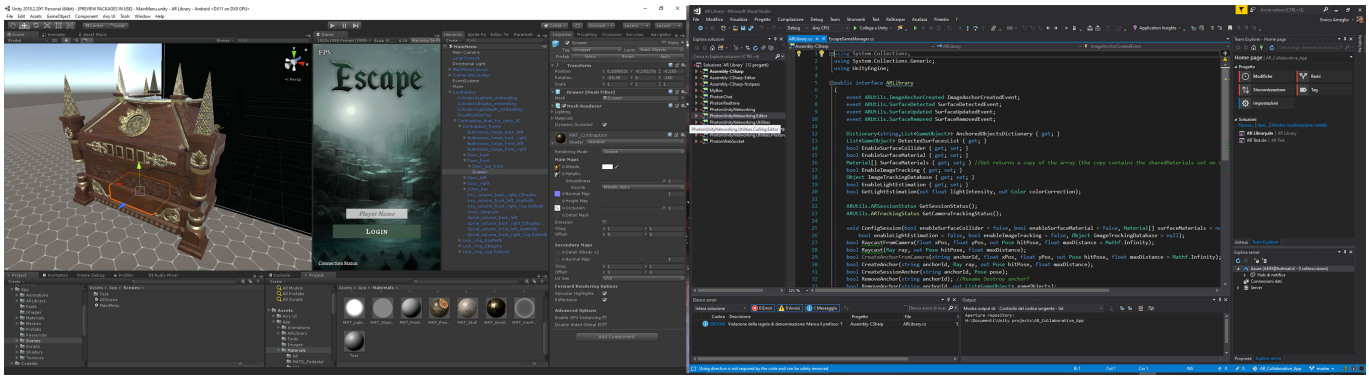


Figura 5.2. A sinistra Unity Game Engine mentre a destra Visual Studio

(si rimanda alle sezioni successive per le motivazioni che ci hanno portato a scegliere questa soluzione). Per facilitare il debug a *run-time* su dispositivi mobili si è deciso di adottare il plugin **Lunar Console** che permette di visualizzare la console di Unity, e quindi tutti i messaggi relativi, direttamente sullo schermo del dispositivo Android su cui si sta eseguendo l'applicazione. Per quanto riguarda l'interfaccia utente, non essendo a nostro parere soddisfacenti gli strumenti messi a disposizione da Unity per il rendering del testo, si è deciso per l'adozione di un plugin apposito che migliorasse sensibilmente la resa. **TextMeshPro** è apparsa subito come una soluzione valida è si è dimostrato un eccellente e versatile strumento. Inoltre, al fine di implementare specifiche *visual cues* precedentemente esaminate, in particolare le tecniche di evidenziazione visiva, si è deciso di sfruttare il plugin **QuickOutline** con alcune modifiche apportare ai suoi script al fine di soddisfare appieno gli scopi dell'applicazione. Per rendere più piacevole l'interazione dell'utene con l'interfaccia si è aggiunto il plugin **AiryUI** che mette a disposizione animazioni per gli elementi della UI. Infine son stati utilizzati altri due plugin: **MyBox** che rendeva disponibile alcune utilità di codice molto comode e infine **Lean Pool** per poter implementare un sistema di *pooling* degli oggetti che si è reso necessario nel corso dello sviluppo dell'applicazione, per evitare tempi di carimento delle scene troppo lunghi.

5.1.2 Blender e Substance Painter

Per poter modellare gli *asset* necessari alla nostra applicazione si è utilizzato il programma di modellazione 3D *open-source* **Blender**. Blender è un software estremamente versatile e molto potente che non ha nulla da invidiare ad altri

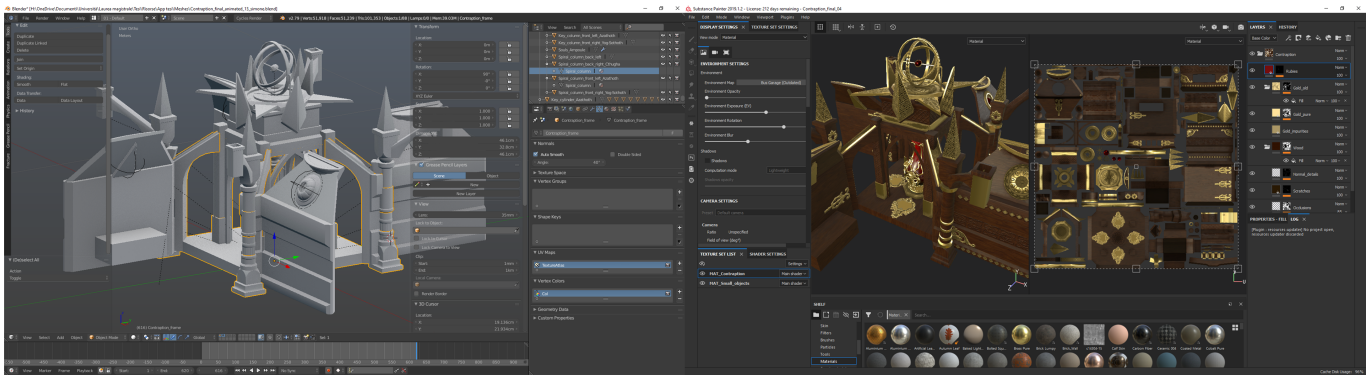


Figura 5.3. A sinistra Blender mentre a destra Substance Painter

programmi commerciali analoghi considerati standard di settore. Grazie alla lunga esperienza maturata a riguardo nel corso degli ultimi anni accademici e alla la grande flessibilità sia dal punto di vista della modellazione che delle animazione, Blender è stata una scelta praticamente ovvia.

Per quanto riguarda invece il *texturing* si è scelto di utilizzare **Substance Painter** di Allegorithmic in versione accademica come principale strumento. Substance Painter può essere considerato senza dubbio lo standard di settore per quanto riguarda la creazione di materiali e texturizzazione ed è infatti utilizzato ampiamente dalle più grandi case produttrici di contenuti digitali. Substance Painter è più che altro utilizzato per la produzione di materiali PBR, cioè basati sulla fisica e di conseguenza con una resa fotorealistica. In figura 5.3 alcune catture schermo delle applicazioni appena descritte.

5.2 Modellazione, animazione e *texturing*

Lo stile che si è voluto adottare per gli *asset* dell'applicazione è stato quello del fotorealismo. Di conseguenza tutti gli oggetti che avrebbero fatto parte della simulazione sono stati modellati seguendo quanto più fedelmente possibile la realtà, mentre i materiali associati sono stati generati seguendo il modello PBR che permette di ottenere i migliori risultati. Queste specifiche vanno a scontrarsi con l'esigenza di poter eseguire l'applicazione su hardware *hand-held* non sempre performante; di conseguenza si è reso necessario abbassare quanto più possibile il numero di poligoni a schermo e la pesantezza delle texture cercando di ottenere un buon *trade-off* tra prestazioni e resa visiva.

Al fine di ottimizzare al massimo la quantità di poligoni delle *mesh*, è stato adottato il metodo di modellazione *high-poly/low-poly*. Questo metodo consiste dapprima nella creazione di un modello non ottimizzato con numerosi poligoni (nell'ordine dei milioni, modello *high-poly*) e poi nel ricavare da questo, attraverso tecniche di ritopologizzazione, un modello ottimizzato con un numero di poligoni ragionevole (nell'ordine delle migliaia, modello *low-poly*). Una volta ottenuto il modello *low-poly* si procede con la tecnica di *UV unwrap* e settaggio dei materiali per predisporre la *mesh* ad essere correttamente texturizzata. Si è cercato di minimizzare il numero di materiali per *mesh* seguendo per quanto possibile la regola "un solo materiale per ogni mesh", questo per alleggerire ulteriormente il carico di lavoro del motore di gioco che si trova così a dover gestire un numero ragionevole di dati. A seguito

della ritopologizzazione però la qualità del modello risulta drasticamente diminuita, si rende quindi necessario adottare la tecnica del *normal baking* per riuscire ad annullare percettivamente la differenza tra il modello *low* e quello *high-poly*, ottenendo infine un'ottima qualità visiva nonostante un numero severamente inferiore di poligoni.

Si procede quindi caricando la *mesh low-poly* modellata in Blender all'interno di Substance Painter. Questo programma, tra le varie caratteristiche che possiede, permette anche di effettuare il *normal baking* a partire da una *mesh high poly* precedentemente impostata in un apposito menù di configurazione del *baking*. In realtà Substance non solo genera una *normal map* a partire da un modello a più alta risoluzione, ma può anche generare numerose altre mappe ricavate dalla forma dell'oggetto. Tali calcoli permettono al programma di "capire" la geometria dell'oggetto e applicare quindi particolari maschere dette *smart mask* ai materiali al fine di simulare per esempio l'accumulo di polvere nelle zone occluse oppure i graffi sui bordi. Una volta ottenute le opportune mappe dal modello *high-poly* si può procedere con l'allestimento dei materiali. I materiali che sono stati utilizzati sono per lo più delle versioni *custom* dei materiali presenti nella libreria base fornita da Allegorithmic oppure ottenuti dal catalogo online Substance Share sempre gestito da Allegorithmic. Alcuni materiali invece sono stati progettati da zero. Una volta completato il materiale, è bastato esportare le relative texture con la configurazione base per Unity ottenendo così la quattro mappe standard (*AlbedoTrasparency, AO, MetallicSmoothness* e *Normal*) da applicare al materiale generato con lo *Standard Shader* di Unity. Sì è constatato che una risoluzione massima pari a 2048×2048 fosse appropriata al fine di ottenere una buona qualità grafica senza inficiare troppo le prestazioni.

Per quanto riguarda invece le animazioni, queste son state pianificate ed eseguite direttamente su Blender. Non dovendo animare movimenti "organici", ma solo aperture meccaniche consistenti in combinazioni di rotazioni e traslazioni, non si sono effettuate operazioni di *rigging* e di conseguenza non si è reso necessario l'utilizzo di armature e altri strumenti di solito utilizzati in questi casi. É stato quindi adottato un metodo di animazione a *key frames*, registrando posizione e rotazione di ciascun elemento da animare in precisi *frame* temporali, con una interpolazione di tipo Bezier.

In seguito si è proceduto ad esportare il modello 3D in formato .fbx da Blender, includendo tutte le animazioni, e ad importarlo su Unity insieme a tutte le texture generate da Substance Painter. Il modello finito si ottiene infine estrapolandone i materiali dall'editor di Unity ed applicandoci le texure opportune. In figura 5.4 un render di alcuni degli *asset* creati, tali render



Figura 5.4. Una serie di oggetti di scena modellati in Blender e texturizzati con Substance Painter

sono stati ottenuti con il motore Iray integrato in Substance.

5.3 Framework di realtà aumentata

Come già anticipato nel precedente capitolo, è stato speso del tempo a studiare le varie librerie di realtà aumentata disponibili sul mercato, esaminandone accuratamente la documentazione al fine di identificare con precisione tutte le funzioni che sarebbe servito esporre all'interno della interfaccia generica AR che si sarebbe dovuta sviluppare. Prima di scendere nel dettaglio dell'implementazione del framework di realtà aumentata, occorre definire alcuni concetti chiave che serviranno a comprendere meglio le scelte fatte e le funzioni che si è deciso di esporre. Come prima cosa occorre definire cosa sia un ancora, quale sia il suo funzionamento, quali tipi di ancore possono essere generate e effettuare una valutazione delle differenze; poi va esaminato il concetto di *detected surface* con i possibili materiali applicabili, va definito il concetto di *light estimation* e riflessioni dinamiche e in seguito va definito l'*image tracking* e le problematiche che esso comporta. Conclusa questa panoramica sulle principali *features* fornite dalle librerie di AR e sugli studi compiuti, si procede con la descrizione dettagliata dell'interfaccia `ARLibrary`

5.3.1 Elementi chiave

Ancore in realtà aumentata

In una simulazione di realtà aumentata di ultima generazione l'utilizzo di un *marker* per il posizionamento degli oggetti virtuali all'interno dell'ambiente reale non è più strettamente necessario in quanto è possibile utilizzare, in sostituzione, quella che viene definita "ancora". Tale ancora non è altro che un insieme di dati visuali di una determinata parte della scena che la sessione AR memorizza e che utilizza come punto di riferimento per il posizionamento degli oggetti virtuali. In sostanza la sessione AR cattura quante più informazioni possibili ricavate dal *feed* della camera del dispositivo per avere un ricostruzione digitale dell'ambiente basata su punti di interesse o *feature points*, che possono essere per esempio gli angoli di un tavolo o un insieme di oggetti diversi che caratterizzano molto un determinato punto dello spazio. Infatti in un ambiente molto povero di punti d'interesse la sessione AR farà molta fatica a distinguere tra loro le diverse inquadrature dell'ambiente, portando di conseguenza ad un decremento della qualità dell'esperienza.

In conclusione quando si genera un ancora, questa può essere vista come un *marker* virtuale formato dalle informazioni visuali ottenute dalla camera; gli oggetti virtuali associati a quest'ancora saranno sovrapposti all'ambiente reale e rimarranno nella posizione prestabilita finché la sessione sarà in grado di riconoscere l'ambiente in cui l'ancora è stata calcolata.

Uno dei principali paradigmi d'interazione della realtà aumentata consiste nel fatto che gli utenti che fanno parte dell'esperienza possono muoversi liberamente all'interno della scena. Questo però comporta alcuni problemi piuttosto complessi da risolvere. Quando un dispositivo si muove nell'ambiente la sessione AR deve tenere conto dello spostamento in modo da mantenere coerente la posizione degli oggetti virtuali; per effettuare questi calcoli però la sessione ha a disposizione i dati posizionali del dispositivo forniti esclusivamente dai suoi sensori inerziali che è risaputo siano affetti da non trascurabili errori di deriva che aumentano con l'aumentare dello spostamento. In questo caso l'ancora o le ancore che sono state inserite all'interno della scena sono di cruciale importanza al fine di "ripristinare" la sessione AR eventualmente andata alla deriva a causa di lunghi spostamenti o spostamenti troppo veloci del dispositivo; la sessione infatti, appena il dispositivo inquadra una scena a cui è stata associata una ancora e si rende conto che i calcoli di spostamento del dispositivo sono affetti da un errore, ripristina immediatamente la scena annullando tale errore.

In generale è possibile distinguere due tipi di ancora: l'ancora propriamente detta, definita **`Anchor`** che viene associata, per esempio in ARCore e ARKit, a determinati elementi chiave ricavati dalla scena detti elementi `trackable` (per il momento sono supportate tutti i tipi di superficie che siano piani orizzontali/verticali, particolari *feature points* oppure immagini rilevate prese da un dizionario) oppure un'ancora non associata ad un determinato elemento, ma alla sessione AR in generale, definita **`SessionAnchor`** (poichè per esempio non è stato trovato alcun `trackable` nei paraggi).

Sono stati effettuali alcuni test qualitativi al fine di verificare le differenti prestazioni tra questi due tipi di ancora, in particolare è stato introdotto anche un terzo tipo di ancora fittizia detta **`WorldAnchor`** che in realtà non è un ancora, ma è semplicemente un riferimento all'interno della scena basato sulle coordinate del mondo di Unity; in realtà non viene mai utilizzata realmente, è stata implementata al solo scopo di verificare l'errore generato dai sensori inerziali quando non sono supportati da alcun tipo di ancora. Di seguito in figura 5.5 alcune immagini tratte dall'applicazione sviluppata per effettuare il test di implementazione della libreria di AR, di cui verranno chiariti i dettagli in successive sezioni. In particolare nella figura di sinistra si sono generate due ancore, una di sessione e una sul `trackable` riferito al piano del tavolo, nello stesso punto identificato con un *tap* dell'utente sullo schermo. Il punto di ancoraggio è stato rappresentato con una *mesh* raffigurante Andy, la mascotte di Android. Dopo aver mosso ragionevolmente il dispositivo all'interno della scena e successivamente reinquadrato la zona delle ancore, si è constatata una differenza seppur lieve nel mantenimento della posizione con una maggiore precisione delle ancore di tipo `trackable`. Nell'immagine di destra è presente invece un test leggermente più avanzato in cui era in fase di implementazione il modulo multi-utente (si rimanda, per una descrizione più dettagliata, alla relativa sezione sul *networking*). In tale immagine sono visualizzate tre ancore distinte. L'indicatore arancione identifica il punto in cui un altro utente all'interno della simulazione ha creato un ancora (l'informazione viene quindi trasmessa agli altri utenti via network), l'ancora di tale indicatore è localizzata sul *marker*, quindi potenzialmente in un luogo lontano da dove effettivamente viene generato l'oggetto. In genere posizionare un oggetto virtuale lontano dall'ancora a cui è associato comporta severe perdite di qualità poiché la camera del dispositivo non può inquadrare in contemporanea sia l'oggetto virtuale che la zona dell'ancora impedendo così alla sessione di AR di effettuare i corretti aggiustamenti. A partire da questo punto definito dall'utente all'interno della simulazione, oltre al suddetto indicatore arancione, si generano anche altri due indicatori

insieme ad altrettante ancore. Il primo indicatore blu viene generato insieme ad un ancora di sessione nel punto esatto di *spawn* dell'indicatore arancione, a partire da questo punto nello spazio vengono effettuati due `raycast` lungo la direzione positiva e negativa delle normali, se tali `raycast` incontrano un oggetto `trackable` allora nel punto di intersezione viene generata un ancora `trackable` ed un indicatore verde.

A seguito di valutazioni qualitative si è riscontrato che: sincronizzare un oggetto lontano dal *marker* e poi effettuando qualche movimento con il dispositivo la posizione dell'indicatore arancione subiva forti variazioni, mentre l'indicatore blu subiva variazioni meno significative. L'indicatore verde invece subiva variazioni trascurabili, ma la posizione non corrispondeva con esattezza al punto indicato dall'utente in quanto opportunamente traslata sul primo piano incontrato. Le conclusioni che si possono trarre da ciò sono che sia le ancora di sessione che quelle associate ad un `trackble` sono affidabili, con una nota a favore di quest'ultime, poiché possono essere piazzate esattamente sul piano rilevato dai diversi utenti e quindi da considerarsi più precise. Questo studio sarà alla base dell'implementazione di un sistema di sincronizzazione robusto tra le varie sessione AR dei diversi utenti partecipanti alla simulazione, come verrà esaustivamente esaminato nelle opportune sezioni.

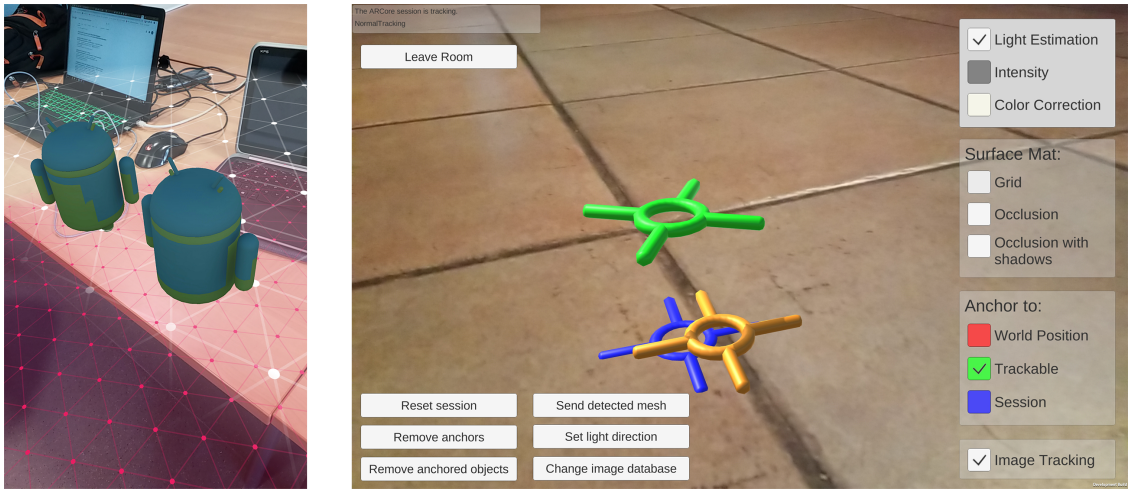


Figura 5.5. Catture schermo dell'applicazione sviluppata per testare `ARLibrary`

Rilevazione delle superfici, *light estimation* e riflessioni dinamiche

ARCore come ARKit permette di generare, a partire dai dati visuali estrapolati dalla fotocamera, delle superfici piane virtuali che vanno a sovrapporsi alle superfici del mondo reale come tavoli, pavimenti o muri. Tali elementi vengono continuamente aggiornati nel corso della sessione, espansi se la camera del dispositivo inquadra porzioni più ampie di una superficie sulla quale era già stato rilevato un piano oppure fusi se la sessione AR ritiene che due porzioni di piano rilevate facciano parte della stessa superficie reale. Queste superfici virtuali, che fanno parte della categoria degli oggetti `trackable`, subiscono continui aggiustamenti al fine di coincidere quanto più possibile con gli effettivi piani reali e risultano fondamentali per una resa quanto più stabile e coerente della sessione AR. Gli oggetti virtuali infatti vengono generalmente "agganciati" a tali piani attraverso le `Anchor` per quanto riguarda ARCore, per ARKit invece ogni piano rappresenta già di per sè una `Anchor`.

Su Unity questi piani vengono trattati come `GameObject` che possono cambiare forma e dimensione a *run-time* e che presentano naturalmente tutte le caratteristiche tipiche di un *asset* classico con le diverse componenti Unity, quali per esempio `MeshRender` o `MeshCollider`. Questo permette di accedere alle caratteristiche dei piani e quindi di modificarne a piacimento e in maniera agevole la resa visiva durante la simulazione.

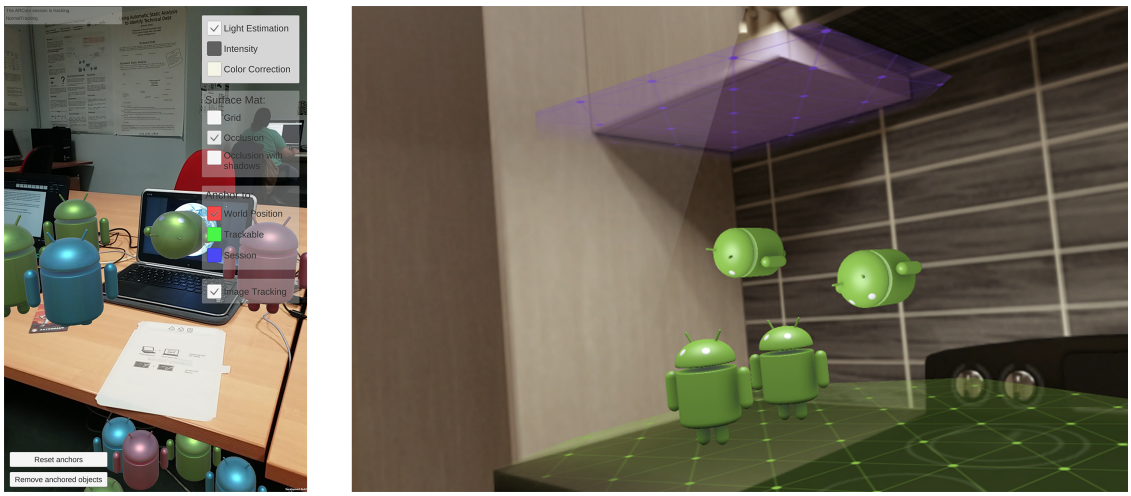


Figura 5.6. A sinstra l'applicazione sviluppata per testare `ARLibrary` che presenta occlusione sui piani rilevati, mentre a destra una immagine di repertorio del *plane detection* `ARLibrary`

É possibile quindi, per esempio, rendere questi piani trasparenti o semi-trasparenti, dotarli di un `collider` per la simulazione fisica oppure far sì che ricevano le ombre proiettate dagli oggetti virtuali. In figura 5.6 a destra una immagine di repertorio raffigurante un progetto di base di ARCore con il rilevamento piani in azione, mentre a sinistra l'applicazione di verifica di `ARLibrary` che adotta un materiale occlusivo sui piani rilevati.

La *light estimation* è una funzione messa a disposizione dalle maggiori librerie di realtà aumentata per incrementare il livello di realismo degli oggetti virtuali presenti in scena. Attraverso questa funzione la sessione AR cattura le informazioni di illuminazione presenti in scena attraverso il *feed* video della fotocamera ed effettua una stima sull'intensità e sul colore della luce presente. Tali dati vengono forniti al programmatore che potrà quindi adattare opportunemante l'illuminazione della scena, per esempio impostando le luci presenti basandosi sull'intensità e sul colore della scena reale (come avviene in maniera automatica in ARKit con la classe `ARSCNView` che usa automaticamente queste informazioni per impostare i paramtri di illuminazione di `SceneKit`) oppure modificando direttamente lo *shader* per la rappresentazione degli oggetti virtuali (come nell'applicazione di test di ARCore fornita da Google). Nelle ultime versioni di queste librerie è possibile anche stimare la direzione della luce principale dell'ambiente.

Un'altra funzione molto interessante per aumentare il realismo degli oggetti virtuali presenti in scena è la riflessione dinamica. In pratica tale caratteristica permette di simulare in modo realistico un oggetto virtuale riflettente all'interno della scena aumentata. Questo oggetto dovrà riflettere di conseguenza l'ambiente reale oltre che eventuali altri oggetti virtuali presenti, per risultare credibile. ARKit fornisce tale opportunità catturando le informazioni della scena reale in una *environment texture* nella forma di una *cube map*; questa texture verrà utilizzata in seguito come *reflection probe*, cioè come campione per il calcolo delle riflessioni. Poiché non è possibile ottenere una completa *cube map* in casi d'uso realistici, le librerie "ricostruiscono" le informazioni mancanti sfruttando opportuni algoritmi di *machine learning* (per poter completare una *cube map* con tutti i dati visuali, infatti, sarebbe richiesto all'utente di effettuare una panoramica a 360 gradi dell'ambiente). Per quanto riguarda invece la piattaforma ARCore, prima dell'ultima versione 1.10 rilasciata nel giugno 2019, non venivano fornite da Google funzionalità analoghe; per questo motivo si era sviluppato appositamente un modulo

per le riflessioni nell'implementazione ARCore della libreria di realtà aumentata, seguendo alcune indicazioni fornite durante una conferenza di Unity [65]. Dopo il rilascio della versione 1.10, questo modulo è stato sostituito direttamente dalle chiamate alla libreria ARCore. In figura 5.7 un esempio di oggetti totalmente riflettenti in un contesto aumentato dove sono state attivate le riflessioni dinamiche: a sinistra la nostra implementazione usando una versione di ARCore precedente alla 1.10, a destra una esempio su ARKit [54].



