Politecnico di Torino

Corso di Laurea Magistrale in
Ingegneria Informatica (Computer Engineering)
Orientamento Grafico e Multimediale

Tesi di Laurea Magistrale

Augmented Reality to support television productions

Realtà aumentata per supportare produzioni televisive

Candidato
Emanuele Maria Munacò

Relatore accademico
Andrea Sanna

Relatori aziendali
Roberto Iacoviello
Davide Zappia

a.a. 2018-2019

Augmented Reality to support television productions

Emanuele Maria Munacò

Abstract

This thesis was developed thanks to the collaboration between **DAUIN** - Department of Automation and Computer Science - of the **Polytechnic of Turin** and **Rai CRITS** - Research, Technological Innovation and Experimentation Center - in the context of an internship at Rai CRITS.

The reference scenario of the thesis is the professional television studio, in particular we want to optimize the creation of a set with the help of new augmented reality devices on the market.

The purpose of the thesis is to design a software on a **Microsoft HoloLens** device capable of assisting the work of the set designer and art director. For this purpose, the help of directors and set designers was requested, in order to meet their needs and understand the motivations and requirements for the development of this innovative application.

The state of the art was initially carried out on **new technologies of extended reality** (**XR**), on their functioning and on their applications in real contexts, focusing attention on the applications of mixed reality realized in the context of television production or with a similar purpose, so such as to create an innovative application that is valid for this type of context but can also be used in other areas.

The project for Microsoft HoloLens has been divided into two parts. This part, in turn, consists of two applications: the first part implements the backend side of the application and part of the frontend side. A desktop application manages the **remote set database** of television sets, and a second application for Microsoft HoloLens **creates virtual TV sets in augmented reality**, searching models in the database by applying certain filters, managing the instantiated models on stage and managing the scenes saved in the database.

The process of creating and loading models in the desktop application is divided into several phases:

1. The real props are previously scanned with the Rai CRITS **3D laser scanner**, in order to recreate the 3D model in a digital format.

2. Models insertions into the database is carried out through a desktop application created on Unity and implemented ad-hoc which, thanks to the ***TriLib*** plugin, overcomes Unity limitations, recovers the model, locally on the PC or remotely via the URL of the object, and instantiates the object in the scene in real-time.

3. Once the object is instantiated, it is possible to set a series of parameters, such as **the description and the tags** (type of object, color and material), useful for finding the model in the application for Microsoft HoloLens. The tags that can be set in this phase are previously inserted in the database.

4. You can create a **preview image** of the model, to allow the user to view the object that he wants to stage. The model is shot from a virtual camera in the application that can be moved in the environment, in order to adjust the frame. The model can also be rotated, in case it has not already been adjusted before loading it.

5. During the **upload**, the preview image is encoded in a base64 string, then the model data is saved into the database and the 3D model is loaded into the server.

The application for Microsoft HoloLens aims to **create and manage virtual sets** using the models previously inserted in the database. It has several features:

- Interaction within the application is done via **gestures** or **speech recognition**. Both have been implemented to take advantage of the two types of inputs. Gestures can be difficult to learn for beginners with Microsoft HoloLens, while speech recognition is easier to use, but could be problematic in case of noisy environments. For this you can enable and disable it using the appropriate menu.

- To **search for a model** filters are applied with keywords, in the name or description, or with the selection of tags. The results are shown in a grid and, once the model has been selected, it will be instantiated asynchronously, in order to impact as little as possible on the application's performance. Real-time streaming of the model from the database to the application could be potentially carried out through **the new 5G technology**.

- The **manipulation of the model** is carried out via grab or bounding box, a grid that encloses the object and able to rotate and scale the model along the axes.

- It is possible to **modify the model** within the application, changing its name, description, tags and rotation and default scaling. The current model can be edited or a copy of the model can be created, that is a child model that depends on the parent model.

- Finally, it is possible to **save, load and delete scenes** using the appropriate menu, to give the possibility to pause the virtual set-up work, to return to work on it later.

The second part of the project is not treated in this context because it is the subject of another thesis. It implements the frontend side, and therefore the spatial mapping, for identifying surfaces where the models can be placed, and the sharing part, for the multi-user sharing of the application, with the help of world anchors.

The two projects were developed in parallel and in the future there will be a merge for the specific use of setting up the TV set of ***Rob-O-Cod***, a program broadcast on Rai Gulp in which Rai CRITS also collaborated. The concept of this thesis is not limited to the design of television sets, but its versatility allows it to fit into a much larger context, the **5G-TOURS**, an European project with the aim of making 5G a key tool for cities of the future.

Realtà aumentata per supportare produzioni televisive

Emanuele Maria Munacò

Sommario

Questa tesi è stata realizzata grazie alla collaborazione tra **DAUIN** - Dipartimento di Automatica e Informatica - del **Politecnico di Torino** e **Rai CRITS** - Centro Ricerche, Innovazione Tecnologica e Sperimentazione - nel contesto di un tirocinio svolto presso quest'ultima.

Lo scenario di riferimento della tesi è lo studio professionale televisivo, in particolare si vuole ottimizzare la creazione di una scenografia con l'aiuto dei nuovi dispositivi di realtà aumentata presenti sul mercato.

Scopo della tesi è quello di progettare un software sul dispositivo **Microsoft HoloLens** capace di coadiuvare il lavoro dello scenografo e del direttore artistico. A questo scopo, è stato richiesto l'aiuto di registi e scenografi, in modo da venire incontro alle loro esigenze e capire le motivazioni ed i requisiti per lo sviluppo di questa innovativa applicazione.

È stato inizialmente effettuato lo stato dell'arte sulle **nuove tecnologie di realtà estesa** (**XR**), sul loro funzionamento e sulle loro applicazioni in contesti reali, focalizzando l'attenzione sulle applicazioni di realtà mixata realizzate nel contesto della produzione televisiva o con uno scopo simile, in modo tale da realizzare un'applicazione innovativa valida per questo tipo di contesto ma utilizzabile anche in altri ambiti.

Il progetto per Microsoft HoloLens è stato suddiviso in due parti: la prima parte implementa il lato backend dell'applicazione e parte di quello frontend. Questa parte, a sua volta, è composta da due applicazioni: un'applicazione desktop gestisce il **database di modelli in remoto** del set televisivo, e una seconda applicazione per Microsoft HoloLens viene usata per **creare i set televisivi virtuali in realtà aumentata**, ricercando i modelli nel database applicando determinati filtri, per gestire i modelli istanziati in scena e per gestire le scene salvate nel database.

Il processo di creazione e caricamento dei modelli nell'applicazione desktop si suddivide in più fasi:

1. Gli oggetti di scena reali vengono precedentemente scansionati con lo **scanner laser 3D** di Rai CRITS, in modo da ricreare il modello 3D in formato digitale.

2. L'inserimento dei modelli nel database avviene attraverso l'applicazione desktop realizzata su Unity ed implementata ad-hoc che, grazie al plugin ***TriLib***, supera le limitazioni di Unity, recupera il modello, in locale nel PC o in remoto tramite l'URL dell'oggetto, e istanzia l'oggetto nella scena in tempo reale.

3. Una volta istanziato l'oggetto, è possibile settare una serie di parametri, come **la descrizione e i tag** (tipo di oggetto, colore e materiale), utili alla ricerca del modello nell'applicazione per Microsoft HoloLens. I tag settabili in questa fase vengono inseriti precedentemente nel database.

4. È possibile creare un'**immagine di anteprima** del modello, per consentire all'utente di visualizzare l'oggetto che si vuole mettere in scena. Il modello viene ripreso da una camera virtuale nell'applicazione che è possibile muovere nell'ambiente, in modo tale da aggiustare l'inquadratura. Il modello può anche essere ruotato, nel caso in cui non sia stato già aggiustato prima di caricarlo.

5. Quando viene effettuato l'**upload**, l'immagine di anteprima viene codificata in una stringa in base64, quindi vengono salvati i dati del modello nel database e il modello 3D viene caricato nel server.

L'applicazione per Microsoft HoloLens ha lo scopo di **creare e gestire dei set virtuali** utilizzando i modelli inseriti in precedenza all'interno del database. Presenta diverse caratteristiche:

- L'interazione all'interno dell'applicazione avviene tramite **gestures** o **speech recognition**. Sono stati implementati entrambi in modo da sfruttare i vantaggi dei due tipi di input. Le gestures possono rivelarsi difficili da imparare per chi è alle prime armi con Microsoft HoloLens, mentre la speech recognition è più semplice da utilizzare, ma potrebbe risultare problematica in caso di ambienti rumorosi. Per questo è possibile abilitarla e disabilitarla tramite l'apposito menu.

- Per **ricercare un modello** vengono applicati dei filtri con parole chiave, presenti nel nome o nella descrizione, o con la selezione di tag. I risultati vengono mostrati in una griglia e, una volta selezionato il modello, questo viene istanziato in modo asincrono, in modo tale da impattare il meno possibile sulle prestazioni dell'applicazione. Lo streaming in tempo reale del modello dal database all'applicazione potrebbe essere potenzialmente effettuata attraverso la **nuova tecnologia del 5G**.

- La **manipolazione del modello** avviene tramite grab o bounding box, una griglia che racchiude l'oggetto in grado di ruotare e scalare il modello lungo gli assi.

- È possibile **modificare il modello** all'interno dell'applicazione, cambiandone nome, descrizione, tag e rotazione e scalamento di default. Può essere modificato il modello attuale oppure può essere creata una copia del modello, ovvero un **modello derivato** che dipende dal modello principale.

- È infine possibile **salvare, caricare ed eliminare scene** tramite l'apposito menu, per dare la possibilità di mettere in pausa il lavoro di allestimento del set virtuale, per tornare a lavorarci in un secondo momento.

La seconda parte del progetto non viene trattata in questo contesto in quanto oggetto di un'altra tesi. Implementa il lato frontend, e quindi lo spatial mapping, per l'individuazione di superfici sulle quali posizionare i modelli, e la parte di sharing, per la condivisione multiutente dell'applicazione, con l'aiuto delle world anchor.

I due progetti sono stati svolti in parallelo e in futuro è previsto un loro merge per l'uso specifico di allestimento del set televisivo di ***Rob-O-Cod***, un programma in onda su Rai Gulp in cui ha collaborato anche Rai CRITS. Il progetto di questa tesi non si limita alla progettazione di set televisivi, ma la sua versatilità gli permette di inserirsi in un contesto molto più grande, il **5G-TOURS**, un progetto a livello europeo con lo scopo di fare del 5G uno strumento chiave per le città del futuro.

Dediche

Vorrei innanzitutto ringraziare il mio relatore accademico, il Prof. Andrea Sanna, per la sua disponibilità, la sua guida e per avermi dato l'opportunità di lavorare a questo progetto.

La tesi è stata svolta nel contesto di un tirocinio presso il RAI CRITS. Vorrei ringraziare tutto il team di RAI CRITS, che in questi mesi di tirocinio mi ha fatto sentire a casa, in particolare i miei relatori aziendali Roberto Iacoviello e Davide Zappia, che mi hanno accompagnato e aiutato per tutta la durata del percorso.

Infine, vorrei ringraziare la mia famiglia e i miei amici, di Parma, di Torino e non solo, per avermi supportato e sopportato, in particolare i miei genitori, mio fratello Giorgio ed Elena Trovato, valida compagna di progetti che mi ha accompagnato durante l'intero percorso di studi al Politecnico.

Indice

Capitolo 1

Introduzione, Obiettivi e Strumenti

Questo capitolo è una sorta di preambolo agli argomenti trattati in questa tesi.

Inizialmente verrà analizzato il contesto (1.1), e quindi il mondo del cinema e della televisione, dando una breve occhiata alle loro origini (1.1.1) e al processo di produzione tipico di un programma televisivo (1.1.2), analizzando uno scenario nel dettaglio come quello di **Rob-O-Cod** (1.1.3).

1.1 Contesto, motivazioni ed obiettivi

1.1.1 Il cinema e la televisione

Negli ultimi anni, il cinema e la televisione sono entrate nella vita della persone, diventando elementi chiave fondamentali nell'ambito dell'intrattenimento, dell'informazione, della narrazione di contenuti, cambiando completamente la percezione delle persone del mondo circostante.

La prima proiezione di una pellicola al pubblico avvenne nel 1895 grazie ad un'invenzione dei fratelli Louis e Auguste Lumière, il cinématographe (fig. 1.1), un apparecchio in grado di proiettare su uno schermo bianco una sequenza di immagini distinte, impresse su una pellicola stampata con un processo fotografico, in modo da creare l'effetto del movimento. Il cinema diverrà il primo mass-media al centro della rivoluzione tecnologica, ma non sarà l'unico.

Figura 1.1: Il cinématographe dei fratelli Lumière.

Il primo prototipo di televisione elettromeccanica (fig. 1.2) è stato costruito da John Logie Baird nel 1925 e presentato al pubblico nel 1926. A seguito di questo prototipo, sono state realizzate televisioni elettroniche, ma al giorno d'oggi sono in uso televisioni digitali.

Figura 1.2: La prima televisione elettromeccanica di John Logie Baird.

Grazie a queste invenzioni, il cinema e la televisione sono state delle vere innovazioni al centro della rivoluzione tecnologica che ha dato il via ad una grande industria, entrando di anno in anno sempre di più nelle case delle persone, diventando parte della vita di tutti i giorni.

1.1.2 Il processo di produzione di un programma televisivo

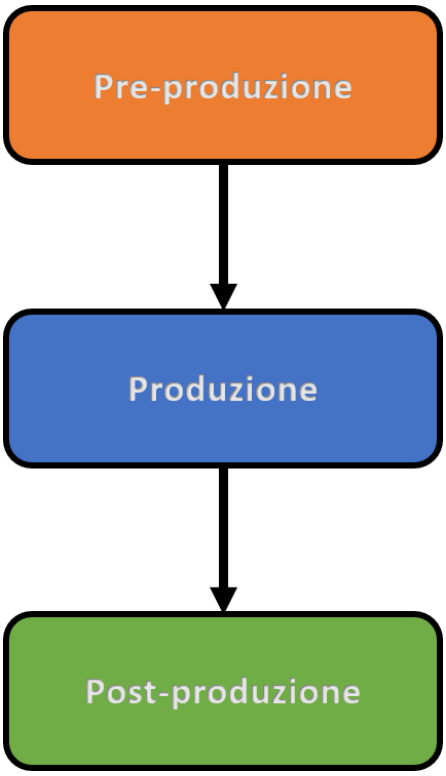

Figura 1.3: Il workflow di una generica produzione televisiva.

Il processo di produzione televisivo non ha un workflow ben definito, in quanto si tratta di un processo creativo, ma è suddiviso generalmente in tre fasi distinte (fig. 1.3):

- **Pre-produzione:** in questa fase, l'autore del programma sviluppa l'idea, il concept del programma, e discute con i registi delle scelte di produzione (fig. 1.4). Nella fase di **script**, vengono scritte tutte le informazioni riguardo la storia, i personaggi e gli eventi che accadono. Successivamente, la fase di **sketch** fornisce una prima idea visiva di come la scena comparirà sullo schermo. La stesura di uno **storyboard** aiuta a visualizzare le transizioni tra le immagini. Si tratta infatti di una serie di frames chiave che rappresentano le scene di una sequenza video, in modo simile ad un fumetto. Fornisce un'idea visiva più concreta delle intenzioni dell'autore agli altri componenti del team, per la **realizzazione del set**. Gli scenografi si curano della realizzazione della scenografia, in modo tale da fare le scelte più opportune per rispettare le esigenze del programma. Certe scelte sul set possono rivelarsi di difficile realizzazione o costose, oppure possono occludere la vista del pubblico in studio o dello spettatore da casa, altre ancora possono essere inopportune ed incoerenti con il tipo di format che si vuole trasmettere. Altre operazioni preparatorie del set televisivo sono ad esempio la gestione dell'illuminazione e degli impianti audio. La pre-produzione è una fase preparatoria alla produzione vera e propria.

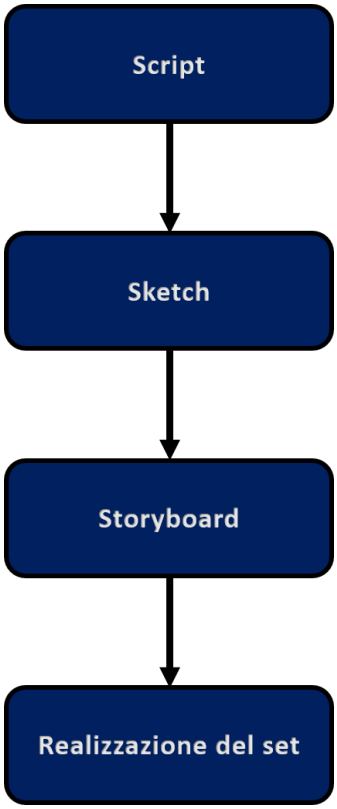

Figura 1.4: Esempio di workflow della fase di pre-produzione..

- **Produzione:** per produzione si intende quel processo che si occupa della messa in scena del programma, sulla base delle scelte effettuate in pre-produzione, il tutto sul set precedentemente allestito. In questa fase, viene registrato il programma vero e proprio, con il presentatore, gli ospiti e il pubblico in studio. Le scelte effettuate nella fase precedente non devono avere ripercussioni negative sul lavoro dei cameramen e dei tecnici, come occlusioni della vista della telecamera.

- **Post-produzione:** una volta registrato il programma, si elabora il prodotto e si monta il video finale prima dell'effettiva messa in onda. Questa è una fase non sempre presente, ad esempio nel caso di dirette televisive.

La produzione televisiva è stata influenzata, nel corso degli anni, dai progressi della tecnologia. La preparazione di programmi televisivi diventa sempre più complessa per un pubblico sempre più esigente. Diventa quindi necessario trovare nuovi mezzi per facilitare il lavoro di registi e scenografi durante la fase di pre-produzione.

1.1.3 Uno scenario in particolare: Rob-O-Cod

Figura 1.5: Logo di *Rob-O-Cod*.

L'applicazione per l'allestimento di set virtuali *Set Builder*, approfondita nel capitolo 4, è stata inizialmente pensata per collaborare alla realizzazione del set televisivo di una trasmissione in particolare: si tratta di **Rob-O-Cod** [1], un programma per ragazzi in collaborazione con **Rai CRITS**, **Rai Gulp** e **Rai Ragazzi** (fig. 1.5), presentato da Matteo Sintucci, Silvia Lavarini e Valeria Cagnina, con la regia di Paolo Severini.

La trasmissione ha come obiettivo quello di avvicinare i ragazzi al mondo della programmazione, un mondo che, al giorno d'oggi, è in continua crescita, e in futuro diverrà una competenza indispensabile in qualsiasi ambito della vita di tutti i giorni.

Il programma è diviso in stagioni. Ogni stagione vede come protagonisti i *Robocoder*, ragazzi tra i 10 e i 15 anni, che si sfidano a squadre composte da due ragazzi per vincere una gara che si svolge per tutta la durata della stagione.

La gara è suddivisa in più gironi da 4, e ad ogni puntata si sfidano due squadre di un girone. Le sfide di ogni puntata consistono nel programmare il proprio robot così da fargli superare ostacoli seguendo determinati percorsi e facendogli svolgere alcune azioni, come colpire dei nemici, spingere oggetti, etc. Il robot dovrà quindi muoversi su dei tavoli esagonali sui quali sono presenti gli ostacoli della sfida. I tavoli sono collegati tra loro da ponti, rampe o altri oggetti. Al termine di ogni sfida, le due squadre riceveranno un punteggio, i *bit-point*: la squadra che a fine puntata avrà il maggior numero di bit-point, sarà la vincitrice della puntata e avrà superato il girone. Nell'ultima puntata della stagione si disputerà la finale che decreterà la squadra vincitrice di tutta la stagione.

Il programma si suddivide in più fasi:

- All'inizio vengono presentati i componenti della squadra attraverso un video di presentazione.
- Inizia la prima sfida: viene illustrato ai robocoder lo scenario, e quindi la sfida che devono affrontare (fig. 1.6).
- Code-on: in questa fase, i robocoder devono programmare i loro robot e testare l'efficacia del codice scritto, affinché svolga i compiti richiesti dalla sfida.
- Code-off, valutazione ed assegnazione del punteggio: finito il tempo per effettuare i test, si verifica il corretto funzionamento del codice definitivo programmato dalle due squadre nella fase di code-on. I robocoder azionano i loro robot, che vengono valutati sulla base del corretto svolgimento degli obiettivi della sfida e dal numero di errori. Verrà quindi assegnato un punteggio corrispondente alla valutazione ricevuta.
- Durante la trasmissione, Valeria Cagnina racconta delle "pills" sul coding, per avvicinare i ragazzi al mondo della programmazione.

Figura 1.6: Estratto di una puntata di *Rob-O-Cod*. Due squadre stanno testando il funzionamento del proprio codice.

I ragazzi, per programmare, non utilizzano linguaggi di programmazione sofisticati come il C++ o il C#, ma usano un linguaggio a blocchi ad alto livello, realizzato ad-hoc da *Lego*, che impartisce comandi elementari ai robot, come ad esempio traslazioni e rotazioni lungo gli assi o altre azioni base utili per lo svolgimento delle sfide.

L'applicazione per HoloLens pensata per questa trasmissione ha come obiettivo l'allestimento del set dei tavoli esagonali sui quali si svolgono le sfide. Il progetto è stato diviso in due sotto-progetti, che in futuro verranno uniti in un progetto unico. In questa tesi è stata realizzata la parte che implementa tutto il lato backend e parte di quello frontend, ma in futuro, dopo aver effettuato il merge delle due applicazioni, saranno aggiunte la parte di scansione degli ambienti reali e di sharing.

- L'applicazione **Asset Loader** (trattata nel capitolo 3), sviluppata per Unity, si occupa dell'inserimento dei modelli e delle loro caratteristiche nel database. Verrà quindi utilizzata da chi si occuperà di gestire i modelli nel database.

- L'applicazione **Set Builder** (trattata nel capitolo 4), sviluppata su Unity per Microsoft HoloLens, viene utilizzata dagli scenografi per allestire il set virtuale sui tavoli esagonali (fig. 1.7, 1.8), ricercando i modelli nel database applicando determinati filtri, per gestire i modelli istanziati in scena e per gestire le scene salvate nel database.

- L'applicazione non trattata in questa tesi si concentra sulla parte frontend. Scansiona ed analizza l'ambiente, individua i tavoli di forma esagonale e consente allo scenografo il posizionamento dei modelli sui ripiani individuati nella fase di scansione tramite l'uso delle *world anchor*. Inoltre, gestisce lo sharing della scena, ovvero sarà possibile condividere, in tempo reale, la vista di chi indossa l'HoloLens, ad esempio tra un addetto specializzato per usare il dispositivo di realtà aumentata, e un altro utente, ad esempio lo scenografo, che potrà impartire indicazioni in remoto e senza dover imparare ad utilizzare il dispositivo.

Figura 1.7: Esempio di un tavolo allestito di *Rob-O-Cod* (1).

Figura 1.8: Esempio di un tavolo allestito di *Rob-O-Cod* (2).

Capitolo 2

La realtà estesa

In questo capitolo, dopo una breve panoramica sulla storia della **realtà virtuale** (2.1) e delle aspettative su questa tecnologia (2.2), ne verranno illustrati i concetti chiave, quali il continuum realtà/virtualità (2.3.1), il senso di presenza (2.3.2) e la stereoscopia (2.3.3). Verrà effettuata una breve panoramica delle tecnologie di realtà aumentata finora esistenti, spiegando brevemente il loro funzionamento generale (2.4) e quali sono i dispositivi di input (2.4.1) e di output (2.4.2), e infine verranno analizzati diversi settori in cui può essere sfruttata (2.5), focalizzando l'attenzione per il settore dell'intrattenimento televisivo e del cinema (2.5.7).

2.1 Breve storia della realtà virtuale

Morton Heilig, già dalla metà del XX secolo, parlò del cosiddetto "cinema esperienza", che poteva coinvolgere tutti i sensi in maniera realistica, immergendo lo spettatore nell'azione che si svolgeva sullo schermo. Costruì un prototipo della sua visione, chiamato **Sensorama** (fig. 2.1), nel 1962, insieme a cinque film che questo apparecchio proiettava e che coinvolgevano molti sensi (vista, udito, olfatto, tatto). Costruito prima dei computer digitali, il Sensorama era un dispositivo meccanico che funziona ancora oggi.

Nel 1968 Ivan Sutherland, con l'aiuto del suo studente Bob Sproull, creò quello che è considerato il primo sistema di realtà virtuale con visore. Era primitivo sia in termini di interfaccia utente sia di realismo, il visore da indossare era così pesante da dover essere appeso al soffitto e la grafica era costituita da semplici stanze in wireframe. L'aspetto di quel dispositivo ne ispirò il nome, **La Spada di Damocle** (fig. 2.2).

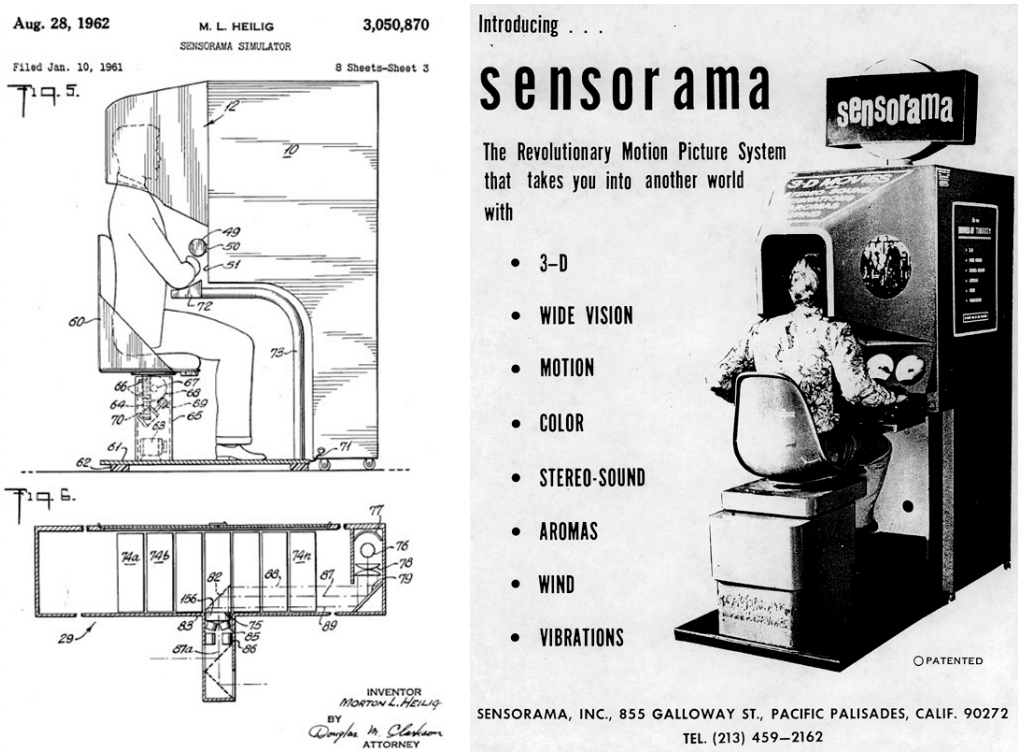

Figura 2.1: Sensorama, il primo prototipo di realtà virtuale.

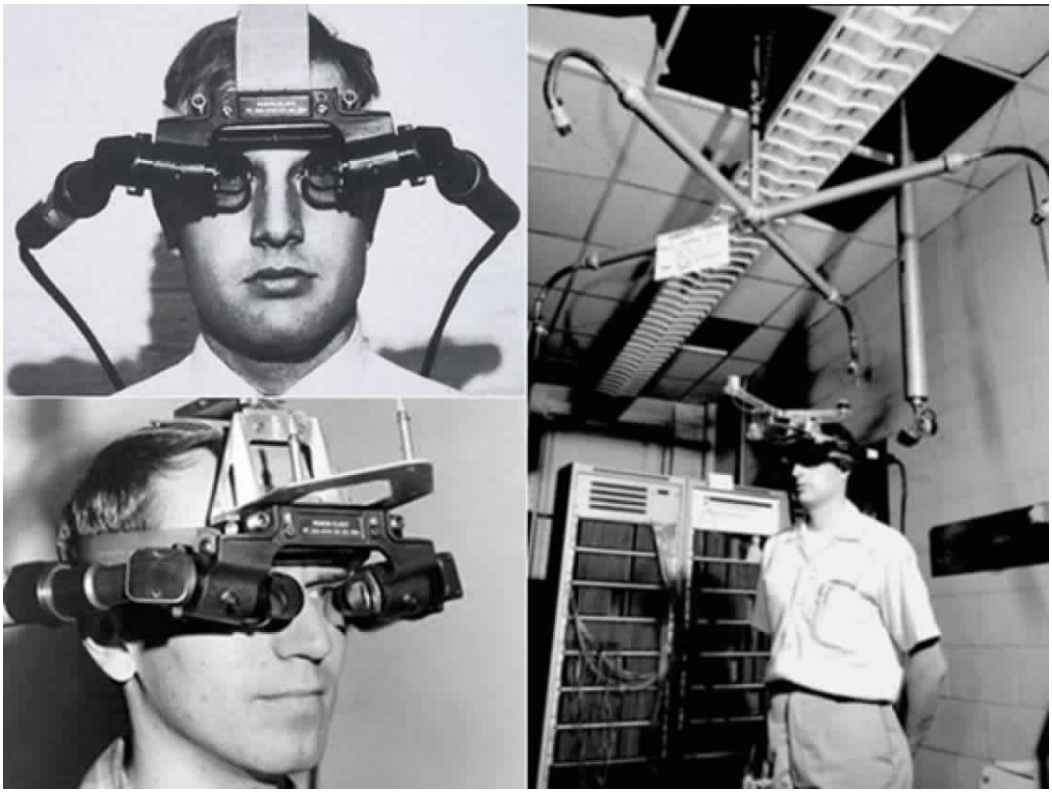

Figura 2.2: La spada di Damocle, il primo primitivo sistema di realtà virtuale con visore.

Figura 2.3: L'Aspen Movie Map, il primo vero sistema di realtà virtuale.

Il primo passo decisivo verso l'ipermedia, e il primo dispositivo che possa essere considerato di realtà virtuale, è stato l'**Aspen Movie Map** (fig. 2.3), realizzato sotto forma di software dal MIT nel 1977. Il principale scopo di questo simulatore era ricreare virtualmente Aspen, cittadina del Colorado; agli utenti era concesso di camminare per le vie in modalità estate, inverno e in modalità poligonale. Mentre le prime due modalità erano indirizzate alla replica di filmati delle strade della cittadina, la terza si basava su una poligonazione tridimensionale, con una grafica scarsa visti i limiti tecnologici di allora. Può essere considerato un antenato di Google Street View.

La nascita del termine VR, Virtual Reality, risale al 1989, anno in cui Jaron Lanier, uno dei pionieri in questo campo, fondò la VPL Research (Virtual Programming Languages, "linguaggi di programmazione virtuale"). Il concetto di cyberspazio, ad esso strettamente collegato, si era originato nel 1982 grazie allo scrittore statunitense William Gibson.

Con le tecnologie attuali (2019), la percezione di un mondo virtuale è ancora distinguibile da quella del mondo reale: il fotorealismo delle immagini rende completa o quasi l'esperienza visiva, tuttavia gli altri sensi sono parzialmente trascurati (olfatto e tatto, ad esempio, sono poco stimolati). È chiaro che tra le varie tipologie di ambiente che possono essere proposte attraverso la realtà virtuale, sono quelli 3D a ricevere e a veicolare oggi un maggior interesse. Questo sembra derivare prevalentemente dal fatto che nell'uomo è la vista il senso dominante, motivo per cui gli ambienti virtuali devono essere caratterizzati innanzitutto da qualità visive eccelse, capaci quindi di presentarsi anche come sostituti della realtà, mentre invece gli altri sensi sembrano avere, almeno agli esordi della realtà virtuale, un peso meno influente.

2.2 Aspettative sulla realtà aumentata

Il livello di aspettative alla fine degli anni 80 e inizio anni 90 era elevatissimo, ma la VR ha presto dimostrato di non esserne all'altezza. Alcune delle cause:

- Le simulazioni in tempo reale di ambienti complessi necessitavano di elevate capacità di calcolo che solo le workstation di fascia alta (e quindi particolarmente costose) potevano garantire.
- La mancanza di interfacce ad alta qualità, in particolare di display ad alta risoluzione e di strumenti di interazione adeguati, determinava una user experience molto povera.
- La mancanza di software adeguati per la progettazione, sviluppo e gestione della complessità di un ambiente virtuale.

Con il progresso della tecnologia nel corso degli anni, sono stati sviluppati dispositivi sempre più sofisticati e con una capacità di calcolo sempre più elevata, abbastanza da poter venire incontro alle esigenze della VR/AR. La VR/AR non è ancora considerabile un dispositivo di massa, in quanto:

- È più difficile ingannare i sensi.
- Il coinvolgimento non è totale, in quanto non riesce ancora a stimolare tutti e cinque i sensi.
- Spesso è monoutente.
- Alcune interfacce sono invasive, oppure un uso prolungato di alcune interfacce possono provare fatica e nausea.
- L'hardware è costoso e non alla portata di tutti.
- L'hardware è da migliore ulteriormente in termini di risoluzione, complessità, velocità e ingombri.

Il *Gartner Hype Cycle* [2] è un grafico che mostra il ciclo di vita di qualunque innovazione tecnologica (fig. 2.4). Un Hype Cycle analizza in dettaglio le cinque fasi chiave del ciclo di vita di una tecnologia:

1. **Trigger dell'innovazione:** nasce una potenziale innovazione tecnologica. Le prime storie di verifiche teoriche e l'interesse dei media scatenano una significativa pubblicità. Spesso non esistono prodotti utilizzabili e la fattibilità commerciale non è dimostrata.

2. **Picco delle aspettative**: la pubblicità iniziale produce numerose storie di successo, spesso accompagnate da decine di fallimenti. Alcune aziende decidono di investire nella nuova tecnologia, ma molti rinunciano.

3. **Depressione della disillusione:** l'interesse diminuisce quando gli esperimenti e le implementazioni non forniscono i risultati sperati. Alcuni produttori di questa tecnologia falliscono. Gli investimenti continuano solo se i fornitori sopravvissuti migliorano i loro prodotti con soddisfazione dei primi utenti.

4. **Inclinazione all'illuminazione**: iniziano a cristallizzarsi e ad essere più ampiamente compresi sempre più esempi di vantaggi che la tecnologia può apportare all'azienda. I prodotti di seconda e terza generazione compaiono dai fornitori e più imprese finanziano le aziende, che comunque rimangono caute sul futuro.

5. **Plateau di produttività**: la tecnologia inizia a decollare. Sono più chiari i criteri per la valutazione del prodotto per il fornitore e la tecnologia comincia a diffondersi sul mercato.

In fig. 2.5 e 2.6 sono illustrati i Gartner Hype Cycle riguardo lo sviluppo rispettivamente della realtà aumentata e delle tecnologie nel 2018.

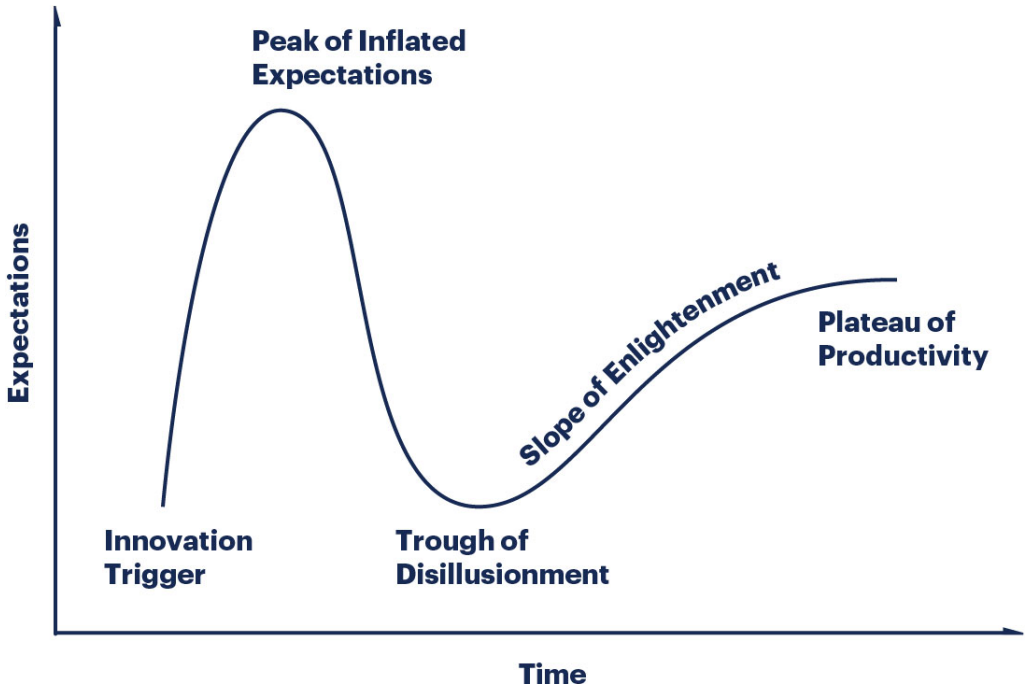

Figura 2.4: Gartner Hype Cycle [2].

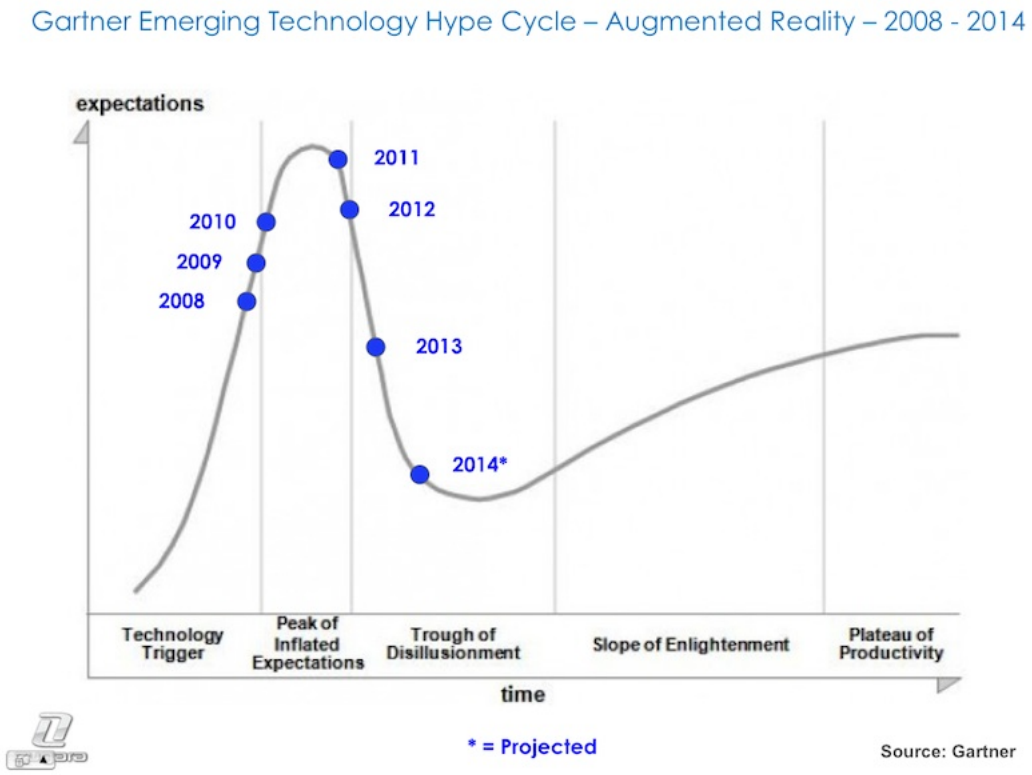

Figura 2.5: Gartner Hype Cycles per la realtà aumentata, dal 2008 al 2015 [3].

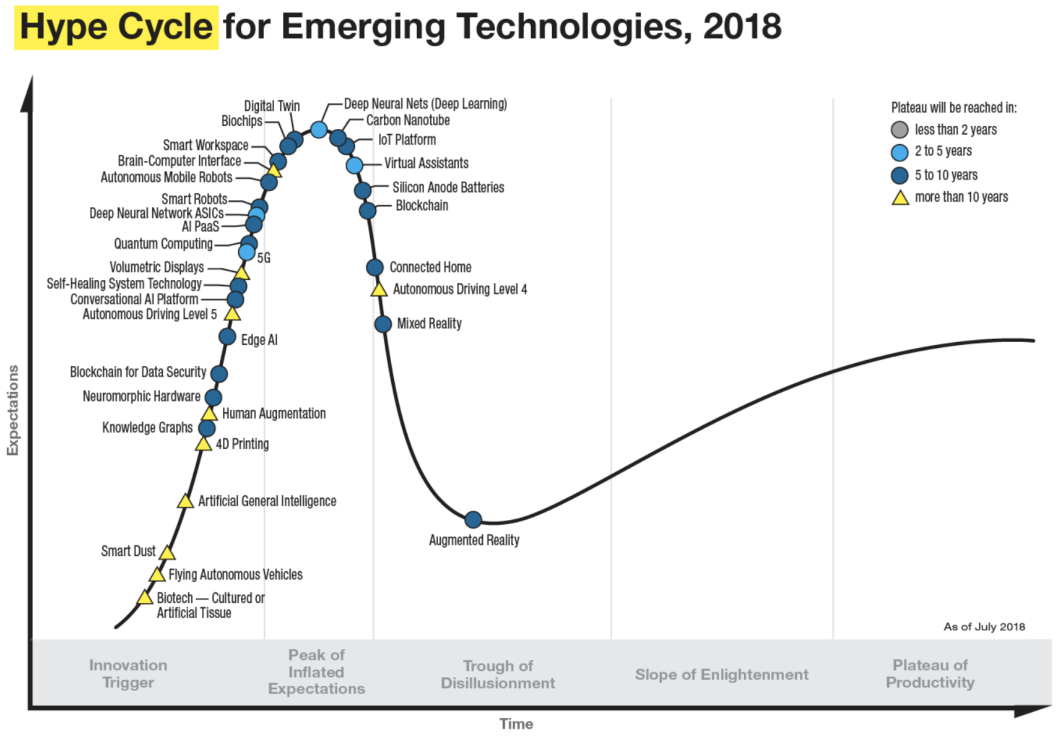

Figura 2.6: Gartner Hype Cycles per le tecnologie emergenti del 2018 [4].

2.3 I concetti base della realtà estesa

2.3.1 Il continuum realtà/virtualità

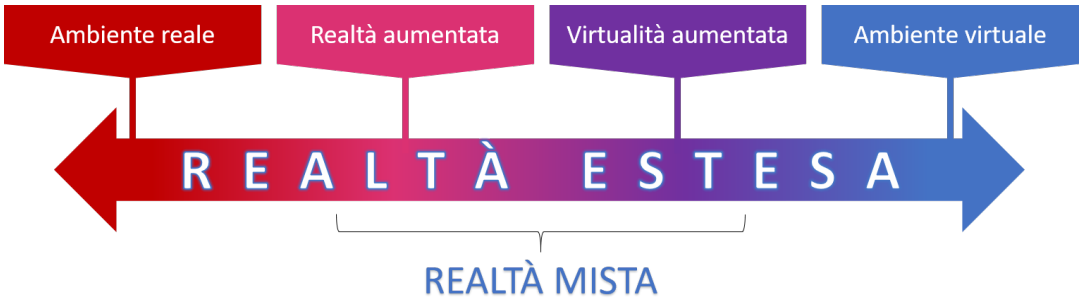

Figura 2.7: Il continuum realtà/virtualità.

La **realtà virtuale** è una realtà simulata, un ambiente tridimensionale costruito al computer che può essere esplorato e con cui è possibile interagire usando dispositivi informatici – visori, guanti, auricolari – che proiettano chi li indossa in uno scenario così realistico da sembrare vero.

Il **continuum realtà/virtualità** è una scala che varia dal completamente reale al completamente virtuale (fig. 2.7). Esprime, quindi il rapporto tra ciò che è reale e ciò che è virtuale. Tutto ciò che è compreso tra i due estremi di questa scala viene definito **realtà mista** (MR) ovvero uno scenario in cui coesistono il reale e il virtuale, mentre la **realtà estesa** (XR) comprende sia la realtà virtuale che la realtà mista.

Nel contesto della realtà mista, possiamo racchiudere principalmente due tipologie di applicazioni:

- **Realtà aumentata:** si basa sull'ampliamento o sull'integrazione della realtà circostante con immagini generate al computer, che modificano l'ambiente originario senza influire sulle possibilità di interazione.
- **Virtualità aumentata:** in questo scenario, persone od oggetti reali sono integrati in ambienti virtuali ed interagiscono con esso. L'integrazione di elementi reali in ambienti virtuali può essere effettuata attraverso diverse tecnologie, quali ad esempio video, digitalizzazioni 3D di oggetti o depth camera.

Il dispositivo utilizzato per sviluppare questa tesi, l'HoloLens, trattato nel paragrafo 3.1.1, è un dispositivo di realtà mista, in quanto consente di avere una visione del set televisivo reale con la possibilità di gestire ed interagire con oggetti di scena virtuali.

2.3.2 Il senso di presenza

La **presenza** esprime il senso di appartenenza in un ambiente virtuale. È tanto più alta quanto sono alti i livelli di:

- **Immersione:** quanto l'utente si sente parte dell'ambiente virtuale. Si ha un maggior livello di immersione quanto più sono coinvolti i 5 sensi (vista, udito, tatto, gusto e olfatto). Si parla di **virtual embodient**, ovvero il feedback

sensoriale relativo al corpo virtuale trasmette i suoi effetti sul sistema cognitivo dell'utente. Il famoso esperimento della "mano di gomma" dimostra questo concetto: la vittima dell'esperimento non vedrà la sua mano reale, ma al suo posto vedrà una mano di gomma. Si ottiene il senso di "appartenza" stimolando in sincronia la mano reale, non visibile, dell'utente e quella di gomma, visibile. Il risultato è che sistema cognitivo fonde le due informazioni "sentendo" come propria la mano di gomma.

- **Interazione:** quanto l'utente può interagire con gli elementi dell'ambiente virtuale. La presenza sarà maggiore quanto più l'utente potrà interagire con questi oggetti.

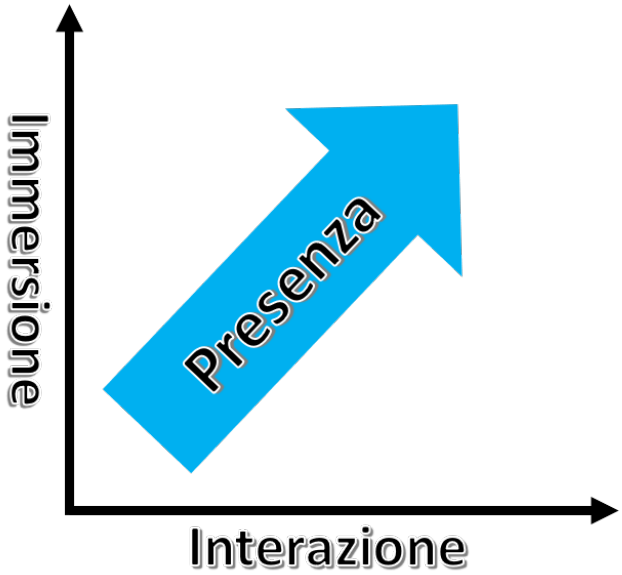

Figura 2.8: Il rapporto tra immersione e interazione fornisce il senso di presenza.

La presenza è anche esprimibile come somma di due fattori:

- **Illusione di luogo:** è la forte illusione di "essere in un luogo" malgrado la consapevolezza di non esserci realmente. L'immersione rappresenta i limiti entro cui l'illusione di luogo può avvenire. Sono necessari feedback sensoriali/motori coerenti con le azioni e i cambiamenti che impongono all'ambiente.

- **Illusione di plausibilità:** Si tratta dell'illusione che ciò che avviene nell'ambiente virtuale stia effettivamente avvenendo. Per plausibilità si intende quanto posso ritenere reali gli eventi a cui assisto. È un concetto separato dalla illusione di luogo, che invece dà la sensazione di essere in un posto, senza per forza credere a quello che accade. L'ambiente deve reagire in maniera plausibile. Per dare l'illusione, non è necessario che l'ambiente sia realistico.

Per migliorare l'illusione di luogo e di plausibilità, bisogna tenere in conto alcuni fattori:

- **Qualità delle informazioni sensoriali:** più è alta la qualità delle informazioni ricevute dai 5 sensi, più è alto il senso di illusione. Ad esempio,

un'applicazione con una grafica ad alta risoluzione sarà più illusiva di una a bassa risoluzione.

- **Mobilità e controllo dei sensori:** i movimenti e i controlli devono essere naturali e l'hardware del dispositivo non deve risultare ingombrante al movimento.
- **Controllo dell'ambiente:** un basso controllo dell'ambiente comporta un basso livello di interazione, e quindi di presenza, in quanto l'utente si sentirà esterno all'esperienza che sta vivendo.

Per poter avere presenza in un ambiente virtuale, è necessario che:

- L'ambiente sia concepito e realizzato in modo tale da massimizzare questa sensazione.
- L'utente sia predisposto ad accettare la sensazione di presenza in un ambiente non reale. Si parla di *sospensione di incredulità*.

La presenza è difficile da misurare. Si può misurare in termini:

- **Quantitativi:** si registrano parametri fisiologici e/o tempi di risposta durante l'esperienza.
- **Qualitativi:** si fanno compilare dei questionari al termine dell'esperienza.

Per questa tesi, la valutazione viene effettuata in termini qualitativi (par. C.1).

2.3.3 La stereoscopia

La **visione 3D** è la percezione del mondo 3D e coinvolge due processi:

- **La parallasse:** è il fenomeno per cui un oggetto sembra spostarsi rispetto allo sfondo se si cambia il punto di osservazione. Gli occhi osservano la scena da due posizioni diverse. L'angolo di convergenza determina la percezione della distanza dell'oggetto dall'osservatore.
- **La stereopsi:** detta anche visione binoculare, è la caratteristica propria del sistema visivo di alcune specie animali. Ognuno dei due occhi ha un campo di vista proprio, ma esiste un'area dove le due aree si sovrappongono: la stereopsi. In quest'area, è possibile combinare le informazioni dei due occhi per ottenere informazioni sull'ambiente.

La **stereoscopia** è una tecnica di realizzazione e visione di immagini, disegni, fotografie e filmati, atta a trasmettere una illusione di tridimensionalità, analoga a quella generata dalla visione binoculare del sistema visivo umano. Sfrutta le **depth cues**, ovvero indizi dell'ambiente come occlusioni, ombre, parallassi e prospettive dinamiche, per ricevere e rafforzare le informazioni sulla profondità e sulla distanza di oggetti nell'ambiente.

Quando si utilizza la stereoscopia, bisogna tenere in considerazione il **conflitto di accomodazione-vergenza**:

- **Accomodazione:** l'atto di messa a fuoco di un oggetto tramite la contrazione del cristallino. È fissa sullo schermo.

- **Vergenza:** rotazione dei bulbi oculari che fa sì che le linee di vista dei due occhi si incrocino sull'oggetto. È variabile a seconda del valore di parallasse (davanti o dietro lo schermo).