Figura 5.7. A sinistra le riflessioni dinamiche implementate da un modulo custom usando una versione di ARCore precedente alla 1.10, a destra un utilizzo di tali riflessioni in ARKit

Rilevazione immagini

Una delle funzioni storiche della realtà aumentata, permette di mostrare oggetti virtuali in determinati punti dell'ambiente dove sono riconosciuti specifici pattern visuali (immagini fisiche) detti comunemente *marker*. Questi *marker*, agli albori dell'AR, risultavano fondamentali per poter rappresentare un oggetto virtuale in un ambiente reale non potendo contare su altri ausili. Al giorno d'oggi, con il rilevamento piani e la possibilità di definire ancore durante la sessione, questa caratteristica è caduta in secondo piano, anche se risulta comunque utile nel caso si vogliano, per particolari requisiti dell'applicazione, mostrare certi oggetti virtuali solo in presenza di determinati *marker*.

Il funzionamento standard dell'*image tracking* consiste nell'inquadrare, per una certa percentuale di schermo, una immagine fisica che sia stata precedentemente inserita all'interno di un database apposito nella logica dell'applicazione AR. Nel momento in cui il *marker* viene riconosciuto dalla sessione AR, sopra di esso viene renderizzato l'oggetto corrispondente a quell'immagine. Alla stessa immagine possono esser associati più oggetti oppure possono essere presenti nella scena molteplici marker ciascuno associato ad un determinato oggetto virtuale. Quando un marker viene mosso all'interno dell'inquadratura, l'oggetto virtuale si muove *real-time* in modo solidale all'immagine, permettendo per esempio paradigmi di interfacce tangibili (esaminare un oggetto virtuale ruotando il*marker* inquadrato). Quando invece il *marker* esce dall'inquadratura, l'oggetto virtuale ad esso associato viene rimosso e solo dopo averlo inquadrato nuovamente riappare.

Analizzando tale caratteristica tra le varie piattaforme prese in esame, si sono tratte alcune conclusioni interessanti. Per prima cosa si è esaminata ARCore e si è scoperto da subito che i *marker* e l'*image traking* in generale hanno un utilizzo diverso rispetto a quello appena esaminato. Sostanzialmente ARCore, appena rileva un *marker* genera un ancora detta `ImageAnchor` su di esso (è un'ancora su `trackable`, col il *marker* in questo caso come oggetto `trackable`) e smette di effettuarne il tracciamento a meno di non inquadrarlo nuovamente molto da vicino. Questo approccio permette di mantenere ancorato l'oggetto virtuale al *marker* senza necessariamente inquadrarlo che però presenta l'aspetto negativo di impedire che una manipolazione fisica del *marker* produca un aggiornamento *real-time* dell'oggetto virtuale associato. Abbiamo analizzato a fondo il problema e si è scoperto che modificando alcuni parametri di configurazione della sessione è possibile ottenere un risultato analogo al comportamento standard dell'*image tracking*. Nel comportamento di base gli script di ARCore cercavano nella scena le `AugmentedImages` solo se erano considerate dal sistemo come `UPDATED` (periodicamente, evidentemente quando sono considerate molto vicine e quindi occupano una determinata percentuale di schermo); modificando tale comportamento e cercando le `AugmentedImages` o i *marker* in ogni frame è possibile aggiornare quasi in tempo reale la posizione degli oggetti virtuali anche qualora il *marker* si sia mosso. Il risultato però non risulta fluido.

Al momento di questa analisi era possibile leggere sul sito ufficiale della documentazione di ARCore che "*ARCore cannot track a moving image, but it can resume tracking that image after it stops moving*", attuamente invece, con l'aggiornamento 1.9 della piattaforma hanno aggiunto il supporto per il *tracking* delle immagini in movimento. ARKit e Vuforia invece supportavano,

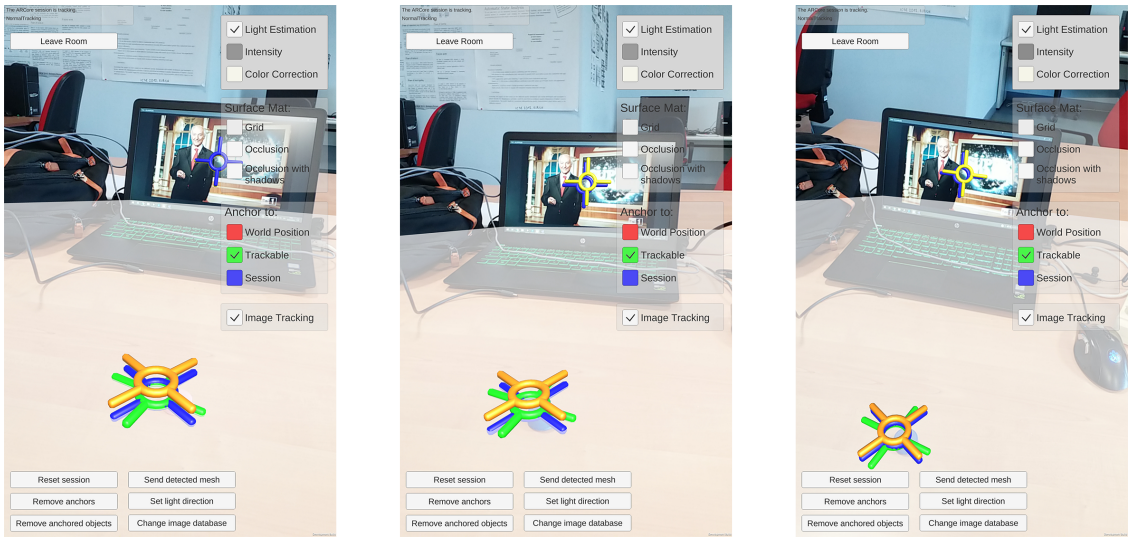


Figura 5.8. Tre catture schermata dall'app di test di `ARLibrary` con *image tracking* di una immagine riprodotta sullo schermo di un PC

già in fase di studio delle piattaforme, *image traking* in movimento.

Si è pensato di sfruttare ARCore per il *surface detection* e Vuforia per *image traking*, cercando di prendere il meglio di ciascuna piattaforma, ma non è stato possibile in quanto Vuforia permette l'utilizzo di ARCore, ma solo come ausilio per funzioni proprietarie. Nella figura 5.8 un esempio di tracciamento dell'immagine dall'applicazione di test di `ARLibrary`.

5.3.2 Dettagli tecnici su `ARLibrary` e `ARManager`

In questa sezione si andrà ad esaminare le classi che sono state implementate al fine di raggiungere lo scopo di creare un framework di realtà aumentata indipendente dalla piattaforma, con una analisi dettagliata delle funzioni esposte. In seguito si procederà con la descrizione di come è stato implementato il modulo relativo ad ARCore e con l'analisi di una applicazione di test sviluppata per verificare il funzionamento corretto del framework. In figura 5.9 una rappresentazione schematica delle principali funzioni ed utilità fornite dal framework, per l'UML completo consultare 5.21 a fine sezione.

Come si può vedere dall'UML semplificato, si è deciso di esporre, per il framework, tutte le funzioni che permettono la completa gestione delle ancore: creazione e rimozione delle ancore (sia normali che di sessione), effettuare *attach* e *detach* di oggetti virtuali a tale ancore e ottenere informazioni di

posa dell'ancora. Inoltre si è deciso di fornire funzioni per: resettare o configurare la sessione, effettuare operazioni di *raycast*, impostare il database delle immagini per l'*image tracking*, selezionare il materiale per le superfici rilevate, definire i parametri di un "visualizzatore" di prossimità, ottenere la *light estimation* della scena ed effettuare trasformate per passare dal sistema di coordinate del mondo a quello dell'ancora e viceversa. Infine si è deciso di fare affidamento su due strutture dati principali per mantenere in memoria informazioni utili per la simulazione AR e di predisporre un sistema ad eventi. Sono disponibili ulteriori funzioni minori che verranno esaminate brevemente nel corso di questa sezione.

`ARManager`

Per prima cosa si procede a descrivere la classe `ARManager`. Lo scopo principale di tale script è quello di creare una istanza, utilizzando un pattern di programmazione di tipo *singleton*, del modulo corretto che implementa

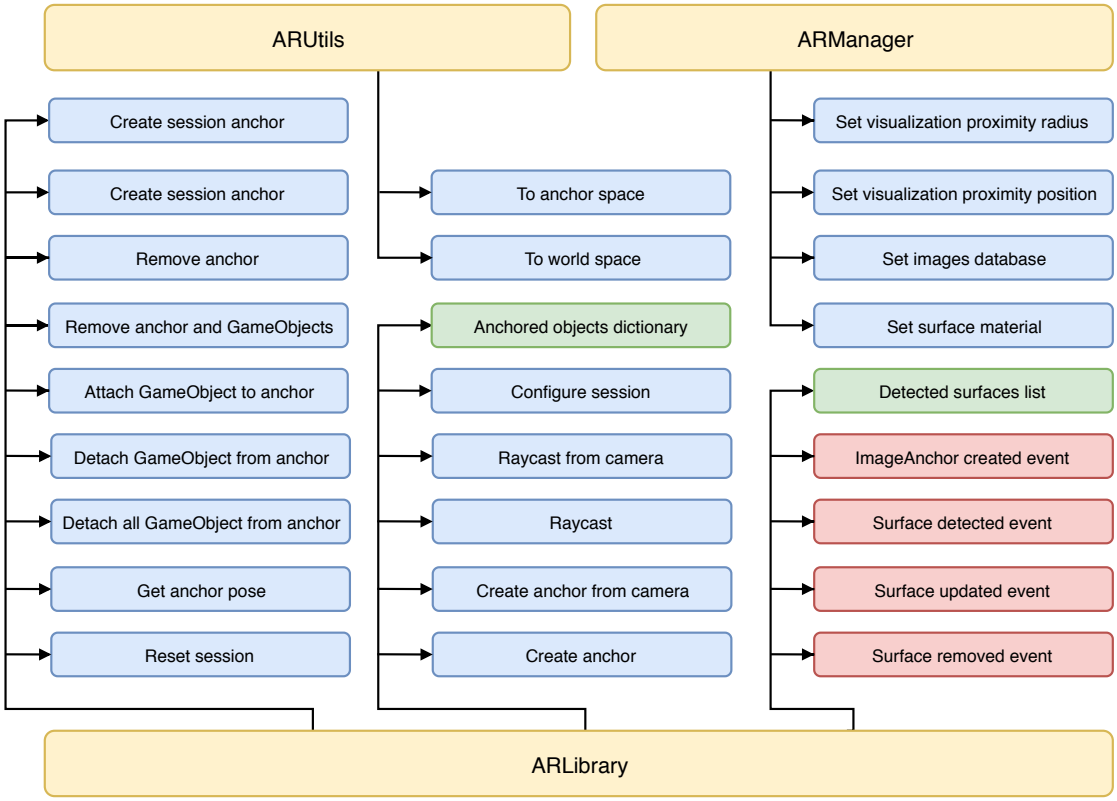


Figura 5.9. UML semplificato

l'interfaccia `ARLibrary` effettuando un controllo opportuno sulla piattaforma su cui il programma è in esecuzione. Come si vedrà sono stati sviluppati due diversi moduli che implementano `ARLibrary`: il modulo ARCore che sfrutta direttamente le API fornite da Google con la classe `ARCoreLibrary` e un modulo di debug da eseguire nell'ambiente `UNITY_EDITOR` con la classe `ARLibraryTest`. Tale modulo di debug si è reso necessario per poter testare direttamente sull'editor il funzionamento corretto dell'applicazione, evitando quindi di fare una *build* ogni volta l'eseguibile su una piattaforma Android. Di seguito, nel codice 5.1 un estratto dello script di `ARManager` dove viene generato il *singleton* e viene inizializzata correttamente il modulo da utilizzare per la sessione AR con la funzione `InitializeARLibrary()`. In questa funzione infatti, attraverso una opportuna direttiva #if al preprocessore di C#, inizializza il modulo corretto in base alla piattaforma di esecuzione. Come è possibile osservare da questo estratto di codice, solo la parte relativa alle due direttive `#define UNITY_EDITOR` e a `UNITY_ANDROID` sono state implementate, lasciando le sezioni relative a `UNITY_IOS` e `UNITY_WINDOWS_UWP` vuote per eventuali implementazioni future.

```
1  private static ARManager _instance = null;
2  private static readonly object Padlock = new object();
3
4  public static ARManager Instance
5  {
6      get
7      {
8          lock (Padlock)
9          {
10             return _instance;
11         }
12     }
13 }
14 private static ARLibrary _arLibraryInstance = null;
15 private static ARUtils.ARPlatform _arPlatform = ARUtils.ARPlatform.
       None;
16
17 private static void InitializeARLibrary()
18 {
19 #if UNITY_EDITOR
20         _arLibraryInstance = _instance.gameObject.AddComponent<
       ARLibraryTest>();
21         _arPlatform = ARUtils.ARPlatform.UnityEditor;
22 #elif UNITY_ANDROID
23         _arLibraryInstance = _instance.gameObject.AddComponent<
       ARCoreLibrary>();
```

```
24          _arPlatform = ARUtils.ARPlatform.ARCore;
25 #elif UNITY_IOS
26 #elif WINDOWS_UWP
27 #endif
28 }
```

Codice 5.1. Implementazione del *singleton* di `ARManager`

Esistono diverse altre funzioni fornite dalla classe `ARManager`. Tali funzioni permettono di modificare il materiale delle superfici rilevate, impostare il database per l'*image tracking* e definire un "raggio di prossimità" quando si avvicinano gli oggetti virtuali ai piani rilevati.

La funzione

```
public void SetOverlaySurfaceMaterials(SurfaceRenderType
    surfaceType1, SurfaceRenderType surfaceType2 =
    SurfaceRenderType.None)
```

con il secondo parametro opzionale, permette di settare il materiale con cui renderizzare le superfici rilevate dalla sessione. I parametri della funzione sono di tipo `SurfaceRenderType` che è un `enum` che può assumere i seguenti valori:

```
public enum SurfaceRenderType : int { None, Visualization,
    VisualizationProximity, Occlusion, OcclusionWithShadows};
```

Tali enum corrispondono ad altrettanti materiali che possono essere settati accedendo alle diverse variabili pubbliche dello script.

Andando più nel dettaglio il primo materiale consente di visualizzare un pattern esagonale su tutti i piani rilevati; il secondo materiale viene mostrato quando un oggetto virtuale afferrato è in prossimità di un piano rilevato e visualizza, solo in quella zona, il pattern esagonale; il terzo materiale è quello occlusivo mentre l'ultimo, oltre ad occludere gli oggetti virtuali, permette di renderizzarne le ombre. La funzione

```
public void SetImageTrackingDatabase();
```

Permette invece di settare un database di immagini per poter effettuare l'*image tracking*. Tale database può essere settato *run-time* e presenta alcune differenze con la versione che adotta ARCore. La piattaforma di Google, quando il database viene settato a `NULL`, mantiene comunque in memoria (e continua quindi a tracciare) le `ImageAnchor` che ha già rilevato, senza però tracciare *marker* che non rilevato prima del settaggio a `NULL` del database. Quando invece di essere settato a `NULL` il database viene sostituito con un altro, ARCore mantiene le `ImageAnchor` già rilevate nell'ultima posa in cui sono state tracciate e smette di effettuarne il *tracking* (non rileverà più altre

immagini che facevano parte del vecchio database). Il nuovo database invece viene rilevato normalmente. Si è deciso di adottare una strategia diversa per quanto riguarda il cambio database *real-time*: con `ARLibrary`, disabilitare o cambiare il database di immagini comporta la rimozione di tutte le `ImageAnchor` precedenti, con tutti i `GameObject` associati a queste.

Le due funzioni

```
public void SetVisualizationProximityPosition(Vector3 position)
public void SetVisualizationProximityRadius(float radius)
```

permettono di configurare, con posizione e raggio, il feedback di prossimità sui piani rilevati. Si è reso possibile, attraverso questa funzionalità, fornire un feedback utile all'utente che in quel momento sta manipolando un oggetto virtuale dinamico e vuole appoggiarlo su un piano, permettendo di visualizzare una variazione visiva significativa sul materiale della superficie rilevata quando un oggetto virtuale è nei paraggi.

Sono state create anche altre due classi, oltre ad `ARManager` e `ARLibrary`. `ARUtils` che non è altro che una piccola classe di utilità con alcune funzione atte a semplificare il calcolo delle matrici di trasformazione per la sincronizzazione attraverso il *marker*, di cui si tratterà nell'apposita sezione di questo capitolo, e `ARLight` che invece effettua alcune operazioni con la *light estimation* per impostare correttamente una luce all'interno della simulazione.

`ARLibrary`

Si fornisce di seguito il codice completo di tale interfaccia dove è possibile notare le funzioni, le strutture dati e gli eventi che si è voluto implementare per lo sviluppo del framework. Nel corso di questa sezione si andranno ad esaminare alcune delle sue principali caratteristiche.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public interface ARLibrary
6  {
7      event ARUtils.ImageAnchorCreated ImageAnchorCreatedEvent;
8      event ARUtils.SurfaceDetected SurfaceDetectedEvent;
9      event ARUtils.SurfaceUpdated SurfaceUpdatedEvent;
10     event ARUtils.SurfaceRemoved SurfaceRemovedEvent;
11
12     Dictionary<string,List<GameObject>> AnchoredObjectsDictionary {
       get; }
13     List<GameObject> DetectedSurfacesList { get; }
```

```
14      bool EnableSurfaceCollider { get; set; }
15      bool EnableSurfaceMaterial { get; set; }
16      Material[] SurfaceMaterials { get; set; } //Get returns a copy
    of the array (the copy contains the sharedMaterials set on
    surfaces objects)
17      bool EnableImageTracking { get; set; }
18      Object ImageTrackingDatabase { get; set; }
19      ARUtils.LightEstimationMode LightEstimationMode { get; set; }
20      bool GetLightEstimation(out float lightIntensity, out Color
    colorCorrection);
21
22      ARUtils.ARSessionStatus GetSessionStatus();
23      ARUtils.ARTrackingStatus GetCameraTrackingStatus();
24
25      void ConfigSession(bool enableSurfaceCollider = false, bool
    enableSurfaceMaterial = false,
26          Material[] surfaceMaterials = null,
27          ARUtils.LightEstimationMode lightEstimationMode = ARUtils.
    LightEstimationMode.Disabled,
28          bool enableImageTracking = false, Object
    imageTrackingDatabase = null);
29      bool RaycastFromCamera(float xPos, float yPos, out RaycastHit
    hitInfo, float maxDistance = Mathf.Infinity);
30      bool Raycast(Ray ray, out RaycastHit hitInfo, float maxDistance
    );
31      bool RaycastFromPose(Pose referencePose, out RaycastHit hitInfo
    , float maxDistance);
32      bool CreateAnchorFromCamera(string anchorId, float xPos, float
    yPos, out Pose hitPose, float maxDistance = Mathf.Infinity);
33      bool CreateAnchorFromRay(string anchorId, Ray ray, out Pose
    hitPose, float maxDistance);
34      bool CreateAnchorFromPose(string anchorId, Pose referencePose,
    out Pose hitPose, float maxDistance);
35      bool CreateSessionAnchor(string anchorId, Pose pose);
36      bool DestroyAnchor(string anchorId);
37      bool DestroyAnchor(string anchorId, out List<GameObject>
    gameObjects);
38      bool DestroyAnchorAndGameObjects(string anchorId);
39      bool AttachGameObjectToAnchor(string anchorId, GameObject
    gameObject, bool worldPositionStays);
40      bool DetachGameObjectFromAnchor(string anchorId, GameObject
    gameObject);
41      bool DetachAllGameObjectsFromAnchor(string anchorId, out List<
    GameObject> gameObjects);
42      bool GetAnchorPose(string anchorId, out Pose anchorPose);
43      void ResetSession();
```

```
44
45 }
```

Codice 5.2. Codice sorgente di `ARLibrary`

Come è possibile leggere dal codice, si è deciso di introdurre quattro eventi fondamentali all'interno del framework di realtà aumentata. Un evento, come si può leggere dalla documentazione ufficiale di `.NET`, è un messaggio inviato da un oggetto per segnalare l'occorrenza di un'azione; le azioni che si possono eseguire all'interno di una simulazione AR sono molteplici e si è deciso di selezionarne alcune principali e legarle a determinati eventi. Tali azioni sono:

- la creazione da parte della sessione AR di una `ImageAnchor`;
- il rilevamento di una superficie da parte della sessione AR;
- l'aggiornamento di una superficie rilevata (cioè l'estensione di una superficie pre-esistente o la fusione di due superfici separate);
- la rimozione di una superficie da parte della sessione AR (quando per esempio due superfici vengono fuse, una delle due vene per forza di cose eliminata.

Dopo aver esaminato la documentazione relativa alle varie piattaforme AR, si è scoperto che ARKit adotta ampiamente un sistema di eventi a supporto della sessione, mentre ARCore no. Ispirandoci quindi alle soluzioni adottate da Apple, che sono apparse più efficaci, si è deciso di adottare un sistema simile all'interno del framework. Sì è deciso di emettere un evento ogni qual volta la sessione AR genera un'ancora tramite *image tracking* e ogni qual volta viene rilevato un nuovo piano o aggiornato/rimosso un piano già rilevato. Questo permette una gestione flessibile del sistema di rilevamento superfici, potendo effettuare controlli oppure operazioni particolari ad ogni aggiornamento dei piani rilevati semplicemente registrandosi ad un opportuno evento.

Per quanto riguarda invece le strutture dati utilizzate, si è deciso di mantenere in memoria l'elenco di tutti gli oggetti ancorati in un opportuno dizionario e l'insieme di tutte le superfici rilevate nel corso della simulazione in una lista. Il dizionario è formato da un elenco di `entry` in formato chiave/valore: un identificativo univoco dell'ancora in formato `string` che corrisponde alla chiave e una lista di `GameObject` che formano il valore. Tale lista di `GameObject` identifica tutti gli oggetti virtuali che sono assegnati ad una determinata ancora. Ogni volta che si genera un'ancora quindi va aggiornato nell'implementazione il dizionario inserendo una nuova chiave e assegnandoci un valore che corrisponde alla lista di oggetti che si desidera legare a tale

ancora; inoltre se un oggetto viene rimosso dalla scena anche il dizionario va aggiornato e se nel caso questo oggetto risultasse l'ultimo della lista, anche l'ancora va rimossa per questioni di efficienza, in quanto ormai inutile non avendo più alcun oggetto virtuale assegnato; analogamente, se si rimuove l'ancora o si effettua qualsiasi altra operazione che modifica lo stato delle ancore o degli oggetti associati, questo dizionario va opportunamente aggiornato. Questi sono i comportamenti previsti per la gestione di tale struttura dati, che diventerà quindi del tutto trasparente al programmatore che utilizza l'interfaccia `ARLibrary`, il quale potrà creare e gestire ancore a piacere senza doversi preoccupare di gestire anche tale struttura dati.

La seconda struttura dati di `ARLibrary` è invece una lista di tutte le superfici rilevate in quel momento dalla sessione AR. Tale lista dev'essere aggiornata ogni volta che un nuovo piano viene rilevato oppure ogni volta che un piano viene rimosso e fuso con uno adiacente Anche in questo caso la struttura dati sarà del tutto trasparente al programmatore che non dovrà gestirla direttamente, ma potrà accedervi per ottenere i `GameObject` relativi ai piani tracciati.

In `ARLibrary` sono presenti anche altre due strutture dati minori elencate nelle seguenti porzioni di codice: un `array` di `Material` che contiene l'insieme dei materiali settati sulle superfici degli oggetti e una variabile di tipo `Object` che contiene il database per l'*image tracking*.

```
Material[] SurfaceMaterials { get; set; }
Object ImageTrackingDatabase { get; set; }
```

Sono presenti inoltre una serie di variabili booleane settabili al fine di attivare o disattivare determinate funzioni; di seguito l'elenco tratto dalla sorgente di `ARLibrary`.

```
1 bool EnableSurfaceCollider { get; set; }
2 bool EnableSurfaceMaterial { get; set; }
3 bool EnableImageTracking { get; set; }
4 bool EnableLightEstimation { get; set; }
```

Tali variabili permettono, nell'ordine in cui sono elencate, di:

- attivare o disattivare il `MeshCollider` sulle superfici rilevate dalla sessione AR;
- attivare o disattivare il *render* dei piani rilevati, cioè mostrare o no i materiali sulle supercifi;

- attivare o disattivare la funzione di *image tracking* all'interno della sessione;

Questi parametri possono anche essere modificati *real-time* e risultano molto utili per una personalizzazione estesa dello scenario di realtà aumentata. In particolare la possibilità di attivare o disattivare i *collider* sulle superfici rilevate permette, per esempio, l'implementazione della fisica in realtà aumentata, mentre la possibilità di attivare i materiali di tali piani da la possibilità di personalizzare a piacere la resa visiva.

Per quanto riguarda invece le funzioni messe a disposizione da questa libreria, le principali che è opportuno menzionare sono le funzioni che riguardano la gestione delle ancore in realtà aumentata elencate di seguito, con alcuni dettagli per ciascuna di queste.

```
1    bool CreateAnchorFromCamera(string anchorId, float xPos, float
     yPos, out Pose hitPose, float maxDistance = Mathf.Infinity);
2
```

Crea un'ancora effettuando un *raycast* a partire da un punto definito sullo schermo. Quando il raggio interseca un oggetto di tipo `trackable` viene generata su di esso un'ancora di tipo `trackable` e vengono aggiornate opportunamente tutte le strutture dati

```
1    bool CreateAnchorFromRay(string anchorId, Ray ray, out Pose
     hitPose, float maxDistance);
2
```

Come la funzione precedente, ma invece di passere le coordinate di un punto sullo schermo, accetta come parametro direttamente un raggio. Questa funzione permette maggiore flessibilità nella creazione delle ancore in quanto non vincola esclusivamente alle coordinate dello schermo, ma permette di utilizzare come raggio per il calcolo della `hit` un qualsiasi raggio.

```
1    bool CreateAnchorFromPose(string anchorId, Pose referencePose,
     out Pose hitPose, float maxDistance);
2
```

Anche questa funzione permette la generazione di un ancora, ma a differenza delle precedenti accetta come parametro una qualsiasi posa. Questa è la funzione che permette massima flessibilità nella creazione delle ancore, basta definire una posizione ed un orientamento nello spazio virtuale e passare tali valori sotto forma di posa a questa funzione. A partire da questo punto nello spazio verranno effettuati due *raycast* nella direzione *up* e nella direzione *up*

negativa rispetto alla posa; il primo punto di `hit` di tali raggi su un *trackable* verrà utilizzato come punto di creazione dell'ancora.

```
1     bool CreateSessionAnchor(string anchorId, Pose pose);
2
```

Funzione base per la creazione di un'ancora di sessione, cioè un ancora slegata da un qualsiasi *trackable*, ma calcolata in base all'ambiente inquadrato.

```
1     bool DestroyAnchor(string anchorId);
2
```

Funzione che distrugge un'ancora specifica definita dal suo identificativo `anchorId`.

```
1     bool DestroyAnchor(string anchorId, out List<GameObject>
    gameObjects);
2
```

Funzione analoga alla precedente, restituisce nel parametro `List<GameObject> gameObjects` l'elenco di tutti i `gameObjects` che erano stati associati all'ancora che si vuole rimuovere. Questo permette di non perdere il controllo sugli oggetti virtuali associati ad una determinata ancora distrutta e di poter per esempio associare tali oggetti ad una nuova ancora o di gestirli in altro modo.

```
1     bool DestroyAnchorAndGameObjects(string anchorId);
2     bool AttachGameObjectToAnchor(string anchorId, GameObject
    gameObject, bool worldPositionStays);
3     bool DetachGameObjectFromAnchor(string anchorId, GameObject
    gameObject);
4     bool DetachAllGameObjectsFromAnchor(string anchorId, out List<
    GameObject> gameObjects);
5     bool GetAnchorPose(string anchorId, out Pose anchorPose);
6     void ResetSession();
```

Le precedenti funzioni permettono rispettivamente di:

- Distruggere l'ancora e tutti i `gameObjects` associati, ritornando l'identificativo dell'ancora.
- Assegnare uno specifico `gameObject` ad una specifica ancora
- Rimuovere ad un ancora uno specifico `gameObject` precedentemente assegnatole
- Rimuovere tutti i `gameObjects` precedente assegnati ad una specifica ancora.

- Ottenere informazioni riguardo la posa di una specifica ancora, quindi ottenere la sua posizione e orientamento all'interno della simulazione

- Resettare la sessione AR rimuovendo tutte le ancore e tutti gli aggetti associati ad esse, ripristinando anche ogni struttura dati relativa alla corrente simulazione.

5.3.3 Implementazione dell'interfaccia con ARCore

Dopo aver definito con precisione tutte le funzioni che sarebbero servite per la creazione e il mantenimento di una sessione in realtà aumentata, si è proceduto con l'implementazione dell'interfaccia relativa a Google ARCore. In particolare sono degni di menzione:

- l'implementazione di un sistema di image tracking ibrido che oltre il *tracking* standard di ARCore per le immagini, sfrutta il rilevamento piani per definire un ancora del marker più stabile;

- lo sviluppo di un modulo per la gestione delle riflessioni in real-time, non disponendo le prime versioni di ARCore di tale funzionalità, poi eliminato;

- la gestione di alcune delle principali funzioni di ARCore tramite il sistema ad eventi definito dalla libreria generica descritta nel precedente paragrafo;

- la gestione e il popolamento di una serie di strutture dati tra cui dizionari e liste di piani rilevati ecc.

Verifica del funzionamento dell'implementazione Per poter verificare il corretto funzionamento di tale implementazione si è proceduto con lo sviluppo di una applicazione di prova che permettesse, con una apposita interfaccia utente, di testare ogni funzione presa in considerazione. In particolare effettuando un *tap* sulle varie superfici rilevate è possibile generare un oggetto virtuale in quel punto in due possibili modalità: generando un'ancora di sessione oppure generando una ancora di tipo *trackable* a cui associare l'oggetto. Come è possibile vedere dalla seguente figura 5.10, è possibile cambiare tale modalità run-time agendo su un apposito pannello dell'interfaccia. Inoltre è possibile cambiare run-time il materiale utilizzato per il rendering

delle superfici rilevate, attivare o disattivare la stima dell'intensità luminosa della scena, attivare o disattivare l'*image tracking*, cambiare run-time il database delle immagini per l'*image tracking*, rimuovere tutte le ancore, rimuovere gli oggetti assegnati alle varie ancore, resettare la sessione AR e definire manualmente la direzione delle fonti luminose.

5.4 Framework di networking

Per lo sviluppo del framework di networking, come si è già ampiamente anticipato, si è deciso di creare una interfaccia generica, chiamata `INetworkManager`, con le principali informazioni e funzioni necessarie per il funzionamento di tutta l'applicazione, rendendo l'applicazione indipendente dalla specifica libreria di networking scelta e permettendo quindi di cambiarla in futuro.

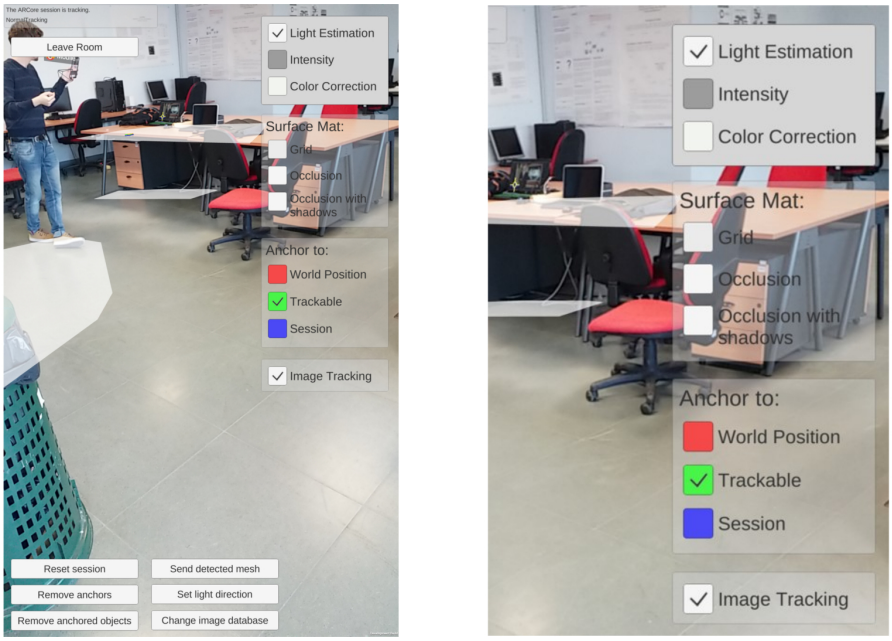


Figura 5.10. Catture schermo raffiguranti l'app di test, a destra il dettaglio dei pannelli con le varie impostazioni

5.4.1 Piattaforme di networking considerate e implementazione con Photon Unity Networking

Il panorama delle librerie di networking per Unity è molto variegato. Si è inizialmente ipotizzato di utilizzare il sistema nativo di sincronizzazione multiutente di Unity, cioè Unity Networking (UNET), ma lo stato di tale libreria al momento dello sviluppo del nostro modulo era di parziale deprecazione in vista di un futuro aggiornamento generale delle funzionalità multiplayer di Unity. Sì è proceduto quindi con l'esaminare alcune delle piattaforme multiplayer più utilizzate tra cui GameSparks e Photon Unity Networking (PUN); si è concluso che quest'ultimo proponeva la soluzione più flessibile e adatta ai nostri scopi.

PUN è sostanzialmente un *package* di Unity per il gioco multiplayer. Un *matchmaking* flessibile permette agli utenti di accedere a "stanze" dove gli oggetti possono essere sincronizzati attraverso la rete sfruttando alcune delle *features* di questa libreria tra cui: *Remote Procedure Calls*, *Custom Properties* oppure eventi Photon di "basso livello". Un'altra caratteristica interessante di tale piattaforma è che la comunicazione può essere opzionalmente *reliable* e viene eseguita attraverso server dedicati.

In conclusione si è deciso di adottare un'architettura di networking di tipo peer-to-peer attraverso la piattaforma Photon Unity Networking. Il sistema che si è sviluppato prevede la presenza di un *master-client* che simula in alcuni casi una un'architettura client-server flessibile. A questo master-client per esempio sono demandati alcuni calcoli od elaborazioni ed ha il compito di effettuare la pre-scansione dell'ambiente con relativo piazzamento degli elementi virtuali. In generale comunque l'applicazione risulta distribuita tra le sue varie istanze e chiunque partecipante alla simulazione può essere promosso a *master-client*.

Si è anche dovuto stabilire quale protocollo di trasmissione dei dati utilizzare e si è optato per l'adozione di un sistema flessibile in cui alcuni dati



Figura 5.11. Le principali piattaforme di networking prese in considerazione per l'implementazione dell'interfaccia multi-utente

vengono condivisi con un sistema "reliable", per esempio informazioni inviate saltuariamente riguardanti variazioni di alcuni parametri condivisi all'interno della simulazione, mentre altri con un sistema "unreliable", per esempio informazioni inviate in modo continuativo come l'aggiornamento delle posizioni degli utenti e degli oggetti virtuali dinamici che può continuare a funzionare anche con perdita di pacchetti.

Partendo da questa struttura è stato possibile creare e poi implementare l'interfaccia `INetworkManager`, che espone tutte le funzionalità necessarie per la nostra applicazione, di cui si elencano le principali:

- creare una stanza, unirsi o lasciarne una;
- informazioni riguardo alle stanze e agli utenti all'interno;
- generare nella stanza un oggetto virtuale visibile a tutti in una posizione specifica (sia esso dinamico o statico);
- afferrare un oggetto dinamico, manipolarlo e posarlo;
- interagire sugli oggetti tramite gesti su touchscreen;
- creazione e distruzione di punti di interesse condivisi da parte di un player;
- numerosi eventi a cui è possibile iscriversi per molte delle azioni che possono accadere relative al networking, come per esempio l'ingresso di un nuovo player nella stanza o l'afferrare di un oggetto da parte di un player nella stanza.

Di seguito una descrizione più dettagliata di come è stata implementata la sincronizzazione degli scenari di realtà aumentata, con particolare attenzione alle soluzioni adottate per la sincronizzazione dei partecipanti alla simulazione, degli oggetti virtuali e del cursore degli utenti.

5.4.2 Sincronizzazione dell'ambiente condiviso

La posa del marker all'interno della simulazione viene utilizzata come origine del sistema di riferimento comune a tutti i partecipanti della sessione AR. Per aumentare la stabilità e quindi l'affidabilità dell'*image tracking*, si è deciso di implementare un sistema che prevede agli utenti di sincronizzarsi con successo solo quando le seguenti due condizioni vengono soddisfatte: l'immagine deve essere rilevata correttamente e deve esser stato rilevato un piano sottostante

al marker. In questo modo è possibile promuovere l'ancora dell'immagine ad un'ancora di tipo *trackable*, decisamente più stabile. Una volta che gli utenti sono sincronizzati tra loro, il sistema sfrutta la posa dell'`imageAnchor` generata sul marker per la creazione di opportune matrici di trasformazione necessarie per passare dal sistema di riferimento del mondo (quindi l'origine della sessione AR) al sistema di riferimento del marker e viceversa. Tali matrici vengono poi applicate a tutte le posizioni inviate e ricevute tramite network; in particolare, quando un client manda in rete un dato di posizione, prima trasforma tale informazione dallo spazio del mondo a quello del marker, applicando una opportuna matrice 4x4, mentre quando riceve un dato di posizione dalla rete lo trasforma dallo spazio del marker a quello del mondo in modo da posizionarlo correttamente all'interno della simulazione. Questo sistema rende quindi possibile una consistenza robusta delle posizioni.

Per quanto riguarda il sistema di sincronizzazione degli oggetti virtuali, si sono dovuti anche superare alcuni ostacoli che avrebbero impedito consistenza all'interno della simulazione. Tra le varie soluzioni che sono state prese in considerazione durante la fase di design del sistema di sincronizzazione dello scenario, figuravano:



Figura 5.12. Catture schermo raffiguranti l'app durante la fase di sincronizzazione con il marker

- un sistema centralizzato in cui un singolo dispositivo effettua il rilevamento superfici e poi condivide la scena ricostruita con tutti i partecipanti alla simulazione;
- un sistema distribuito in cui ciascun utente ricostruisce la scena localmente.

Ciascuna di queste soluzioni presenta aspetti positivi e negativi. La prima soluzione permette intrinsecamente consistenza degli oggetti virtuali all'interno della simulazione, ma impedisce di utilizzare il sottosistema di rilevamento delle superfici su più di un dispositivo. La seconda soluzione invece non assicura la consistenza dello scenario, ma permette di sfruttare appieno il sistema di scansione dell'ambiente reale. Sì è optato per la seconda soluzione in modo da sfruttare completamente tutte le funzionalità messe a disposizione dalla libreria di AR, con qualche accorgimento usato per migliorare la consistenza spiegato di seguito.

La fase iniziale di pre-scansione compiuta dal *master-client* prevede che egli disponga gli oggetti virtuali all'interno della simulazione prima del caricamento effettivo della scena. Poiché in tale fase vengono sfruttati solo i piani rilevati dal *master-client* per il posizionamento degli oggetti, è altamente probabile che la loro posizione comunicata in rete non corrisponda a nessun piano rilevato dagli altri utenti. Per ovviare a questo problema si sono adottate alcune soluzioni che variano leggermente in base alla tipologia di oggetto virtuale da sincronizzare.

Se un client riceve la posizione sincronizzata di oggetto statico per prima cosa effettua un controllo per verificare se nei dintorni della posa ricevuta sono stati rilevati o no dei piani locali. Questa operazione viene effettuata eseguendo due `raycast` nelle direzioni `up` e `-up` della posa. Se viene rilevato un piano, tale oggetto virtuale statico viene ancorato e quindi traslato sulla superficie, permettendo quindi di "appoggiarsi" meglio sul piano rilevato localmente, altrimenti viene creata un'ancora di sessione. Dato che nel corso della simulazione il client può rilevare un nuovo piano o aggiornare quelli esistenti, periodicamente vengono effettuati dei controlli sulle ancore di sessione eseguendo nuovamente i due `raycast` e riposizionando l'oggetto nel caso in cui tali `raycast` intersechino una superficie.

Per quanto riguarda invece gli oggetti virtuali dinamici e i cursori degli utenti viene adottata una soluzione simile a quelli statici sfruttando però il concetto di copia *dummy*. In pratica la posa dell'oggetto sincronizzata tramite rete rimane sempre invariata e non viene mai traslata effettivamente per appoggiarsi su un piano locale, come invece avviene per gli oggetti

statici. Ad essere traslato ed ancorato ai piani locali vicini è invece il *dummy* dell'oggetto, cioè la sua rappresentazione 3D. Questo perché gli oggetti statici, una volta stabilita la posizione all'interno della simulazione, non vengono più sincronizzati mentre per quanto riguarda gli oggetti dinamici e i cursori, potendo essere manipolati e posizionati *run-time* dagli utenti ovunque nella scena e in qualunque momento, la loro posizione e rotazione sono continuamente sincronizzate.

In generale l'implementazione di una libreria di networking che funziona anche non il LAN ci ha permesso di estendere le funzionalità dell'applicazione anche ad uno scenario multi-utente remoto, cioè un ambiente in cui uno o più utenti non sono presenti fisicamente. Basta che ciascuno dei partecipante abbia a disposizione un *marker* e che i vari ambienti reali siano abbastanza simili, quantomeno nella disponibilità di superfici piane. Per consentire comunque la comunicazione verbale nell'eventualità di uno scenario remoto è stato anche implementato un sistema di chat vocale.

5.5 Scenari sperimentali

Dopo aver definito, implementato e testato i due framework di realtà aumentata e networking, si è proceduto con il design e l'implementazione di un'applicazione adatta ad un utilizzo all'interno di un protocollo sperimentale. Per poter far ciò è stato necessario definire tre scenari collaborativi distinti, rappresentante ciascuno una delle tre interazioni definite nei capitoli precedenti, ovvero basate su sguardo, manipolazione di oggetti e sulla posizione degli utenti. In particolare ogni scenario è stato progettato affinché esemplificasse i possibili casi d'uso di ciascuna interazione, presentando una serie di task (due o tre) da portare a termine in maniera collaborativa.

In seguito, come caso d'uso di questi studi sulla realtà aumentata collaborativa, è stato sviluppato un estratto della *escape room* virtuale definita nel capitolo precedente. Come già anticipato, sono stati selezionati quattro degli enigmi più significativi dal punto di visto della collaborazione multi-utente; tali enigmi sono stati implementati in quattro scene distinte, anch'esse somministrate, ma in modo opzionale, al completamento degli scenari sperimentali. Tutti gli scenari sono stati progettati pensando a gruppi formati da tre individui e oltre agli scenari citati è stato necessario anche implementare tre ulteriori piccole scene per permettere agli utenti di effettuare del *training* prima di partire con l'esecuzione effettiva dei *task*.

Si rimanda al capitolo successivo per i dettagli riguardo il protocollo sperimentale utilizzato; di seguito una breve descrizione degli scenari.

5.5.1 Interazione collaborativa basata sullo sguardo

Come prima cosa è stato implementato un brevissimo scenario di *training* per abituare l'utente all'interazione con il mondo virtuale attraverso lo sguardo / cursore. Tale scenario consiste in 4 cubi bianchi interattivi posizionati su una superficie, quando un utente posa il proprio cursore per alcuni istanti su uno qualsiasi di questi cubi, questi cambiano colore ed emettono un suono.

Per quanto riguarda invece i compiti veri e propri relativi a questo tipo di interazione, sono stati definiti tre differenti *task*, ciascuno esemplificativo di uno specifico caso d'uso.

Task 1 Il primo *task*, di cui è possibile osservare delle catture in figura 5.13, consiste nel seguente scenario: una matrice di sedici cubi bianchi appare su una superficie rilevata e uno di essi preso casualmente viene evidenziato rosso; il compito che i partecipanti alla simulazione devono portare a termine consiste nell'osservare contemporaneamente (quindi posare sopra il cursore) il cubo rosso. Una volta che tutti i partecipanti avranno osservato tale cubo per qualche istante, esso tornerà ad essere bianco emettendo un suono e un altro cubo, scelto casualmente dalla matrice, diventerà rosso. Gli utenti quindi dovranno osservare nuovamente il cubo rosso finché anche lui non tornerà bianco e un altro cubo casuale sarà diventato rosso. A questo punto gli utenti osserveranno per un ultima volta il cubo rosso portando a termine il primo compito relativo all'interazione collaborativa basata sullo sguardo.

Tale scenario esemplifica il caso d'uso in cui è richiesto agli utenti di osservare contemporaneamente punti specifici della simulazione, tale azione è ripetuta più volte per aumentare leggermente la complessità e il livello di sfida, nonché fornire più dati riguardo tale tipo di interazione. Inoltre si è definito tale *task* come di tipo *system commanded*, cioè un *task* in cui è il sistema a definire quale azione devono compiere gli utenti (in questo caso la scelta di un cubo casuale tra i sedici disponibili).

Task 2 Il secondo *task*, di cui è possibile osservare alcune catture in figura 5.14, consiste in uno scenario molto simile al precedente. Anche in questo caso abbiamo una matrice di sedici cubi posta su una superficie rilevata, ma, a differenza del *task* precedente, sono presenti tre cubi colorati, uno di verde,

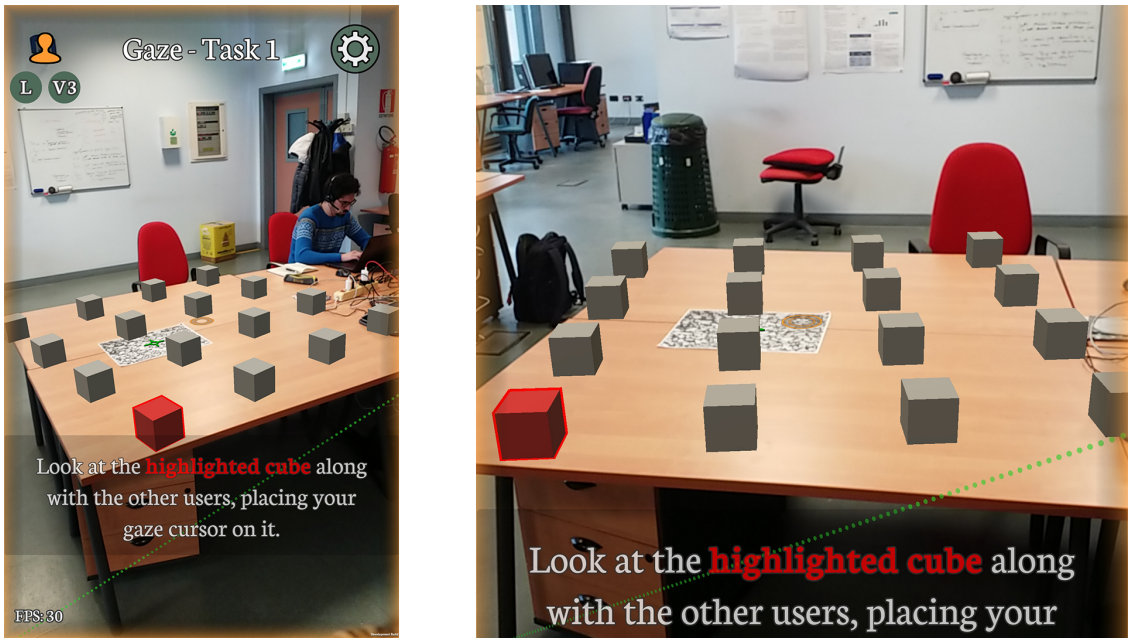


Figura 5.13. Catture schermo raffiguranti l'app durante l'esecuzione del primo task dello sguardo