L'oggetto visualizzato su uno schermo può creare o no disagio in base a dove si trova rispetto alla vista di chi osserva, in quanto può alimentare il conflitto accomodazione-vergenza. Basandosi sulla figura 2.9, le zone della vista stereoscopica sono di diversi tipi:

- **Zona di comfort:** è una gamma di parallassi positive e negative in cui il disaccoppiamento di accomodazione-vergenza non crea disagio. Varia da persona a persona, ma si trova approssimativamente intorno al piano zero, ovvero lo schermo del dispositivo.
- **Alta parallasse:** sono aree lontane dal piano zero che danno un basso livello di comfort.
- **Rivalità retiniche:** sono le aree al di fuori della stereopsi, ovvero sono aree osservabili sono da uno dei due occhi. Possono essere:
  - Positive (con parallasse positiva): si trovano oltre il piano zero, sono tollerabili per bassi valori di parallasse.
  - Negative (con parallasse negativa): si trovano tra il piano e l'osservatore, oggetti in quest'area sono estremamente fastidiosi. Può essere usato solo per ingressi o uscite di scena.

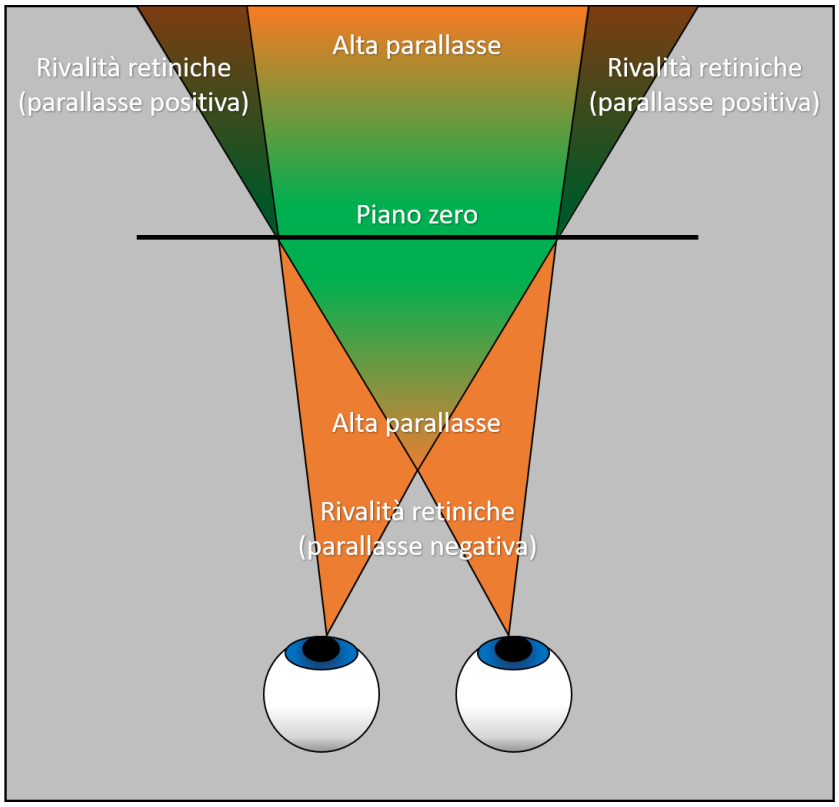

Figura 2.9: Schematizzazione delle zone di comfort stereoscopico.

2.3.4 Il problema della motion sickness

La **motion sickness**, o cinetosi, è un senso di malessere provocato dall'alterazione della relazione cinestetica tra il corpo e l'ambiente. In realtà virtuale ed aumentata costringe l'interruzione dell'esperienza.

Questo disturbo fisiologico è causato da un conflitto di più segnali sensoriali: nel caso della motion sickness dovuta alla realtà virtuale ed aumentata, il conflitto avviene da parte del sistema visivo, che percepisce un movimento corporeo, e dal senso dell'equilibrio, che invece non percepisce movimento. È quindi comparabile al mal d'auto o al mal di mare.

Tanto più il livello di immersione è elevato, tanto più sarà elevato il coinvolgimento sensoriale dell'utente, spesso con le controindicazioni come il senso di nausea che, con l'avanzare delle tecnologie sensoriali per le esperienze virtuali, andranno ad affievolirsi sempre di più, rendendo l'esperienza virtuale sempre più simile a quella reale.

È l'orecchio interno che include il labirinto auricolare, composto a sua volta dalla coclea, l'organo dell'udito, e dall'organo dell'equilibrio, quest'ultimo responsabile della percezione cinestetica in grado di regolare le relazioni tra le parti del corpo e tra il corpo e l'ambiente circostante.

Sono stati effettuati degli studi riguardo la motion sickness in realtà mista su Microsoft HoloLens, attraverso un simulatore di motion sickness [5]: le sensazioni degli utenti sono state misurate per mezzo del Simulation Sickness Questionnaire (SSQ)[1] i cui risultati sono stati riportati nella tabella in figura 2.10. La principale causa di malessere è stato l'affaticamento degli occhi, seguito da mal di testa e difficoltà nella messa a fuoco. In generale, la realtà aumentata comporta un livello basso di motion sickness, e che il senso di malessere è crescente con l'aumentare della virtualità dell'ambiente.

SSQ	*Mean*	*SD*	*>0*	*N*	*O*	*D*
General discomfort	0.15	0.41	12%	×	×	
Fatigue	0.14	0.37	13%		×	
Headache	0.17	0.43	14%		×	
Eyestrain	0.28	0.53	24%		×	
Difficulty focusing	0.16	0.41	14%		×	×
Salivation	0.05	0.21	5%	×		
Sweating	0.06	0.24	6%	×		
Nausea	0.05	0.26	5%	×		×
Difficulty concentrating	0.08	0.27	8%	×	×	
Fullness of the head	0.09	0.32	8%			×
Blurred vision	0.10	0.30	10%		×	×
Dizziness (eyes open)	0.05	0.21	5%			×
Dizziness (eyes closed)	0.04	0.19	4%			×
Vertigo	0.03	0.17	3%			×
Stomach awareness	0.05	0.24	4%	×		
Burping	0.02	0.15	2%	×		

Figura 2.10: Risultati dello studio [5] sulle principali cause di motion sickness.

[1] N: Nausea - D: Disorientamento - O: Nervi Oculomotori (occhi).

2.4 Tecnologie

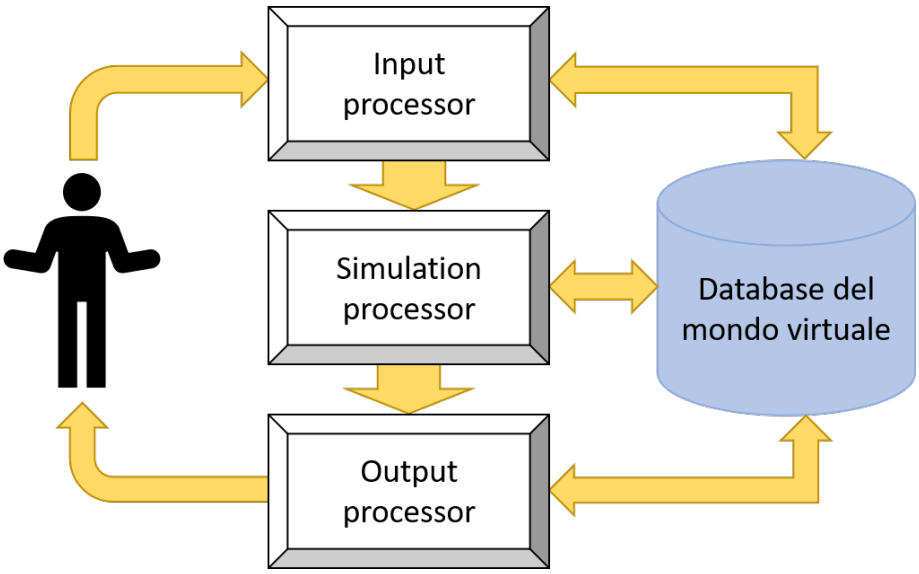

Figura 2.11: Funzionamento di un dispositivo di VR/AR.

Il funzionamento di un dispositivo VR/AR si suddivide principalmente in tre fasi (fig. 2.11):

- **Raccolta degli input:** in questa fase, vengono raccolti gli input provenienti dall'ambiente oppure forniti direttamente dall'utente ad esempio tramite trackers o videocamere.
- **Elaborazione degli input:** in questa fase, vengono prodotti dei valori di output elaborando gli input della fase precedente, ad esempio calcolando la nuova immagine da mostrare a schermo sulla base della nuova posizione dell'utente.
- **Restituzione degli output:** in questa fase, l'utente riceverà un feedback dal dispositivo sulla base delle sue azioni.

Sia i valori di input che quelli di output possono coinvolgere i 5 sensi. Finora, ci si concentra sui sensi della vista, dell'udito e del tatto, ma esistono anche dispositivi I/O di altro tipo, come quelli inerziali.

2.4.1 Dispositivi di input

I dispositivi di input vengono classificati sulla base di:

- **Caratteristiche dei dati:**
  - Discreti/ad eventi (attivi): generano un evento per volta in base alle azioni dell'utente (es. tastiera, pulsanti, ...).
  - Continui/campionati: generano informazioni in modo autonomo (passivi) o in base alle azioni dell'utente (attivi) (es. tracker, dataglove, ...).
  - Ibridi: ad esempio il mouse.
- **Per gradi di libertà:**
  - 1 DoF: tastiera, bottoni.
  - 2 DoF: posizione del mouse.
  - 3 DoF: mouse + tasti, posizione 3D, orientamento 3D.
  - 6 DoF: posizione + orientamento.
  - 6+N DoF: posizione + orientamento + altri controlli.
- **Per tecnologia/tipo di sensori:** es. sensori elettronici, magnetici, ottici, inerziali, meccanici, ibridi, onde radio e onde sonore.

Alcune tipologie di dispositivi di input sono:

- **Desktop input devices:** tastiera, mouse, joystick, trackballs, tavolette grafiche, spaceMouse, spaceBall e dispositivi tattili a 6+ DoF.
- **Tracker:** si tratta di sensori responsabili della cattura di posizione e orientamento di un oggetto 3D. Si differenziano per DoF, range di misurazione, precisione e frequenza di misurazione. Le misurazioni sono relative ad un sistema di riferimento fisso e noto. Includono componenti di acquisizione (sensori) e di elaborazione. Le prestazioni di un tracker si misurano in termini di:
  - Precisione (accuracy): differenza tra la posizione reale di un oggetto e quella misurata dal sensore.
  - Rumore (jitter): variazione del valore misurato dal sensore quando l'oggetto è fermo.
  - Deriva (drift): aumento dell'errore fisso introdotto nella misura del sensore in funzione del tempo. Può essere controllata e azzerata tramite un dispositivo secondario fisso.
  - Latenza (latency): ritardo di risposta del sensore nel misurare una variazione di posizione o orientamento dell'oggetto.
  - Velocità di acquisizione dei dati (update rate): numero di misurazioni al secondo.

  I trackers sono di diverso tipo a seconda di come ottengono valori in input, e possono essere meccanici, magnetici, acustici, ottici o ibridi.

- **Motion capture:** molto usato in ambito cinematografico, cattura il movimento dettagliato del corpo umano tramite sistemi live, se il controllo del movimento è in tempo reale, oppure offline, ad esempio per animazioni. Il **motion capture facciale** è più complesso di quello fatto su tutto il corpo e può essere effettuato con sistemi marker-based o markerless (computer vision).
- **Interfacce aptiche:** es. 3D probe, guanti sensibili e leap motion, ovvero si usa la mano nuda nell'interazione. Gli HoloLens usano quest'ultima interfaccia.
- **Interfacce inerziali (vestibolari):** catturano gli spostamenti dell'utente, in particolare la camminata.
- **Interfacce sperimentali:** es. riconoscimento del parlato, biosensori, Brain Computer Interface. Gli HoloLens sfruttano il riconoscimento del parlato.
- **Interfacce utente 3D:** mappano dispositivi 2D e bottoni su elementi del mondo 3D. L'idea che sta alla base è quella delle metafore di interazione, cioè riproducono concetti, conosciuti dall'utente in un determinato contesto, con lo scopo di trasferire questa conoscenza in un nuovo contesto legato all'esecuzione di un certo task, per effettuare operazioni di navigazione nell'ambiente, selezione, manipolazione e controllo del sistema. Nel caso dell'applicazione realizzata per HoloLens, le metafore utilizzate sono *real walking* per la navigazione e *raycasting/speech* per la selezione, il *leap motion* per la manipolazione e un *menu* per il controllo del sistema.

2.4.2 Dispositivi di output

I dispositivi di output stimolano i sensi dell'utente sulla base degli stimoli ambientali ricevuti in input in precedenza, in modo tale da aumentare il senso di presenza all'interno dell'applicazione. Finora, si possono distinguere principalmente quattro categorie di display: visuali, sonori, tattili ed inerziali.

Display visuali

Ricostruiscono l'immagine virtuale sul dispositivo.
I **display stereoscopici** vengono classificati in:

- **Sistemi stereoscopici (basati su occhialini):**
  - Head mounted display: possono essere ricostruite due immagini separate per due microdisplay davanti agli occhi.
  - Sistemi multiplexed: viene ricostruita un'immagine su un'unica superficie di visualizzazione, usando stereoscopia attiva (o temporale), che usa occhiali sincronizzati con un monitor, o passiva (o spaziale), che separa le immagini per i due occhi sulla base di luce polarizzata o color multiplexing (es. anaglifi).
- **Sistemi autostereoscopici (non intrusivi):** si tratta di sistemi basati su parallasse e multiplexing spaziale, che separano in modo distinto i canali che raggiungono i due occhi, a patto che l'osservatore sia posizionato in un determinato angolo ad una determinata distanza. I display di questo tipo sono:

  - Passivi: usano ottiche speciali poste di fronte al monitor, come le barriere di parallasse e le lenti lenticolari. Sono multiutente.
  - Attivi: generano immagini sulla base della posizione e della testa dell'utente. Sono monoutente.
  - Volumetrici: creano una effettiva rappresentazione tridimensionale di un oggetto invece di simulare la stereoscopia.
  - Olografici: si basa sull'olografia, cioè la percezione dell'oggetto è data dalla luce che colpisce l'oggetto e dalla luce riflessa (o rifratta) sull'osservatore. Si riproduce la luce emessa dall'oggetto, in modo da ricrearne la sua presenza.

I **display grafici** si suddividono in:

- **Personal graphic display:** ad esempio hand supported display, floor supported display e head mounted display. Gli HoloLens fanno parte dell'ultima tipologia.
- **Desk supported display:** sono monitor fissi per più utenti, con risoluzioni, dimensioni e qualità elevate ma con mobilità nulla e area attiva ridotta.
- **Large volume display:** sono dispositivi multiutente generalmente basati su proiettori. Alcuni LVD sono i workbench, i wall-type display, i diplay curvi, i dome display e i CAVE.

Il **rendering grafico** è quel processo che trasforma una scena 3D in una 2D da mostrare su un display. Ha tre diversi stadi funzionali:

1. **Application**: legge i dati nel database e i dati in input, calcola le risposte agli input, individua le collisioni e calcola il feedback tattile.
2. **Geometry:** effettua le trasformazioni geometriche (rotazioni, traslazioni e scalamenti), calcola le illuminazioni, effettua le proiezioni della scena, il clipping e il mapping delle textures.
3. **Rasterizer:** crea l'immagine 2D, assegna il colore ai pixel, determina le superfici visibili con Z-buffer ed effettua il rendering con il quad-buffer.

Display sonori

I display sonori possono fornire un suono monoaurale, ovvero uguale per entrambe le orecchie, o biaurale, diverso per ciascun orecchio.

L'audio 3D restituisce informazioni sonore basandosi sui concetti di localizzazione del suono, cioè come il cervello percepisce la provenienza di un suono, e sulla spazializzazione del suono, per generare un suono come se provenisse da un punto ben preciso dello spazio. Il suono viene generato sulla base della *Head Related Transfer Function*, trasformata di Fourier della Head Related Impulse Response.

Display tattili

Ad oggi, i display tattili restituiscono feedback praticamente solo sulla mano. I feedback possono essere:

- Touch feedback: danno informazioni su temperatura, scivolosità, rugosità e geometria della superficie.
- Force feedback: danno informazioni su resistenza, inerzia e peso.

Il rendering tattile è suddiviso in tre stadi con un update rate molto maggiore della pipeline grafica:

- Collision detection stage: carica i dati dal database e i valori in input e identifica le collisioni. Passa allo stato successivo solo se identifica almeno una collisione.
- Force computation stage: calcola il valore delle forze di collisione, aggiusta la direzione delle forze calcolate e le mappa sul dispositivo tattile.
- Tactile computation stage: calcola la componente di touch feedback, la somma alle forze calcolate in precedenza e restituisce l'output al dispositivo.

Display inerziali

Questi sistemi hanno lo scopo di permettere all'utente di percepire le forze di inerzia in accordo con le azioni e i movimenti effettuati nell'ambiente virtuale. Esempi di display inerziali sono le motion platform e i locomotion.

2.5 Alcune aree di utilizzo

Quando si pensa alla realtà mista, il primo ambito di utilizzo che viene in mente è quello dell'intrattenimento, in particolare videoludico. In realtà, le sue applicazioni spaziano su diversi ambiti. Qui di seguito ne vedremo alcuni esempi.

2.5.1 Intrattenimento

Il settore dell'intrattenimento è sicuramente il più immediato quando si pensa alla realtà mista, ed investire in quest'area è importante per avvicinare le persone a questa nuova tecnologia emergente. Sullo store dei dispositivi di realtà aumentata sono già presenti numerose applicazioni che sfruttano nuovi paradigmi e nuove modalità di gaming. Un progetto interessante è quello trattato nell'articolo "*Towards Emergent Play in Mixed Reality*" [6]:

> L'esperienza di gioco può includere una combinazione di gameplay, narrativa e interazioni basate sugli agenti. Il giocatore può interagire in modo creativo con il sistema per migliorare l'esperienza. Utilizziamo Microsoft HoloLens come interfaccia di realtà mista. Fornisce la mappatura spaziale per posizionare oggetti virtuali all'interno dell'ambiente fisico reale. Rileviamo inoltre oggetti reali nell'ambiente e integriamo le proprietà di oggetti reali all'interno dell'esperienza di gioco.

Gli oggetti virtuali nella scena interagiscono in tempo reale con gli stimoli dell'ambiente reale e modificano il loro comportamento in base a questi ultimi (fig. 2.12). Questo comporta un maggiore senso di presenza dell'utente all'interno dell'applicazione.

Figura 2.12: "Towards Emergent Play in Mixed Reality".

Un videogioco in realtà mista può essere anche utilizzato per far prendere confidenza con diverse tecnologie, come nel caso di un progetto del dipartimento di informatica dell'Università Tecnica di Cluj-Napoca [7], che simula un videogioco basato sull'utilizzo di un drone. In questo modo, l'utente impara a conoscere i meccanismi per apprendere le basi per utilizzare un drone reale.

2.5.2 Medicina

Esistono numerose applicazioni in realtà mista sviluppate in ambito medico, in modo da facilitare il lavoro del medico, del chirurgo o del ricercatore. Ad esempio, esiste un progetto per Microsoft HoloLens che riguarda la patologia anatomica, trattato nell'articolo "*Augmented Reality Technology Using Microsoft HoloLens in Anatomic Pathology*" [8] (fig. 2.13), che lavora su diversi casi d'uso della patologia:

- Autopsia: tramite Skype, più persone hanno la possibilità di comunicare in tempo reale, anche a distanza, durante una procedura di autopsia.
- Campioni olografici 3D di patologia lorda: l'utente può lavorare e manipolare organi del corpo umano ad esempio estratti durante un'autopsia e successivamente scansionati in 3D (fig. 2.14). In questo modo, è possibile ricercare elementi patologici sull'organo senza avere a disposizione l'organo fisico.
- Telepatologia: più persone in contemporanea hanno la possibilità di accedere in remoto a dati comuni in modo da parallelizzare le comunicazioni e migliorare l'efficienza della procedure.
- Coregistrazione radiografica del campione: è possibile visualizzare delle radiografie ad una qualità ed accuratezza abbastanza elevate da poter individuare dettagli chiave per l'individuazione di patologie.
- Visualizzazione completa di slide di immagini: l'applicazione consente di navigare tra slide di immagini, utilizzando gesti per scorrere le immagini o per effettuare operazioni di zoom-in e zoom-out.
- Patologia volumetrica: vengono visualizzati modelli 3D di oggetti anche microscopici, in modo da individuare con più facilità patologie altrimenti difficili da individuare.

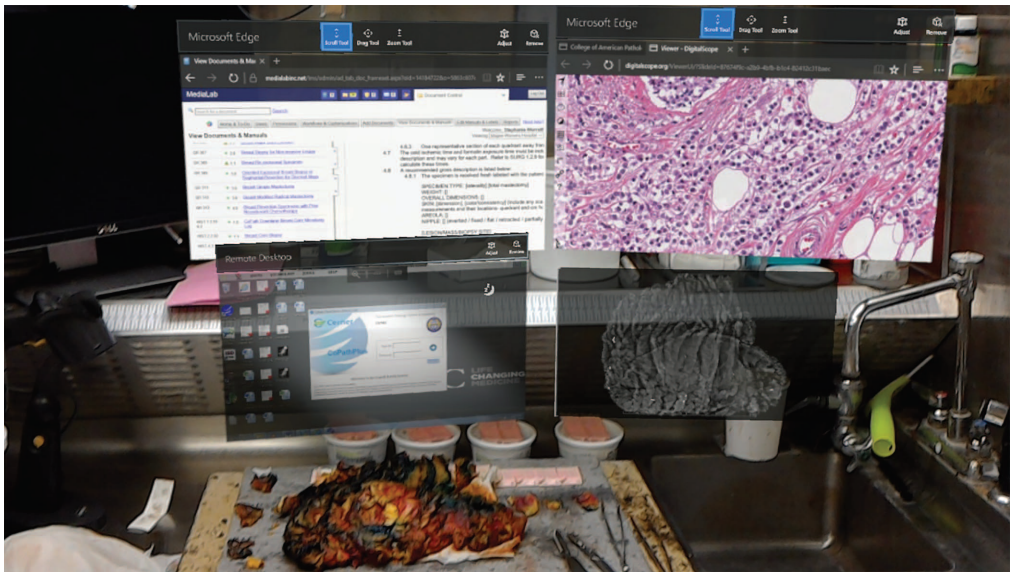

Figura 2.13: Realtà mista in patologia anatomica.

Figura 2.14: "Esempio di visualizzazione di un modello 3D nel progetto descritto in "Augmented Reality Technology Using Microsoft HoloLens in Anatomic Pathology".

La telepresenza è un ambito medico che in futuro potrebbe salvare tantissime vite umane, come nel caso del progetto discusso nell'articolo "*Augmented Reality as a Telemedicine Platform for Remote Procedural Training*" [9]. Quest'applicazione ha come obiettivo quello di dare la possibilità di effettuare operazioni chirurgiche in remoto, in modo sicuro e controllato (fig. 2.15).

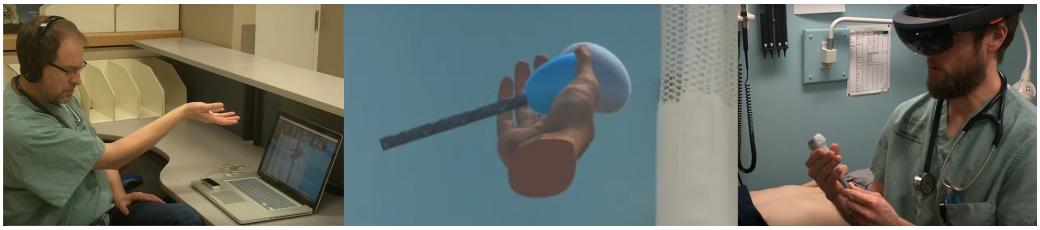

Figura 2.15: "Augmented Reality as a Telemedicine Platform for Remote Procedural Training".

A volte, invece, può essere d'aiuto un'applicazione in realtà mista per facilitare la comunicazione tra medico e paziente, come nel caso del progetto nell'articolo *3D visualisation of breast reconstruction using Microsoft HoloLens* [10]. In questo caso, l'applicazione viene utilizzata per mostrare al paziente un'anteprima del risultato finale prima di effettuare l'operazione di ricostruzione del seno, in modo tale da sceglierne con maggior consapevolezza la forma più opportuna prima di iniziare l'operazione chirurgica.

2.5.3 Psicologia

Le applicazioni di realtà virtuale e mista possono avere grande peso nel trattamento di disturbi psicologici. Un esempio è il progetto esposto nell'articolo "*Augmented Reality Phobia Treatment including Biofeedback*" [11], che ricrea uno scenario virtuale specifico per il trattamento di una fobia del paziente. Il vantaggio principale di questo tipo di approccio è il poter operare in un ambiente sicuro. Per esempio, per il trattamento dell'aracnofobia (fig. 2.16) vengono inseriti nella scena dei ragni virtuali in modo graduale a seconda dello stato del livello di trattamento. Pur essendo lo scenario realistico, l'esperienza è sempre interrompibile in modo sicuro, in caso di notevole stress da parte del paziente, mantenendo quindi un ambiente protetto e controllato. L'applicazione permette, inoltre, di tenere traccia di feedback biologici del paziente, come il battito cardiaco, per registrare il livello di paura e di stress dovuto al suo utilizzo.

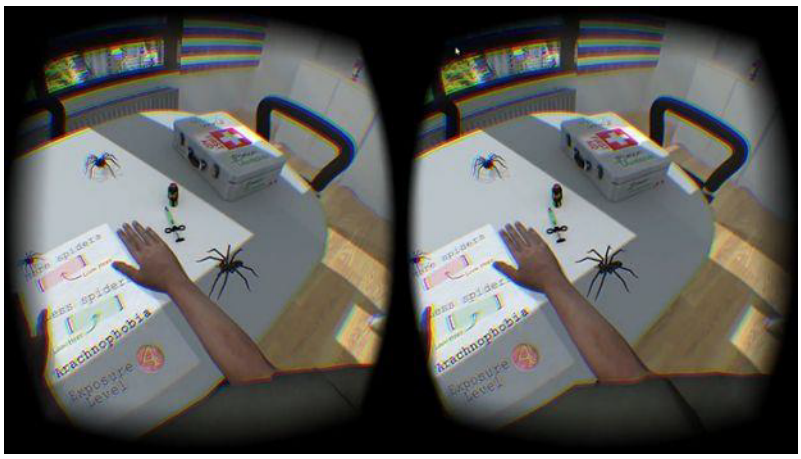

Figura 2.16: Esempio di trattamento dell'aracnofobia in realtà aumentata.