uno di arancio e uno di ciano (gli stessi colori associati agli utenti partecipanti). Il compito quindi che gli utenti devono portare a termine consiste nell'osservare contemporaneamente, posandoci sopra il cursore, ciascuno il cubo del proprio colore. Una volta che tutti i partecipanti avranno osservato i rispettivi cubi colorati per qualche istante, essi torneranno ad essere bianchi emettendo un suono e altri tre cubi, scelti casualmente dalla matrice, diventeranno colorati. Quindi, in modo del tutto analogo al precedente *task*, gli utenti dovranno ripetere altre due volte questa operazione finché il task non sarà completato.

Tale scenario esemplifica il caso d'uso in cui è richiesto agli utenti di osservare contemporaneamente punti diversi all'interno della simulazione, come per il task precedente tale operazione viene eseguita più volte per aumentare complessità e livello di sfida oltre che dar modo agli utenti di aver più tempo per sfruttare appieno l'interfaccia messa a disposizione. Anche in questo caso il *task* è di tipo *system commanded*, ovvero è il sistema che sceglie casualmente tre cubi dalla matrice di sedici.

Task 3 Il terzo *task* dell'interazione collaborativa basata sullo sguardo, di cui è possibile osservare alcune catture in figura 5.15, consiste anch'esso in

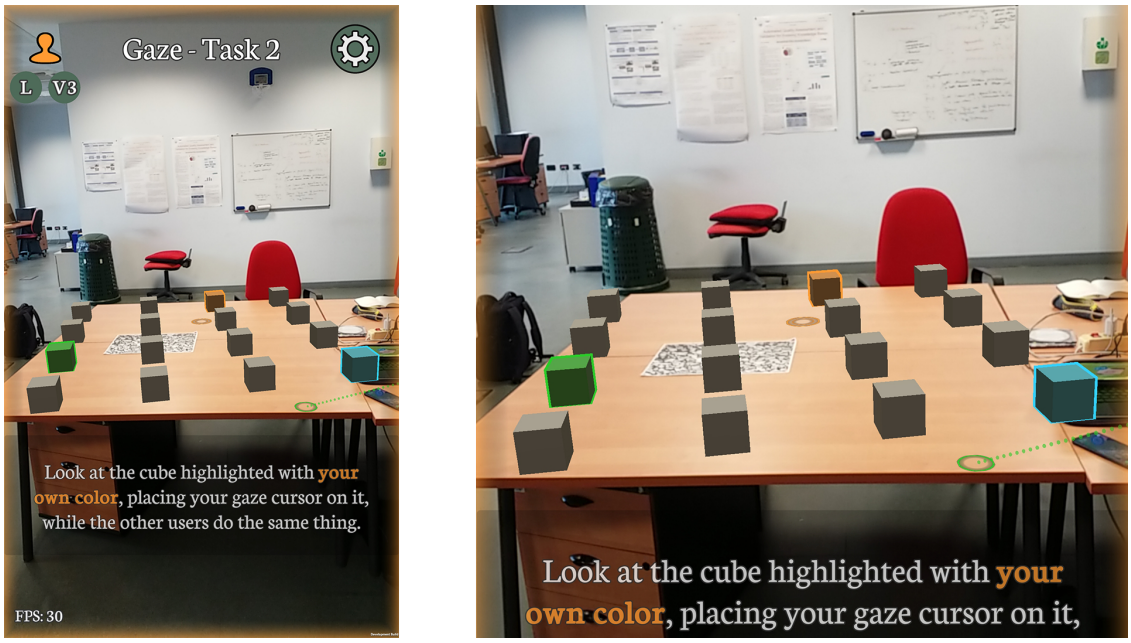


Figura 5.14. Catture schermo raffiguranti l'app durante l'esecuzione del secondo task dello sguardo

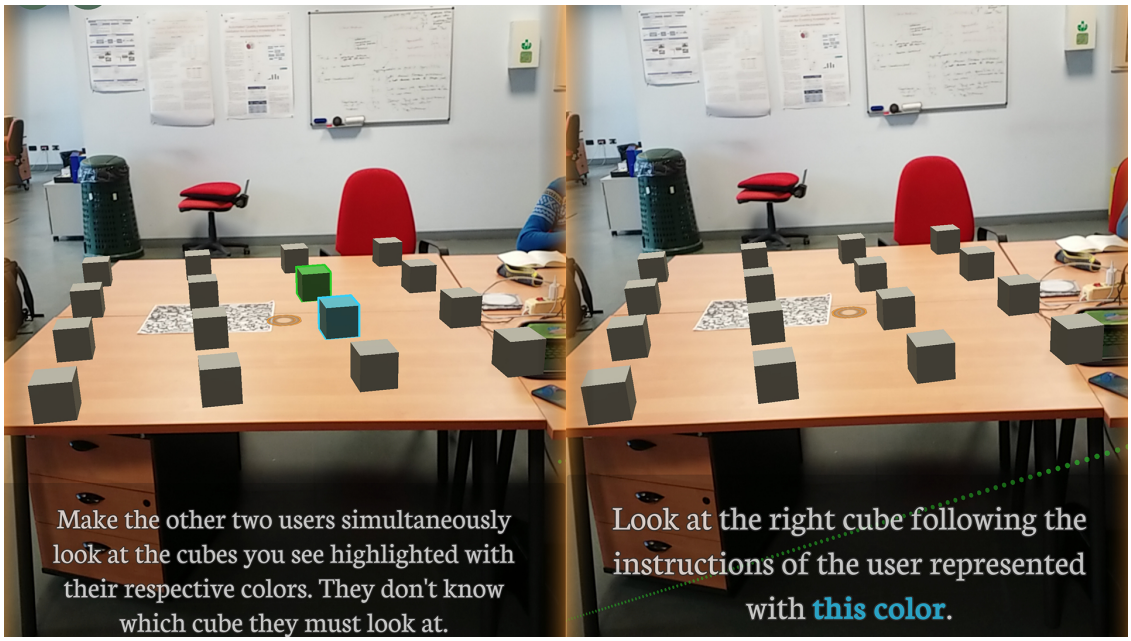


Figura 5.15. Catture schermo raffiguranti l'app durante l'esecuzione del terzo task dello sguardo, a sinistra l'utente che ha il comando

uno scenario molto simile ai precedenti. Anche in questo caso è presente una matrice di sedici cubi posizionati sopra un piano rilevato, ma, a differenza dei precedenti *task* dove tutti gli utenti partecipanti disponevano delle stesse informazioni (posizione nella matrice del cubo rosso o dei cubi colorati), ora solo un partecipante dispone di tali dati. Descrivendo più nel dettaglio questo *task*, un solo utente vedrà i cubi colorati dei colori degli altri utenti (identificheremo questo utente come colui che attualmente ha il comando sulla simulazione), mentre gli altri partecipanti vedranno solamente una matrice di cubi bianchi. Il compito da portare a termine è far si che i due utenti che non dispongono di nessuna informazione visuale posino contemporaneamente lo sguardo sui cubi del proprio colore; l'utente che ha il comando deve quindi fornire agli altri utenti le informazioni necessarie affinché possano portare a termine il compito, sfruttando come meglio crede gli elementi d'interfaccia messi a disposizione. Questo *task* viene eseguito due volte in modo tale che due dei tre gli utenti assumano il comando della simulazione. Viene eseguito solo due volte invece che tre per motivi che verranno chiariti quando si affronterà più nel dettaglio la fase di sperimentazione.

Tale scenario, come quello precedente, esemplifica il caso d'uso in cui è richiesto agli utenti di osservare contemporaneamente punti diversi all'interno della simulazione. La differenza sostanziale con il task precedente sta nel fatto che ora non è più *system commanded*, ma *user commanded*.

5.5.2 Interazione collaborativa basata sullo manipolazione

Come prima cosa è stato implementato uno scenario di *training* per permettere agli utenti di familiarizzare con l'interfaccia e con i metodi di interazione con gli oggetti virtuali dinamici. Tale scenario presenta sei elementi virtuali, tutti appoggiati su un piano rilevato: tre oggetti dinamici interattivi e tre oggetti virtuali statici. Gli oggetti virtuali statici non sono altro che dei "fori virtuali" che danno l'impressione di essere ricavati da un intaglio sulla superficie e hanno un numero casuale stampato sul fondo, mentre gli oggetti dinamici sono delle forme geometriche che corrispondono ai fori. Questi ultimi oggetti hanno ulteriori caratteristiche tra cui: un numero stampato sulla superficie e un elemento interattivo rappresentato da una rotella rossa a lato. Interagendo con la rotella rossa è possibile scorrere i numeri stampati sopra l'oggetto. Essendo questi oggetti virtuali di tipo dinamico, possono essere raccolti, manipolati e spostati all'interno della simulazione AR. In particolare il training permette agli utenti di "incastrare" le forme geometriche

all'interno dei fori corrispondenti. L'incastro avviene solo se sono rispettati quattro parametri: l'oggetto corrisponde alla forma del foro nel quale lo si vuole incastrare, il numero stampato sull'oggetto corrisponde al numero stampato sul fondo del foro virtuale, l'orientamento dell'oggetto corrisponde all'orientamento del foro e infine la posizione dell'oggetto corrisponde alla posizione del foro. Per agevolare questa procedura si è deciso di implementare una soglia di tolleranza ragionevole per quando riguarda la posizione e l'orientamento.

Per quanto riguarda invece i compiti veri e propri relativi a questo tipo di interazione, sono stati definiti due differenti *task*, ciascuno esemplificativo di uno specifico caso d'uso.

Task 1 Il primo *task* di tale interazione collaborativa basata sulla manipolazione, di cui è possibile osservare alcune catture in figura 5.16, consiste in uno scenario del tutto identico a quello definito per il *training*: anche in questo caso sono presenti sei oggetti virtuali di cui tre oggetti dinamici da incastrare nei rispettivi tre fori statici. L'unica differenza con lo scenario di *training* è che ora gli oggetti virtuali dinamici sono colorati con i tre colori corrispondenti agli utenti. Il compito quindi che i partecipanti devono portare a termine consiste nell'incastrare solo l'oggetto che gli appartiene identificandolo attraverso il codice colori.

Tale scenario esemplifica il caso d'uso in cui è richiesto agli utenti di manipolare contemporaneamente oggetti virtuali diversi all'interno della simulazione. In questo caso il *task* è di tipo *system commanded*.

Task 2 Il secondo *task* di questa tipologia di interazione, di cui è possibile osservare alcune catture in figura 5.15, si presenta concettualmente simile al *task* precedente, cioè anche in questo caso gli utenti dovranno incastrare degli oggetti virtuali all'interno di appositi fori. La differenza sostanziale con il task precedente sta nel fatto che ora solo un utente all'interno della simulazione è in grado di vedere i fori virtuali nello scenario, mentre gli altri utenti non vedranno nulla. Andando più nel dettaglio, l'utente che ha il comando, cioè colui che ha informazioni in più rispetto agli altri, vedrà comparire su una superficie una griglia popolata da sedici fori virtuali di forma triangolare, orientati in modo casuale. Inoltre due di questi fori sono colorati del colore degli altri due utenti e presentano un numero stampato sul fondo. Gli altri utenti vedranno la stessa griglia, ma priva di fori virtuali e per portare a termine il compito (cioè incastrare correttamente i due oggetti virtuali

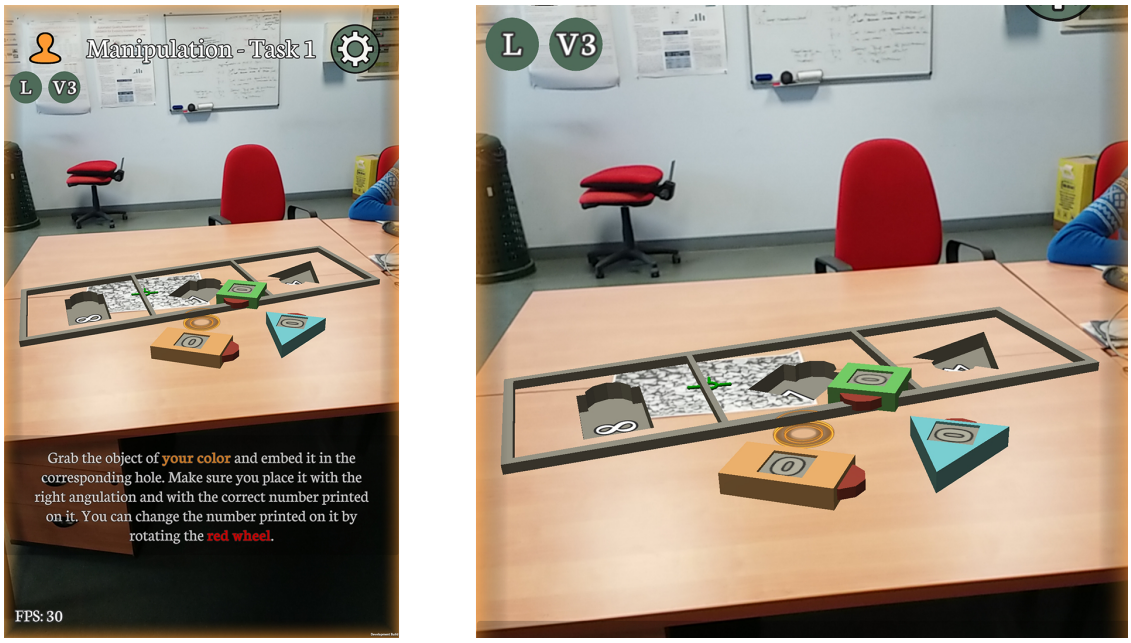


Figura 5.16. Catture schermo raffiguranti l'app durante l'esecuzione del primo task della manipolazione

nei corrispondenti fori) dovranno seguire le direttive dell'utente in comando. Tali istruzioni dovranno comprendere quindi tre informazioni: la posizione dell'oggetto virtuale all'interno della griglia, l'orientamento di tale oggetto (che potrà assumere per semplicità solamente otto orientamenti) e il numero da impostare. Analogamente al terzo *task* dell'interazione collaborativa basata sullo sguardo, anche questo *task* viene eseguito due volte in modo tale che a turno due dei tre utenti assumano il comando della simulazione. Viene eseguito solo due volte invece che tre per motivi che verranno chiariti quando si affronterà più nel dettaglio la fase di sperimentazione.

Tale scenario, come quello precedente, esemplifica il caso d'uso in cui è richiesto agli utenti di manipolare contemporaneamente oggetti virtuali diversi all'interno della simulazione. La differenza sostanziale con il task precedente sta nel fatto che ora l'attività non è più *system commanded*, ma *user commanded*.

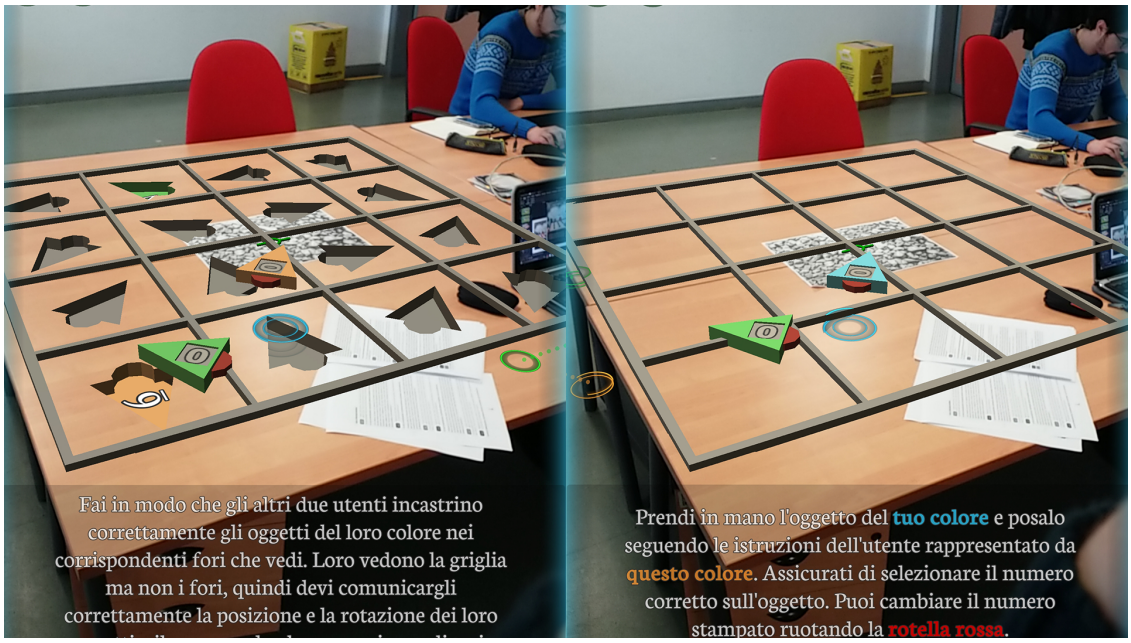


Figura 5.17. Catture schermo raffiguranti l'app durante l'esecuzione del secondo task della manipolazione, a sinistra l'utente che ha il comando. Come è possibile notare, solo l'utente nell'immagine di sinistra può vedere i fori virtuali all'interno della griglia

5.5.3 Interazione collaborativa basata sulla posizione

Analogamente alle altre tipologie di interazione collaborativa, anche per quella relativa alla posizione si è deciso di implementare un brevissimo scenario di *training* per illustrarne agli utenti il funzionamento. Questo scenario prevede la disposizione su una superficie di quattro oggetti virtuali statici raffiguranti delle piattaforme blu. Per dimostrare agli utenti che la posizione dei dispositivi viene tenuta in considerazione all'interno della simulazione e che può essere utilizzata come forma di interazione, quando un utente qualsiasi posiziona il proprio device in linea d'aria sopra una delle quattro piattaforme, queste si attivano mostrando sopra di esse una cilindro blu semi-trasparente

Per quanto riguarda invece i compiti veri e propri relativi a questo tipo di interazione, sono stati definiti tre differenti *task*, ciascuno esemplificativo di uno specifico caso d'uso.

Task 1 Il primo *task*, di cui è possibile osservare delle catture in figura 5.18, consiste nel seguente scenario: otto oggetti virtuali statici rappresentanti delle piattaforme grige vengono posizionate su una superficie rilevata,

in particolare sul pavimento; una di queste piattaforme è rossa e presenta sopra di essa un cilindro semi-trasparente del medesimo colore. Il compito che gli utenti dovranno portare a termine è quello di posizionarsi contemporaneamente sopra la piattaforma rossa. Una volta che tutti gli utenti avranno posizionato per qualche istante il proprio dispositivo sopra tale piattaforma, questa tornerà ad essere grigia e un'altra piattaforma, scelta casualmente dal sistema tra le otto disponibili, diventerà rossa inducendo i partecipanti a raggiungere la nuova piattaforma per progredire nel completamento del *task*. Questa procedura verrà eseguita un'ultima volta quando si attiverà la terza piattaforma rossa: appena tutti gli utenti la raggiungeranno il *task* sarà completato.

Tale scenario esemplifica il caso d'uso in cui è richiesto agli utenti di dirigersi contemporaneamente in un punto specifico della simulazione. Come nelle altre tipologie di interazione, tale azione è ripetuta più volte per aumentare leggermente la complessità e il livello di sfida, nonché fornire più dati riguardo tale tipologia di interazione e dare modo agli utenti di sfruttare meglio l'interfaccia utente. Tale *task* è di tipo *system commanded*, cioè è il sistema nel corso della simulazione a scegliere casualmente quale piattaforma attivare.



Figura 5.18. Catture schermo raffiguranti l'app durante l'esecuzione del primo task della posizione

Task 2 Il secondo *task* relativo all'interazione collaborativa basata sulla posizione, di cui è possibile osservare delle catture in figura 5.19, consiste in uno scenario molto simile al precedente. Anche in questo caso sono presenti otto oggetti virtuali posti sul pavimento raffiguranti altrettante piattaforme. A differenza del *task* precedente, ma analogamente al secondo *task* relativo allo sguardo, al posto di una singola piattaforma evidenziata di rosso ora sono evidenziate tre piattaforme dei colori associati ai vari utenti. Il compito che i partecipanti alla simulazione devono portare a termine consiste nel posizionarsi contemporaneamente ognuno sulla piattaforma del proprio colore. Una volta che tutti e tre gli utenti si saranno posizionati sulle rispettive piattaforme per alcuni istanti, queste torneranno ad essere grigie emettendo un suono e altre tre piattaforme, scelte casualmente dal sistema, verranno colorate di verde, arancio e ciano. Quindi, in modo del tutto analogo al precedente *task*, gli utenti dovranno ripetere altre due volte questa operazione finché il task non sarà infine completato.

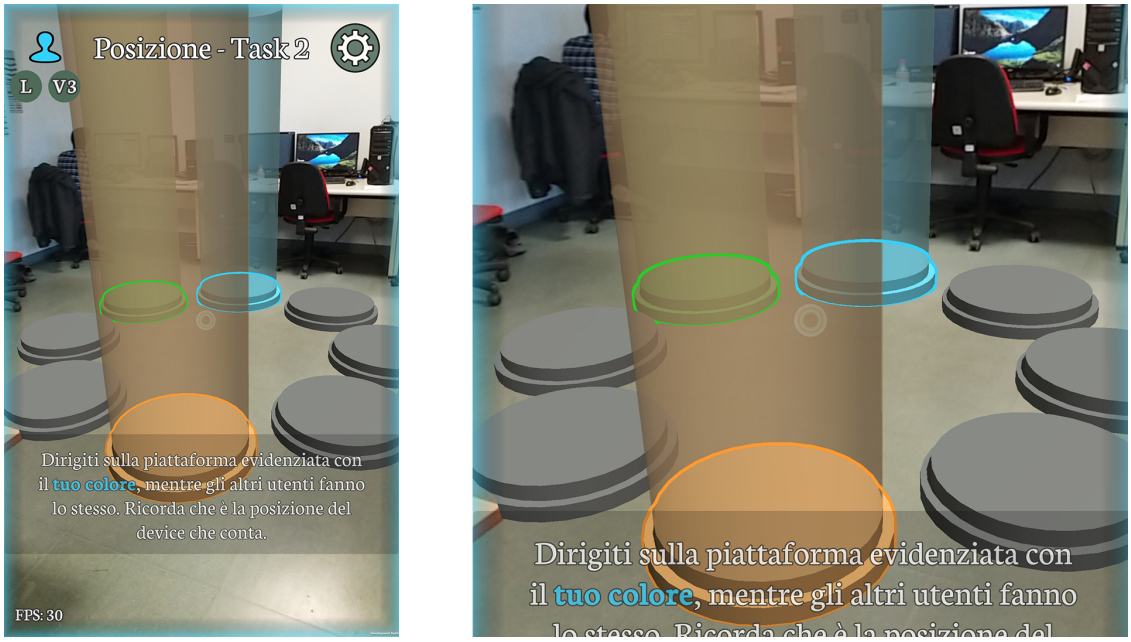


Figura 5.19. Catture schermo raffiguranti l'app durante l'esecuzione del secondo task della posizione

Tale scenario esemplifica il caso d'uso in cui è richiesto agli utenti di posizionarsi contemporaneamente in punti diversi all'interno della simulazione; come per il task precedente tale operazione viene eseguita più volte per aumentare complessità e livello di sfida oltre che dar modo agli utenti di aver

più tempo per sfruttare appieno l'interfaccia messa a disposizione. Anche in questo caso il *task* è di tipo *system commanded*, ovvero è il sistema che sceglie casualmente tre piattaforme dalle otto disponibili.

Task 3 Il terzo *task* relativo alla posizone, di cui è possibile osservare delle catture in figura 5.20, consiste in uno scenario simile ai precedenti *task*. Anche in questo caso sono presenti otto piattaforme posizionate sopra un piano rilevato sul pavimento, ma, a differenza dei precedenti *task* dove tutti gli utenti partecipanti disponevano delle stesse informazioni (posizione della piattaforma rossa o di quelle colorate), ora solo un partecipante dispone di tali dati. Descrivendo più nel dettaglio questo *task*, un solo utente vedrà le piattaforme colorate dei colori degli altri utenti, mentre gli altri partecipanti vedranno solamente piattaforme grige. Il compito da portare a termine è far si che i due utenti che non dispongono di nessuna informazione visuale si posizionino contemporaneamente sulle piattaforme del proprio colore; l'utente che ha il comando deve quindi fornire agli altri utenti le informazioni necessarie affinché possano portare a termine il compito, sfruttando come meglio crede gli elementi d'interfaccia messi a disposizione. Questo *task* viene eseguito due volte in modo tale che a turno tutti gli utenti assumano il comando della simulazione.

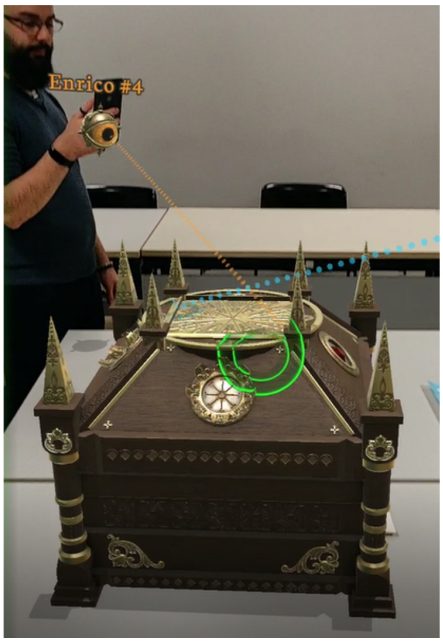


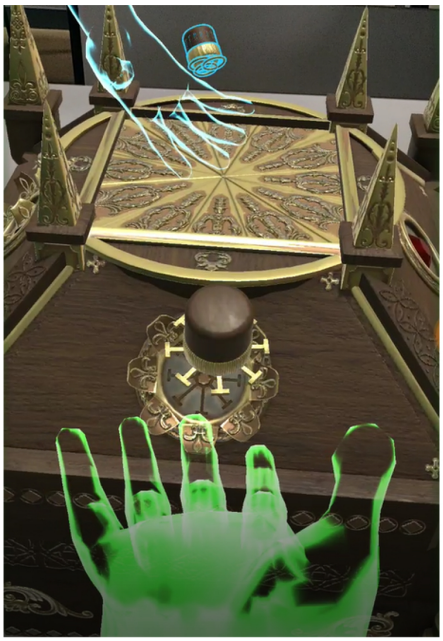
Figura 5.20. Catture schermo raffiguranti l'app durante l'esecuzione del terzo task della posizione

Tale scenario, come quello precedente, esemplifica il caso d'uso in cui è richiesto agli utenti di posizionarsi contemporaneamente in punti diversi all'interno della simulazione. La differenza sostanziale con il task precedente sta nel fatto che ora non è più *system commanded*, ma *user commanded*.

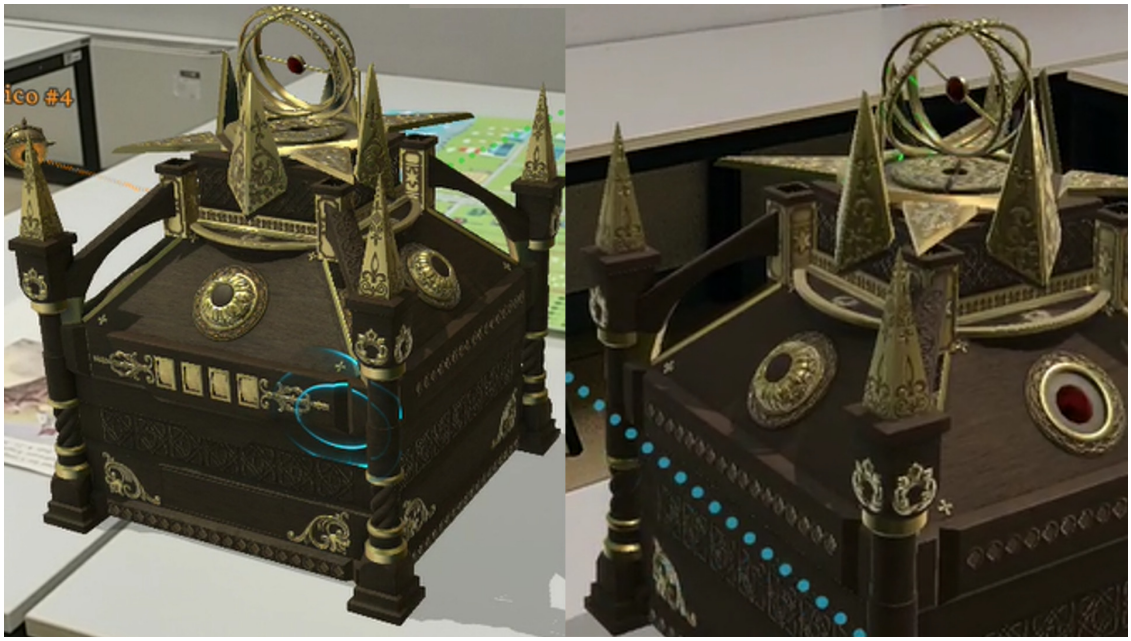
5.5.4 Scenari di tipo escape room

Scenario 1: Tutorial Lo scenario virtuale è composto unicamente dalla scatola, che presenta sulla parte superiore tre fori rotondi e in un quella inferiore decorazioni varie, tra cui alcuni strani cilindri agli angoli. Gli utenti sono per prima cosa tenuti ad osservare un punto preciso della scatola (sommità) per poter attivare tutte le interazioni (**interazione collaborativa basata sullo sguardo**). Una volta che tutti avranno osservato il punto indicato tramite un punto di interesse condiviso, gli utenti potranno interagire con i cilindri alla base che diventeranno afferrabili. Questi cilindri potranno essere manipolati e tramite interazione con una ghiera di metallo su di essi potranno essere trasformati in chiavi inseribili all'interno dei fori della forma corretta sulla parte superiore della scatola. Una volta inseriti tali cilindri all'interno degli appositi alloggiamenti, essi si trasformeranno in pulsanti interattivi. Quando tutti e tre i pulsanti risulteranno premuti contemporaneamente per un certo periodo di tempo si avvierà una animazione che aprirà la sommità



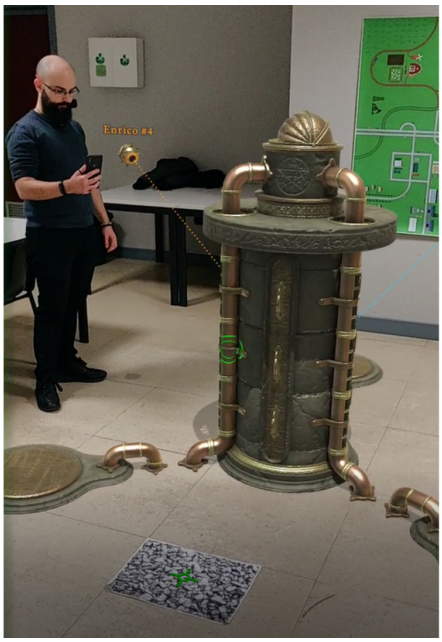


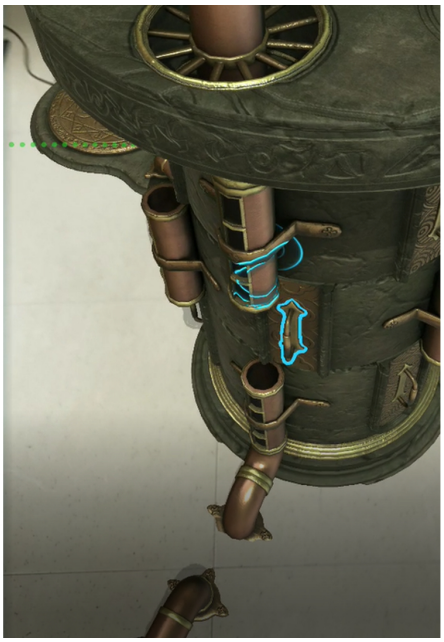
della scatola (**interazione collaborativa basata sulla manipolazione di oggetti virtuali**).



Scenario 2: Piedistallo Questo scenario è composto da quattro elementi: un piedistallo e tre pedane di metallo. Il piedistallo è di pietra ed è piuttosto imponente alto circa 1,2 m e piatto in cima. La superficie laterale del piedistallo presenta tre sezioni con alcuni intarsi particolari che appaiono disallineati e delle maniglie interattive. Questa struttura presenta inoltre tre tubazioni di rame che la percorrono e si immettono nel terreno, riaffiorando in prossimità delle piastre metalliche e collegandosi ad esse. Gli utenti dovranno accorgersi che le sezioni del basamento del piedistallo non sono allineate e procedere con il loro riallineamento, afferrando una maniglia e spostando fisicamente il dispositivo per ruotarle. Tali operazioni devono essere effettuate contemporaneamente in quanto, quando un blocco viene rilasciato, esso ritorna da solo nella posizione di partenza (**interazione collaborativa basata sulla manipolazione di oggetti virtuali e sulla posizione**). Una volta effettuato il riallineamento, una animazione indicherà che l'operazione ha avuto successo e le tre pedane si illumineranno ciascuna di un colore associato agli utenti. Posizionandosi su queste pedane, una barra di completamento si riempirà sul tubo associato fissato sul piedistallo, mentre

spostandosi al di fuori della pedana tale barra si svuoterà. Solo quando tutti gli utenti si saranno posizionati sulle rispettive pedane e avranno atteso il completamento della barra, un'animazione indicherà che l'operazione ha avuto successo, terminando lo scenario (**interazione collaborativa basata sul posizionamento degli utenti in luoghi diversi**).





Scenario 3: Dipinto Gli oggetti virtuali presenti in questo scenario saranno:

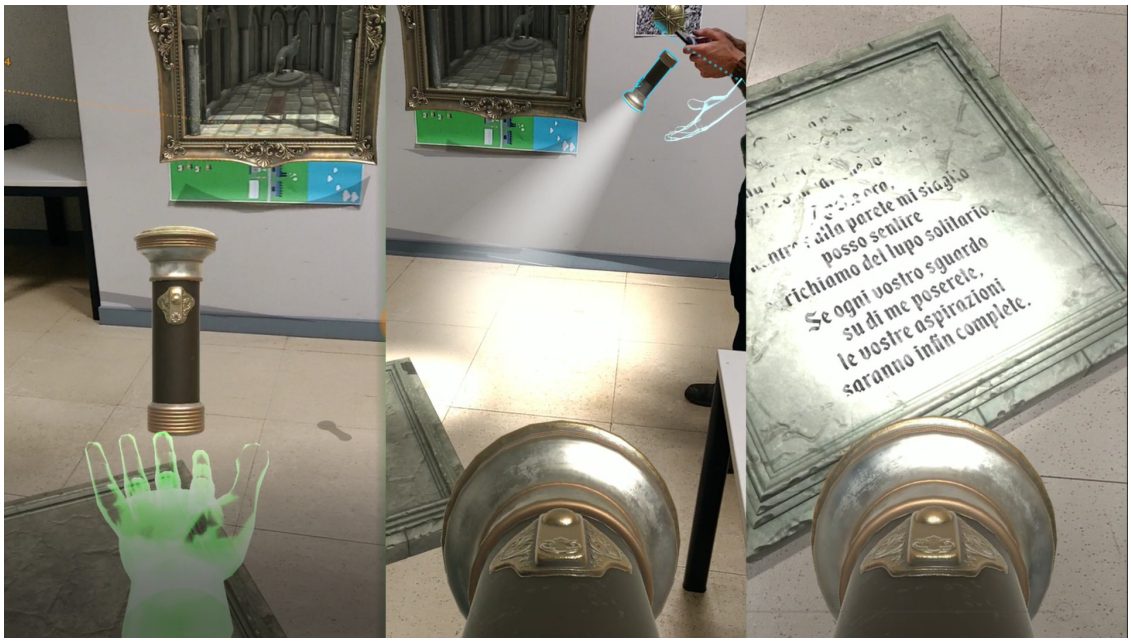
- quello che sembra quadro antico appeso a un muro, raffigurante l'interno di una torre gotica, con la particolarità però che l'ambiente "sprofondi" nel muro e che quindi sia tridimensionale; sono visibili alcuni dettagli particolari nella scena raffigurata come la statua di un lupo ululante, una raffigurazione di un dio abissale con occhi luminosi che sembrano seguire gli utenti e alcune finestre laterali che, se inquadrate dalla giusta prospettiva, rivelano un cielo notturno con la luna piena;
- tre torce elettriche equipaggiabili, che possono essere accese e spente tramite un interruttore su di esse;
- una lastra di pietra quadrata.

Gli utenti dovranno innanzi tutto afferrare le tre torce, accenderle e puntarle contemporaneamente sulla tavola di pietra per poter riuscire a leggere l'iscrizione incisa sopra (**interazione collaborativa basata sullo sguardo**).

La scritta è un indovinello e recita:

"*Quando nasco quasi non posso essere vista,*
ma al calar di ogni notte puoi confidar che io cresca.
Anche ora, mentre sulla parete mi staglio,
posso sentire il richiamo del lupo solitario.
Se ogni vostro sguardo su di me poserete,
le vostre aspirazioni saranno infin complete."

Dopo aver letto l'iscrizione i giocatori dovranno capire che è necessario combinare lo sguardo sulla luna rappresentata nel dipinto per poter proseguire (**interazione collaborativa basata sullo sguardo**). Dopo che tutti gli utenti avranno concentrato l'attenzione sulla luna, il dipinto si animerà aprendo una piccola botola al centro della scena rappresentata con uno strano simbolo nascosto e diventando bidimensionale.



Scenario 4: Traduzione Nella scena sono presenti la scatola iniziale, quattro oggetti dinamici di scena (una daga, un calice, un teschio e un portalume) e nuovamente il dipinto. Dopo aver combinato il proprio sguardo

tramite un punto di interesse condiviso sul dipinto, gli utenti osserveranno nuovamente il quadro diventare bidimensionale e poi ruotare su un suo cardine, rivelando una nicchia vuota e un macchinario agganciato al retro del quadro. Questo macchinario presenta una manopola e un foro che mostra una lettera dell'alfabeto; la rotazione della manopola permette di far scorrere la lettera mostrata e contemporaneamente anche il simbolo all'interno del quadro dal lato opposto di esso, associando così le lettere dell'alfabeto a dei simboli.

La scatola, invece, presenta una nuova zona in cui è possibile inserire una combinazione di quattro lettere, in cui a ciascuna lettera è associata una gemma di colore diverso.

Esaminando gli oggetti di scena gli utenti dovranno capire che su ognuno di essi è individuabile un simbolo particolare con un colore assegnato dato dalle gemme incastonate. Una volta trovati i quattro simboli occorrerà tradurli in lettere dell'alfabeto utilizzando il marchingegno installato dietro il dipinto (la parola tradotta è RLUH, *r'luh* in *R'lyehian* significa segreto). È necessario quindi che i tre giocatori cooperino verbalmente in modo tale che un utente giri la manovella dietro il dipinto per far scorrere i simboli nella parte frontale di esso dove sarà presente un altro utente alla ricerca dei simboli corrispondenti a quelli trovati sotto gli oggetti e il terzo utente sarà sulla scatola ad inserire la combinazione corretta comunicata dagli altri due utenti (**interazione collaborativa basata sulla comunicazione verbale**).





Figura 5.21. UML riassuntivo delle principali funzioni, strutture dati ed eventi forniti dalla libreria

Parte IV

Valutazioni

Capitolo 6

Sperimentazione

Dopo la definizione dei paradigmi di interazione e di *awareness*, a seguito dello studio dello stato dell'arte, dopo il design e infine lo sviluppo di una app in realtà aumentata che superasse le varie criticità incontrate e soddisfacesse tutti i requisiti che ci si è posti, si è arrivati infine ad avere tutti gli strumenti necessari per poter effettuare la validazione degli studi compiuti attraverso l'applicazione di un rigoroso e accuratamente studiato protocollo sperimentale. Tale protocollo verrà esaminato nel dettaglio nella prossima sezione, di seguito alcune informazioni generali sulla sperimentazione.

Il *pool* di volontari a cui è stato somministrato il protocollo ammonta a 42 persone di cui 11 di sesso femminile e 31 di sesso maschile, in un range d'età che varia tra i 18 e 64 anni. Al fine di agevolare l'esecuzione del protocollo e di contenere il numero di soggetti necessari, si è deciso di limitare a due persone il numero di componenti per ciascun gruppo di lavoro, anziché tre persone come invece ipotizzato nei capitoli precedenti. Per mantenere comunque continuità con il lavoro svolto e al contempo esercitare un maggiore controllo sulla simulazione, si è deciso di mantenere comunque la terza persona, ma di impersonarla direttamente, in qualità di *super utenti*. Di conseguenza si è deciso di agire come meri esecutori delle direttive impartite dal sistema o dagli stessi utenti in modo da non influenzare minimamente i dati quantitativi o qualitativi ricavati dalla sperimentazione.

Come anticipato questa fase di test prevede l'esecuzione di tre scenari collaborativi, uno per ogni paradigma di interazione nelle modalità che saranno a breve esaminate. Al termine degli scenari sperimentali è stata data anche la possibilità di provare opzionalmente gli scenari di tipo *escape room*. La durata di ciascuna sessione sperimentale è stata di circa 60/90 minuti.

6.1 Definizione del protocollo sperimentale

Inizialmente si sono delineati con la massima chiarezza gli obiettivi di tale sperimentazione e in base a quanto stabilito è stato pianificato un protocollo che permettesse di ottenere risultati il più significativi possibile, sia dal punti di vista quantitativo attraverso opportuni log che dal punto di vista qualitativo attraverso la somministrazione di questionari. A seguito della definizione esaustiva del protocollo si è proceduto con una valutazione di fattibilità dello stesso e con un fase preliminare di test. A seguito di ciò, si è proceduto nell'effettuare alcune semplificazioni e riadattamenti che hanno condotto al protocollo finale. Di seguito una descrizione più dettagliata di questo processo.

6.1.1 Obiettivi della sperimentazione e considerazioni iniziali

L'obiettivo principale di questo lavoro, come descritto dettagliatamente nei precedenti capitoli, consiste nella valutazione di paradigmi di interazione collaborativa e di *awareness* in un contesto di realtà aumentata multi-utente, multi-dispositivo, co-locato o remoto. Come già visto, per effettuare tali valutazioni si è proceduto con l'identificazione di una serie di interazioni collaborative e di *visual cues* per garantire l'*awareness* multi-utente. A questo punto, dopo aver effettuato tutti gli studi del caso e aver identificato con chiarezza tutti gli elementi in gioco riguardanti tale argomento, è possibile definire in modo puntuale gli aspetti da porre come obiettivi per la fase di testing:

- valutazione qualitativa dell'interfaccia utente in contesto co-locato, permettendo agli utenti di testare separatamente alcune configurazioni di *visual cues* predefinite;
- valutazione qualitativa dell'interfaccia utente in contesto remoto;
- valutazione qualitativa generale di usabilità degli scenari di test;
- valutazione qualitativa generale sugli scenari di tipo "escape room";
- valutazioni quantitative attraverso l'analisi dei dati di log forniti dall'applicazione.

In particolare, per quanto riguarda il primo obiettivo, si sono stabiliti tre configurazioni principali di visualizzazioni, definite visualizzazione minima (V1), visualizzazione base (V2) e visualizzazione completa (V3). Il criterio di definizione di tali configurazioni è stato il seguente: si sono delineati tre livelli di complessità dell'interfaccia e per ciascuno di questi livelli è stato ipotizzato quanto gli utenti lo avrebbero apprezzato in base sia al buon senso che agli studi effettuati nel corso di questa tesi. Di seguito una breve descrizione di tali pre-set di *visual cues* e le ipotesi formulate a riguardo.

(V1) Visualizzazione minima Visualizzazione più semplice che include solamente due *visual clues*, cioè il frustum e il cursore proiettato. Tale visualizzazione è stata appositamente scelta come caso limite, dato che molte delle *visual cues* più importanti infatti non sono state inserite. È stato scelto di inserire il cursore in quanto considerato a priori un elemento d'interfaccia fondamentale in qualsiasi scenario collaborativo, quindi non avrebbe contribuito a trarre conclusioni significative la scelta di non inserirlo all'interno di questa visualizzazione. Il frustum invece è stato scelto in quanto considerato l'elemento di interfaccia meno significativo in quanto non fornisce informazioni precise sugli altri utenti e risulta oltretutto molto ingombrante e invasivo. In conclusione tale visualizzazione è stata scelta come caso zero della sperimentazione, ipotizzando a priori una valutazione piuttosto negativa da parte degli utenti.

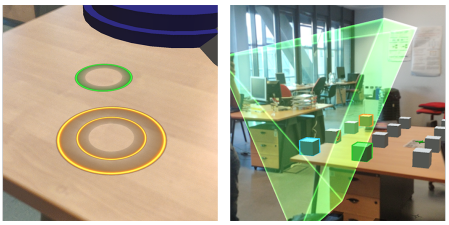


Figura 6.1. Dettagli di alcune catture schermo raffiguranti il cursore e il frustum

(V2) Visualizzazione base Visualizzazione intermedia, include la maggior parte delle *visual cues* esaminate. Questa configurazione infatti fornisce: il cursore, la linea visuale, l'avatar con nome, la mano virtuale e l'evidenziazione visiva degli oggetti manipolati. Tale visualizzazione permette agli utenti di avere un'interfaccia quasi completa dato che l'unica *visual cue* mancante è il sistema per attirare l'attenzione, cioè il punto di interesse condiviso.

Questo tipo di pre-set, così com'è stato strutturato, permette quindi di valutare in modo preciso l'utilità o meno del punto di interesse condiviso, che è stato ipotizzato come uno degli elementi d'interfaccia più utile e significativo all'interno di un ambiente collaborativo.



Figura 6.2. Dettagli di alcune catture schermo raffiguranti il cursore, la linea visuale, l'avatar e la mano virtuale

(V3) Visualizzazione completa Visualizzazione più complessa, include tutte le *visual cues* esaminate nel corso di questo lavoro. Si differenzia da V2 solamente per l'inclusione del punto di interesse condiviso. L'ipotesi preliminare è che sia questa la visualizzazione più apprezzata dagli utenti.

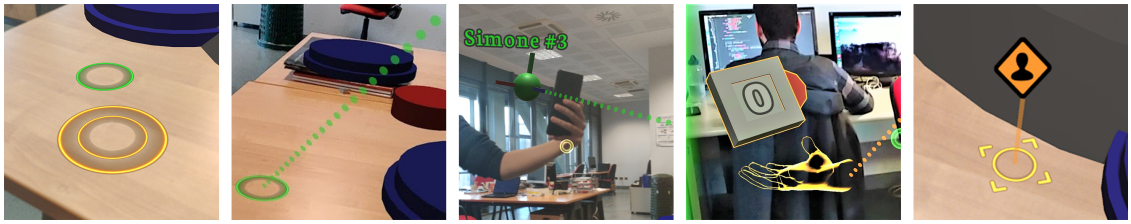


Figura 6.3. Dettagli di alcune catture schermo raffiguranti il cursore, la linea visuale, l'avatar, la mano virtuale e il punto di interesse condiviso

Per la valutazione dei dati qualitativi si è fatto uso esteso di appositi questionari, in particolare sono stati selezionati otto questionari da somministrare in momenti precisi della sperimentazione. Per i dettagli di tali questionari e su come sia stati inseriti all'interno del protocollo sperimentale si rimanda alle apposite sezione.

Per la valutazione dei dati quantitativi invece sì è predisposto un sistema di registrazione automatica di eventi all'interno dell'applicazione. Tale sistema si collega ad un documento online popolandolo opportunamente di tutti i dati utili. Di seguito un elenco di tutti i possibili eventi registrati:

- `RoomCreated`, registra il momento in cui è stata generata una determianta stanza o *lobby*;
- `PlayerEntered`, registra il momento in cui un determinato utente si è unito ad una stanza;
- `PlayerLeft`, registra l'istante in cui un determinato utente abbandona una stanza;
- `PlayerStateChanged`, registra il momento in cui uno specifico utente cambia stato e passa dall'essere non ancora sincronizzato con il *marker* ad essere sincronizzato;
- `PlayerFocalPointCreated`, registra il momento in cui un determinato giocatore genera un punto di interesse condiviso;
- `PlayerFocalPointDestroyed`, registra il momento in cui un determinato giocatore distrugge un punto di interesse condiviso;
- `StaticObjectSpawned`, registra l'istante in cui viene generato all'interno della stanza un oggetto virtuale statico;
- `DynamicObjectSpawned`, registra l'istante in cui viene generato all'interno della stanza un oggetto virtuale dinamico;
- `RoomSceneChanged`, registra il momento in cui viene caricata una determinata scena durante la simulazione;
- `SceneStarted`, registra il momento in cui una determinata scena è stata caricata;
- `SceneCompleted`, registra l'istante in cui una determinata scena viene completata;
- `ObjectGrabbed`, registra l'istante in cui un determinato oggetto virtuale dinamico viene afferrato;
- `ObjectPlaced`, registra l'istante in cui un determinato oggetto virtuale dinamico viene posato;
- `GazeCompleted`, registra l'istante in cui l'interazione collaborativa basata sullo sguardo viene completata;
- `ManipCompleted`, registra l'istante in cui l'interazione collaborativa basata sulla manipolazione viene completata;
- `PosCompleted`, registra l'istante in cui l'interazione collaborativa basata sulla posizione viene completata.

Infine, per chiarezza espositiva, poiché, come anticipato, la sperimentazione prevede l'esecuzione di tre tipologie diverse di interazione collaborative, tali interazioni verranno identificate nel seguente modo:

- l'interazione collaborativa basata sullo sguardo verrà codificata come I1 e i suoi tre *task* verranno codificati come T1.1 (osservare contemporaneamente lo stesso cubo), T1.2 (osservare contemporaneamente cubi diversi in contesto *system commanded*) e T1.3 (osservare contemporaneamente cubi diversi in contesto *user commanded*;

- analogamente l'interazione collaborativa basata sulla manipolazione verrà codificata come I2 e i suoi due *task* come T2.1 (manipolazione contemporanea di oggetti diversi in contesto *system commanded*) e T2.2 (manipolazione contemporanea di oggetti diversi in contesto *user commanded*;

- infine l'interazione collborativa basata sulla posizione verrà codificata come I3 e i suoi tre *task* come T3.1 (posizionarsi contemporaneamente sulla stessa piattaforma), T3.2 (posizionarsi contemporaneamente su piattaforme diverse in contesto *system commanded*) e T3.3 (posizionarsi contemporaneamente su piattaforme diverse in contesto *user commanded*).

6.2 Questionari

Come già anticipato, per la valutazione qualitativa dei dati, si è fatto largo uso di questionari. In particolare sono stati definiti otto questionari diversi in modo tale che potessero essere somministrati all'occorrenza nel corso della simulazione. Si è ipotizzato infatti che proporre un questionario unico al termine di tutta l'esperienza, considerando la lunghezza della stessa, avrebbe compromesso le valutazioni fornite dai soggetti in quanto quest'ultimi avrebbero probabilmente avuto serie difficoltà a distinguere i vari pre-set di *visual cues* o di identificare con esattezza il *task* a cui alcune domande avrebbero fatto riferimento. Per questo si è deciso di suddividere le domande in più questionari tematici da proporre al termine di ogni interazione collaborativa.

In particolare è stato definito un questionario iniziale conoscitivo, consultabile in appendice A, con alcune domande personali e una serie di domande sulle esperienze pregresse riguardanti l'utilizzo di smartphone, l'utilizzo di app in generale, di app in realtà aumentata, di videogiochi in generale, di

videogiochi collaborativi ed eventuali esperienza collaborative di tipo *escape room*.

Per quanto riguarda la stesura dei successivi questionari, sono state effettuate diverse ricerche per trovare in letteratura uno standard che risultasse utile ai nostri scopi. Dopo aver esaminato alcuni dei più celebri questionari sull'usabilità tra cui USE [6] e SUS [2] e sul carico cognitivo come il NASA-TLX [1] si è passato ad esaminare questionari più specifici come il VRUSE [4] per l'usabilità in contesto di realtà virtuale che con le dovute approssimazioni può essere parzialmente applicabile al nostro caso. A seguito di ulteriori ricerche, è stato trovato un questionario piuttosto recente e molto promettente che si presenta come uno standard di valutazione dell'usabilità di una applicazione in realtà aumentata su dispositivi *hand-held*. Tale questionario, chiamato HARUS da Handheld Augmented Reality Usability Scale [28], si suddivide in due sezioni principali: una parte dedicata alla valutazione della comprensibilità di una applicazione AR, cioè la facilità di comprensione delle informazioni fornite dal sistema, e una parte dedicata alla valutazione della manipolabilità di un sistema AR, cioè l'ergonomia dei dispositivi utilizzati. Poiché tale questionario è risultato il più adatto ai nostri scopi, si è deciso di integrarlo in fase di valutazione dell'interfaccia e dell'ergonomia in generale. In particolare sono state selezionate alcune domande di comprensibilità che sono apparse da subito più rilevanti per i sistemi di visualizzazione e quindi utili a fornire una valutazione dei diversi set di *visual cues*. Queste domande sono state riutilizzate più volte all'interno dei vari questionari. Per quanto riguarda invece le domande più generiche sulla comprensibilità e sulla manipolabilità si è deciso di creare due questionari appositi visibili in appendice E e in appendice F.

Per la valutazione delle interazioni collaborative e dei relativi set di visualizzazioni, si è proceduto a progettare tre questionari appositi che valutino singolarmente le varie *visual cues* chiedendo ai soggetti se le hanno trovate utili oppure no, sia in generale che ai fini della collaborazione con gli altri soggetti. In particolare per ciascun set di visualizzazione è presente, come anticipato nel precedente paragrafo, un sottoinsieme di domande tratte dalla comprensibilità di HARUS. Questi questionari sono consultabili in appendice, in particolare il questionario sulla interazione collaborativa basata sullo sguardo è disponibile in appendice B, quello relativo all'interazione collaborativa basata sulla posizione è disponibile in appendice C, mentre quello relativo all'interazione collaborativa basata sulla posizione è disponibile in appendice D. Inoltre, per avere una valutazione più "standard" del lavoro svolto, si

è deciso comunque di somministrare ai soggetti un questionario SUS relativo all'esperienza in generale. Questo questionario può essere consultato in appendice G.

Infine si è deciso di predisporre un questionario opzionale relativo al caso d'uso che si è deciso di implementare, cioè l'*escape room* virtuale. Tale questionario presenta diverse domande utili per fornire una panoramica generale sul livello di gradimento dell'app sviluppata ed è possibile consultarlo in appendice H. Si rimanda alle sezione riguardante la valutazione dei risultati per esaminare in dettaglio le risposte fornite dai soggetti della sperimentazione. Si procede di seguito con la formulazione del protocollo sperimentale completo.