La realtà mista in ambito psicologico non serve solo al trattamento delle fobie: ad esempio, nell'articolo "*Facial Emotion Recognition: A Survey and Real-World User Experiences in Mixed Reality*" [12] vengono esposti strategie e meccanismi per il riconoscimento delle emozioni basandosi sulle espressione dei volti delle persone. Viene quindi individuato il volto grazie ad algoritmi di computer vision, quindi alcuni punti chiave del viso come gli occhi e la bocca, ed infine vengono applicati meccanismi per il riconoscimento dell'emozione in base alle espressioni.

2.5.4 Istruzione e addestramento

Applicazioni di realtà mista possono essere adoperate per uso scolastico, per facilitare l'apprendimento in numerose discipline. In chimica e biologia, ad esempio, può essere adoperata per la visualizzazione 3D delle strutture molecolari [13] (fig. 2.17), che sulle pagine 2D di un libro potrebbero essere difficili da rappresentare. Oppure in anatomia, per la visualizzazione del corpo umano [14], dove il professore può interagire con le varie parti del corpo e isolarne alcune per mostrare agli studenti più nel dettaglio l'oggetto della sua spiegazione (fig. 2.18).

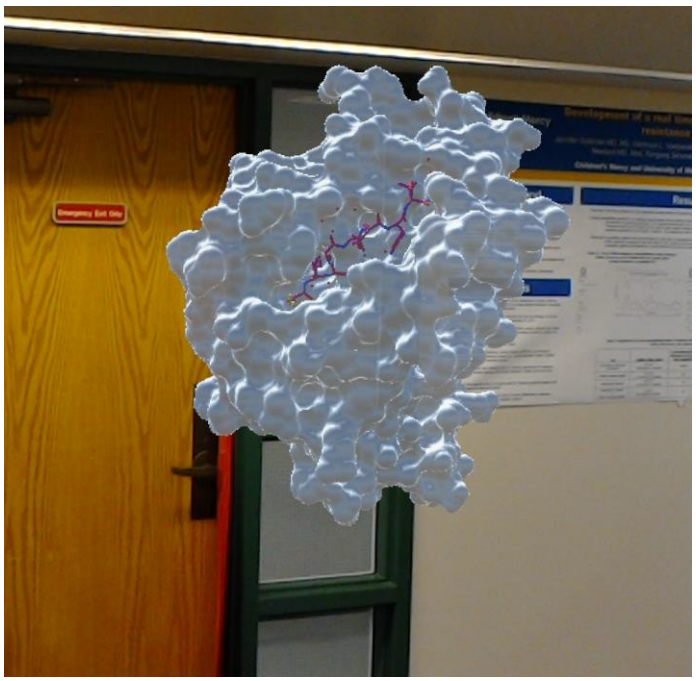

Figura 2.17: Visualizzazione di una struttura molecolare in realtà aumentata.

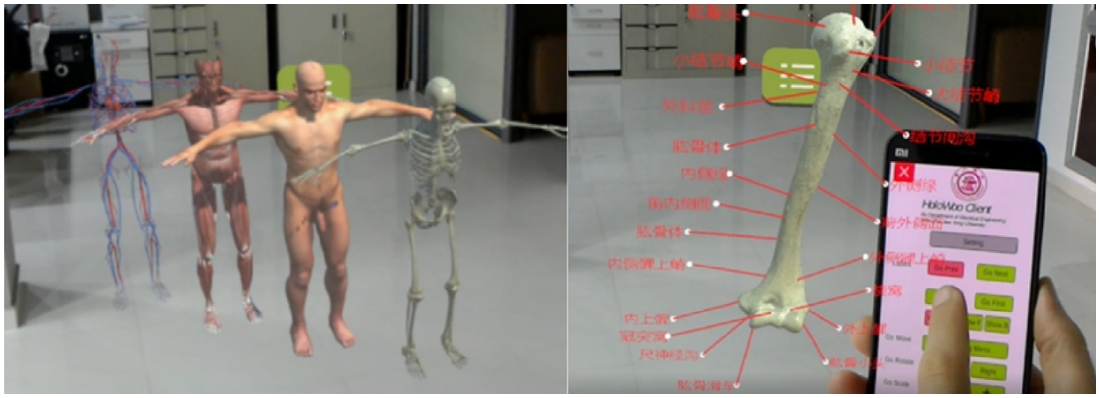

Figura 2.18: Visualizzazione a strati di un corpo umano in realtà aumentata.

Oltre che per istruire, la realtà mista può essere d'aiuto per addestrare ed allenare gli utenti a compiere determinati tasks, sia in ambienti particolari come ad esempio simulatori di volo, ma anche in casa. Per esempio può essere d'aiuto un'applicazione che aiuti in cucina [15], mostrando video delle ricette oppure delle azioni specifiche da seguire quando si guarda un determinato ingrediente (fig. 2.19).

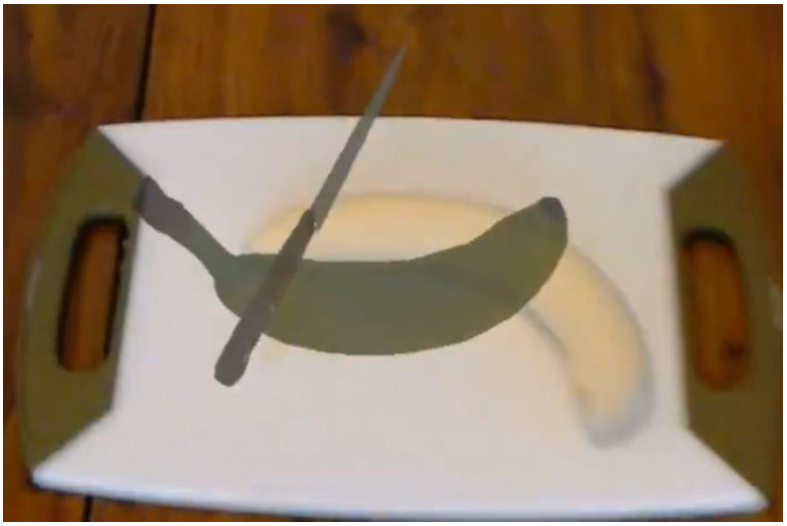

Figura 2.19: Modello 3D dell'istruzione "tagliare".

2.5.5 Industria

La realtà mista in ambito industriale è oggetto di quella che viene chiamata *Industria 4.0* [16], un nuovo modo di approcciarsi al mondo dell'industria con l'ausilio dei robot. Fondamentale, a tal proposito, è la fiducia che si deve instaurare tra l'uomo e la macchina, in modo da permettere la collaborazione (Human Robot Collaboration, HRC) tra queste due entità, e quindi di far lavorare i robot a fianco degli umani in modo sicuro.

Per ridurre i costi, la realtà mista viene sfruttata anche per tasks per la manutenzione, l'assemblaggio e la riparazione [17]. Il problema di questi tasks è però legato alla loro complessità. In questo senso, la realtà mista viene incontro alle esigenze di tecnici che altrimenti dovrebbero fare affidamento a manuali tecnici per completare le procedure a loro assegnate. Oltre all'esecuzione di questi tasks, la realtà mista può essere utilizzata ad esempio per il training dei tecnici, per il controllo di qualità di un prodotto o per monitorare la struttura e la costruzione di ambienti.

Esistono diversi ambienti in cui l'industria 4.0 si è diffusa, uno tra questi è quello navale [18]. Alcuni tra i possibili casi d'uso della IAR (Industrial AR) sono il controllo di qualità, l'assistenza nel processo di produzione, la visualizzazione della posizione di prodotti e strumenti, la gestione di magazzini, la manutenzione predittiva attraverso il data mining, il miglioramento delle comunicazioni, la visualizzazione di installazioni in aree nascoste e il funzionamento in remoto di dispositivi o prodotti IIoT e smart connected.

2.5.6 Architettura e turismo

Con il progresso delle nuove tecnologie, le città "tradizionali" si evolveranno in smart cities, ovvero si attueranno un insieme di strategie di pianificazione urbanistica tese all'ottimizzazione e all'innovazione dei servizi pubblici così da mettere in relazione le infrastrutture materiali delle città con il capitale umano, intellettuale e sociale di chi le abita grazie all'impiego diffuso delle nuove tecnologie della comunicazione, della mobilità, dell'ambiente e dell'efficienza energetica, al fine di migliorare la qualità della vita e soddisfare le esigenze di cittadini, imprese e istituzioni.

Un progetto che implementa questo concetto è quello descritto nell'articolo " *Visualizing Toronto City Data with HoloLens*" [19], nel quale, attraverso una ricostruzione in scala della città di Toronto, è possibile visionare, in realtà mista, un'insieme di dati legati alla mappa della città, con i quali si può interagire attraverso l'uso di gaze, gestures e voce (fig. 2.20).

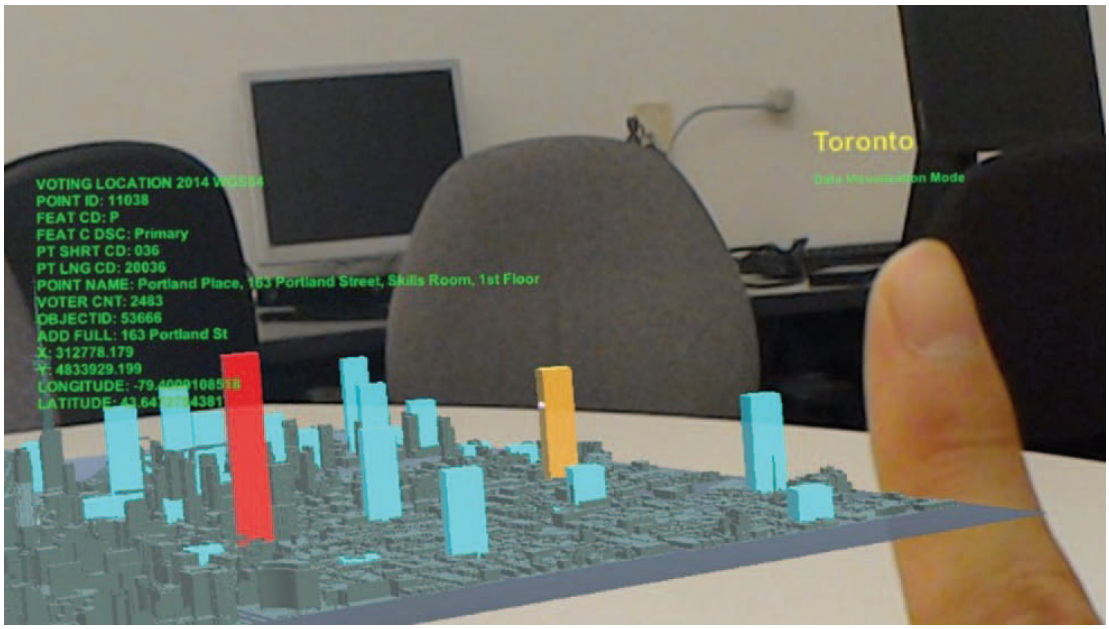

Figura 2.20: Visualizzazione della città di Toronto in realtà aumentata.

In generale, l'inserimento della realtà mista in contesti legati al turismo e al patrimonio culturale vengono percepiti dalle persone in modo positivo, spesso incrementandone il livello di interesse [20]. Nei musei, ad esempio, la realtà aumentata può dare informazioni riguardo gli oggetti osservati dall'utente [21], guidandolo nella visita in modo coinvolgente ed immersivo.

2.5.7 Televisione e cinema

Esistono innumerevoli applicazioni della realtà virtuale e mista in ambito televisivo e cinematografico. In molte trasmissioni televisive, sia in diretta che in differita, viene spesso utilizzata la tecnica del **Chroma Key**, ovvero vengono inseriti nella scena degli oggetti di un determinato colore, solitamente verdi (green screen), successivamente rimpiazzati da oggetti virtuali. Un esempio di uso di questa tecnica è nei telegiornali, in cui un inviato che sta riportando una notizia avvenuta in una determinata città ha alle spalle il paesaggio in movimento di quella città, anche se, di fatto, non si trova lì.

La **European Broadcasting Union** (**EBU**) [22] è la principale alleanza mondiale di media di servizio pubblico, che rappresenta 116 organizzazioni in 56 paesi. Nel 2017 ha indagato sull'utilizzo dei video 360 e delle tecnologie della XR, ma come si può osservare dalla fig. 2.21, non molte aziende investono ancora nella VR, e ancora meno nell'AR. Alcune stanno sperimentando queste tecnologie, altre hanno in programma di farlo in futuro, ma molte di loro non hanno neanche pianificato l'utilizzo di queste nuove tecnologie.

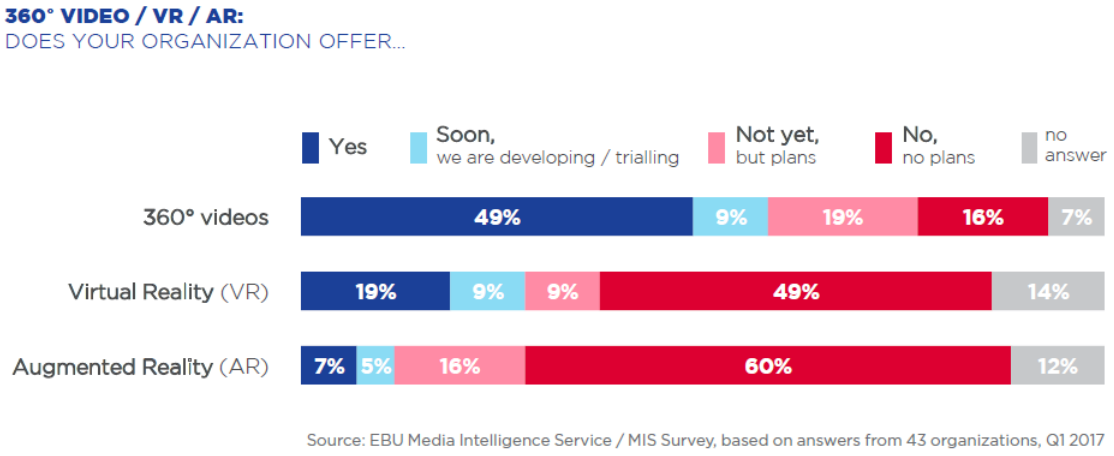

Figura 2.21: La diffusione delle diverse tecnologie delle aziende in EBU nel 2017.

Le aziende che decidono di investire in queste nuove tecnologie intravedono dei vantaggi, in quanto la VR e i video 360 offrono maggiori opportunità di raccontare storie, ed è responsabilità di un'emittente pubblica esplorare tutte le possibili strategie per migliorare l'esperienza per il pubblico. Inoltre, si possono ricevere maggiori feedback da parte del pubblico in modo da migliorare sempre di più la qualità dell'esperienza.

Altre aziende preferiscono tirarsi indietro da questo investimento, per mancanza di conoscenze, competente e strumenti, perché è una tecnologia troppo recente, senza un workflow predefinito, ancora di bassa qualità, e il mercato VR è molto dinamico e in rapido cambiamento. Uno dei problemi più grandi è senz'altro quello legato ai canali di distribuzione, ancora limitati in quanto si tratta di tecnologie ancora poco diffuse tra il pubblico.

Si vogliono sperimentare nuovi approcci per lo storytelling: ogni storia va raccontata con gli strumenti giusti, e il VR e i video 360 non sono compatibili per raccontare tutte le tipologie di storie. Altre, invece, possono migliorare di gran lunga l'esperienza con il pubblico se raccontate tramite VR o video 360. Ad esempio, l'interazione dell'utente all'interno del racconto aumenta il senso di presenza e fa sì che l'utente si senta parte della storia.

Nel 2018, EBU ha pubblicato una prima guida per le emittenti sull'utilizzo della realtà aumentata [23], fornendo strumenti, workflows, sfide, esperienze ed esempi di applicazioni.

Nel corso degli anni, anche il cinema ha sviluppato numerose tecnologie per venire incontro alle sempre più esigenti richieste di produzione di film e programmi televisivi. In questo paragrafo, vedremo alcuni dei più recenti ed innovativi esempi di combinazione tra il reale e il virtuale nei film e nelle serie tv più di successo nell'ultimo anno.

Game Of Thrones [24] è stata sicuramente una delle serie tv più di successo negli ultimi tempi, ed è stata particolarmente lodata per i suoi effetti speciali. Quasi tutte le scene hanno del virtuale, alcune addirittura sono completamente ricostruite in digitale, ma uno degli elementi di rilievo nei VFX di Game Of Thrones è la realizzazione dei draghi. Per la realizzazione dei modelli 3D in computer grafica è stato condotto uno studio su animali reali, come ad esempio i pipistrelli, per studiarne i movimenti delle ali, o le lucertole, per le caratteristiche fisiche. I modelli 3D dei draghi sono stati inseriti nelle scene spesso tramite Chroma Key, come in fig. 2.22. Sono stati adoperati anche alcuni macchinari simili a dei tori meccanici per simulare la cavalcata dei personaggi sui draghi.

Figura 2.22: Estratto di una scena di *Game of Thrones* con un drago, a sinistra durante le riprese, a destra il risultato della postproduzione.

Ogni scena di qualunque film che richieda degli effetti speciali è oggetto di studio da parte dei supervisori SFX, che pensano a tutte le strategie possibili per la realizzazione di scene complesse sulla base dei costi e delle disponibilità a livello tecnologico. Un film come quello di **Avengers: Endgame** [25] è stato molto complicato da realizzare, in particolare nella celebre scena della battaglia finale, dove è stato necessario combinare le performance degli attori reali a performance virtuali di ricostruzioni 3D di personaggi reali (fig. 2.23).

Figura 2.23: Una delle scene più complesse realizzate in *Avengers: Endgame*.

I personaggi virtuali non vengono animati a mano a computer, ma recuperano i movimenti degli attori reali. È il caso del live action di **Aladdin** [26]. L'attore *Will Smith*, che interpreta il genio, indossa per buona parte del film una tuta particolare (fig. 2.24), in grado di catturare i suoi movimenti e di riportarli al modello 3D del genio. Questa tecnica viene chiamata **Motion Capture**. Il motion capture può essere effettuato sia sul corpo che sul volto, le cui espressioni sono più complesse da catturare. Il motion capture facciale viene effettuato grazie a delle telecamere sopra la testa dell'attore che riprendono il suo viso, e catturano il movimento di specifici punti che permettono di ricostruire digitalmente le sue espressioni facciali.

Figura 2.24: Esempio di scena con il genio nel live action di *Aladdin*.

Esistono casi particolari in cui un personaggio virtuale con caratteristiche ben precise diventa una necessità ai fini della trama di film e serie tv. È il caso di **Star Wars: Rogue One** [27], dove il personaggio del *Grand Moff Tarkin*, presente nelle prime trilogie della saga, è stato interpretato dall'attore *Peter Wilton Cushing*, deceduto nel 1994. Il film, del 2019, ha avuto il bisogno di richiamare questo personaggio. Il modello 3D dell'attore (fig. 2.25) è stato ricostruito sulla base delle scene dei precedenti film, studiando e riportando con precisione le deformazioni facciali che avrebbe subito il volto dell'attore a seguito di determinate espressioni, come le rughe di espressione o i movimenti della bocca. È pressoché impossibile distinguere, all'interno del film, l'attore virtuale dagli altri attori reali.

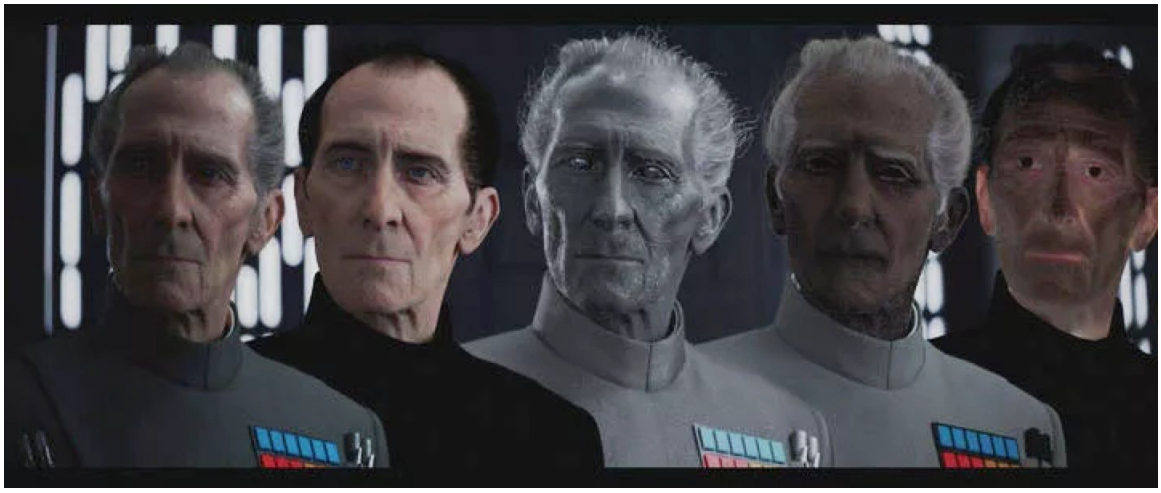

Figura 2.25: Ricostruzione del personaggio *Tarkin* in *Star Wars: Rogue One*.

Per determinate produzioni può essere necessario ricreare un set completamente virtuale: è il caso del live action **The Lion King** [28]. Il film è stato realizzato interamente in computer grafica, ma le riprese sono state effettuate con una tecnica molto particolare: il set reale, in fig. 2.26, è un'area delimitata all'interno di una stanza nella quale vengono utilizzate speciali telecamere che non inquadrano il set reale, ma quello virtuale. Viene quindi realizzato un mapping tra le coordinate del set reale con quello virtuale.

Figura 2.26: Set del live action *The Lion King*.

Capitolo 3

Strumenti per lo sviluppo

In questo capitolo si tratteranno gli strumenti utilizzati per lo sviluppo, sia hardware (3.1), e quindi il **Microsoft HoloLens** (3.1.1) e i metodi di realizzazione dei modelli 3D (3.1.2), che software (3.2), ovvero la gestione del server e la struttura del database (3.2.1) e gli ambienti di sviluppo, quindi verranno analizzati il **Mixed Reality ToolKit** (3.2.2) e le difficoltà riscontrate su Unity (3.2.3), risolte dal plugin **TriLib** (B.1).

3.1 L'hardware

3.1.1 Microsoft HoloLens

Figura 3.1: Microsoft HoloLens (1st Gen).

Microsoft HoloLens [29] è un dispositivo per la realtà mista con un computer olografico incorporato che esegue il sistema operativo di Windows 10.

Svelati in occasione del lancio di Windows 10, il progetto di questo visore, ancora in fase di prototipo, è sviluppato in collaborazione con la NASA e funziona in modo autonomo, ossia non necessita di alcun collegamento con uno smartphone o altro dispositivo. Si tratta, in pratica, di un vero e proprio computer olografico indossabile dotato di sensori di movimento, microfono e audio surround – o meglio spatial sound – che consente di capire da dove proviene il suono, oltre a una videocamera di profondità simile a quella in dotazione su Microsoft Kinect. Il funzionamento dei Microsoft HoloLens è piuttosto complesso: dei fotoni colpiscono le lenti formate da

strati di vetro blu, verde e rosso, fino a raggiungere la parte posteriore degli occhi. Le particelle di luce, a questo punto, attraversando il Light Engine del dispositivo in una data angolazione, generano gli ologrammi visualizzati dall'utente. È stato scelto di sviluppare l'applicazione per l'HoloLens di prima generazione, in quanto l'HoloLens di seconda generazione, ad oggi, non è stato ancora reso disponibile in Italia.

L'hardware dell'HoloLens [30] presenta diversi componenti (fig. 3.2):

- **Visore:** contiene lo schermo sul quale viene mostrato l'ologramma e i sensori. Lo schermo ha un field-of-view ridotto, di circa 35 gradi, una grossa limitazione che riduce l'effetto di immersione da parte dell'utilizzatore. Con l'HoloLens di seconda generazione, il field-of-view verrà ampliato a circa 70 gradi.
- **Headband:** per poter fissare correttamente l'HoloLens alla testa, è possibile regolare la lunghezza della headband con una rotellina posta nella parte posteriore della stessa.
- **Bottoni per la luminosità:** per regolare il livello di luminosità.
- **Bottoni per il volume:** per regolare il volume.
- **Device arms:** usati per afferrare correttamente il dispositivo quando viene messo o rimosso.

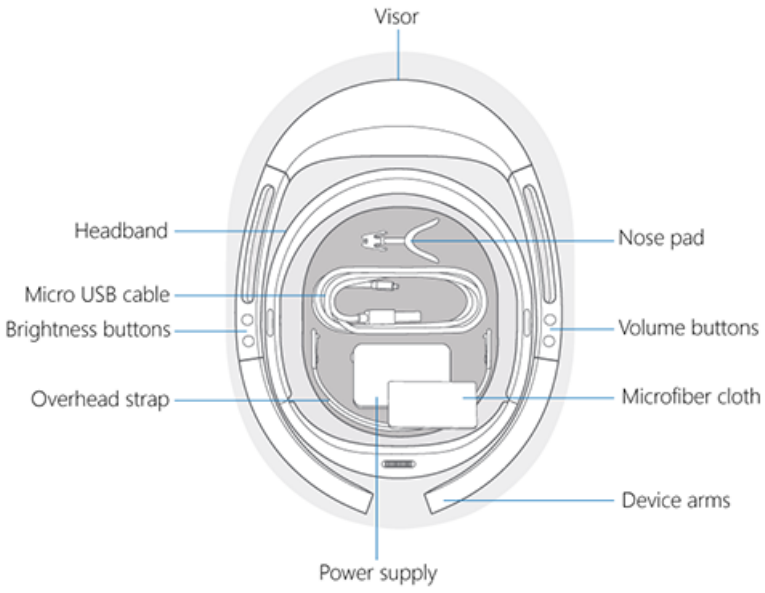

Figura 3.2: Hardware di Microsoft HoloLens (1st Gen).

L'HoloLens fa uso di diversi **sensori** (fig. 3.3):

- 4 microfoni.
- 4 telecamere per lo spatial mapping.
- 1 telecamera di profondità.
- 1 fotocamera 2MP / videocamera HD.
- 1 sensore per la luce ambientale.
- 1 Inertial Measurement Unit (IMU): si tratta di un dispositivo elettronico che misura la forza specifica, la velocità angolare e l'orientamento di un corpo, combinando l'uso di accelerometri, giroscopi e, talvolta, magnetometri.

Figura 3.3: I sensori di Microsoft HoloLens (1st Gen).

3.1.2 Metodi di riproduzione dei modelli 3D

Solitamente, i modelli 3D virtuali vengono realizzati utilizzando software specifici per la modellazione 3D, come **Blender** [31]. La riproduzione di un modello reale in virtuale, però, potrebbe essere onerosa, sia per quanto riguarda la fase di realizzazione della mesh, sia per la realizzazione e la corretta mappatura delle textures sulla mesh.

Un'alternativa alla classica modellazione 3D tramite software è lo scanner laser 3D. In particolare, il *Rai CRITS* ha utilizzato lo scanner laser 3D **Artec Eva**[32] (fig. 3.4), di *Artec3D*, che scannerizza un oggetto reale tridimensionale, inanimato o animato, grande fino ad 8 metri circa, utilizzando una tecnologia a luce strutturata in grado di scansionare anche oggetti neri e lucidi. La scansione realizza ed assegna una texture alla mesh dell'oggetto appena scansionato, in modo tale da riprodurre graficamente le caratteristiche dell'oggetto reale. In questo modo, il processo di creazione di un modello virtuale identico ad un oggetto reale è molto più rapido, garantendo una qualità ed una precisione di riproduzione del modello accettabile.

Figura 3.4: Lo scanner laser 3D *Artec Eva*.

3.2 Il software

3.2.1 Il server e il database

Il server locale è stato realizzato tramite **XAMPP** (A.1), una distribuzione di Apache gratuita contenente MySQL, PHP e Perl. Le applicazioni collegate alla stessa LAN del server potranno contattarlo all'indirizzo IP del computer su cui viene eseguito XAMPP. **PhpMyAdmin** si occupa invece della gestione dell'amministrazione del database MySQL, e quindi della visualizzazione del contenuto e dell'esecuzione di alcune operazioni elementari come aggiunta, modifica e rimozione di tabelle e tuple. L'utilizzo di questi due importanti tool viene approfondito nell'appendice A.

Il database è stato pensato per essere il più scalabile e versatile possibile. Si è inoltre cercato di minimizzare la mole di dati memorizzata per ciascuna tabella in modo da evitare ridondanza ed evitare una diminuzione delle prestazioni. Le tabelle contenute nel database sono sei e sono illustrate nel modello logico della fig. 3.6. Il modello Entità-Relazione della fig. 3.5, invece, illustra le relazioni tra le varie tabelle[1].

[1]La tabella *Feature*, che relaziona *Model* con le tre tabelle dei tag *ObjectType*, *Color* e *Material*, non è stata creata in quanto ridondante. È stato sufficiente aggiungere l'ID dei tre tag corrispondenti all'interno della tabella *Model* stessa per fare riferimento alle altre tre tabelle.

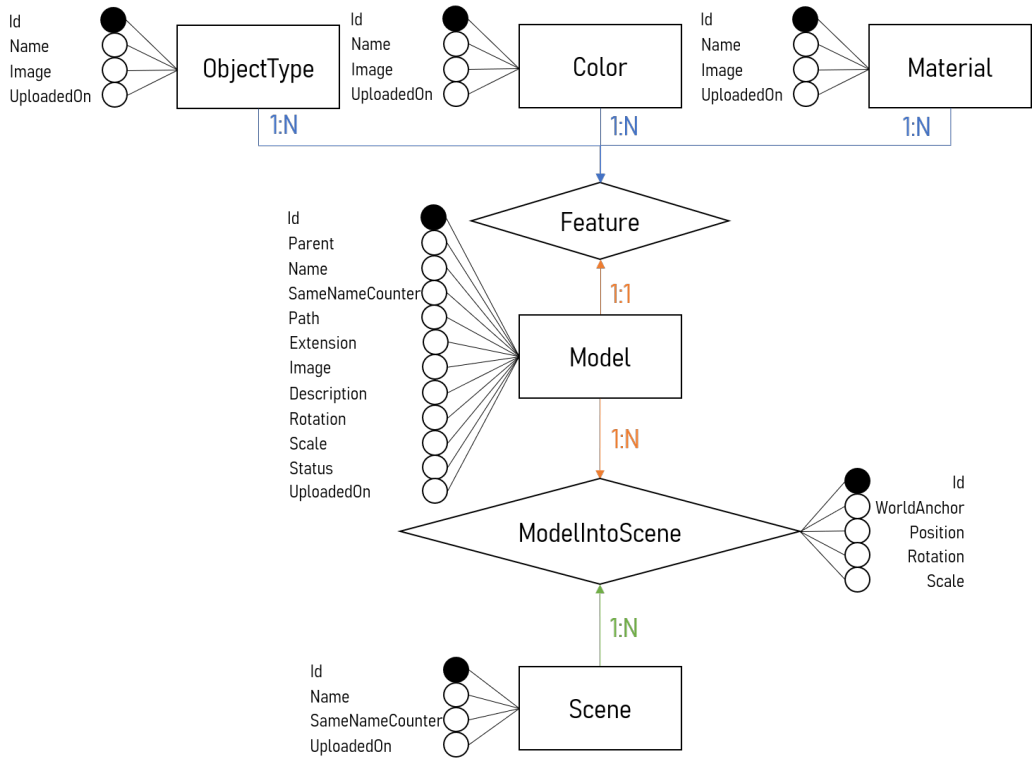

Figura 3.5: Modello Entità-Relazione del database.

Model(id, parent, name, same_name_counter, path, extension, base64_image,
description, object_type, color, material, rotation, scale, status, uploaded_on)
ObjectType(id, name, base64_image, uploaded_on)
Color(id, name, base64_image, uploaded_on)
Material(id, name, base64_image, uploaded_on)
Scene(id, name, same_name_counter, uploaded_on)
Model_into_scene(id, idscene, idmodel, world_anchor, position, rotation, scale, uploaded_on)

Figura 3.6: Modello logico del database.

Le tabelle presenti nel database sono:

- **Model:** la tabella Model contiene le informazioni legate ad un modello caricato nel database. Contiene sia i modelli base che quelli derivati (par. 5.5.4). I modelli base possono essere aggunti dall'applicazione *Asset Loader*, mentre i modelli derivati dall'applicazione *Set Builder*. Entrambe le tipologie di modelli, invece, possono essere modificate dall'applicazione *Set Builder*.

  È la tabella con il maggior numero di campi nel database, e sono:

  - **ID:** identifica in modo univoco il modello. Un modello base e i suoi modelli derivati hanno ID diversi fra loro.
  - **Parent:** se è uguale a zero, il modello è un modello base. Se è diverso da zero, il modello è un modello derivato e questo campo indica l'ID del modello base a cui fa riferimento.

- **Name:** il nome del modello. È un campo utilizzato per la ricerca testuale nell'applicazione *Set Builder* (par. 5.5.1).
- **Same name counter:** questo campo, di tipo int, è stato inserito per gestire modelli con nomi uguali in modo automatico. Quando viene inserito un nuovo modello con lo stesso nome di un modello già esistente, lo si salva con un valore di *same_name_counter* pari a max + 1, dove max è il più alto valore di *same_name_counter* presente nella tabella e relativo a quel nome. Nomi con un valore di *same_name_counter* pari a 1 manterranno inalterato il proprio nome, altrimenti verrà affiancato loro questo valore, il nuovo nome, pertanto, diverrà "*name same_name_counter*". Esempio: se ho due modelli con il nome "robot", quello con *same_name_counter* = 1 sarà visualizzato con il nome "robot", il secondo invece, con *same_name_counter* = 2, verrà mostrato con il nome "robot 2".
- **Path:** il nome del file .zip che contiene la mesh, le textures ed eventuali metadati del modello, così com'è salvato nel database.
- **Extension:** l'estensione del file a cui fa riferimento la tupla. Tipicamente si tratta di un file .zip, ma in caso di modelli privi di textures è possibile salvare direttamente i file .obj, .fbx, etc.
- **Image:** l'immagine di anteprima del modello, realizzata nell'applicazione *Asset Loader* al momento dell'upload (par. 4.3). L'immagine viene codificata in base64 e quindi in stringa, così da salvare l'immagine direttamente come campo all'interno del database. Solo i modelli base hanno un'immagine di anteprima, i modelli derivati hanno questo campo a NULL e utilizzano la stessa immagine del modello base. Quando nell'applicazione *Set Builder* viene effettuata una ricerca, verrà visualizzata, per i modelli base, l'immagine corrispondente al modello ricercato (par. 5.5.3). La stringa viene riconvertita in base64, poi in Image e infine in Texture2D, così da poter essere applicata come materiale nella casella della griglia dei risultati della ricerca per caricare il modello nell'applicazione per HoloLens.
- **Description:** la descrizione del modello. È un campo utilizzato per la ricerca testuale nell'applicazione *Set Builder* (par. 5.5.1).
- **Object type:** specifica l'ID corrispondente alla tipologia di oggetto che rispecchia il modello.
- **Color:** specifica l'ID del colore dominante del modello.
- **Material:** specifica l'ID del materiale dominante del modello.
- **Rotation/scale:** è possibile modificare la rotazione e la scala per future istanziazioni del modello, indipendentemente dalla scena in cui si trova il modello. Si tratta di 6 campi (3 per la rotazione, 3 per lo scalamento) di tipo float che indicano questa "correzione" da applicare al modello nel momento in cui viene istanziato nella scena.
- **Status:** se uguale a 1, il modello è valido e può essere utilizzato all'interno dell'applicazione. Se è zero, il modello non è più valido e viene ignorato e la tupla deve essere modificata o rimossa. È stato inserito questo campo per utilizzi futuri, viene usato come filtro dei modelli usati all'interno

delle applicazioni ma non è stato implementato un modo per settarlo a 0 se non modificando direttamente la tupla all'interno del database.

- **Uploaded on:** indica l'istante temporale in cui è stata effettuata l'ultima modifica alla tupla.

- **ObjectType/Color/Material:** sono le tre tabelle che contengono le informazioni dei tag. Queste informazioni sono:
  - **ID:** identifica in modo univoco il tag.
  - **Name:** il nome del tag.
  - **Image:** il path dell'immagine relativa al tag. Il path, interno al server, punta a un'immagine che verrà mostrata nella griglia per la scelta dei tag nel menu dei tag dell'applicazione *Set Builder* (par. 5.5.2)
  - **Uploaded on:** indica l'istante temporale in cui è stata effettuata l'ultima modifica alla tupla.

- **Scene:** contiene l'elenco di tutte le scene realizzate nell'applicazione *Set Builder* per HoloLens. Contiene solo i dati relativi alla scena e non ai modelli contenuti. I suoi campi sono:
  - **ID:** identifica in modo univoco la scena.
  - **Name:** il nome della scena.
  - **Same name counter:** come con il nome del modello, anche il nome della scena ha lo stesso metodo automatico per la gestione di nomi uguali.
  - **Uploaded on:** indica l'istante temporale in cui è stata effettuata l'ultima modifica alla tupla.

- **Model into scene:** è la tabella che esprime la relazione tra le tabelle *Model* e *Scene*, ovvero contiene le informazioni dei modelli contenuti in una scena salvata. Ogni tupla può quindi fare riferimento ai valori in *Model* e *Scene* corrispondenti agli ID. Più copie di uno stesso modello possono essere presenti in una stessa scena. I suoi campi sono:
  - **ID:** identifica in modo univoco la tupla.
  - **Model ID:** ID del modello a cui fa riferimento la tupla.
  - **Scene ID:** ID della scena in cui è stato istanziato questo modello.
  - **World Anchor:** la World Anchor del modello istanziato. Questo argomento viene approfondito nel par. 5.6.3.
  - **Position/rotation/scale:** le trasformazioni del modello così com'è stato posizionato, ruotato e scalato nella scena.

3.2.2 Mixed Reality ToolKit

Il **Mixed Reality ToolKit** [33] è un toolkit realizzato dalla community di sviluppatori di Microsoft che offre agli utenti degli strumenti utili per lo sviluppo di applicazioni per HoloLens.

Il toolkit per Unity consiste principalmente in due cartelle: quella contenente il toolkit vero e proprio, e quella con alcune scene d'esempio in cui viene usato questo toolkit.

Il toolkit a sua volta è suddiviso in diverse categorie di asset:

- **Boundary:** scripts che sfruttano le API di Unity per i confini, utili per renderizzare il pavimento per i dispositivi immersivi, oppure per verificare se un particolare oggetto rientra entro certi confini.
- **Build and deploy:** automation window di build e deploy per buildare soluzioni di Visual Studio, installare e avviare APPX e creare file di log.
- **Common:** contiene classi necessarie per la maggior parte delle classi delle altre cartelle del toolkit.
- **Input:** contiene un sistema di input completo di tutte le funzioni, che consente di gestire vari tipi di input e di inviarli a qualsiasi gameObject che si sta puntando con il cursore. Include anche alcuni cursori di esempio che sfrutta il sistema di animazione di Unity. Questo sistema di input si basa sull'utilizzo dell'EventSystem di Unity, perciò non è necessario un modulo di input personalizzato.
- **Sharing:** questa libreria consente a diverse istanze dell'applicazione di "collaborare" tra loro, e quindi di far collaborare anche più utenti fra loro in remoto. È stata pensata per l'utilizzo di più HoloLens sincronizzati tra loro, per migliorare lo strumento di pianificazione dei Mars rover, in particolare per permettere a più dispositivi di eseguire un task.
- **Spatial mapping:** scripts che sfruttano lo spatial mapping.
- **Spatial sound:** scripts relativi alle caratteristiche audio.
- **Spatial understanding:** scripts, prefabs e scene di prova che sfruttano le caratteristiche relative al riconoscimento spaziale.
- **Utilities:** componenti utili per rendere lo sviluppo e il debugging più semplice.
- **UX:** controlli legati alla User Experience, come i bottoni, utilizzabili nelle applicazioni.

La cartella degli esempi è suddivisa in altre categorie e contiene numerose scene che fanno uso di questi elementi, per facilitare la comprensione e rendere più familiari i toolkit da parte dell'utente che sta prendendo confidenza per la prima volta con questo tipo di programmazione.

Il repository del toolkit è scaricabile direttamente dall'account Microsoft di **GitHub** [34].

3.2.3 SDK e ambienti di sviluppo per Microsoft HoloLens

Un SDK (Software Development Kit) consiste in un insieme di strumenti per lo sviluppo e la documentazione di software. Microsoft Hololens fa uso del **Windows 10 SDK** [35], che fornisce gli headers, le librerie, i metadati e gli strumenti più recenti per realizzare build di applicazioni UWP (Universal Windows Platform), in questo caso di applicazioni di Mixed Reality con il sistema operativo di Windows 10 su Microsoft HoloLens.

Le applicazioni sono state realizzate su **Unity** [36] in C# usando come ambiente **Visual Studio 2017** [37], che utilizza il Windows 10 SDK per realizzare la build e caricarla direttamente su Microsoft HoloLens.

È stato preso spunto dalla guida offerta da *Microsoft* [38] per il settaggio delle impostazioni dell'applicazione su Unity per renderla compatibile con gli HoloLens, e per l'implementazione di alcune classi come *SpeechManager.cs*, per il riconoscimento vocale, e *GazeGestureManager.cs*, per l'identificazione degli oggetti puntati dal cursore, in seguito modificate ad-hoc per l'applicazione.

Uno degli obiettivi di questo progetto è il potenziale utilizzo del 5G per poter istanziare in tempo reale modelli che non sono presenti all'interno dell'applicazione ma all'interno di un server remoto. Unity non consente di importare modelli durante l'esecuzione dell'applicazione, né tantomeno di creare prefab da istanziare, se non con l'ausilio di oggetti come gli **AssetBundles** [39].

Gli AssetBundles sono contenitori di modelli identificati da un tag. Unity è in grado di inviare in remoto e ricevere questi contenitori da remoto impacchettando ed estrapolando i modelli contenuti in esso.

Inizialmente era stato adottato questo tipo di approccio, inserendo un solo modello all'interno di ogni AssetBundle, in seguito è stato abbandonato in quanto sarebbero stati complicati da gestire per esempio all'interno del database. L'approccio definitivo utilizzato all'interno di Unity è stato il plugin **TriLib** (B.1).

Capitolo 4

Asset Loader: caricamento del modello nel database

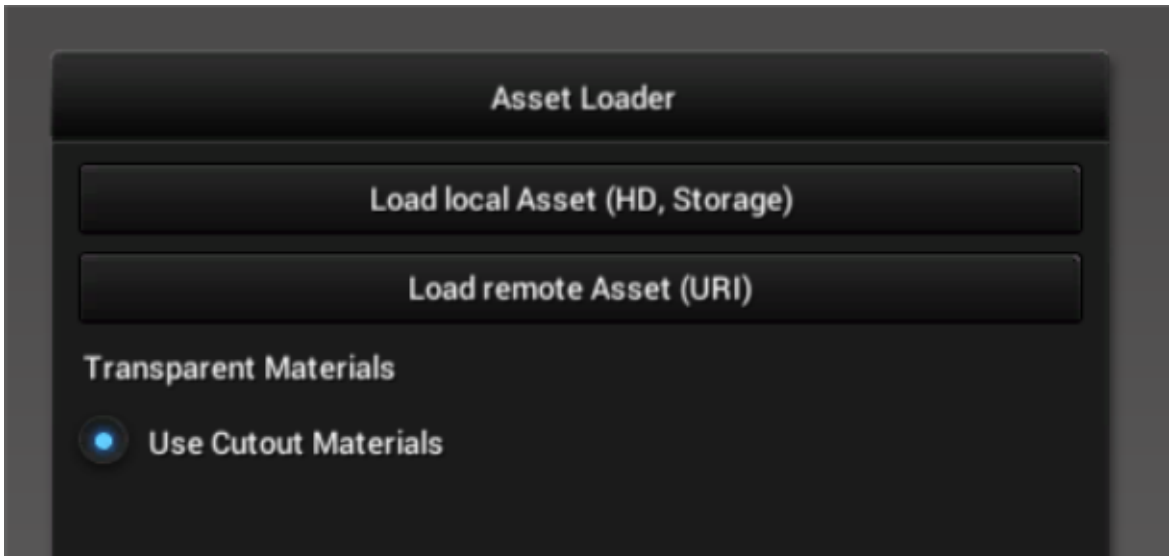

Figura 4.1: Schermata iniziale di **Asset Loader**.

La prima applicazione ha come obiettivo di essere un adapter fra il modello da caricare e il database, ovvero ha il compito di trasferire i modelli sul server, inserendo eventualmente alcune caratteristiche utili per la ricerca, ovvero la descrizione e i tag. Questo progetto è stato sviluppato su Unity (3.2.3) e la piattaforma d'esecuzione scelta è il PC.

Verrà illustrato il funzionamento della schermata iniziale, il caricamento del modello (4.1), il settaggio dei tag (4.2), la creazione dell'immagine di anteprima (4.3) e infine il caricamento del modello sul server (4.4).

4.1 La finestra di caricamento

Il progetto è stato realizzato sulla base di un progetto di esempio del plugin TriLib (vedi B.1), modificato ad-hoc per adattarlo agli scopi del progetto. La prima finestra visibile (fig. 4.1) permette di scegliere il metodo di caricamento del modello nel progetto:

- **Load Local Asset(HD, Storage):** questo bottone apre un'ulteriore finestra (fig. 4.2) che permette di navigare sul disco locale del computer per recuperare il path del modello da caricare in locale.

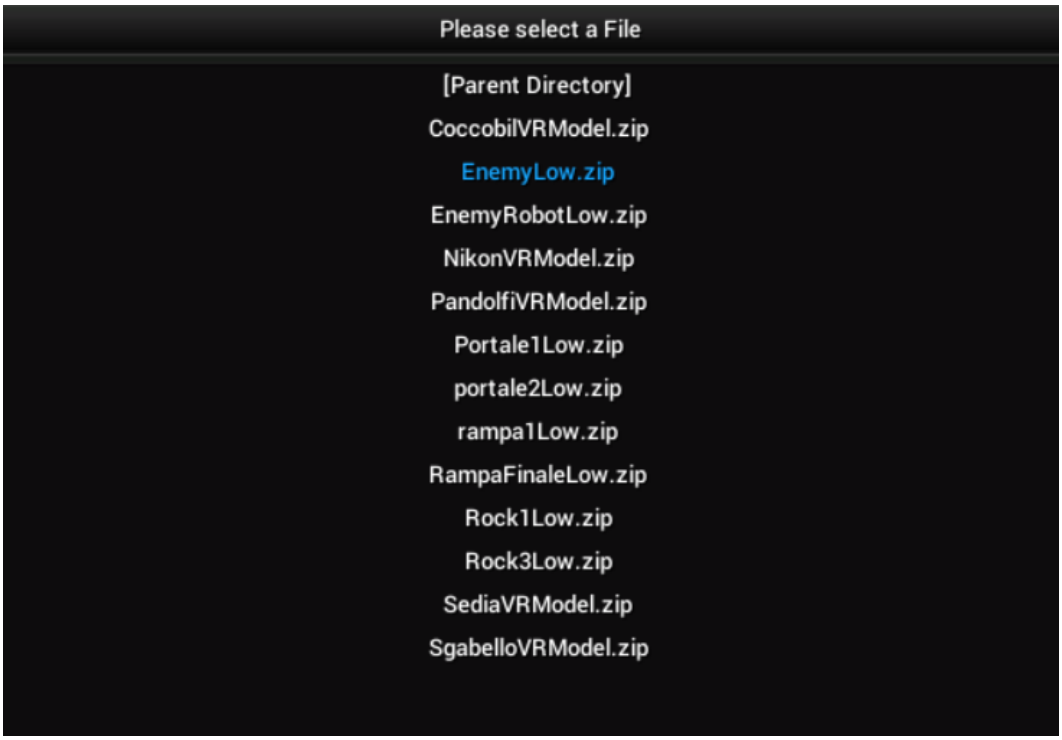

Figura 4.2: Schermata di selezione del modello in locale.

- **Load Remote Asset (URI):** questo bottone apre un'ulteriore finestra (fig. 4.3) che richiede l'inserimento di un URL dal quale andare a recuperare il modello.

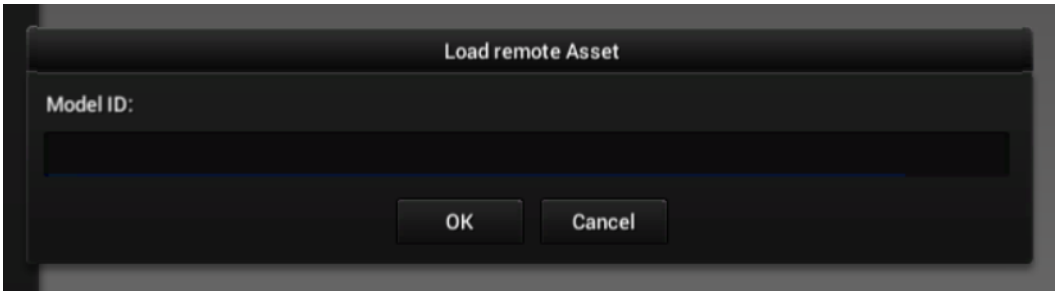

Figura 4.3: Schermata di selezione del modello in remoto.

Entrambi i metodi richiamano la classe *AssetLoaderWindowWithSearch*, che gestisce ciò che accade prima e dopo il download e l'upload di un modello, e a sua volta richiama la classe *AssetDownloaderWindowWithSearch*, che avvia la coroutine per il download effettivo del modello (fig. 4.4.

Come già spiegato nel paragrafo 3.2.3, è stato oggetto di ricerca la modalità di creazione del prefab nella scena, in quanto Unity non consente l'importazione nell'editor di modelli dall'esterno in tempo reale. TriLib recupera il path passato attraverso uno dei due bottoni sopra citati e ricostruisce la mesh del modello e le sue textures. La costruzione del modello viene effettuata in tempi molto ridotti, in meno di qualche secondo.

Una volta ricostruito il modello, sarà subito visualizzabile all'interno dell'ambiente con la texture applicata.

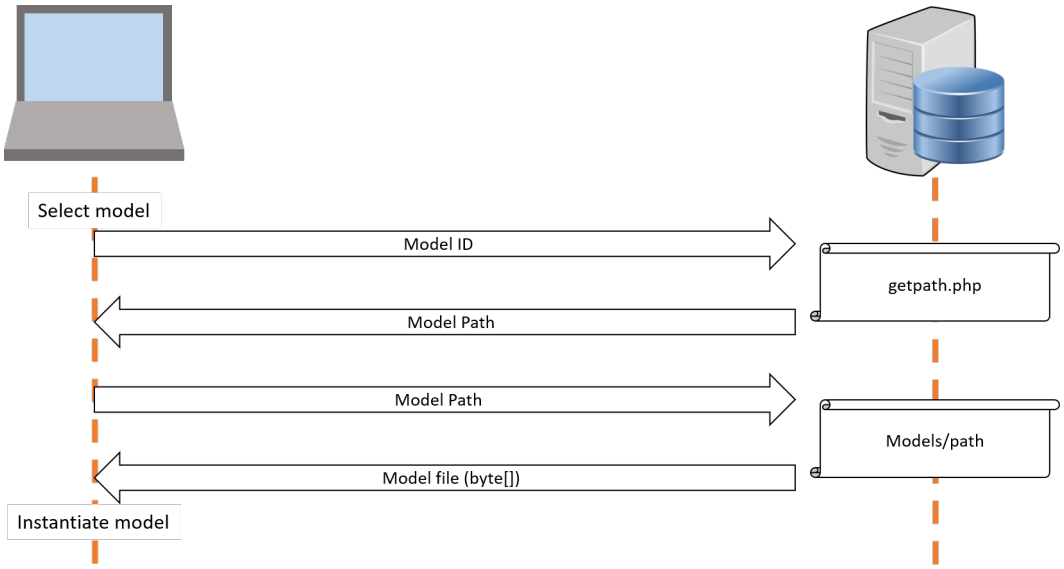

Figura 4.4: Flusso di esecuzione quando viene istanziato un modello remoto.