6.2.1 Protocollo sperimentale completo

Dopo aver definito con chiarezza gli obiettivi della sperimentazione e le modalità per ottenere i dati quantitativi e qualitativi necessari per poter trarre conclusioni significative, si è proceduto con la definizione rigorosa di un protocollo sperimentale.

Innanzi tutto, come già detto precedentemente, i vari soggetti devono eseguire tre tipologie di interazione collaborativa: I1, I2 e I3. Per poter valutare tutti i vari pre-set di visualizzazioni, si è deciso di far ripetere ciascuna interazione I tante volte quanti sono le varie configurazioni di visualizzazioni. Essendo tre tali configurazioni, cioè V1 (minima), V2 (base) e V3 (completa),

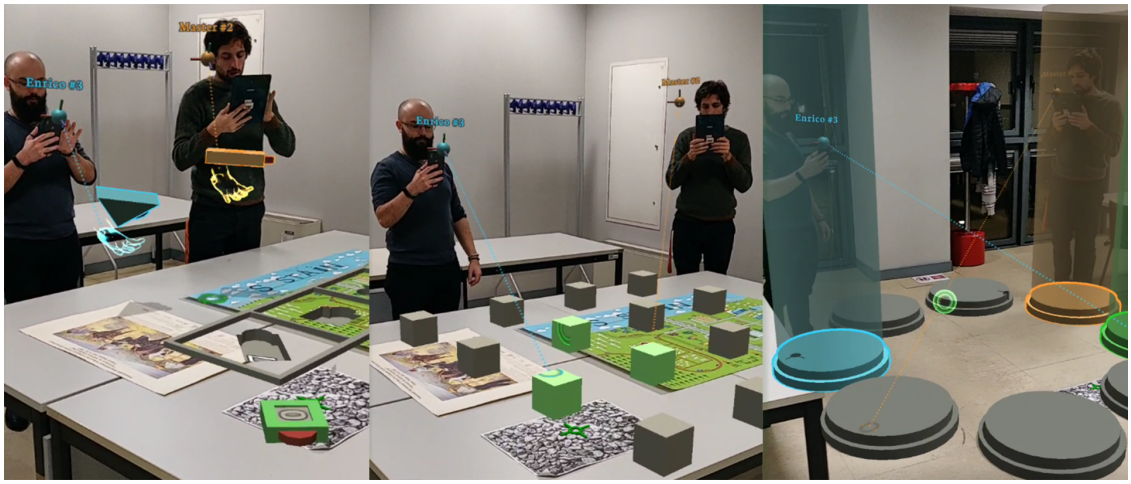


Figura 6.4. Alcune catture schermo raffingurante le tre interazioni collaborative nel *setting* della sperimentazione

ciascuna interazione I viene ripetuta tre volte, ognuna con un diverso tipo di visualizzazione. Al termine di ciascuna interazione I viene chiesto ai soggetti di compilare un questionario con il quale è possibile valutare e confrontare le varie *visual cues* di ciascun pre-set di visualizzazioni. Al termine dell'ultima interazione, cioè I3, viene chiesto ai soggetti di compilare, oltre al questionario relativo a I3, ulteriori tre questionari relativi alla compresibilità, alla manipolabilità e all'usabilità. Dopodiché viene chiesto ai soggetti la disponibilità ad effettuare un esperienza di *escape room* virtuale opzionale, e, in caso acconsentano, sono invitati, dopo aver concluso l'esperienza, a compilare un ulteriore questionario relativo a questo caso d'uso videoludico.

Per poter valutare il tutto anche in contesto remoto, si è pensato quindi di far ripetere tutta l'esperienza in contesto remoto a ciascun gruppo oppure dividere il gruppo di volontari in due, somministrando a metà la sperimentazione in contesto co-locato e metà in contesto remoto, chiedendo ad un membro del gruppo di recarsi in un'altra posizione sufficientemente distante da quella originale per simulare una situazione in cui uno dei partecipanti non è fisicamente presente.

Un problema che potrebbe sorgere nell'applicare un protocollo del genere è il *learning effect* che si andrebbe a creare nel momento in cui le stesse interazioni vengono ripetute più volte. Questo effetto potrebbe influire in maniera significativa sui dati raccolti in quanto i soggetti, potendo ripetere così tante volte gli stessi compiti, ad un certo punto sapranno già quello che dovranno fare riducendo inevitabilmente l'efficacia e l'utilità delle varie *visual cues*. È possibile però risolvere in parte questo problema inserendo nel protocollo un sistema di randomizzazione nella somministrazione delle visualizzazioni per ciascun gruppo. Si sono quindi stabilite sei possibili sequenze delle tre visualizzazioni e si è definito a priori quale sequenza somministrare ai vari gruppi, distribuendo le sequenze uniformemente. Per quanto riguarda invece gli scenari di tipo *escape room* si è deciso di fornire alternativamente ai vari gruppi le visualizzazioni V2 e V3, avendo stabilito a priori la bassa rilevanza di V1.

Riassumendo, dopo aver convocato un gruppo formato da due soggetti il protocollo prevede l'esecuzione nell'ordine dei seguenti passaggi:

1. selezionare la corretta sequenza di visualizzazioni da somministrare al gruppo attuale;
2. illustrare brevemente ai soggetti cosa si intende per realtà aumentata e informarli su ciò che dovranno affrontare;

3. chiedere ai soggetti di compilare il primo questionario, cioè il questionario conoscitivo (appendice A);
4. eseguire i task relativi all'interazione I1, con la visualizzazione corretta selezionata;
5. ripetere il punto precedente altre due volte seguendo la sequenza di visualizzazioni;
6. chiedere ai soggetti di compilare il successivo questionario, cioè il questionario relativo all'interazione I1 (appendice B);
7. ripetere i punti 3 e 4 eseguendo i task relativi all'interazione I2 invece che all'interazione I1;
8. chiedere ai soggetti di compilare il successivo questionario, cioè il questionario relativo all'interazione I2 (appendice C);
9. ripetere i punti 3 e 4 eseguendo i task relativi all'interazione I3 invece che all'interazione I1;
10. chiedere ai soggetti di compilare il successivo questionario, cioè il questionario relativo all'interazione I3 (appendice D);
11. chiedere ai soggetti di compilare i successivi questionari, cioè quelli relativi alla comprensibilità, alla manipolabilità e all'usabilità (appendice E, F, G);
12. eseguire opzionalmente gli scenari di tipo *escape room*;
13. chiedere ai soggetti di compilare il successivo questionario, cioè il questionario relativo all'*escape room* (appendice H);
14. eventualmente ripetere i punti dal 4 al 13 passando da contesto co-locato a contesto remoto.

6.2.2 Semplificazioni e protocollo sperimentale finale

In seguito alla definizione del protocollo sperimentale completo, si sono effettuate alcune valutazioni sulla fattibilità di tale procedura e sono state decise alcune semplificazioni al fine di alleggerire sia il numero di parametri da tenere in considerazione che la durate della sperimentazione stessa.

La prima semplificazione attuata è stata quella di rimuovere il caso di esperienza collaborativa in remoto. Questo ha permesso di dimezzare i tempi di esecuzione del protocollo, ma ha al contempo impedito una valutazione

effettiva dell'efficacia della collaborazione nel caso in cui uno o più utenti partecipino alla simulazione da remoto. Si è decido di rimandare tale valutazione ad un eventuale ulteriore fase di sperimentazione postuma alla pubblicazione di questo lavoro.

Dopo questa semplificazione si sono svolti alcuni test preliminari col fine di validare l'efficacia del protocollo. Dopo aver somministrato questa procedura sperimentale ad alcuni soggetti e aver raccolto alcuni pareri preliminari, sia direttamente dai soggetti che da alcuni professionisti del settore, si è giunti alla conclusione che la visualizzazione minima, cioè V1, poteva essere scartata a priori, perché meno interessante. Questo avrebbe ulteriormente semplificato il protocollo sperimentale, evitando di ripetere tre volte ogni interazione collaborativa. La rimozione di V1 è giustificata dal fatto che le *visual cues* utilizzate non avrebbero apportato alcun contributo significativo sulla valutazione dell'interfaccia utente.

Il protocollo sperimentale finale quindi corrisponde per lo più a quello completo con due significative semplificazioni: non vengono più ripetute tre volta le varie interazioni, ma solo due volte; non si deve eventualmente ripetere una seconda volta l'esperienza per la valutazione di un ambiente remoto. La rimozione di V1 comporta inoltre una ridefinizione delle sequenze di visualizzazione per contrastare il *learning effect*: infatti non si hanno più sei possibili sequenze, ma solo due. Un'altra conseguenza della rimozione di V1 è che ora i soggetti non hanno più un caso zero con il quale confrontare le visualizzazioni V2 e V3, l'unica cosa che cambia ora è la presenza o meno del punto di interesse condiviso; i soggetti potranno quindi valutare tale *visual cue* facendo riferimento ai casi in cui era permesso utilizzarlo e nei casi in cui non era permesso, mentre per le altre *visual cues* non avranno alcun caso nullo in cui non era possibile utilizzarle. Nonostante ciò questa semplificazione non inficia la buona riuscita della sperimentazione, anche senza la presenza del caso nullo, i soggetti dovrebbero essere comunque in grado di fornire pareri affidabili sull'utilità o meno delle varie *visual cues* sia in generale che ai fini della collaborazione. Di seguito l'elenco numerato aggiornato con il protocollo finale utilizzato per la sperimentazione:

1. selezionare la corretta sequenza di visualizzazioni da somministrare al gruppo attuale, tale sequenza ora può essere solo V2-V3 (base-completa) oppure V3-V2 (completa-base);

2. illustrare brevemente ai soggetti cosa si intende per realtà aumentata e informarli su ciò che dovranno affrontare;

3. chiedere ai soggetti di compilare il primo questionario, cioè il questionario conoscitivo (appendice A);
4. eseguire i task relativi all'interazione I1, con la visualizzazione corretta selezionata;
5. ripetere il punto precedente con la successiva visualizzazione;
6. chiedere ai soggetti di compilare il questionario relativo all'interazione I1 (appendice B);
7. ripetere i punti 3 e 4 eseguendo i task relativi all'interazione I2 invece che all'interazione I1;
8. chiedere ai soggetti di compilare il questionario relativo all'interazione I2 (appendice C);
9. ripetere i punti 3 e 4 eseguendo i task relativi all'interazione I3 invece che all'interazione I1;
10. chiedere ai soggetti di compilare il questionario relativo all'interazione I3 (appendice D);
11. chiedere ai soggetti di compilare i questionari relativi alla comprensibilità, alla manipolabilità e all'usabilità (appendice E, F, G);
12. avviare opzionalmente gli scenari di tipo *escape room*
13. chiedere ai soggetti di compilare il questionario relativo all'*escape room* (appendice H);

6.2.3 Somministrazione del protocollo

La sessione sperimentale è durata due settimane durante le quali sono stati convocati 21 gruppi. Come *setting* per la sperimentazione è stata utilizzata



una sala riunioni all'interno del Dipartimento di Automatica ed Informatica (DAUIN) del Politecnico di Torino opportunamente allestita. Per aumentare la stabilità della sessione AR si è proceduto ad incrementare le *features* ambientali presenti all'interno della sala riunioni con poster e disegni opportunamente fissati su tavoli e pareti. Durante l'esecuzione del protocollo si è fatto uso esteso della proiezione di slide esemplificative. È risultato imperativo preparare correttamente i soggetti affinché affrontassero i vari *task* sapendo esattamente a cosa sarebbero andati incontro; tale preparazione è necessaria per non inficiare i log sulle tempistiche che altrimenti sarebbero stati sporcati dal tempo necessario ai soggetti per capire cosa avrebbero dovuto fare. In figura 6.2.3 e in figura 6.2.3 alcune foto raffiguranti il *setting* allestito per la sperimentazione.



6.3 Valutazione dei risultati

Prima di tutto alcune premesse. I grafici in questa sezione sono dei *boxplot*, in cui le linee centrali rappresentano le mediane, i limiti del rettangolo indicano il primo quartile e il terzo quartile, mentre i baffi si estendono 1.5 volte la lunghezza del rettangolo; i valori "estremi" vengono rappresentati con dei punti e le croci rappresentano i valori medi. Per quanto riguarda i

dati dei questionari, le domande con accezione negativa sono state invertite e rappresentate con la lettera "i" e quelle che hanno ottenuto significatività statistica sono state contrassegnate con un asterisco. La significatività statistica è stata stabilita mediante t-test con p-value < 0.05.

Date le modalità di somministrazione, la significatività statistica è stata calcolata anche comparando sequenze di visualizzazione diverse (base-completa e completa-base) e nel caso non sia stata rilevata significatività statistica si è proceduto a unire i dati relativi.

Si riportano quindi i risultati dei test.

Informazioni generali sui partecipanti I partecipanti che sono stati reclutati per questa sperimentazione sono abbastanza eterogenei (29 maschi e 11 femmine) con età compresa tra i 18 e i 34 anni (con l'eccezione di una persona tra i 55 e i 64 anni).

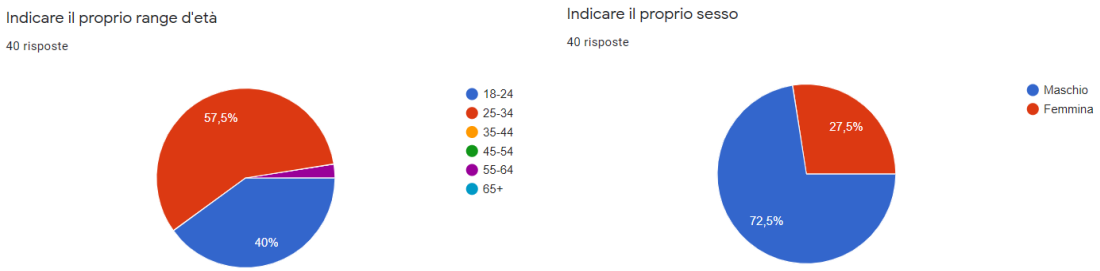


Figura 6.5. Grafici a torta su informazioni riguardo a sesso e età.

Per quanto riguarda le esperienze in campo tecnologico, il nostro campione di volontari ha molta familiarità con gli *smartphone* e più della metà dei partecipanti gioca spesso a videogiochi, anche se solo un quarto del totale gioca spesso a videogiochi con componenti collaborative multi-giocatore. L'esperienza riguardo al mondo della realtà aumentata è in genere bassa, con più del 75% delle persone che non ha mai o quasi mai utilizzato applicazioni in AR.

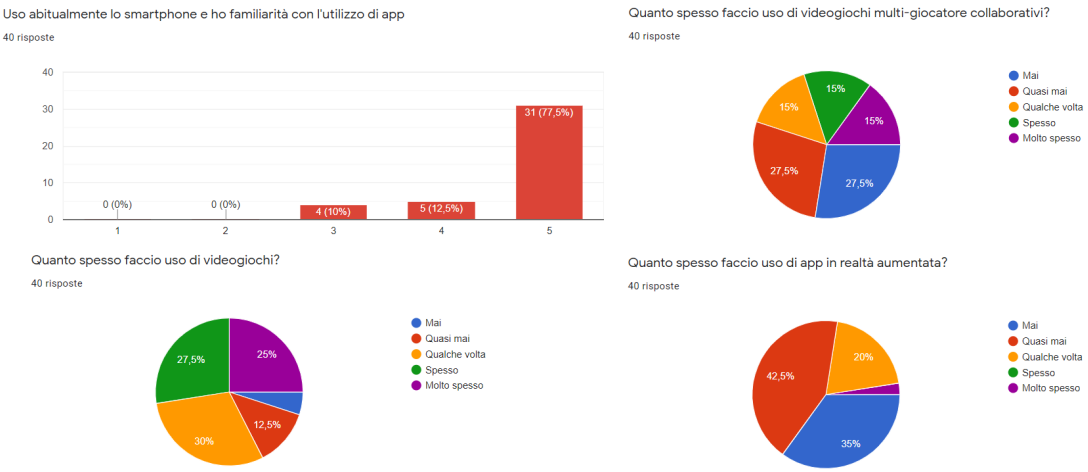


Figura 6.6. Grafici a torta riguardo alle esperienze pregresse.

Valutazione dell'usabilità dell'applicazione tramite SUS In generale i risultati del SUS sono soddisfacenti ottenendo un risultato di 86.75/100, considerando che la media dell'accettabilità dei risultati è di 68 [2]. In particolare si è notato come alcune risposte (3 delle 10) fossero statisticamente diverse tra i due tipi di sequenze di visualizzazione, suggerendo che il fatto di aver ottenuto un punteggio medio più alto con sequenza base-completa (ovvero 90.13) indichi che eseguire i task prima con una visualizzazione più semplice e poi con una più complessa abbia aiutato gli utenti ad assimilare più facilmente le informazioni e migliorato l'usabilità.

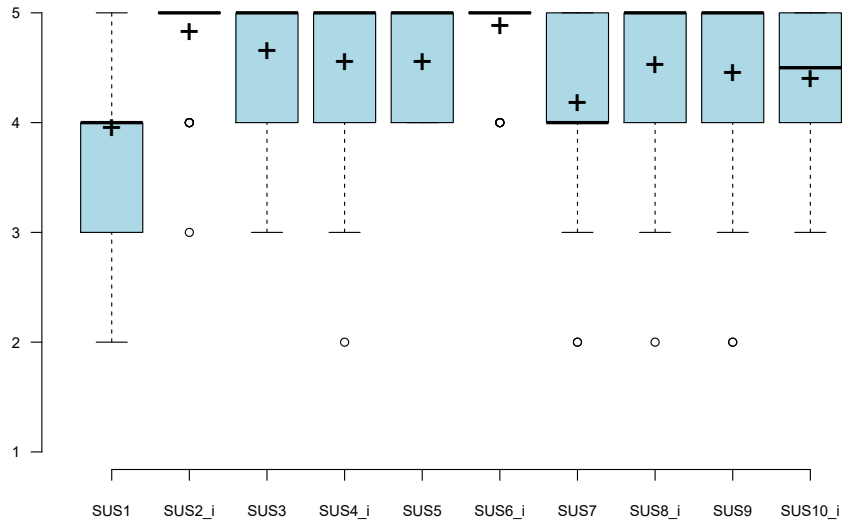


Figura 6.7. Boxplot delle domande relative al questionario SUS.

	Punteggio	Punteggio associato alla sequenza di visualizzazione base-completa	Punteggio associato alla sequenza di visualizzazione completa-base
SUS	86.75	90.13	83.38

Tabella 6.1. Punteggi SUS

Valutazione dell'usabilità dell'applicazione tramite HARUS Analizzando per prima cosa la comprensibilità, in generale nonostante le maggiori informazioni presenti sullo schermo nella visualizzazione completa, esse non sembrano rendere l'esperienza particolarmente più confusionaria o invasiva.

I questionari relativi alle interazioni con lo sguardo e la posizione mostrano la visualizzazione completa leggermente più comprensibile, anche se è non statisticamente significativa la differenza tra i due tipi di visualizzazione. Per quanto riguarda invece la manipolazione di oggetti anche qui la visualizzazione completa è più comprensibile, ma la differenza per due domande su tre è significativa statisticamente. I task di manipolazione richiedevano, infatti, più precisione nell'indicare i punti e quindi può essere questo il motivo per cui viene ritenuta più comprensibile e usabile quella completa.

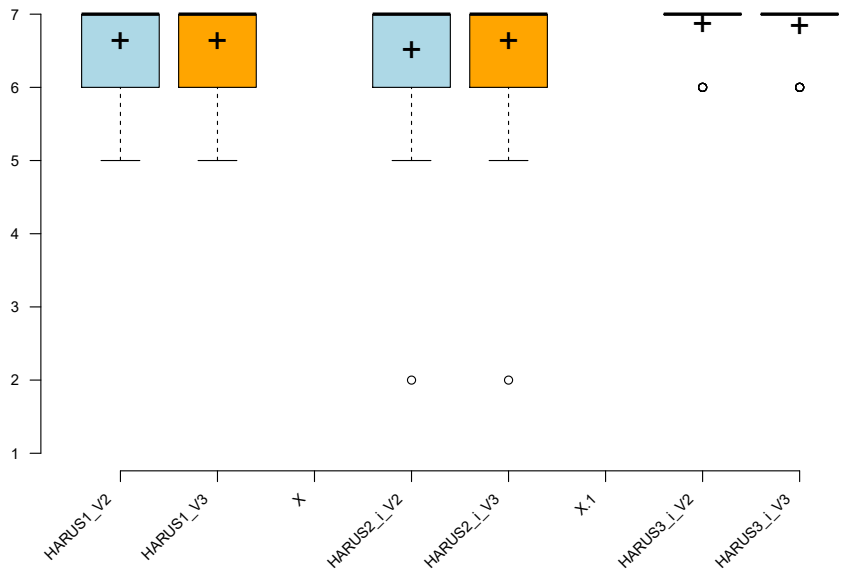


Figura 6.8. Boxplot delle prime tre domande dell'HARUS per quanto riguarda i task relativi allo sguardo, comparando la visualizzazione base (V2) e completa (V3).

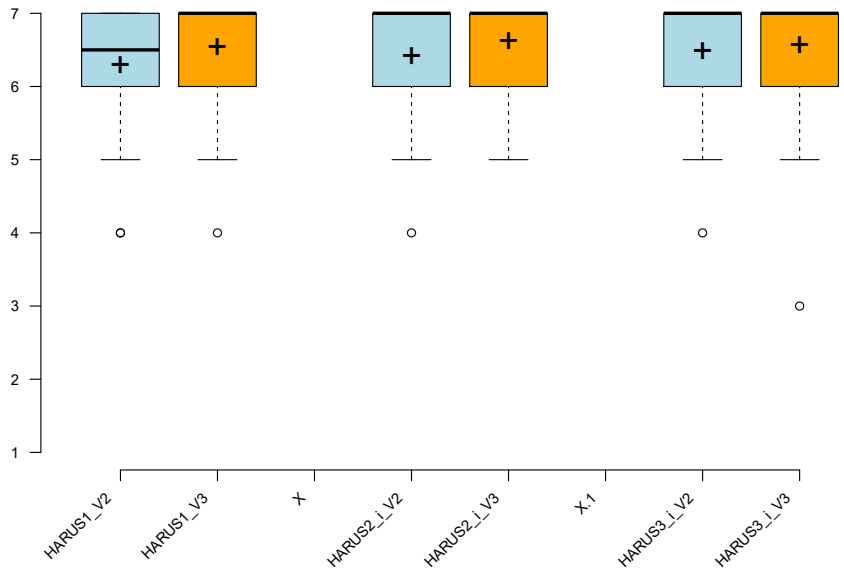


Figura 6.9. Boxplot delle prime tre domande dell'HARUS per quanto riguarda i task relativi alla manipolazione, comparando la visualizzazione base (V2) e completa (V3).

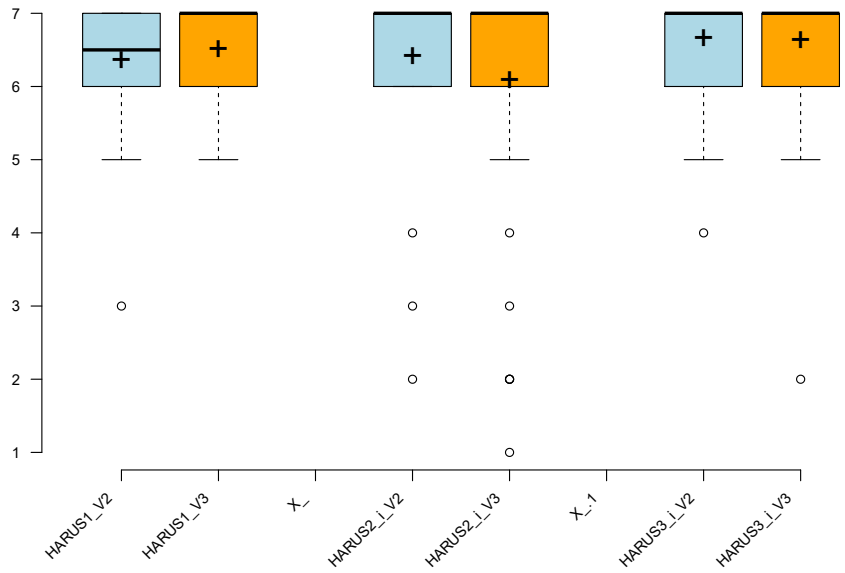


Figura 6.10. Boxplot delle prime tre domande dell'HARUS per quanto riguarda i task relativi alla posizione, comparando la visualizzazione base (V2) e completa (V3).

Comprendendo anche le 5 domande comuni sulla comprensibilità, il punteggio finale medio è di 87.47/100 per la visualizzazione base e 87.76/100 per la visualizzazione completa. Come per il SUS, anche se il numero di dati statisticamente significativi non è alto, il fatto di avere punteggi medi più alti in riferimento alla sequenza base-completa (89.79 base e 90 completa) potrebbe indicare che eseguire i task con questa sequenza abbia aiutato gli utenti ad assimilare più facilmente le informazioni.

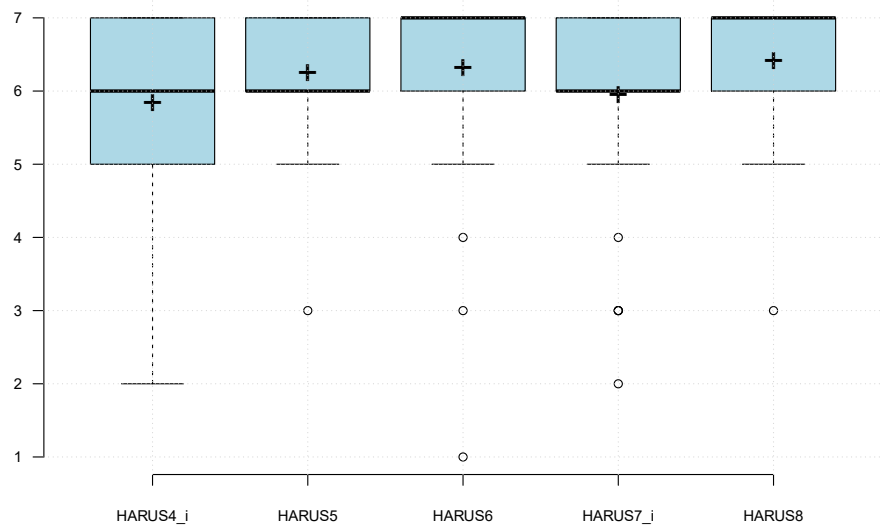


Figura 6.11. Boxplot delle cinque domande generica sulla comprensibilità dell'HARUS.

La manipolabilità dell'applicazione ha ottenuto, invece, un punteggio medio di 82.6/100, senza significatività statistiche tra le due sequenze di visualizzazione, mostrando un valore minore della comprensibilità. I valori più bassi sono stati rilevati in particolare sulla comodità per mani e braccia e sulla comodità di impugnare i dispositivi, elementi comprensibili data la durata dell'esperimento.

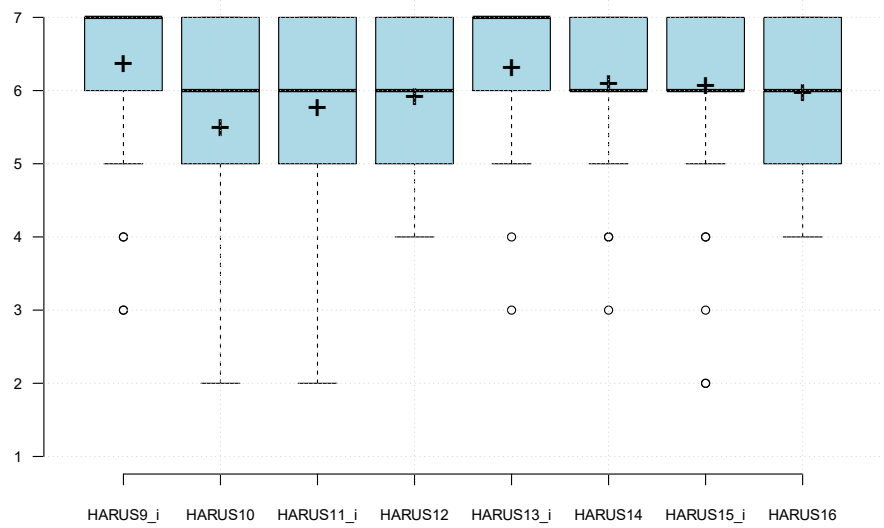


Figura 6.12. Boxplot delle cinque domande generiche sulla manipolabilità dell'HARUS.

Complessivamente quindi l'applicazione ha ottenuto un punteggio medio dell'HARUS di 85.03/100 per la visualizzazione base e 85.18/100 per quella completa, risultato più che soddisfacente.

	Punteggio	Punteggio associato alla sequenza di visualizzazione base-completa	Punteggio associato alla sequenza di visualizzazione completa-base
HARUS Comprensibilità con visualizzazione base	87.47	89.79	85.14
HARUS Comprensibilità con visualizzazione completa	87.76	90	85.52
HARUS Manipolabilità	82.6	82.29	82.92
HARUS Totale con visualizzazione base	85.03	86.04	84.03
HARUS Totale con visualizzazione completa	85.18	86.15	84.22

Tabella 6.2. Punteggi HARUS

Valutazione delle *visual cues* di *awareness* Per quanto riguarda le *visual cues* non sono state rilevate differenze significative statisticamente tra sequenze di visualizzazioni diverse, quindi i dati sono stati considerati assieme.

Tra le *visual cues* proposte quelle considerate senza dubbio più utili sono state, come ci si aspettava, il cursore e la linea visiva. Per quanto riguarda quelle relative alle attività di manipolazione, invece, sia il contorno colorato sia la mano virtuale trasparente sono state considerate leggermente utili, con una preferenza verso il contorno. Anche la rappresentazione degli altri utenti, come ci si aspettava in contesti co-locati, è stata ritenuta solo vagamente utile.

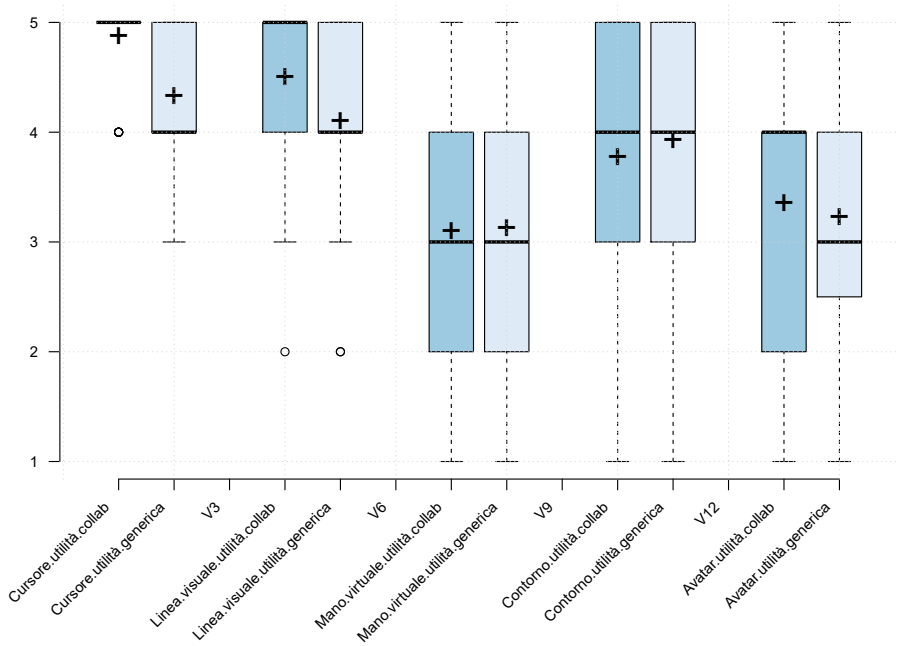


Figura 6.13. Boxplot delle domande relative all'utilità delle *visual cues*.

Valutazione dei punti di interesse condivisi e preferenze tra le visualizzazioni In generale il poter creare i punti di interesse condivisi è

stato considerato molto utile, in particolare negli scenari relativi alla posizione, dove non si possono facilmente inquadrare tutte le piattaforme e la componente del *distractor* si rivela più efficace.

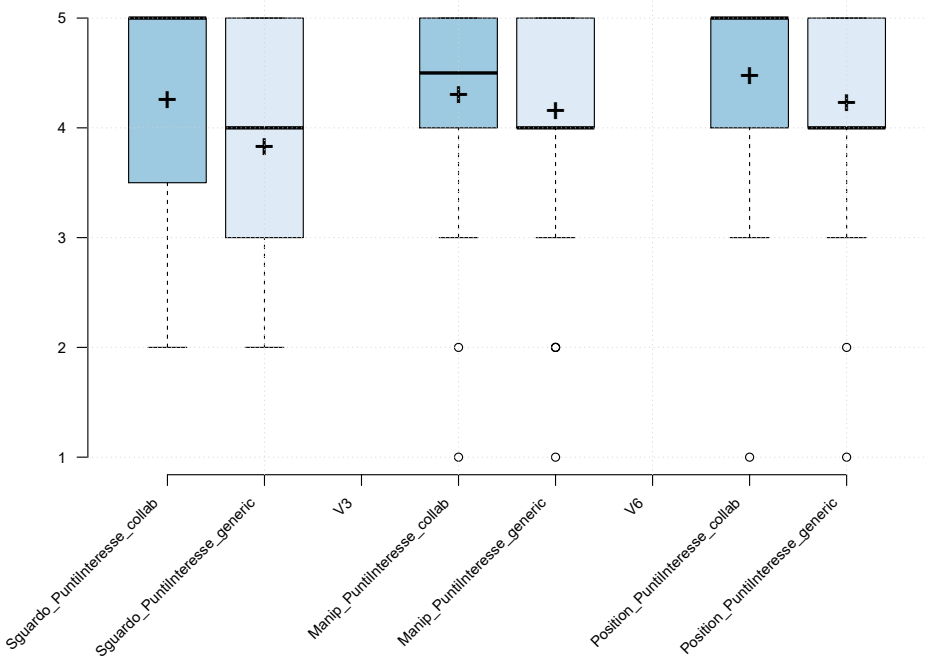


Figura 6.14. Boxplot delle domande relative all'utilità dei punti di interesse condivisi.

Anche guardando le preferenze tra le due visualizzazioni, viene confermata questa tesi, mostrando una netta preferenza (più dell'80% dei partecipanti) in tutte le tipologie di interazioni verso quella completa.

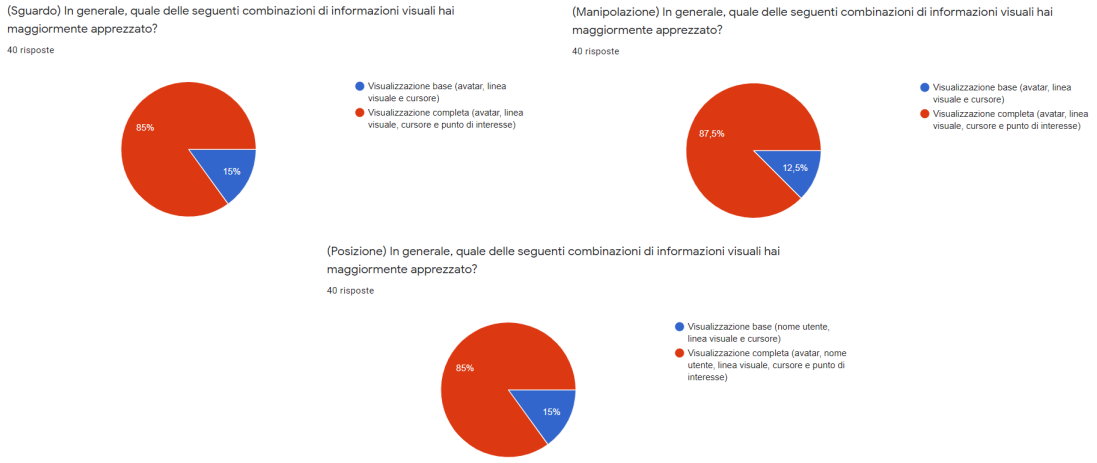


Figura 6.15. Grafici a torta sulla preferenza di visualizzazione nei task relativi a sguardo, manipolazione e posizione.

Domande sulle informazioni visive e sulla collaborazione In linea con gli altri risultati, in generale le informazioni visive aggiuntive, nonostante si fosse in un contesto co-locato, sono state considerate estremamente utili per l'awareness delle altre persone ai fini della collaborazione in ogni tipologia di interazione.

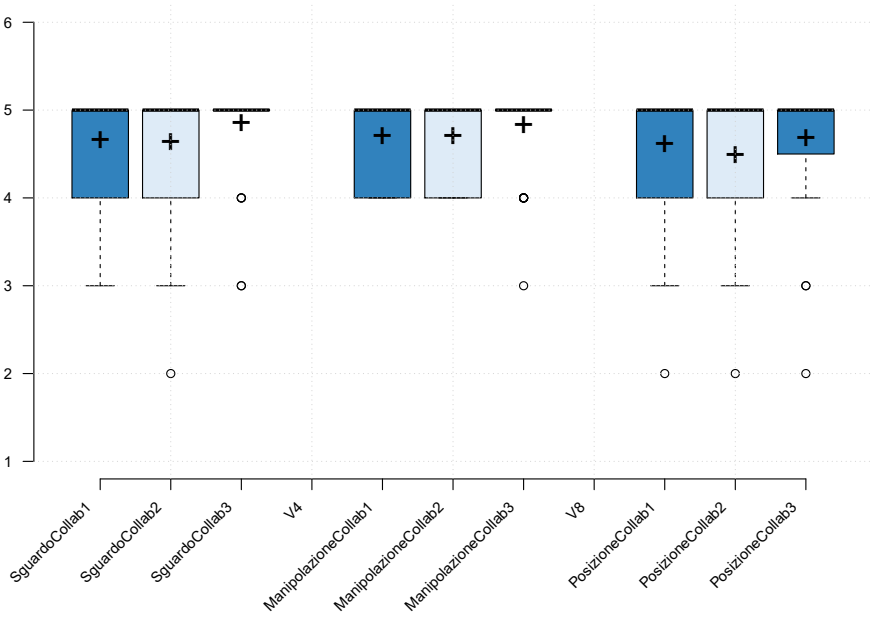


Figura 6.16. Boxplot delle domande relative alla collaborazione divise per tipo di interazione.

Valutazione dei tempi di completamento dei task Guardando in particolare i dati relativi ai task più collaborativi ovvero il terzo task di sguardo e posizione e il secondo task di manipolazione, si possono notare dei trend. In generale si osserva come la prima ripetizione dei task richieda più tempo rispetto alla seconda, come ci si aspetta a causa del *learning factor*. Considerando la media di tutti i dati (mitigando così il learning factor), sembra anche che la visualizzazione completa richieda in genere più tempo rispetto alla visualizzazione base, cosa che può essere motivata dal tempo in più necessario ad eseguire il gesto del doppio tap, usato per creare i punti di interesse condivisi.

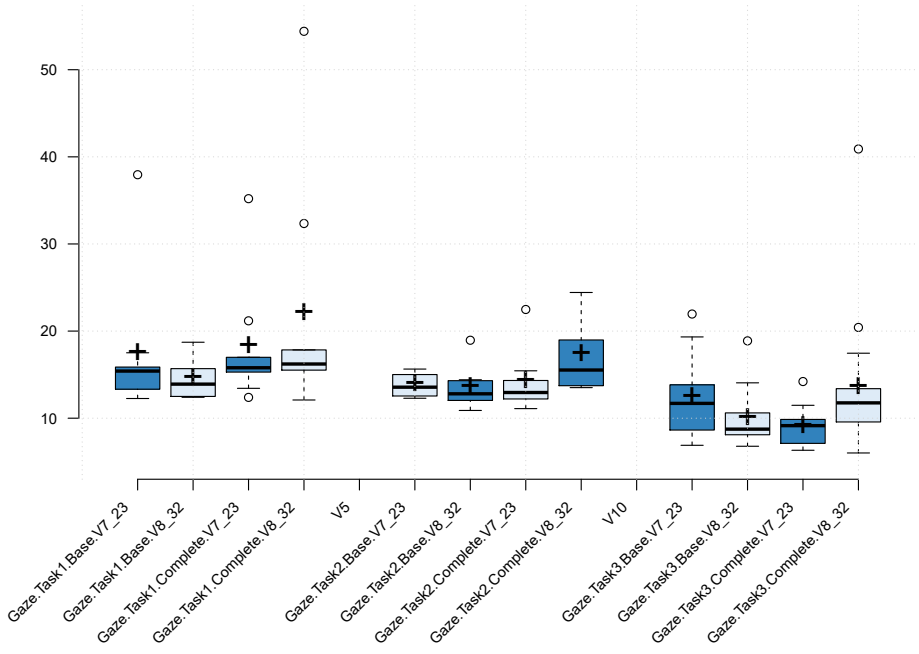


Figura 6.17. Boxplot dei tempi di completamento relativi ai task dello sguardo, divisi per visualizzazione e sequenza di visualizzazione.

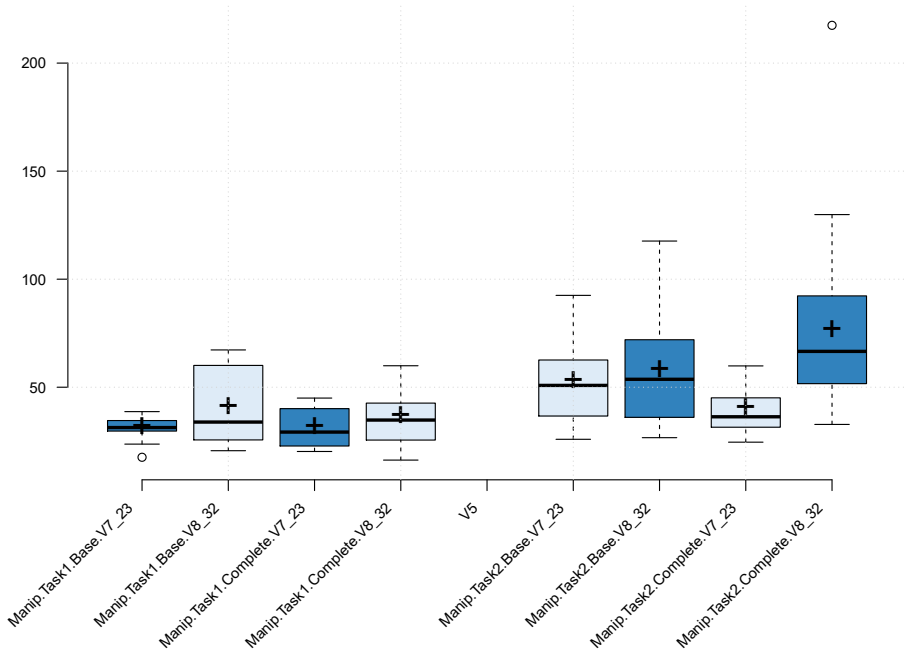


Figura 6.18. Boxplot dei tempi di completamento relativi ai task dello sguardo, divisi per visualizzazione e sequenza di visualizzazione.

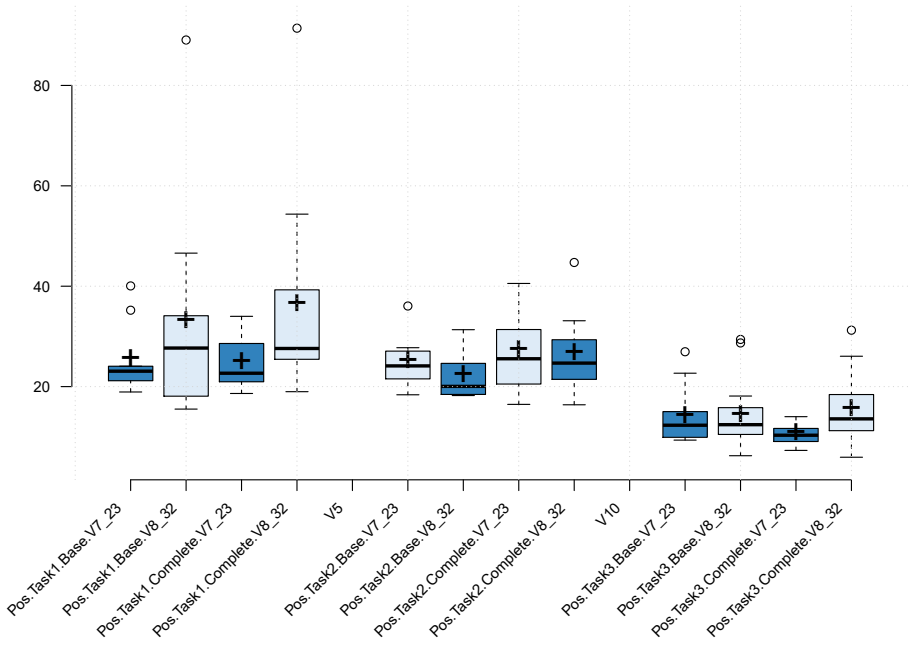


Figura 6.19. Boxplot dei tempi di completamento relativi ai task dello sguardo, divisi per visualizzazione e sequenza di visualizzazione.

	Tempi medi
Sguardo Task1 VBase	15.81444444
Sguardo Task1 VCompleta	19.93777778
Sguardo Task2 VBase	13.55944444
Sguardo Task2 VCompleta	15.63055556
Sguardo Task3 VBase	11.0525
Sguardo Task3 VCompleta	11.16972222
Manipolazione Task1 VBase	35.3045
Manipolazione Task1 VCompleta	33.412
Manipolazione Task2 VBase	54.703
Manipolazione Task2 VCompleta	57.66625
Posizione Task1 VBase	28.946
Posizione Task1 VCompleta	30.315
Posizione Task2 VBase	23.316
Posizione Task2 VCompleta	26.622
Posizione Task3 VBase	13.7815
Posizione Task3 VCompleta	12.7815

Tabella 6.3. Tempi medi task

Valutazione degli scenari della *escape room* Nonostante solo il 10% dei partecipanti effettui spesso esperienze di tipo *escape room*, in generale la *escape room* è stata molto apprezzata, con una media di 9.13/10 (deviazione standard = 0.83), e lo scenario preferito da quasi il 40% dei partecipanti è stato l'ultimo, ovvero quello di traduzione. Gli enigmi sono stati considerati impegnativi ma non in maniera eccessiva e gli oggetti di scena sono stati molto graditi per il loro realismo. Sebbene alcune persone abbiano lamentato specifici controlli poco gestibili, questi scenari sono stati accolti in modo estremamente positivo, tanto che numerosi partecipanti hanno espresso l'interesse nel voler provare in futuro nuove ulteriori scene tratte dalla escape room.

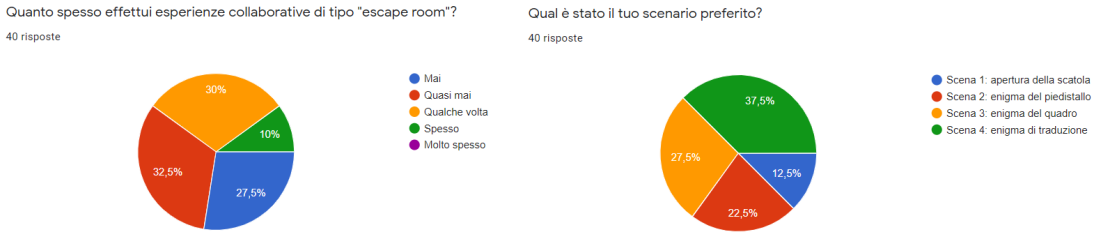


Figura 6.20. Grafici a torta delle esperienze pregresse degli utenti per quanto riguarda le escape room e delle preferenze sugli scenari.

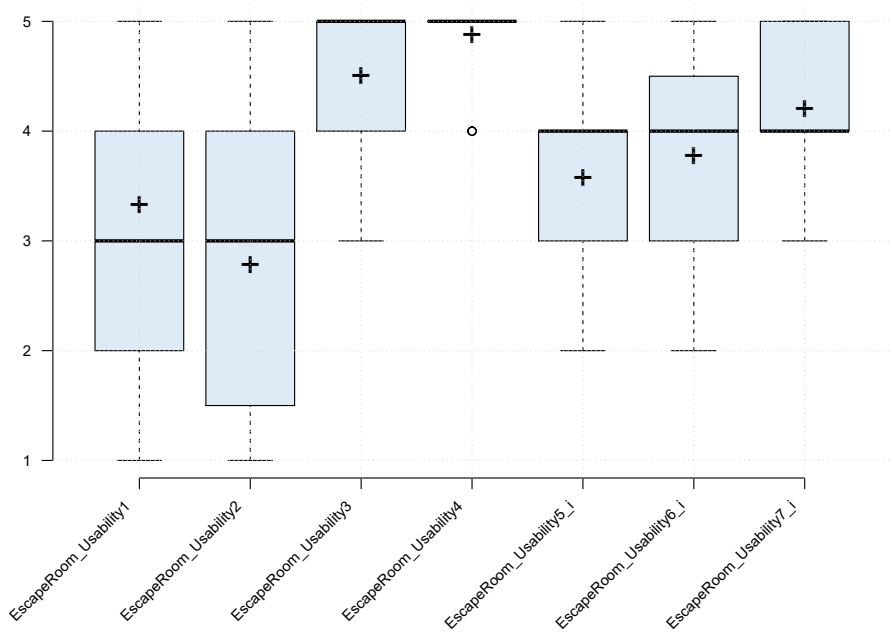


Figura 6.21. Boxplot delle domande relative alla usabilità della escape room.

Capitolo 7

Conclusioni

Al termine di questo lungo lavoro è opportuno trarre le dovute conclusioni ripercorrendo brevemente tutte le tappe che hanno portato al completamento di questa tesi. Come prima cosa sono state effettuate innumerevoli ricerche nel campo della realtà aumentata e più precisamente in abito collaborativo. Dopo aver estratto dallo stato dell'arte i lavori più significativi, si sono delineati una serie di concetti chiave che hanno portato alla definizione di tre interazioni collaborative fondamentali e una serie di informazioni visuali o *visual cues*.

Dopodiché si è proceduto con la definizione dei requisiti che una applicazione di realtà aumentata collaborativa multi-utente avrebbe dovuto soddisfare: l'app sarebbe dovuta essere multi-dispositivo, multi-piattaforma, multi-utente, sia co-locata che remota e avrebbe dovuto implementare soluzioni per realizzare le tre interazioni collaborative precedentemente definite e le varie *visual cues*.

Conclusa questa fase si è stabilito quale caso d'uso si sarebbe implementato nella nostra applicazione e dopo diverse proposte si è deciso per una *escape room* in realtà aumentata. Successivamente, avendo ben chiaro i requisiti precedentemente definiti si è passati alla fase implementativa. Si è per prima cosa strutturato un framework di realtà aumentata che fungesse da substrato unificante per le varie librerie di AR attualmente disponibili, poi si è implementato tale framework sviluppando un'interfaccia per Google ARCore. Si è verificato in seguito il funzionamento corretto di tale interfaccia sviluppando una apposita app basta sulla libreria AR appena creata. Successivamente si è proseguito sviluppando il framework di networking implementando un interfaccia per la piattaforma Photon Unity Networking e si sono gettate le basi per lo sviluppo di un ambiente di realtà aumentata

condiviso. Proseguendo nello sviluppo, si sono studiate soluzioni per la sincronizzazione degli ambienti AR e degli oggetti virtuali e si è ottenuto un primo prototipo di applicazione AR multi-utente.

Successivamente si è iniziato lo sviluppo dell'*escape room* virtuale, selezionando alcuni degli scenari più significativi ai fini della collaborazione. Terminato lo sviluppo dell'applicazione si è iniziato a delineare un protocollo sperimentale preliminare che prevedeva lo sviluppo di un'applicazione che esemplificasse nella maniera più semplice possibile tutti i possibili casi d'uso delle tre interazioni collaborative fondamentali, in modo da poter ottenere risultati più chiari possibili in fase di sperimentazioni.

Si è proseguito quindi sviluppando i tre scenari per la sperimentazione, implementando altresì tre scenari di *training* per ciascuna interazione. Una volta pronta l'applicazione sperimentale si è proceduto con la ricerca in letteratura di questionari che facessero al caso nostro e al contempo si è definito con maggior precisione un protocollo sperimentale valido. Dopo numerose ricerche si sono definiti i vari questionari che si sarebbero utilizzati per ottenere valutazioni qualitative e si sono predisposti meccanismi automatici per la registrazione di opportuni log al fine di ottenere i risultati quantitativi. A seguito di ciò si è stabilito, dopo aver effettuato alcune semplificazioni, un protocollo sperimentale definitivo e si è proceduto con l'avvio della fase di sperimentazione. Tale sperimentazione è stata effettuata su 42 soggetti ed è durata circa due settimane.

Dopo questa fase si è proseguito con l'analisi dei dati raccolti e la stesura di opportuni report a riguardo, terminando questo lavoro di tesi. I dati raccolti hanno permesso di giungere ad alcune conclusioni valide tra cui: c'è una netta preferenza da parte degli utenti per un sistema di visualizzazione rispetto ad un altro (in particolare viene generalmente preferita la visualizzazione completa) al netto di una differenza di tempi di completamento comparabile. Da questo si può per esempio dedurre che gli utenti preferiscono avere un elemento d'interfaccia utile in più nonostante la sua presenza influisca in modo marginale sui tempi di esecuzione dei task collaborativi.

7.1 Contributi significativi

I contributi più significativi che questo lavoro di tesi lascia in eredità sono sostanzialmente due: lo sviluppo di un framework unico per l'implementazione di una applicazione AR multi-utente indipendente dalla piattaforma

e lo studio ed implementazione in un'unica soluzione (facilmente espandibile ed aggiornabile) dei paradigmi più efficaci di interazione e di awareness multi-utente in AR.

Per quanto riguarda il primo contributo, è stato molto valido per la maggior parte del tempo di sviluppo di questo lavoro, ma allo stato attuale, con il rilascio di alcuni potenti tool come ARFoundation, la sua rilevanza sul panorama generale dell'AR è venuta meno. Nonostante ciò può comunque essere considerata come una valida alternativa alle soluzioni più commerciali e potrebbe comunque ancora essere definito come un contributo, magari non così significativo, nell'ambiente AR multi-piattaforma

7.2 Sviluppi futuri

Gli eventuali sviluppi futuri che potrebbero arricchire ulteriormente questo lavoro sono molteplici. Innanzi tutto si potrebbe espandere il protocollo sperimentale includendo le sessioni di testing in remoto, rimosse dall'attuale protocollo per altrimenti eccessiva complessità.

Oltre a questo si potrebbe pensare di includere un sistema di interfacce tangibili per quanto riguarda la definizione dei paradigmi di interazione collaborativa in realtà aumentata, valutandone la fattibilità per dispositivi *hand-held*; come si era dedotto dallo studio dello stato dell'arte, tali tipi di interazioni tangibili potrebbero risultare un notevole salto in avanti per quanto riguarda l'immersione e le possibilità di interazione.

Sicuramente un possibile sviluppo futuro è l'implementazione delle interfacce delle maggiori librerie di AR disponibili. In particolare un contributo particolarmente significativo consisterebbe nell'implementazione di un interfaccia per dispositivi di tipo *head-mounted-display*. Tale implementazione porterebbe ad esplorare e a definire un nuovo sistema di interazioni tra HHD ed HMD.

Appendice A

Questionario conoscitivo

Questionario preliminare conoscitivo da eseguire prima dell'esecuzione dei test

A.1 Informazioni personali

Per favore indicare di seguito le proprie informazioni personali

1. Inserire identificativo utente (Risposta breve precompilata)
2. Inserire nome utente (Risposta breve)
3. Indicare il proprio range d'età
4. Indicare il proprio sesso

A.2 Esperienze pregresse

Dichiarare se si è d'accordo o meno con la seguente affermazione indicando un numero su una scala da 1 (totalmente in disaccordo) a 5 (totalmente d'accordo)

5. Uso abitualmente lo smartphone e ho familiarità con l'utilizzo di app

Rispondere alle seguenti domande indicando con che frequenza si eseguono alcune attività (mai, quasi mai, qualche volta, spesso, molto spesso)

6. Quanto spesso faccio uso di app in realtà aumentata?

7. Quanto spesso faccio uso di videogiochi?
8. Quanto spesso faccio uso di videogiochi multi-giocatore collaborativi?
9. Quanto spesso effettui esperienze collaborative di tipo “escape room”?

Appendice B

Questionari interazione collaborativa basata sullo sguardo

Di seguito una serie di domande sulla comprensibilità e utilità delle varie informazioni visive fornite

B.1 Visualizzazione base

B.1.1 Visualizzazione base: comprensibilità

Dichiarare se si è d'accordo o meno con le seguenti affermazioni indicando un numero su una scala da 1 (totalmente in disaccordo) a 7 (totalmente d'accordo)

10. (Sguardo, Visualizzazione base) Ritengo che il numero delle informazioni mostrate a schermo sia appropriato

11. (Sguardo, Visualizzazione base) Ritengo che le informazioni mostrate a schermo siano difficili da leggere
interpretare

12. (Sguardo, Visualizzazione base) Ritengo che le informazioni mostrate a schermo siano confusionarie

B.1.2 Visualizzazione base: utilità delle informazioni visuali

Indicare se le informazioni fornite sono risultate utili oppure no selezionando un numero su una scala da 1 (per niente utile) a 5 (molto utile)

13. (Sguardo) Ai fini della collaborazione quanto è risultato utile visualizzare il cursore degli altri utenti?

14. (Sguardo) In generale quanto è risultato utile visualizzare il cursore degli altri utenti?

15. (Sguardo) Ai fini della collaborazione quanto è stato utile visualizzare la linea visuale degli altri utenti?

16. (Sguardo) Quanto è stato utile in generale poter visualizzare la linea visuale degli altri utenti?

B.2 Visualizzazione completa

B.2.1 Visualizzazione completa: comprensibilità

Dichiarare se si è d'accordo o meno con le seguenti affermazioni indicando un numero su una scala da 1 (totalmente in disaccordo) a 7 (totalmente d'accordo)

17. (Sguardo, Visualizzazione completa) Ritengo che il numero delle informazioni mostrate a schermo sia appropriato

18. (Sguardo, Visualizzazione completa) Ritengo che le informazioni mostrate a schermo siano difficili da leggere
interpretare

19. (Sguardo, Visualizzazione completa) Ritengo che le informazioni mostrate a schermo siano confusionarie

B.2.2 Visualizzazione completa: utilità delle informazioni visuali

Indicare se le informazioni fornite sono risultate utili oppure no selezionando un numero su una scala da 1 (per niente utile) a 5 (molto utile)

20. (Sguardo) Ai fini della collaborazione, quanto è stato utile poter impostare dei punti di interesse personalizzati?

21. (Sguardo) Quanto è stato utile in generale poter impostare dei punti di interesse personalizzati all'interno della simulazione?

B.3 Considerazioni generali su questo tipo di interazione

Indicare di seguito quale delle due combinazioni di visualizzazione si ha maggiormente apprezzato e rispondere alle domande successive riferendosi alla visualizzazione scelta

22. (Sguardo) In generale, quale delle seguenti combinazioni di informazioni visuali hai maggiormente apprezzato? (Visualizzazione base (avatar, linea visuale e cursore), Visualizzazione completa (avatar, linea visuale, cursore e punto di interesse))

23. (Sguardo) Ai fini della collaborazione ho trovato la presenza di informazioni visuali importanti per riuscire a portare a termine i task

24. (Sguardo) La presenza di informazioni visuali mi ha permesso di capire meglio le attività compiute dagli altri partecipanti

25. (Sguardo) In generale la presenza di informazioni visuali ha migliorato l'efficacia della collaborazione rispetto a non aver alcuna informazione

Appendice C

Questionari interazione collaborativa basata sulla manipolazione

Di seguito una serie di domande sulla comprensibilità e utilità delle varie informazioni visive fornite

C.1 Visualizzazione base

C.1.1 Visualizzazione base: comprensibilità

Dichiarare se si è d'accordo o meno con le seguenti affermazioni indicando un numero su una scala da 1 (totalmente in disaccordo) a 7 (totalmente d'accordo)

26. (Manipolazione, Visualizzazione base) Ritengo che il numero delle informazioni mostrate a schermo sia appropriato

27. (Manipolazione, Visualizzazione base) Ritengo che le informazioni mostrate a schermo siano difficili da leggere
interpretare

28. (Manipolazione, Visualizzazione base) Ritengo che le informazioni mostrate a schermo siano confusionarie

C.1.2 Visualizzazione base: utilità delle informazioni visuali

Indicare se le informazioni fornite sono risultate utili oppure no selezionando un numero su una scala da 1 (per niente utile) a 5 (molto utile)

29. (Manipolazione) Ai fini della collaborazione quanto è stato utile poter visualizzare la mano virtuale degli altri utenti?

30. (Manipolazione) Quanto è stato utile in generale poter visualizzare la mano virtuale degli altri utenti?

31. (Manipolazione) Ai fini della collaborazione quanto è stato utile il contorno colorato sugli oggetti manipolati dagli altri utenti?

32. (Manipolazione) In generale quanto è stato utile il contorno colorato sugli oggetti manipolati dagli altri utenti?

C.2 Visualizzazione completa

C.2.1 Visualizzazione completa: comprensibilità

Dichiarare se si è d'accordo o meno con le seguenti affermazioni indicando un numero su una scala da 1 (totalmente in disaccordo) a 7 (totalmente d'accordo)

33. (Manipolazione, Visualizzazione completa) Ritengo che il numero delle informazioni mostrate a schermo sia appropriato

34. (Manipolazione, Visualizzazione completa) Ritengo che le informazioni mostrate a schermo siano difficili da leggere
interpretare

35. (Manipolazione, Visualizzazione completa) Ritengo che le informazioni mostrate a schermo siano confusionarie

C.2.2 Visualizzazione completa: utilità delle informazioni visuali

Indicare se le informazioni fornite sono risultate utili oppure no selezionando un numero su una scala da 1 (per niente utile) a 5 (molto utile)

36. (Manipolazione) Ai fini della collaborazione, quanto è stato utile poter impostare dei punti di interesse personalizzati?

37. (Manipolazione) Quanto è stato utile in generale poter impostare dei punti di interesse personalizzati all'interno della simulazione?

C.3 Considerazioni generali su questo tipo di interazione

Indicare di seguito quale delle due combinazioni di visualizzazione si ha maggiormente apprezzato e rispondere alle domande successive riferendosi alla visualizzazione scelta

38. (Manipolazione) In generale, quale delle seguenti combinazioni di informazioni visuali hai maggiormente apprezzato? (Visualizzazione base (avatar, linea visuale e cursore), Visualizzazione completa (avatar, linea visuale, cursore e punto di interesse))

39. (Manipolazione) Ai fini della collaborazione ho trovato la presenza di informazioni visuali importanti per riuscire a portare a termine i task

40. (Manipolazione) La presenza di informazioni visuali mi ha permesso di capire meglio le attività compiute dagli altri partecipanti

41. (Manipolazione) In generale la presenza di informazioni visuali ha migliorato l'efficacia della collaborazione rispetto a non aver alcuna informazione

Appendice D

Questionari interazione collaborativa basata sulla posizione

Di seguito una serie di domande sulla comprensibilità e utilità delle varie informazioni visive fornite

D.1 Visualizzazione base

D.1.1 Visualizzazione base: comprensibilità

Dichiarare se si è d'accordo o meno con le seguenti affermazioni indicando un numero su una scala da 1 (totalmente in disaccordo) a 7 (totalmente d'accordo)

42. (Posizione, Visualizzazione base) Ritengo che il numero delle informazioni mostrate a schermo sia appropriato