4.2 Settaggio dei tag

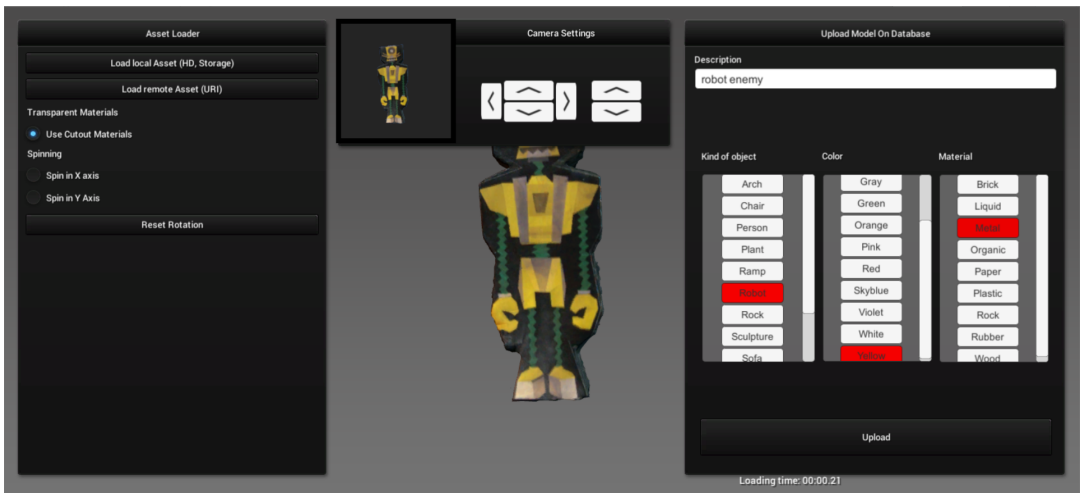

Figura 4.5: Schermata di settaggio delle caratteristiche del modello prima di caricarlo sul server

Una volta caricato il modello in scena, la classe *FillScrollViews* si collega al server per recuperare i tag salvati in precedenza nel database tramite lo script *getscroll-viewlists.php* e li carica nella finestra *Upload Model on Database* posta a destra della schermata.

I tag vengono raggruppati in tre principali categorie:

- **Tipo di oggetto:** identifica la tipologia di oggetto, se si tratta di una sedia, un divano, un tavolo, ecc.
- **Colore:** per ogni oggetto è possibile selezionare il colore predominante.
- **Materiale:** il materiale di cui è composto principalmente l'oggetto (plastica, metallo, carta, ...).

Per ognuna delle tre categorie è possibile salvare sul database un numero illimitato di elementi, ma è possibile scegliere non più di un tag per ognuna delle tre categorie. È anche possibile non selezionare nessun tag, oppure selezionarne solo per alcune categorie. Oltre alla scelta dei tag, è possibile inserire una **descrizione**, facoltativa, del modello in questione. Il testo contenuto nella descrizione verrà utilizzato nell'applicazione per Hololens per la ricerca testuale dei modelli (par. 5.5).

4.3 Creazione dell'immagine di anteprima del menu di ricerca

È possibile personalizzare l'immagine che verrà visualizzata nella griglia di anteprima 5.5.3 tra i risultati della ricerca nell'app Hololens.

Una **telecamera**, posta di fronte al modello caricato, lo riprende frontalmente in modo prospettico, escludendo dalla sua vista tutti gli altri oggetti esterni (ovvero l'interfaccia grafica). È possibile visualizzare ciò che la telecamera riprende da un'**immagine raw** in una nuova finestra posta in alto al centro della schermata (fig. 4.6). L'immagine visualizzata sarà l'icona che verrà salvata per il modello.

È possibile regolare la posizione della telecamera utilizzando le frecce nella stessa finestra, sotto la dicitura *Camera Settings*, grazie alla classe *CameraHandle*. I movimenti possibili sono traslazioni lungo gli assi orizzontale e verticale[1] e lo zoom-in/zoom-out.

Per aggiustare la posizione dell'oggetto, si fa uso di due nuovi bottoni nella sezione *Spinning* nella finestra iniziale di selezione del modello a sinistra della schermata. In questa sezione, è possibile ruotare l'oggetto, in senso orario, lungo gli assi X e Y relativi dell'oggetto.

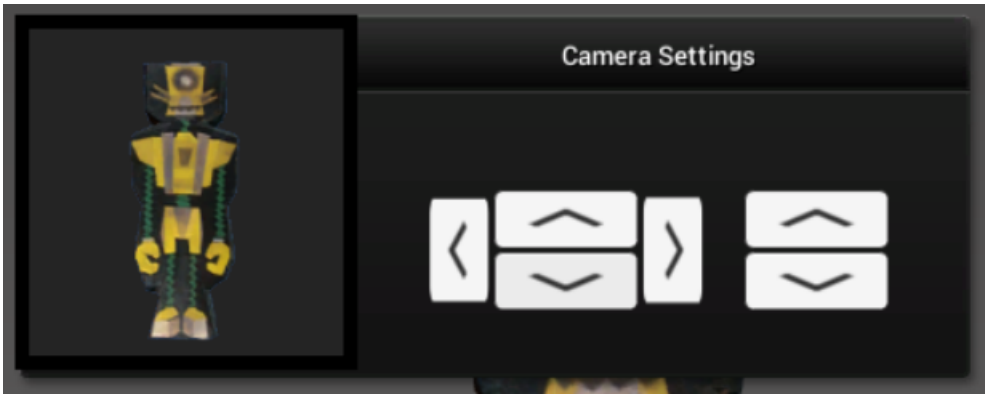

Figura 4.6: Particolare della finestra delle impostazioni della telecamera.

[1]Per assi orizzontali e verticali, in questa applicazione, si intendono rispettivamente gli assi X e Y dell'editor.

4.4 Caricamento sul server

Una volta scelti i tag, inserito una descrizione e creata l'immagine di anteprima dell'oggetto, per caricare il modello sul server si clicca sul bottone *Upload* posto in fondo alla finestra *Upload Model on Database* (fig. 4.7). Cliccando su questo bottone, si attiva la classe *AssetUploader* che invia, tramite lo script *upload.php*, il modello al server insieme agli ID dei tag, alla descrizione e all'immagine di anteprima.

L'immagine è un oggetto Texture2D codificato prima in JPG, che consente una maggiore compressione dei dati multimediali, e poi convertita in una stringa in base64, così da avere l'immagine direttamente nella tupla dell'oggetto e non in un file nel server. Questo processo viene effettuato dalla classe *TakePhotoFromCamera*.

Il nuovo modello viene quindi inserito nella tabella *model* con un ID generato automaticamente. L'ID è autoincrementale, ovvero viene assegnato un ID pari al valore dell'ID massimo contenuto nella tabella + 1.

Un dialog informerà l'utente se l'inserimento è avvenuto con successo o se ha riscontrato dei problemi.

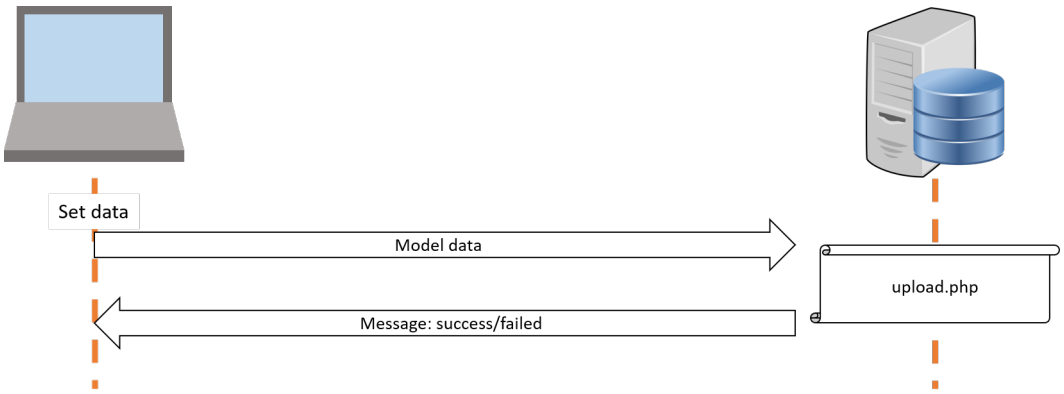

Figura 4.7: Flusso di esecuzione quando viene caricato il modello sul server.

Capitolo 5

Set Builder: l'applicazione per HoloLens

L'applicazione sviluppata per HoloLens ha come obiettivo la realizzazione di un set televisivo virtuale in un ambiente reale, utilizzando come input gestures o comandi vocali (5.2), con la possibilità di modificare parametri generali dell'applicazione come l'indirizzo IP del server o l'abilitazione dei comandi vocali (5.4), di effettuare la ricerca di oggetti di scena dal database (5.5), di gestire le loro trasformazioni, quindi traslazione, rotazione e scala (5.6.1), pur rimanendo legato alla realtà tramite le world anchor (5.6.3), ma anche i dettagli dell'oggetto, con la possibilità di modificare l'oggetto corrente o di crearne uno nuovo nel database (5.6.2), e infine di salvare il set corrente o di caricarne uno esistente salvato in precedenza sul database (5.7).

5.1 L'interfaccia dell'applicazione

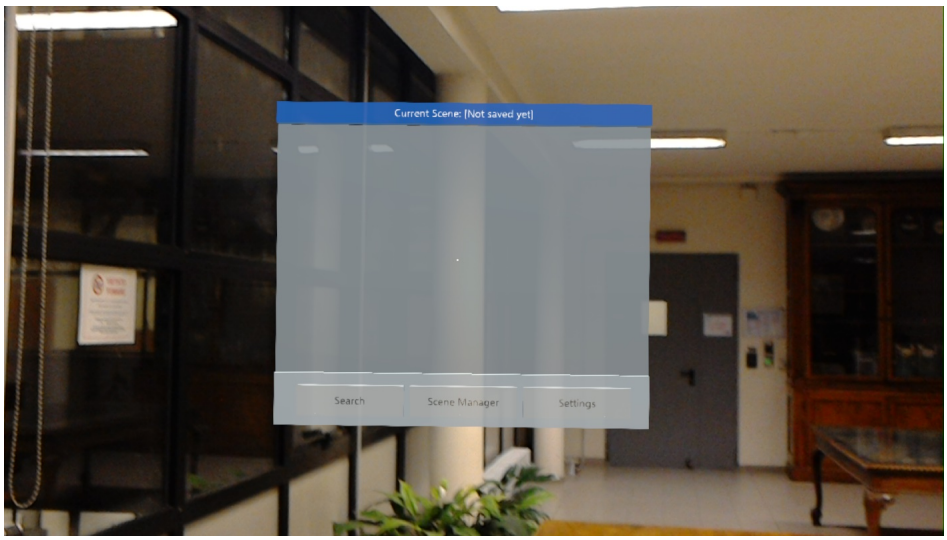

Figura 5.1: Schermata iniziale di **Set Builder**.

All'avvio, l'utente vedrà di fronte a sé una finestra vuota (fig. 5.1) sulla quale possono comparire, uno per volta, i diversi menu descritti nel corso di questo capi-

tolo[1] (par. 5.4, 5.5, 5.5.2, 5.6.2, 5.7). Ciascun menu è apribile o tramite interazione dell'utente con i bottoni interattivi posti in fondo alla finestra principale utilizzando gestures o, in alternativa, utilizzando specifici comandi vocali (par. 5.2). Il menu di impostazioni del modello, a differenza degli altri menu, è apribile tramite gestures cliccando direttamente sul modello interessato e non interagendo con i bottoni della finestra principale. Il menu dei tag, invece, è un sottomenu apribile sia dal menu per la ricerca di un modello, per specificare i tag per la ricerca, che dal menu delle impostazioni dell'oggetto, per settare i tag di uno specifico modello.

La finestra segue lo sguardo dell'utente grazie al component *Billboard* dell'HoloToolkit, ed è possibile spostarla afferrandola e poi trascinandola dalla barra blu in alto grazie al component *Hand draggable*.

La barra blu superiore contiene il nome della scena attualmente aperta. Se il nome della scena è racchiusa tra parentesi quadre, le modifiche effettuate alla scena non sono state salvate. La gestione del nome della finestra viene effettuata dalla classe *SceneNameManager*.

5.2 I comandi vocali

L'utente potrebbe non essere a suo agio ad usare solo le gestures come input, in particolare per un uso prolungato dell'applicazione per il quale il braccio potrebbe sentirsi affaticato. Per questo motivo, sono stati inseriti i comandi vocali, che, d'altro canto, potrebbero essere difficili da interpretare in ambienti rumorosi. Per evitare ciò, è possibile disattivare gli input vocali nel menu delle impostazioni della scena (par. 5.4).

Per unire i vantaggi dei due metodi di input, si è pensato di dare la possibilità di sfruttare entrambe le modalità.

Esistono una serie di comandi vocali generici e altri più specifici, entrambi specificati nella classe *SpeechManager*. Viene attivato un oggetto di tipo **KeywordRecognizer**, inizializzato all'avvio dell'applicazione con una serie di keywords generiche, cioè valide per diverse categorie di bottoni, oppure specifiche, cioè legate alla label del bottone. Per alcuni comandi generici è necessario individuare l'oggetto puntato dal cursore: a questo scopo si fa uso della classe *GazeGestureManager*.

I **comandi generici** sono i seguenti:

- **Ok:** comando generico che esegue la action del bottone "ok" delle finestre di dialog oppure conferma una data operazione.
- **Close:** comando generico che chiude qualunque finestra aperta.
- **Search:** apre il menu di ricerca del modello (par. 5.5).
- **Set tags:** apre il menu di selezione dei tag, sia per la ricerca del modello (par. 5.5.1) che per la modifica delle impostazioni.
- **Left/Right e Back/Next:** scorre avanti o indietro le pagine dell'elenco quando viene ricercato un modello o una scena.

[1]La classe *Global* gestisce le variabili globali dell'intera applicazione, per facilitare l'interazione tra tutte le classi e, quindi, tra i diversi menu dell'applicazione.

- **Model settings:** se il cursore punta un modello istanziato, apre il menu di impostazioni relativo al modello in scena puntato (par. 5.6.2).
- **Scene:** apre il menu di gestione delle scene, per poterle salvare, caricare o eliminare (par. 5.7).
- **Load:** nel menu di salvataggio/caricamento delle scene, apre il sottomenu che recupera le scene salvate sul server e le mostra all'utente (par. 5.7.3).
- **Save:** nel menu di salvataggio/caricamento delle scene, apre il sottomenu per modificare il nome della scena corrente per poi successivamente salvarla (par.5.7.2).
- **Save new:** nel menu delle impostazioni del modello o di salvataggio delle scene, salva un nuovo modello o una nuova scena sul database. Nel caso di salvataggio di un nuovo modello, crea un oggetto derivato senza icona, con la stessa mesh dell'oggetto base ma con nome, descrizione, tag e trasformazioni diverse (par. 5.5.4).
- **Save edited:** nel menu delle impostazioni del modello o di salvataggo delle scene, salva le modifiche ad un modello o ad una scena già presente sul database.
- **Settings:** apre il menu delle impostazioni della scena (par. 5.4).

I **comandi specifici** sono legati ai singoli bottoni. Un metodo in *SpeechManager* legge la label contenuta nei bottoni di tutte le finestre e la registra come keyword del keywordRecognizer. In caso di keywords duplicate, si mantiene un'unica keyword aggiungendo eventualmente una nuova action alla keyword. Infatti, la action non verrà attivata se il bottone non è visibile, pertanto verrà attivata al più una sola action della keyword. Non sono mai visibili contemporaneamente nella scena due bottoni con la stessa label.

5.3 La finestra di dialog

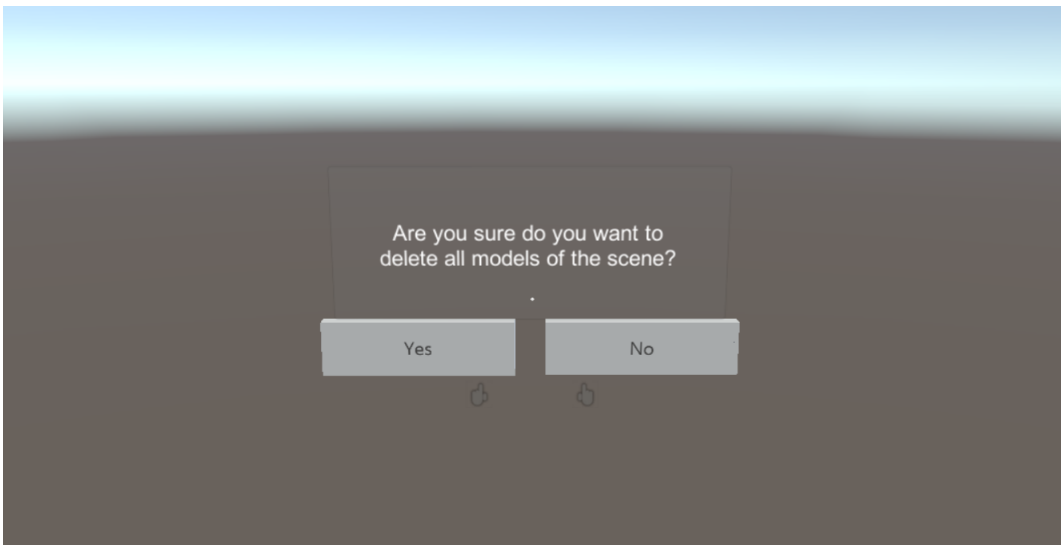

Figura 5.2: Esempio di finestra di dialog nell'applicazione. In questo caso, l'utente ha selezionato il bottone per la cancellazione di tutti i modelli in scena, il dialog quindi chiede conferma all'utente prima di effettuare un'operazione così "pericolosa".

La finestra di dialog compare a seconda se l'applicazione ha bisogno di segnalare dei messaggi, di successo o di errore, all'utente. Segue gli spostamenti dell'utente grazie ai components Billboard e Tagalong. Il contenuto della finestra di dialog viene gestito dai metodi statici della classe *LoadingWindowManager*:

- **StartLoading()/EndLoading():** mostra un dialog rispettivamente senza e con il bottone di "ok", utilizzato per chiudere la finestra. Richiamati durante i caricamenti dei modelli.
- **ShowMessage():** mostra un dialog con il bottone di "ok" per chiuderlo.
- **ShowDialogForModelsDeletion():** come *ShowMessage()*, ma mostra due opzioni, *yes* e *no* (fig. 5.2). Viene usato per richiedere la conferma per la cancellazione di tutti i modelli della scena.
- **ShowDialogForSceneDeletion():** come *ShowDialogForModelsDeletion()*, ma viene usato per richiedere la conferma per la cancellazione di una scena (5.7.3).

Ognuno di questi metodi riceve un parametro di tipo string corrispondente al messaggio da mostrare nel dialog.

5.4 Menu delle impostazioni della scena

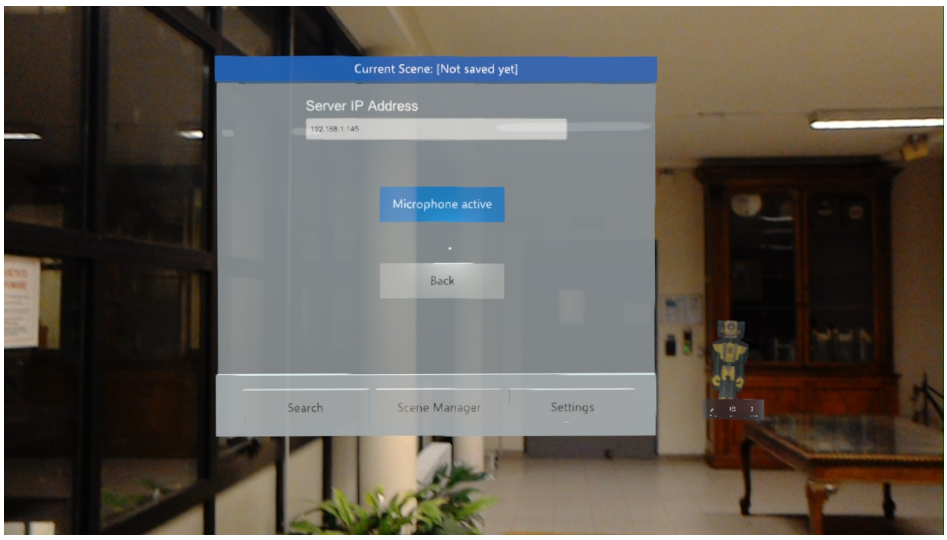

Figura 5.3: Menu delle impostazioni della scena.

Il menu delle impostazioni della scena (fig. 5.3) è apribile cliccando sul bottone "*Settings*" della finestra principale oppure con il comando vocale "*Settings*".

Obiettivo di questo menu è il settaggio di alcuni parametri globali di tutta l'applicazione, in particolare:

- **L'indirizzo IP del server:** è necessario specificare il base path del server in cui andare a recuperare la cartella di base contenente tutti i dati legati all'applicazione (l'app. A.1 illustra il funzionamento del server di XAMPP). Il campo è un InputTextField, con il quale è possibile scrivere, tramite tastiera virtuale, il base path. Il valore viene inizializzato dal metodo *UpdateIPAddressInputField()* della classe *EditInputField* e modificato dal metodo *SetIPAddress()* della stessa classe.

- **L'attivazione del microfono:** per abilitare o no il microfono per i comandi vocali (par. 5.2). Potrebbe essere indispensabile, in caso di utilizzo dell'applicazione in ambienti rumorosi, la sua disabilitazione. Il campo è un bottone di tipo InteractiveToggle, che sarà selezionato o no a seconda se il microfono è attivato o disattivato. Sulla base di questo valore, la classe *MicrophoneManager* gestirà l'attivazione del microfono.

Le impostazioni vengono salvate automaticamente una volta modificati i campi. Il salvataggio viene effettuato sia nell'applicazione per la sessione corrente che nel file *mypreferences.txt*, all'interno del dispositivo, così da salvare le impostazioni per successivi utilizzi dell'applicazione. La gestione del salvataggio e del caricamento delle impostazioni è gestito dalla classe *MyPreferences*.

Per chiudere il menu, l'utente può cliccare sul bottone "*Back*" oppure usando i comandi vocali "*Back*" o "*Ok*".

5.5 Menu per la ricerca di un modello

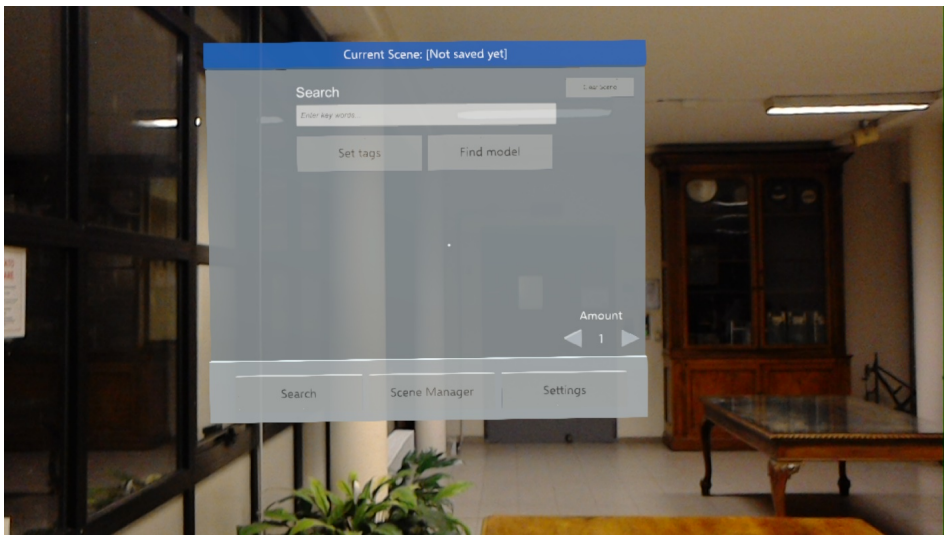

Figura 5.4: Menu per la ricerca di un modello.

Obiettivo di questo menu è la ricerca di un modello specifico nel database usando determinati parametri per la ricerca (par. 5.5.1, 5.5.2) così da poterne istanziare un determinato numero di copie specificate dal valore di "*Amount*" in basso a destra.

Per aprire il menu per la ricerca di un modello si può cliccare sul bottone "*Search*" della finestra principale o usare il comando vocale "*Search*".

5.5.1 I parametri per la ricerca

La finestra mostra due campi da settare, utilizzati per la ricerca del modello:

- **Keywords:** in questo InputFieldText è possibile specificare alcune parole chiave presenti o nel nome del modello e/o nella sua descrizione.
- **Set tags:** cliccando su questo bottone si apre il menu dei tag, dove è possibile selezionarne il tipo di oggetto, il colore e il materiale. Lo switch tra il menu della ricerca e il menu dei tag è gestito dalla classe *SwitchMenuWindows*.

Questi campi sono tutti facoltativi, e possono anche essere lasciati non settati. In questo caso, la ricerca restituirà, senza filtri, tutti i modelli presenti nel database. Per avviare la ricerca, è sufficiente cliccare sul bottone "*Find Models*".

5.5.2 Il menu dei tag

Figura 5.5: Menu per la selezione dei tag.

Il menu dei tag (fig. 5.5) permette di impostare le tre caratteristiche intrinseche dell'oggetto:

- **Tipo di oggetto:** la tipologia dell'oggetto (es. sedia, tavolo, pianta, etc.).
- **Colore:** il colore predominante dell'oggetto.
- **Materiale:** il materiale di cui è composto principalmente l'oggetto.

I tre tag appartenenti ad un modello possono essere inizializzati ed associati ad un modello in due modi:

- Se si tratta di un modello base, i tag vengono scelti al momento del caricamento del modello nel database (par. 4.2).
- Se si tratta di un modello derivato da un modello base, i tag vengono selezionati nel menu delle impostazioni dell'oggetto, prima di salvare una nuova versione del modello base (par. 5.6.2).

In entrambi i casi, è sempre possibile modificare i valori delle tre categorie importando un modello in scena, modificandone i parametri e salvandone i cambiamenti. Questi cambiamenti si ripercuoteranno su tutti i modelli dello stesso tipo.

Si può scegliere un solo tag per ognuna delle tre categorie, e non è obbligatorio che un modello abbia impostati tutte e tre le tipologie di tag.

La classe *InitTagsGrids* richiama la classe *FillScrollViews*, che recupera dal database l'elenco dei tag grazie allo script *getscrollviewlists.php*, e istanzia le tre liste nelle tre griglie (fig. 5.6). Ciascun elemento della lista è il prefab *Buttontags* e consiste in un bottone quadrato contenente l'immagine legata a quel tag e nella parte inferiore una TextMesh con il loro nome. Se il menu viene aperto attraverso il menu di impostazioni di un modello, verranno selezionati automaticamente i tag legati a quel modello. La griglia di oggetti verrà visualizzata una volta ultimato il caricamento di tutti gli oggetti.

La classe *Settags* modifica l'aspetto delle griglie quando l'utente seleziona uno qualunque degli elementi delle griglie. Se in quella categoria non è stato selezionato nessun tag, il tag scelto diventa il tag selezionato. Se in quella categoria è già stato selezionato un tag in precedenza, il nuovo tag sostituirà il vecchio e diventerà il nuovo tag selezionato. Se si sceglie il tag già selezionato, questo verrà deselezionato.

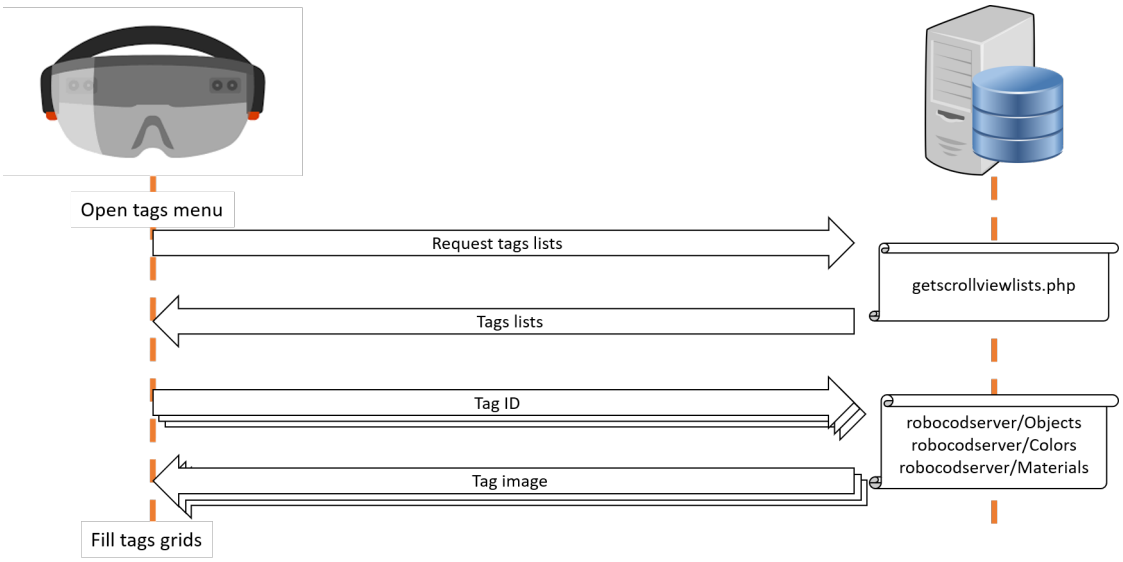

Figura 5.6: Flusso di esecuzione per recuperare i tag salvati nel database.

5.5.3 Griglia dei risultati della ricerca

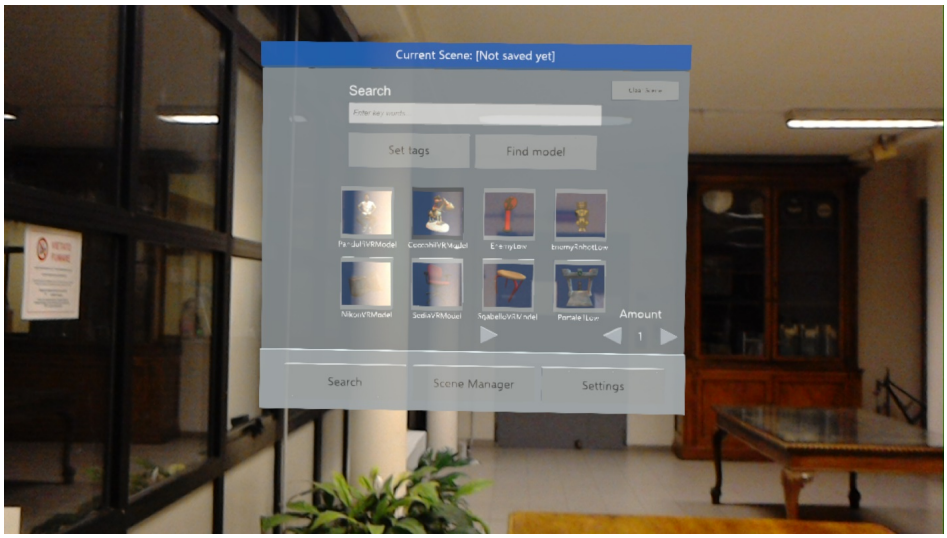

Figura 5.7: Schermata con i risultati di una ricerca sulla base delle keywords e dei tag selezionati.

La ricerca dei modelli parte dal metodo *LoadQueryAssetButtonClick()* della classe *AssetLoaderWindowWithSearch* che gestisce la ricerca, il pre-load e il post-load dei modelli da istanziare, e poi dalla classe *AssetDownloaderWithText*, che si occupa di avviare le relative coroutines che interrogano il database e di avviare la costruzione del modello tramite i metodi del plugin *TriLib*. Il database viene interrogato tramite lo script *getthumbnails.php*, e quindi inizializza e istanzia i risultati nella scena.

I risultati della ricerca verranno visualizzati in una griglia ordinata (fig. 5.7) contenente dei blocchi raffiguranti un'immagine con l'anteprima del modello (par. 4.3) e una label sotto i blocchi con il nome del modello. I nomi e le immagini mostrate nei risultati della ricerca riguardano sempre il modello base, in quanto i modelli derivati (par. 5.5.4) non hanno un'immagine di anteprima.

Nel caso in cui i modelli trovati superino il valore massimo di oggetti che può contenere la griglia, compaiono in basso nella parte centrale della finestra base due frecce, utilizzare per scorrere le pagine dell'elenco di oggetti trovati. Il comportamento delle pagine e delle relative frecce viene gestito dalla classe *ArrowsManager*.

Per selezionare un oggetto tra quelli nell'elenco, si clicca sull'immagine di anteprima oppure si punta l'immagine e si usa il comando vocale "*Select*", che azionerà il metodo *OnObjectSelected()* della classe *ObjectInstantiateButton*. A questo punto, se il modello non ha modelli derivati, oppure se il filtro di ricerca è pertinente con uno solo dei modelli tra quello base e quelli derivati, il modello trovato viene istanziato subito. Altrimenti, se al modello selezionato fanno riferimento uno o più modelli derivati pertinenti con la ricerca, verrà visualizzato un ulteriore elenco di modelli che ne mostrerà solo i nomi (fig. 5.8). L'elenco dei modelli derivati viene recuperato dallo script *getchildren.php*. Anche in questo caso, se il numero di modelli derivati supera il valore massimo di oggetti che la griglia può contenere, verranno mostrate le due frecce in basso al centro per scorrere le pagine della lista di oggetti trovati.

5.5.4 Modelli derivati

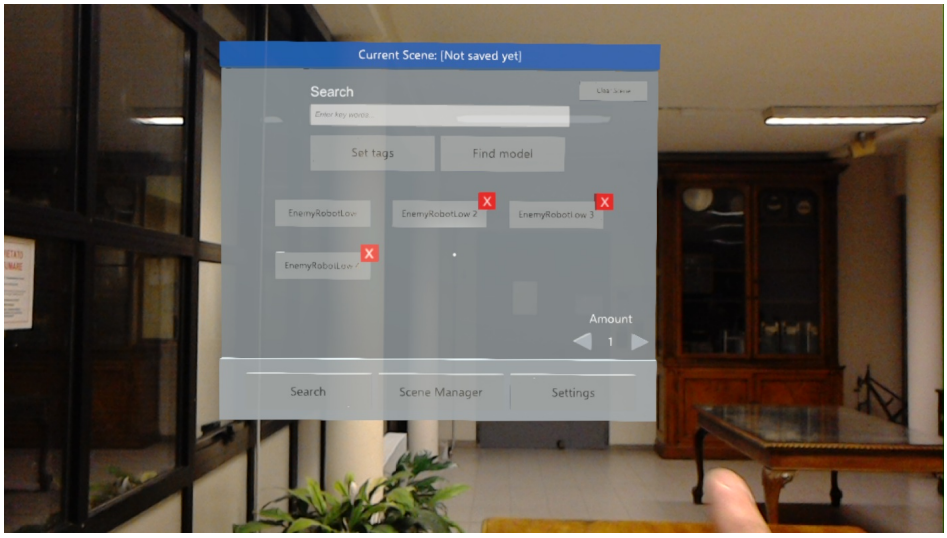

Figura 5.8: Una volta selezionato un modello tra quelli comparsi nella ricerca, si apre questa schermata con l'elenco dei modelli derivati pertinenti a quelli della ricerca.

I modelli derivati sono modelli base modificati. I modelli derivati hanno le stesse mesh e textures del modello base, ma possono avere nome, descrizione, tag, rotazione e scala differenti. Questo può essere utile nel caso si voglia tenere una copia modificata di un modello e usata ricorrentemente. La posizione non è contemplata nei parametri da salvare, in quanto è un valore mutabile in caso di più copie del modello in una stessa scena o in più scene.

La creazione di un modello derivato viene effettuata nel menu delle impostazioni del modello stesso[2] (par. 5.6.2).

È possibile eliminare un determinato modello derivato ricercandolo nel menu di ricerca (par. 5.5.3) e selezionando la "X" rossa sul bottone corrispondente al modello[3]. Una volta eliminato il modello, verranno eliminati anche i suoi riferimenti nelle scene salvate sul database e verranno rimosse tutte le eventuali istanziazioni nella scena aperta all'interno dell'applicazione. L'eliminazione del modello è gestita dalla classe *DeleteModelsManager*, che richiama lo script *deletemodel.php*.

[2]Se si crea un nuovo modello derivato a partire da un altro modello derivato, il nuovo modello prenderà come riferimento il modello base e non quello derivato. Non è quindi possibile creare ulteriori gerarchie di modelli.

[3]La "X" non compare per i modelli base, pertanto solo i modelli derivati possono essere eliminati direttamente dall'applicazione

5.5.5 Caricamento del modello in remoto e real-time

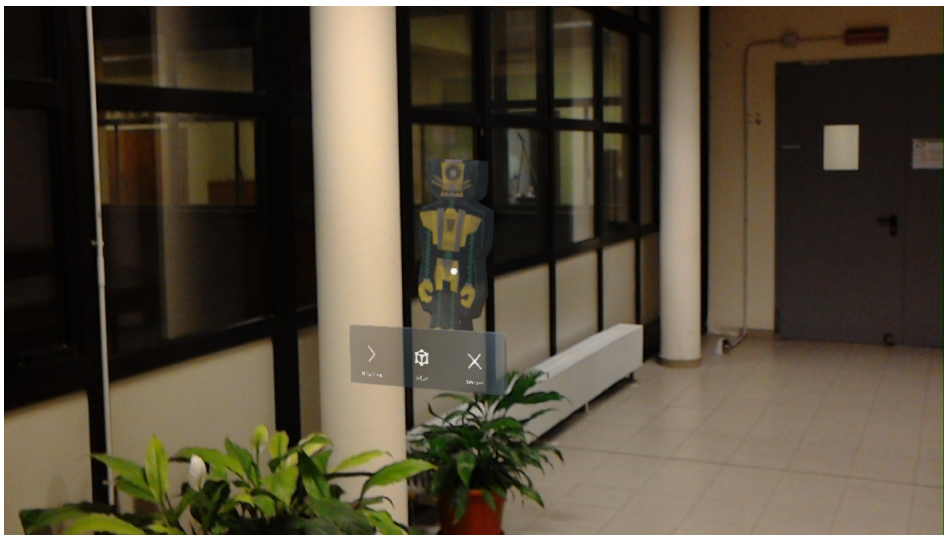

Figura 5.9: Modello istanziato nella scena insieme alla sua AppBar.

Una volta selezionato il modello, si attiverà il metodo *LoadRemoteAssetButtonClick()* di *AssetLoaderWindowWithSearch* che aprirà una finestra di dialog che informerà l'utente che il modello è in fase di caricamento. Partirà quindi la corrispondente coroutine in *AssetDownloaderWithText* che dapprima, sulla base dell'ID del modello selezionato, recupererà i dati del modello tramite lo script *receivefromdatabase.php*, dati che verranno successivamente memorizzati nella classe **ModelDataContainer**[4], unica per ogni modello istanziato. Tra questi dati è presente il path relativo del modello da recuperare su server. Si recupera tale modello e lo si ricostruisce in un nuovo prefab di Unity, e quindi viene istanziato.

La costruzione del prefab viene effettuata in modo asincrono partendo da zero, come illustrato nel paragrafo B.1, aggiungendo prima la mesh e poi le textures. Il caricamento del modello in scena potrebbe metterci qualche secondo a seconda del numero di poligoni della mesh o della dimensione delle textures.

Il metodo *PostLoadSetup()* di *AssetLoaderWindowWithSearch* inizializza alcuni parametri del modello, come i mesh collider e le trasformazioni iniziali dell'oggetto come somma dei valori salvati nel database e dei valori di correzione preimpostati per il corretto posizionamento del modello in scena. In caso di più copie del modello da istanziare, i modelli verranno istanziati uno in fila all'altro, con una leggera variazione preimpostata della posizione. La gestione delle quantità di oggetti dello stesso tipo da istanziare viene effettuata nella classe *AmountManager*. Una volta instanziati (fig. 5.9), oppure in caso di errore, la finestra di dialog notificherà l'esito delle operazioni all'utente.

Nella figura (fig. 5.10) viene riassunto l'intero processo di esecuzione dalla ricerca del modello alla sua istanziazione in scena.

[4]La classe *ModelDataContainer* si appoggia ad un'altra classe, *CurrentModelData*, che contiene la sua stessa tipologia di dati. La differenza è che *ModelDataContainer* è un contenitore dei dati, proprio di ogni modello istanziato, mentre *CurrentModelData* contiene le variabili statiche dell'oggetto su cui attualmente si sta lavorando, ad esempio quando viene istanziato un modello o quando si cambiano le sue impostazioni.

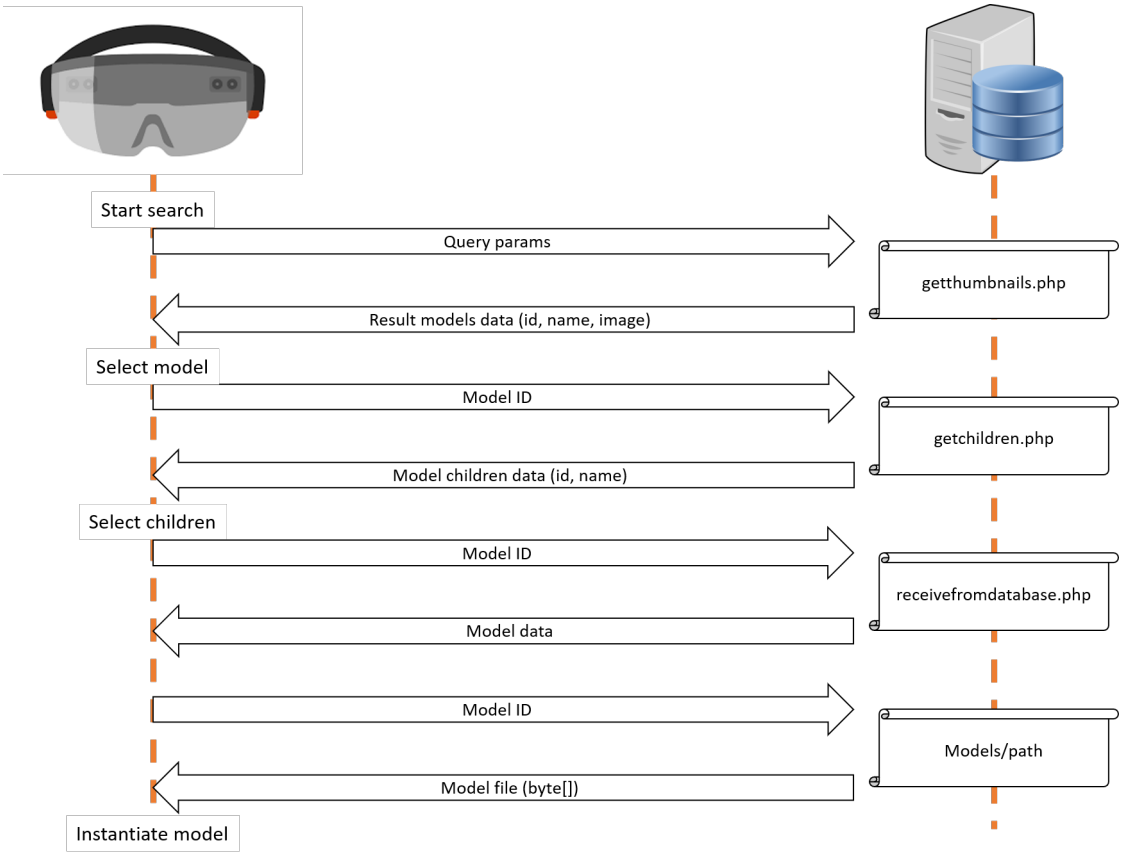

Figura 5.10: Flusso di esecuzione quando viene ricercato e poi caricato un modello salvato nel database.

5.5.6 Eliminazione rapida di tutti i modelli in scena

Potrebbe essere scomoda l'eliminazione di tutti gli oggetti della scena se fatta uno per uno, soprattutto se la scena presenta un gran numero di oggetti. In questo caso, è sufficiente cliccare sul bottone "*Clear Scene*", che azionerà i metodi della classe *DeleteModelsManager*. Si aprirà un dialog che chiederà conferma all'utente dell'azione appena selezionata. Se l'azione verrà confermata, tutti i modelli istanziati nella scena verranno istantaneamente eliminati [5].

[5]La scena nel database non verrà modificata finchè non verrà salvata dall'utente (par. 5.7.2).

5.6 Gestione del modello

5.6.1 Trasformazioni nella scena

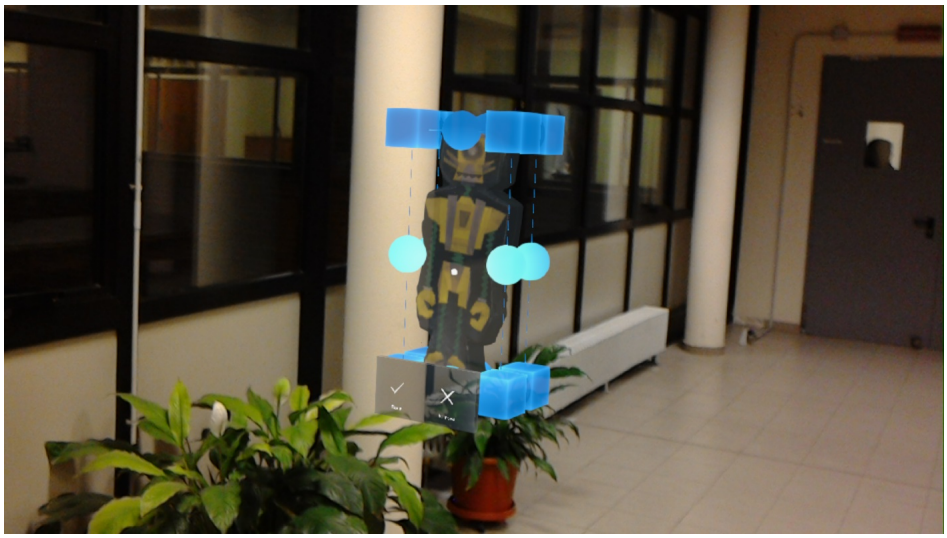

Figura 5.11: Bounding box di un modello istanziato.

Una volta istanziato il modello nella scena, è possibile manipolarlo nell'ambiente, in particolare è possibile svolgere le tre trasformazioni principali:

- **Traslazione:** afferrando il modello con il gesto di grab, è possibile spostarlo nello spazio. Ciò è possibile grazie al component *Hand Draggable* dell'HoloToolkit. Con le world anchor (par. 5.6.3), in futuro questo movimento sarà limitato.
- **Rotazione e scalamento:** il component *BoundingBoxRig* dell'HoloToolkit associa una **AppBar** ad ogni modello istanziato (fig. 5.11). La AppBar presenta diverse opzioni:
  - **Show/Hide:** mostra/nasconde la AppBar.
  - **Adjust:** mostra/nasconde il reticolo per effettuare la rotazione e lo scalamento lungo gli assi relativi del modello. Per effettuare la rotazione, è necessario afferrare, e quindi trascinare, una delle sfere lungo i lati del reticolo. Per lo scalamento il procedimento è analogo, ma bisogna afferrare i cubi ai vertici del reticolo.
  - **Delete:** elimina il modello associato.

5.6.2 Menu delle impostazioni del modello

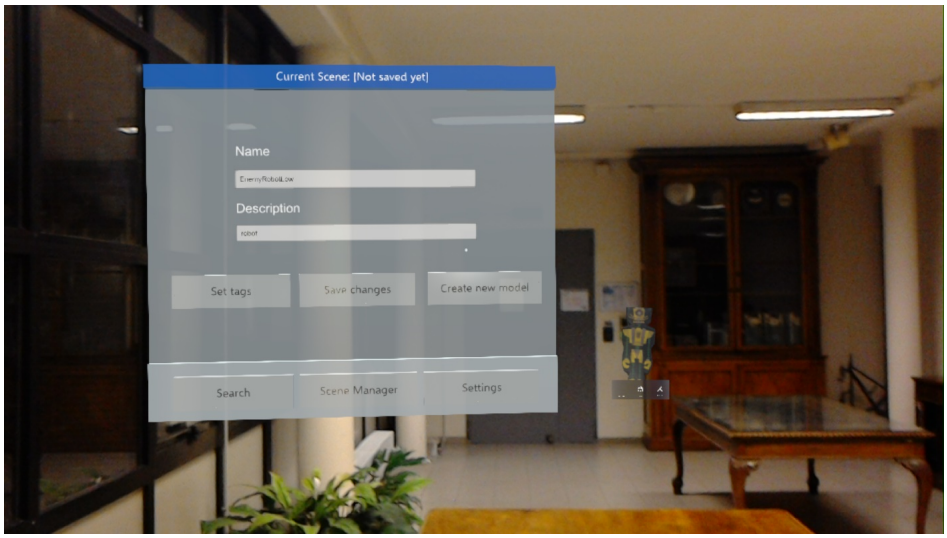

Figura 5.12: Menu delle impostazioni legate ad uno dei modelli istanziati.

È possibile aprire il menu delle impostazioni di un modello istanziato in due modi: si può fare tap sul modello oppure, dopo averlo puntato con il cursore, si usa il comando vocale *Model Settings*.

Il menu viene inizializzato grazie alla classe *InitSettingsWindow*, mentre le classi *SettingsWindow* e *EditInputField* gestiscono gli InputTextField.

Il menu presenta diversi campi:

- **Name:** InputTextField nel quale è possibile modificare il nome del modello. Le variazioni di questo campo vengono gestite dalla classe *EditInputField*. Non è possibile modificare la numerazione del modello in caso di omonimie, che invece viene gestita in modo automatico.

- **Description:** InputTextField nel quale è possibile modificare la descrizione legata al modello. Anche in questo caso, le variazioni di questo campo vengono gestite dalla classe *EditInputField*. Come spiegato nel paragrafo 5.5.1, la ricerca per parola chiave viene effettuata sia sul nome che sulla descrizione.

- **Set tags:** bottone che apre un menu identico a quello descritto nel paragrafo 5.5.2, con la differenza che, alla sua apertura, selezionerà automaticamente i tag già assegnati in precedenza a quel modello.

Qualunque modifica effettuata su uno di questi campi andrà persa se non verranno salvate le modifiche del modello prima ancora di chiudere questo menu o di aprirne un altro.

Il salvataggio viene effettuato dalla classe *SaveModelOnDB*, che implementa due diverse coroutines per due diversi approcci (fig. 5.13):

- **Save changes:** attiva la coroutines richiamando lo script *saveeditedmodel.php*, che salva i cambiamenti di quel modello nel database. Le future istanziazioni di quello stesso modello avranno le stesse impostazioni dell'ultima modifica salvata.
- **Save new model:** la coroutines richiama lo script *upload.php*, che crea una copia del modello, ovvero un modello derivato, illustrato nel paragrafo 5.5.4. Il nuovo modello avrà le stesse caratteristiche del modello base, ma il nome, la descrizione, i tag, la rotazione e la scala saranno differenti.