43. (Posizione, Visualizzazione base) Ritengo che le informazioni mostrate a schermo siano difficili da leggere
interpretare

44. (Posizione, Visualizzazione base) Ritengo che le informazioni mostrate a schermo siano confusionarie

D.1.2 Visualizzazione base: utilità delle informazioni visuali

Indicare se le informazioni fornite sono risultate utili oppure no selezionando un numero su una scala da 1 (per niente utile) a 5 (molto utile)

45. (Posizione) Ai fini della collaborazione quanto è stato utile poter visualizzare l'avatar degli altri utenti?

46. (Posizione) Quanto è stato utile in generale poter visualizzare l'avatar degli altri utenti?

D.2 Visualizzazione completa

D.2.1 Visualizzazione completa: comprensibilità

Dichiarare se si è d'accordo o meno con le seguenti affermazioni indicando un numero su una scala da 1 (totalmente in disaccordo) a 7 (totalmente d'accordo)

47. (Posizione, Visualizzazione completa) Ritengo che il numero delle informazioni mostrate a schermo sia appropriato

48. (Posizione, Visualizzazione completa) Ritengo che le informazioni mostrate a schermo siano difficili da leggere
interpretare

49. (Posizione, Visualizzazione completa) Ritengo che le informazioni mostrate a schermo siano confusionarie

D.2.2 Visualizzazione completa: utilità delle informazioni visuali

Indicare se le informazioni fornite sono risultate utili oppure no selezionando un numero su una scala da 1 (per niente utile) a 5 (molto utile)

50. (Posizione) Ai fini della collaborazione, quanto è stato utile poter impostare dei punti di interesse personalizzati?

51. (Posizione) Quanto è stato utile in generale poter impostare dei punti di interesse personalizzati all'interno della simulazione?

D.3 Considerazioni generali su questo tipo di interazione

Indicare di seguito quale delle due combinazioni di visualizzazione si ha maggiormente apprezzato e rispondere alle domande successive riferendosi alla visualizzazione scelta

52. (Posizione) In generale, quale delle seguenti combinazioni di informazioni visuali hai maggiormente apprezzato? (Visualizzazione base (avatar, linea visuale e cursore), Visualizzazione completa (avatar, linea visuale, cursore e punto di interesse))

53. (Posizione) Ai fini della collaborazione ho trovato la presenza di informazioni visuali importanti per riuscire a portare a termine i task

54. (Posizione) La presenza di informazioni visuali mi ha permesso di capire meglio le attività compiute dagli altri partecipanti

55. (Posizione) In generale la presenza di informazioni visuali ha migliorato l'efficacia della collaborazione rispetto a non aver alcuna informazione

Appendice E

Questionario sulla comprensibilità in generale

Comprensibilità dell'esperienza

Dichiarare se si è d'accordo o meno con le seguenti affermazioni indicando un numero su una scala da 1 (totalmente in disaccordo) a 7 (totalmente d'accordo)

56. Ritengo che interagire con questa applicazione richieda parecchio sforzo mentale
57. Ritengo che le informazioni mostrate siano state sufficientemente reattive
58. Ritengo che i simboli e le parole presenti a schermo fossero facili da leggere
59. Ritengo che l'esperienza tremolava troppo, era instabile
60. Ritengo che le informazioni mostrate a schermo fossero consistenti

Appendice F

Questionario sulla manipolabilità in generale

Manipolabilià dell'eseprienza

Dichiarare se si è d'accordo o meno con le seguenti affermazioni indicando un numero su una scala da 1 (totalmente in disaccordo) a 7 (totalmente d'accordo)

61. Ritengo che l'interazione con questa applicazione richieda un grande sforzo muscolare
62. Ritengo che l'uso di questa applicazione sia comodo per mani e braccia
63. Ho trovato difficile impugnare il dispositivo durante l'esperienza
64. Ho trovato semplice fornire input all'applicazione
65. Ho sentito le mie braccia o mani affaticarsi dopo aver usato l'applicazione
66. Penso che l'applicazione sia semplice da controllare
67. Ad un certo punto mi è sembrato che il device mi scivolasse dalle mani e che potesse cadere a terra
68. Penso che utilizzare questa applicazione sia semplice e senza complicazioni

Appendice G

System usability scale

Dichiarare se si è d'accordo o meno con le seguenti affermazioni indicando un numero su una scala da 1 (totalmente in disaccordo) a 5 (totalmente d'accordo)

69. Mi piacerebbe utilizzare questo sistema frequentemente
70. Ho trovato questo sistema inutilmente complesso
71. Penso che questo sistema sia facile da usare
72. Penso che avrei bisogno del supporto di un tecnico per essere in grado di usare questo sistema
73. Penso che le varie funzioni di questo sistema siano ben integrate
74. Ho pensato ci fosse troppa incoerenza in questo sistema
75. Penso che la maggior parte delle persone imparerebbe ad usare questo sistema molto velocemente
76. Ho trovato questo sistema molto macchinoso da utilizzare
77. Ero molto sicuro di me nell'utilizzare questo sistema
78. Ho avuto bisogno di imparare molte cose prima di poter utilizzare questo sistema

Appendice H

Questionario sulla escape room virtuale

Di alcune domande relative all'esperienza di escape room virtuale che hai appena provato assieme ad alcune affermazioni per quali va indicato se si è d'accordo o meno su una scala da 1 (totalmente in disaccordo/poco utile) a 5 (totalmente d'accordo/molto utile)

79. Come valuteresti su una scala da 1 a 10 la tua esperienza con questo videogioco? (scala lineare da 1 a 10)

80. Se avevi la possibilità di usare il punto di interesse condiviso nel corso dell'esperienza di escape room, l'hai trovato utile?

81. Non avendo la possibilità di usare il punto di interesse condiviso nel corso dell'esperienza di escape room, ne ho sentito il bisogno

82. La presenza di informazioni visuali mi ha aiutato molto a portare a termine i compiti che richiedevano collaborazione

83. Gli oggetti di scena erano molto realistici ed hanno aumentato il mio senso di immersione

84. Ho trovato difficile portare a termine alcuni enigmi

85. Ho trovato difficile capire cosa avrei dovuto fare

86. Ho trovato difficile l'interazione con gli oggetti virtuali

87. Qual è stato il tuo scenario preferito? (Scena 1: apertura della scatola, Scena 2: enigma del piedistallo, Scena 3: enigma del quadro, Scena 4: enigma di traduzione)

Bibliografia

Articoli

[2] John Brooke. "SUS: A quick and dirty usability scale". In: *Usability Eval. Ind.* 189 (nov. 1995).

[3] M. Billinghurst, S. Weghorst e T. Furness III. "Shared Space: An Augmented Reality Approach for Computer Supported Collaborative Work". In: *Virtual Reality* 3.1 (mar. 1998), pp. 25–36. DOI: https://doi.org/10.1007/BF01409795.

[4] Roy S. Kalawsky. "VRUSE a computerised diagnostic tool: for usability evaluation of virtual/synthetic environment systems". In: *Applied Ergonomics* 30.1 (1999), pp. 11–25. ISSN: 0003-6870. DOI: https://doi.org/10.1016/S0003-6870(98)00047-7. URL: http://www.sciencedirect.com/science/article/pii/S0003687098000477.

[5] T. Kawashima et al. "Magic Paddle: A Tangible Augmented Reality Interface for Object Manipulation". In: *ISMR 2001* (2001), pp. 194–195.

[6] A. Lund. "Measuring Usability with the USE Questionnaire". In: *Usability and User Experience Newsletter of the STC Usability SIG* 8.1 (2001).

[8] M. Billinghurst e H. Kato. "Collaborative augmented reality". In: *Communication of the ACM* 45.7 (2002), pp. 64–70.

[10] M. Billinghurst et al. "Experiments with Face-To-face Collaborative AR Interfaces". In: *Virtual Reality* 6 (2002), pp. 107–121.

[11] N.R. Hedley et al. "Explorations in the use of Augmented Reality for Geographic Visualization". In: *Presence* 11.2 (apr. 2002), pp. 119–133. DOI: 10.1162/1054746021470577.

[12] H.T. Regenbrecht, M.T. Wagner e G. Baratoff. "Magic Meeting: A Collaborative Tangible Augmented Reality System". In: *Virtual Reality* 6.3 (ott. 2002), pp. 151–166. DOI: 10.1007/s100550200016.

[13] C. Ulbricht e D. Schmalstieg. "Tangible Augmented Reality for Computer Games". In: (apr. 2003).

[14] M. Billinghurst, R. Grasset e J. Looser. "Designing Augmented Reality Interfaces". In: *ACM SIGGRAPH Computer Graphics* 39.1 (2005), pp. 17–22. DOI: 10.1145/1057792.1057803.

[15] F. Biocca et al. "Attention Funnel: Omnidirectional 3D Cursor for Mobile Augmented Reality Platforms". In: *Proceedings of the 2006 Conference on Human Factors in Computing Systems* (apr. 2006). DOI: 10.1145/1124772.1124939.

[16] R. Sidharta, J. Oliver e A. Sannier. "Augmented Reality Tangible Interface for Distributed Design Review". In: *Third International Conference on Computer Graphics, Imaging and Visualization* (gen. 2006). DOI: 10.1109/CGIV.2006.25.

[17] J. Chastine et al. "Studies on the Effectiveness of Virtual Pointers in Collaborative Augmented Reality". In: *2008 IEEE Symposium on 3D User Interfaces* (mar. 2008). DOI: 10.1109/3DUI.2008.4476601.

[18] T. Peck, H. Fuchs e M. Whitton. "Evaluation of Reorientation Techniques and Distractors for Walking in Large Virtual Environments". In: *IEEE Transactions on Visualization and Computer Graphics* 15.3 (mar. 2009), pp. 383–394. DOI: 10.1109/TVCG.2008.191.

[19] D. Ta Huynh et al. "Art of Defense: A Collaborative Handheld Augmented Reality Board Game". In: *Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games* (ago. 2009), pp. 135–142. DOI: 10.1145/1581073.1581095.

[20] Wang Xiangyu e P.S. Dunston. "Comparative Effectiveness of Mixed Reality Based Virtual Environments in Collaborative Design". In: *IEEE International Conference on Systems, Man and Cybernetics* (ott. 2009). DOI: 10.1109/ICSMC.2009.5346691.

[21] E. Prytz, S. Nilsson e A. Jonsson. "The Importance of Eye-contact for Collaboration in AR Systems". In: *IEEE International Symposium on Mixed and Augmented Reality* (ott. 2010). DOI: 10.1109/ISMAR.2010.5643559.

[22] J. Gu, N. Li e H. Been-Lirn Duh. "A Remote Mobile Collaborative AR System for Learning in Physics". In: *IEEE Virtual Reality* (mar. 2011). DOI: 10.1109/VR.2011.5759496.

[23] T. Piumsomboon et al. "Poster: Physically-Based Natural Hand and Tangible AR Interaction for Face-To-Face Collaboration on a Tabletop". In: *2012 IEEE Symposium on 3D User Interfaces (3DUI)* (mar. 2012). DOI: 10.1109/3DUI.2012.6184208.

[27] D. Jo, K. Kim e G.J. Kim. "Avatar Motion Adaptation for AR based 3D Tele-Conference". In: *2014 International Workshop on Collaborative Virtual Environments (3DCVE)* (mar. 2014). DOI: 10.1109/3DCVE.2014.7160932.

[29] A. S. Lee et al. "3D Collaboration Method over HoloLens™ and Skype™ End Points". In: *The 3rd International Workshop* (ott. 2015). DOI: 10.1145/2814347.2814350.

[30] C. Yusof et al. "A Review of 3D gesture interaction for handheld augmented reality". In: *Jurnal Teknologi* 78.2 (dic. 2015), pp. 15–20. DOI: 10.11113/jt.v78.6923.

[31] P. Boonbrahm, C. Kaewrat e S. Boonbrahm. "Interactive Augmented Reality: A New Approach for Collaborative Learning". In: *International Conference on Learning and Collaboration Technologies* (giu. 2016), pp. 115–124. DOI: https://doi.org/10.1007/978-3-319-39483-1_11.

[32] P. Boonbrahm, C Kaewrat e S. Boonbrahm. "Interactive Augmented Reality: A New Approach for Collaborative Learning". In: *International Conference on Learning and Collaboration Technologies* (lug. 2016), pp. 115–124. DOI: 10.1007/978-3-319-39483-1_11.

[34] K. Lien et al. "PPV: Pixel-Point-Volume Segmentation for Object Referencing in Collaborative Augmented Reality". In: *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)* (set. 2016). DOI: 10.1109/ISMAR.2016.21.

[36] S. Orts-Escolano et al. "Holoportation: Virtual 3D Teleportation in Real-time". In: *ACM Symposium on User Interface Software and Technology* (ott. 2016). DOI: 10.1145/2984511.2984517.

[37] C. Brown et al. "Coordinating Attention and Cooperation in Multi-user Virtual Reality Narratives". In: *2017 IEEE Virtual Reality (VR)* (mar. 2017). DOI: 10.1109/VR.2017.7892334.

[40] I. Mavridou et al. “FACETEQ Interface Demo for Emotion Expression in VR”. In: *2017 IEEE Virtual Reality (VR)* (mar. 2017). DOI: 10.1109/VR.2017.7892369.

[41] T. Piumsomboon et al. “CoVAR: A Collaborative Virtual and Augmented Reality System for Remote Collaboration”. In: *SIGGRAPH Asia 2017 Emerging Technologies* (nov. 2017). DOI: 10.1145/3132818.3132822.

[42] T. Piumsomboon et al. “Exploring Enhancements for Remote Mixed Reality Collaboration”. In: *SIGGRAPH Asia 2017 Mobile Graphics & Interactive Applications* (nov. 2017). DOI: 10.1145/3132787.3139200.

[44] S. Gunther et al. “CheckMate: Exploring a Tangible Augmented Reality Interface for Remote Interaction”. In: *Extended Abstracts of the 2018 CHI Conference* (apr. 2018). DOI: 10.1145/3170427.3188647.

[45] Figen Gül L. “Studying gesture-based interaction on a mobile augmented reality application for co-design activity”. In: *Journal on Multimodal User Interfaces* 12.2 (giu. 2018), pp. 109–124. DOI: 10.1007/s12193-017-0252-0.

[46] T. Piumsomboon et al. “Mini-Me: An Adaptive Avatar for Mixed Reality Remote Collaboration”. In: *The 2018 CHI Conference* (apr. 2018). DOI: 10.1145/3173574.3173620.

Conferenze

[24] R. Budhiraja, G. A. Lee e M. Billinghurst. “Using a HHD with a HMD for Mobile AR Interaction”. In: *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. Ott. 2013. DOI: 10.1109/ismar.2013.6671837.

[25] S.A. Chowdhury et al. “Handheld Augmented Reality Interaction Technique”. In: *International Visual Informatics Conference*. Nov. 2013. DOI: 10.1007/978-3-319-02958-0_38.

[26] S. Chowdhury et al. “Handheld Augmented Reality Interaction Technique”. In: *International Visual Informatics Conference 2013: Advances in Visual Informatics*. Nov. 2013, pp. 418–426. DOI: 10.1007/978-3-319-02958-0_38.

[28] Marc Ericson Santos et al. “A Usability Scale for Handheld Augmented Reality”. In: nov. 2014. DOI: 10.1145/2671015.2671019.

[33] A. Golombek, M. Lankes e J. Hagler. "Vancouver Maneuver: Designing a Cooperative Augmented Reality Board Game". In: *ICEC 2016: Entertainment Computing*. Set. 2016, pp. 286–289. DOI: 10.1007/978-3-319-46100-7_31.

[35] Baron N. "CollaborativeConstraint: UI for collaborative 3D Manipulation Operations". In: *2016 IEEE Symposium on 3D User Interfaces (3DUI)*. Mar. 2016. DOI: 10.1109/3DUI.2016.7460076.

[38] J.G. Grandi et al. "Collaborative Manipulation of 3D Virtual Objects in Augmented Reality Scenarios using Mobile Devices". In: *IEEE Symposium on 3D User Interfaces (3DUI)*. Mar. 2017, pp. 18–19.

[39] Grandi J.G. "Design of Collaborative 3D User Interfaces for Virtual and Augmented Reality". In: *IEEE Virtual Reality (VR)*. Mar. 2017, pp. 18–22.

[43] J.G. Grandi et al. "Design and Assessment of a Collaborative 3D Interaction Technique for Handheld Augmented Reality". In: *IEEE Conference on Virtual Reality and 3D User Interfaces*. Mar. 2018, pp. 49–56. DOI: 10.1109/vr.2018.8446295.

[62] I. Poupyrev et al. "Tiles: A Mixed Reality Authoring Interface". In: *INTERACT 2001 Conference on Human Computer Interaction*.

Libri

[1] Sandra G. Hart e Lowell E. Staveland. "Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research". In: *Human Mental Workload*. A cura di Peter A. Hancock e Najmedin Meshkati. Vol. 52. Advances in Psychology. North-Holland, 1988, pp. 139–183. DOI: https://doi.org/10.1016/S0166-4115(08)62386-9. URL: http://www.sciencedirect.com/science/article/pii/S0166411508623869.

Siti

[47] Apple. *ARKit Overview*. URL: https://developer.apple.com/documentation/arkit. (ultimo accesso: 3 luglio 2019).

[48] Apple. *Get ready for the latest advances in augmented reality.* URL: https://developer.apple.com/augmented-reality/. (ultimo accesso: 3 luglio 2019).

[49] Engel J. e Cremers D. *LSD-SLAM: Large-Scale Direct Monocular SLAM.* URL: https://vision.in.tum.de/research/vslam/lsdslam. (ultimo accesso: 4 luglio 2019).

[50] Rumen F. *Multi-AR Examples for AR-Core, AR-Kit and HoloLens.* URL: https://rfilkov.com/2017/10/04/multi-ar/. (ultima modifica: 4 ottobre 2017).

[51] Facebook. *Facebook Spaces.* URL: https://www.facebook.com/spaces/. (ultimo accesso: 24 giugno 2019).

[52] Google. *ARCore overview.* URL: https://developers.google.com/ar/discover/. (ultima modifica: 28 febbraio 2019).

[53] Google. *Shared AR Experiences with Cloud Anchors.* URL: https://developers.google.com/ar/develop/java/cloud-anchors/overview-android. (ultima modifica: 7 maggio 2019).

[54] Nesterenco I. *Realistic reflections and environment textures in AR-Kit 2.0.* URL: https://medium.com/@ivannesterenko/realistic-reflections-and-environment-textures-in-arkit-2-0-d8d0f1332eed. (ultima modifica: 13 giugno 2018).

[55] Kunze K. *Affective Wear: Recognizing facial expressions.* URL: http://kaikunze.de/2015/08/18/-affective-wear/. (ultima modifica: 18 agosto 2015).

[56] Kudan. *Kudan Main Page.* URL: https://www.kudan.io. (ultimo accesso: 4 luglio 2019).

[57] Miesnieks M. *Dawn of the AR Cloud.* URL: https://medium.com/6d-ai/dawn-of-the-ar-cloud-1b31eb4b52ac. (ultima modifica: 29 maggio 2018).

[58] Miesnieks M. *Introducing 6D.ai and Our Master Plan.* URL: https://medium.com/6d-ai/introducing-6d-ai-and-our-master-plan-a39b58ce58e8. (ultima modifica: 29 marzo 2018).

[59] MAXST. *MAXST Main Page.* URL: http://www.maxst.com. (ultimo accesso: 4 luglio 2019).

[60] Microsoft. *AltspaceVR: The Social VR App.* URL: https://altvr.com/. (ultimo accesso: 24 giugno 2019).

[61] Microsoft. *Microsoft HoloLens: Skype*. URL: https://www.youtube.com/watch?v=4QiGYtd3qNI. (ultima modifica: 29 febbraio 2016).

[63] PTC. *Vuforia Overview*. URL: https://library.vuforia.com/getting-started/overview.html. (ultimo accesso: 3 luglio 2019).

[64] Mur-Artal R. *ORB-SLAM2*. URL: https://github.com/raulmur/ORB_SLAM2. (ultima modifica: 13 gennaio 2017).

[65] Lukasz Pasek (Unity Technologies). *Unite Berlin 2018 - Rendering Techniques for Augmented Reality*. URL: https://www.youtube.com/watch?v=jD1gBuVRtZs&t=. (ultimo accesso: 7 dicembre 2019).

[66] Unity. *About AR Foundation*. URL: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@1.0/manual/index.html. (ultimo accesso: 3 luglio 2019).

[67] Wikitude. *Wikitude Main Page*. URL: https://www.wikitude.com. (ultimo accesso: 4 luglio 2019).