In entrambi i casi, una finestra di dialog (par. 5.3) informerà l'utente dell'esito delle operazioni.

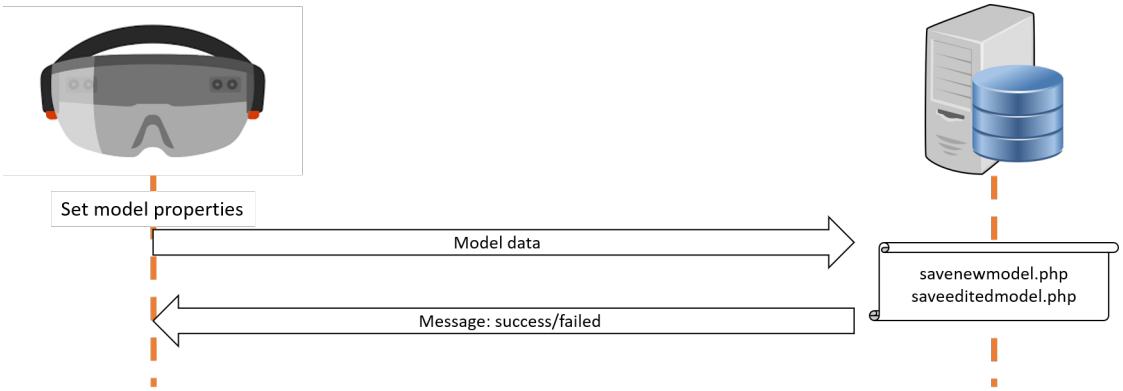

Figura 5.13: Flusso di esecuzione quando viene salvato un nuovo modello derivato o quando vengono salvate le modifiche ad un modello esistente.

5.6.3 Le World Anchor

Le **world anchor** [40] sono oggetti presenti in Unity in grado di creare un legame tra un gameObject (e quindi un oggetto nel mondo virtuale) che possiede il component corrispondente, con un punto preciso nel mondo reale. Le world anchor fissano e mappano un determinato oggetto virtuale nello spazio reale, viene quindi perso il controllo del suo component *Transform*. Un oggetto con le world anchor non può avere un comportamento dinamico, non può quindi essere un *RigidBody*. Vanno usate con parsimonia, in quanto sono computazionalmente onerose e fanno diminuire le prestazioni dell'intera applicazione.

Nel contesto dell'applicazione, uno dei requisiti era garantire la possibilità di poter salvare e caricare la scena per poter tornare a lavorarci in giorni successivi, e le world anchor sono necessarie per ricollegare i modelli delle scene salvate ad un contesto reale. In questo modo, al caricamento di una scena precedentemente salvata, gli oggetti virtuali vengono caricati nello stesso punto reale dove erano stati posizionati durante la precedente apertura della scena.

5.7 Menu di gestione delle scene

Uno dei requisiti dell'applicazione è quello di dare la possibilità di salvare e caricare scene in modo tale da tornare a lavorarci in un secondo momento. Una volta posizionati gli oggetti nella scena, si vuole offrire la possibilità di memorizzare sul database le scene e le trasformazioni dei modelli nella scena. Il menu di gestione delle scene offre la possibilità di salvare le modifiche alla scena corrente o di crearne una nuova (par. 5.7.2), di caricare una scena salvata in precedenza e di eliminare scene salvate nel database (par. 5.7.3).

La gestione del menu viene effettuata dalla classe *SceneWindowManager*, che contiene diversi parametri globali, come l'ID e il nome della scena, e alcuni metodi comuni ai diversi menu, come il reset degli elementi nella griglia dei risultati della ricerca per il caricamento e l'eliminazione delle scene.

Per aprire il menu per la gestione delle scene si può cliccare sul bottone "*Scene Manager*" della finestra principale o usare il comando vocale "*Scene*".

5.7.1 La finestra

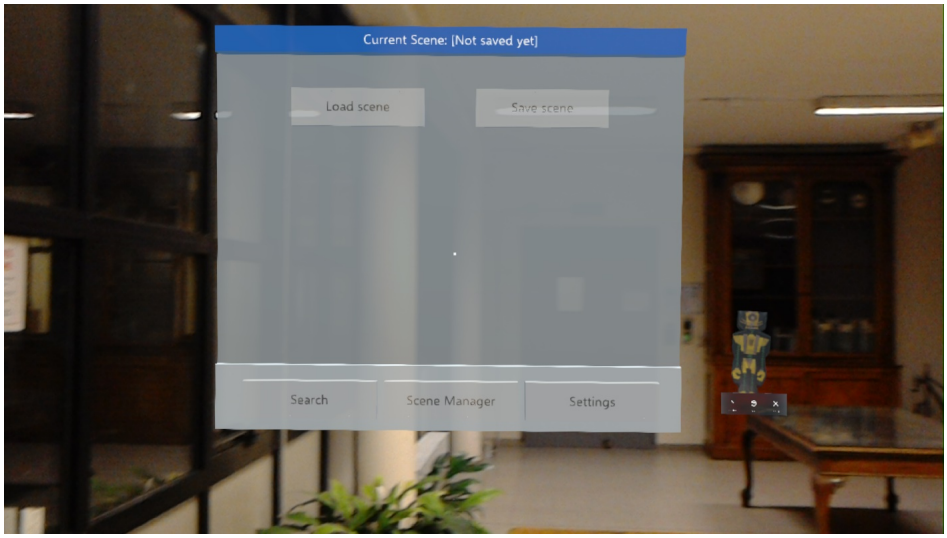

Figura 5.14: Menu di gestione delle scene.

Il menu (fig. 5.14) che si presenta all'utente è minimale e presenta due bottoni:

- **Save Scene:** apre il sottomenu per poter salvare la scena, modificando quella aperta o creando una nuova scena (par. 5.7.2). Si può cliccare sul bottone o usare il comando vocale "*Save*".
- **Load Scene:** apre il sottomenu che mostra l'elenco delle scene da poter caricare o eliminare (par. 5.7.3). Si può cliccare sul bottone o usare il comando vocale "*Load*".

5.7.2 Salvataggio

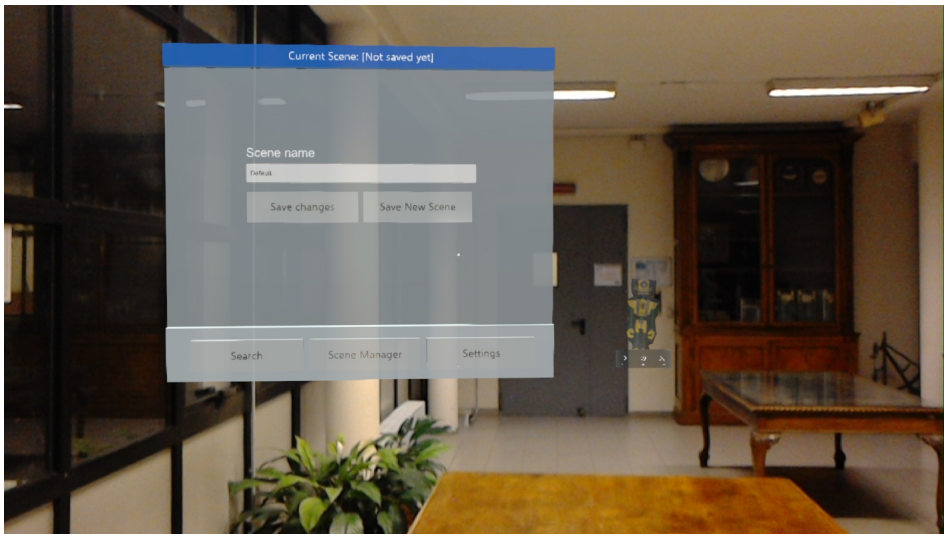

Figura 5.15: Menu di salvataggio della scena.

Il menu offre la possibilità di modificare il nome da dare alla scena da salvare, altrimenti verrà usato un valore di default (fig. 5.15). Una volta scelto il nome, la classe *SaveScene* avvierà la coroutine che si occuperà del salvataggio della scena nel database (fig. 5.16). Ciascuno dei due bottoni implementa un diverso approccio di salvataggio della scena:

- **Save Changes:** salva le modifiche della scena corrente, e ricerca, svuota e reinserisce i nuovi valori nella tabella *model_into_scene*, che contiene le trasformazioni dei singoli modelli relative alla scena. La coroutine richiama gli script *saveeditedscene.php*, per salvare la scena, e *savemodelofscene.php*, per salvare le trasformazioni relative dei modelli nella scena.
- **Save New Scene:** crea una nuova scena nel database, salvando le trasformazioni dei modelli in scena. La coroutine richiama gli script *savenewscene.php* e *savemodelofscene.php*. I due bottoni richiamano la stessa coroutine, leggermente modificata a seconda del bottone che viene selezionato, pertanto il procedimento di salvataggio è analogo all'altro bottone.

Una finestra di dialog informerà l'utente sia che il salvataggio è in corso, sia che è terminato, informandolo dell'esito delle operazioni (par. 5.3).

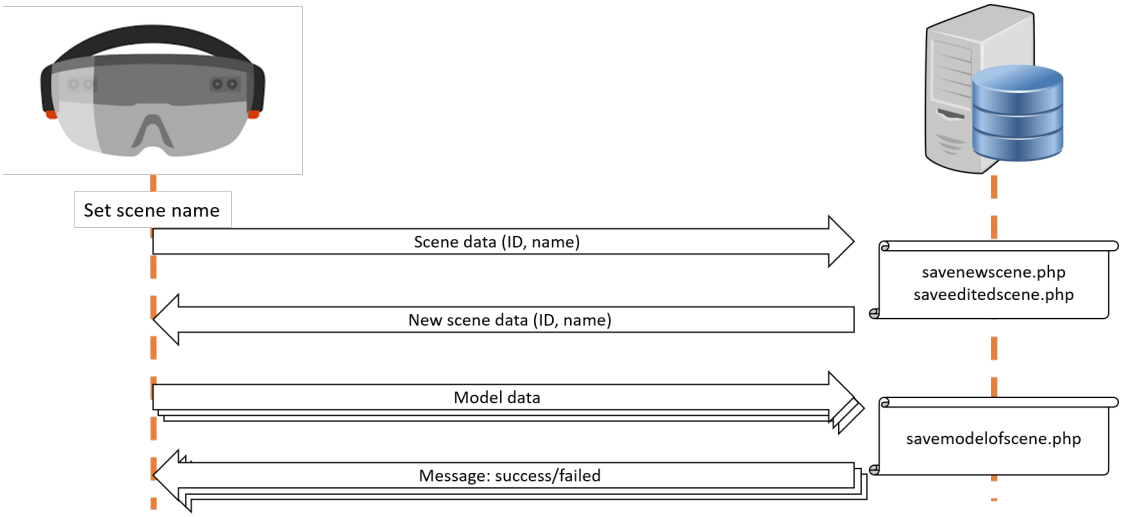

Figura 5.16: Flusso di esecuzione quando viene salvata una nuova scena o viene modificata una scena esistente.

5.7.3 Caricamento ed eliminazione

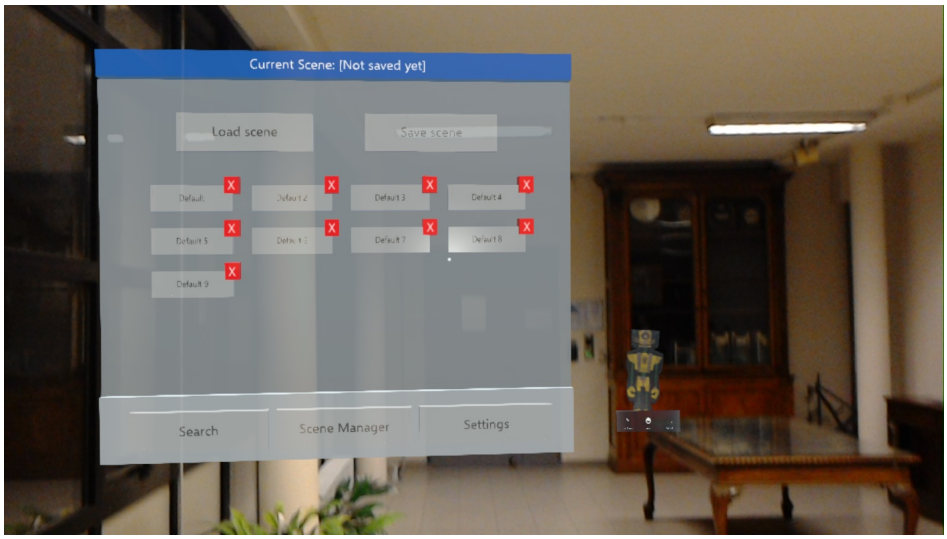

Figura 5.17: Menu di caricamento/eliminazione della scena.

All'apertura del menu, verrà visualizzata una griglia di bottoni, uno per ogni scena da poter caricare (fig. 5.17). Esattamente come nella griglia del menu della ricerca dei modelli (par. 5.5.3), anche in questo caso è possibile sfogliare le pagine con le frecce nella parte inferiore della finestra per scorrere la lista delle scene trovate nel database.

La lista delle scene viene gestita da una coroutine della classe *SaveScene*, che le recupera tramite lo script *getsceneslist.php*. Verranno quindi istanziati tanti prefab "*Button*" quante sono le scene trovate. Ogni bottone, infatti, conterrà il nome e corrisponderà ad una scena.

Cliccando su uno di questi bottoni, inizierà il caricamento di tutti i modelli della scena, e verranno istanziati uno per volta. La coroutine che gestisce il caricamento, nella classe *SaveScene*, dapprima svuota la griglia, svuota la scena attuale eliminando tutti i modelli in scena e inizia a caricarli uno alla volta richiamando il metodo

DownloadAsset() della classe *AssetDownloaderWithText* (fig. 5.18). La modalità di istanziazione dei modelli è identica a quella descritta nel paragrafo 5.5.5.

Una finestra di dialog informerà l'utente che il caricamento della scena è iniziato e lo notificherà quando il caricamento sarà completato.

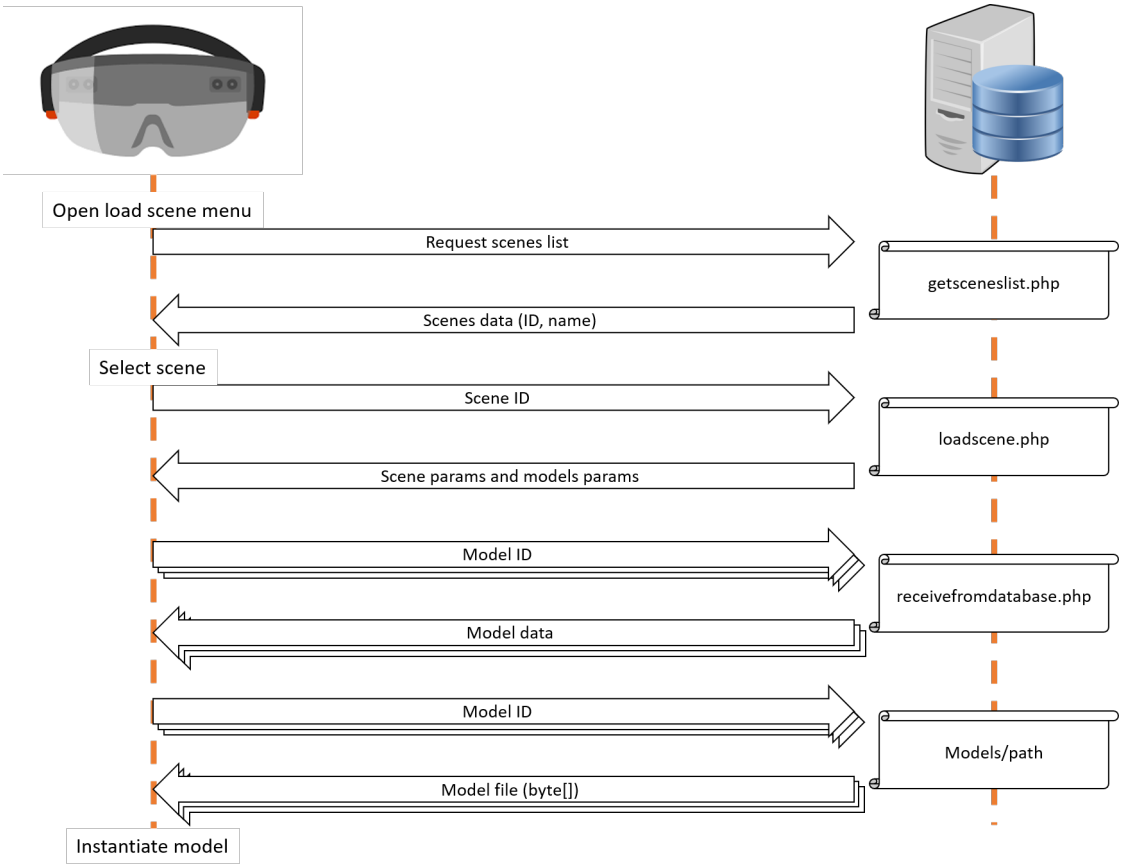

Figura 5.18: Flusso di esecuzione quando viene caricata una scena esistente.

È possibile anche eliminare una scena salvata nel database selezionando la "X" rossa corrispondente alla scena che si vuole eliminare. Una volta selezionata la "X", si aprirà un dialog che chiederà conferma di voler eliminare quella determinata scena. In caso di conferma, una coroutine nella classe *SaveScene* si occupa di eliminare la scena e i relativi modelli in scena (senza modificare gli originali) tramite lo script *deletescene.php*.

Una finestra di dialog informerà l'utente che l'eliminazione della scena è iniziato e lo notificherà quando l'eliminazione è stata completata.

Capitolo 6

Valutazione del Lavoro

In questo capitolo verranno presentati l'analisi dei requisiti e un primo test in un set reale presso le sedi RAI (6.1), il test di usabilità presso il Politecnico (6.2) e i risultati del test (6.3).

6.1 Analisi dei requisiti e test in sede RAI

Figura 6.1: Prova dal vivo dell'applicazione *Set Builder* in uno studio RAI.

Sono stati effettuati due incontri con registi e scenografi che lavorano nelle sedi RAI, in modo tale da avere un riscontro diretto con gli utilizzatori finali dell'applicazione.

Il primo incontro si è svolto presso la sede del RAI CRITS, nel quale sono state illustrate loro gli obiettivi le potenzialità del progetto ed è stata effettuata una prima dimostrazione del prototipo. In questo incontro sono state raccolte idee, spunti e suggerimenti, è stata quindi svolta un'analisi dei requisiti sulla base delle loro esigenze sul set, in particolare l'aggiunta di tags per la ricerca dei modelli, i casi d'uso dell'applicazione, i problemi principali dell'allestimento di un set e altre migliorie.

Il secondo incontro si è svolto presso il centro di produzione RAI di Torino in via Verdi, nel quale l'applicazione è stata testata da uno scenografo direttamente in uno studio televisivo reale. Ad oggi, l'allestimento di un set viene dapprima ricostruito su programmi di modellazione e computer grafica, e ciò comporta la ricostruzione dei modelli di oggetti reali via software e il rendering di particolari inquadrature della scena virtuale che richiedono tempi piuttosto lunghi. Inoltre, le inquadrature dei rendering, che simulano quelle reali delle diverse telecamere sul set, non sempre rispecchiano il pensiero dei registi. L'applicazione HoloLens rende più efficiente la comunicazione tra registi e scenografi, che possono immergersi completamente nel set virtuale ed osservare la stessa scena in tempo reale da diverse angolazioni, in modo da studiare i diversi punti di vista delle telecamere in modo molto più rapido.

6.2 Il test di usabilità al Politecnico

Successivamente agli incontri presso le sedi RAI, è stato svolto in giornata un test di usabilità soggettivo sull'usabilità dell'applicazione per Microsoft HoloLens presso il Politecnico. Il test, della durata di 15 minuti circa a persona, è stato così strutturato:

- È stata fatta una panoramica generale su Microsoft HoloLens, in particolare su gestures e comandi vocali, e poi è stato illustrato il funzionamento dell'applicazione da testare (*Set Builder*).
- Sono stati assegnati una serie di semplici tasks da svolgere. L'obiettivo era la realizzazione di una determinata scena, ricercando modelli dal database, spostandoli, ruotandoli e scalandoli nell'ambiente in modo da ricreare una scena ben precisa (fig. 6.2) e poi salvando la scena sul database. In questa fase, è stato calcolato per ciascuno il tempo di esecuzione complessivo dei tasks.
- È stato fatto compilare un questionario di valutazione soggettivo, il **System Usability Scale** (app. C.1), in modo da evidenziare punti forti e debolezze dell'applicazione.

Figura 6.2: Scena che i candidati dovevano ricostruire per il test di usabilità.

6.3 Risultati

Il test è stato sottoposto a 18 soggetti, uomini e donne, di età compresa tra i 18 e i 30 anni, ed ha avuto una durata media di 7 minuti. La maggior parte dei soggetti aveva già usato applicazioni di realtà aumentata (fig. 6.3).

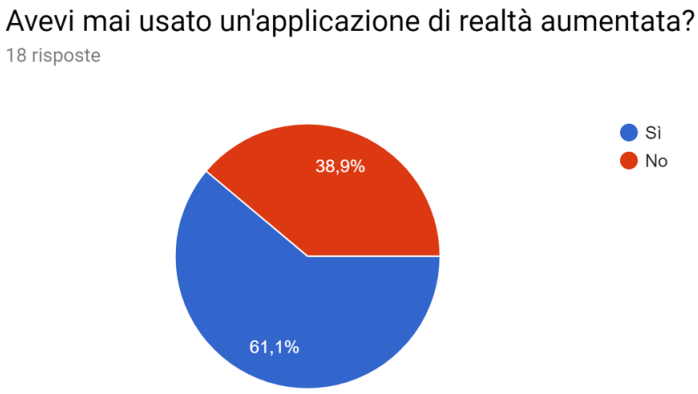

Figura 6.3: Grafico che mostra quanti candidati avevano già usato applicazioni di realtà aumentata.

In generale, la preferenza per quanto riguarda l'interazione (fig. 6.4) è stata quella con gestures rispetto a quella con i comandi vocali, mentre altri ancora non hanno espresso preferenze rispetto ai due tipi di interazioni. Un dato interessante che è emerso è che quasi tutti quelli che non avevano mai avuto a che fare con la realtà aumentata e mista, e quindi che hanno dovuto imparare per la prima volta a usare i gesti per interagire, hanno preferito l'interazione tramite comandi vocali, mentre la maggior parte delle persone che aveva già dimestichezza con questo tipo di dispositivi, e che quindi aveva già in precedenza imparato ad usare i gesti, ha preferito l'interazione con gestures.

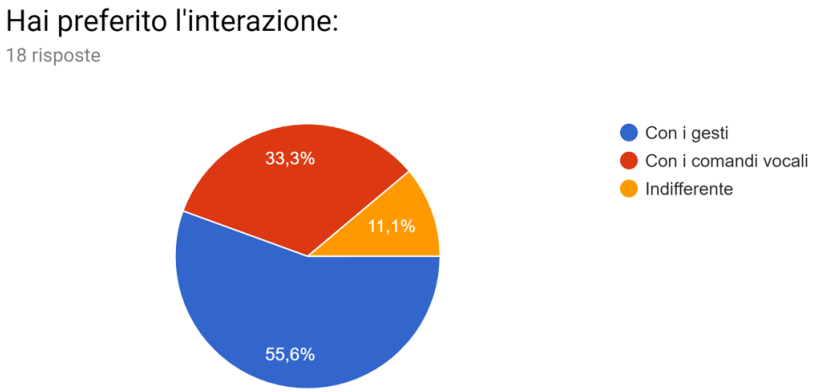

Figura 6.4: Preferenze tra le gestures e i comandi vocali.

Il questionario compilato al termine dell'esperienza è il **System Usability Scale** (SUS), una lista di 10 domande che permette di calcolare il **SUS score** (app. C.2), un punteggio che serve ad avere una valutazione soggettiva dell'applicazione.

Question	1	2	3	4	5
1. I think that I would like to use this system frequently.	0	0	3	10	5
2. I found the system unnecessarily complex.	9	8	0	0	1
3. I thought the system was easy to use.	0	1	4	4	9
4. I think that I would need the support of a technical person to be able to use this system.	3	5	4	5	1
5. I found the various functions in this system were well integrated.	0	0	2	5	11
6. I thought there was too much inconsistency in this system.	13	2	2	0	1
7. I would imagine that most people would learn to use this system very quickly.	0	0	3	8	7
8. I found the system very cumbersome to use.	12	6	0	0	0
9. I felt very confident using the system.	0	1	1	6	10
10. I needed to learn a lot of things before I could get going with this system.	14	3	0	1	0

Legend: 1: Strongly Disagree – 2: Partially Disagree – 3: Neutral – 4: Partially Agree – 5: Strongly Agree

Figura 6.5: Risultati del questionario.

Le risposte sono state valutate tutte positivamente, alcune in modo più marcato di altre (in fig. 6.5 e 6.9 è possibile visionare l'elenco completo delle risposte), ma particolare attenzione va alla domanda numero 4 riguardo il bisogno di un supporto da parte di una persona che conosca l'applicazione (fig. 6.6), nella quale si evidenzia, per la metà dei soggetti, la necessità di un supporto tecnico per imparare ad utilizzare l'applicazione. Da notare, però, che praticamente tutte le persone che necessitano un supporto sono le stesse che non avevano mai utilizzato un'applicazione di realtà aumentata/mista, inoltre quasi tutti i soggetti pensano che la maggior parte delle persone possano imparare ad utilizzare l'applicazione facilmente (fig. 6.8), pertanto il bisogno di aiuto nasce soprattutto dall'inesperienza nei confronti di queste nuove tecnologie, ancora molto poco diffuse, e non è legata direttamente all'applicazione.

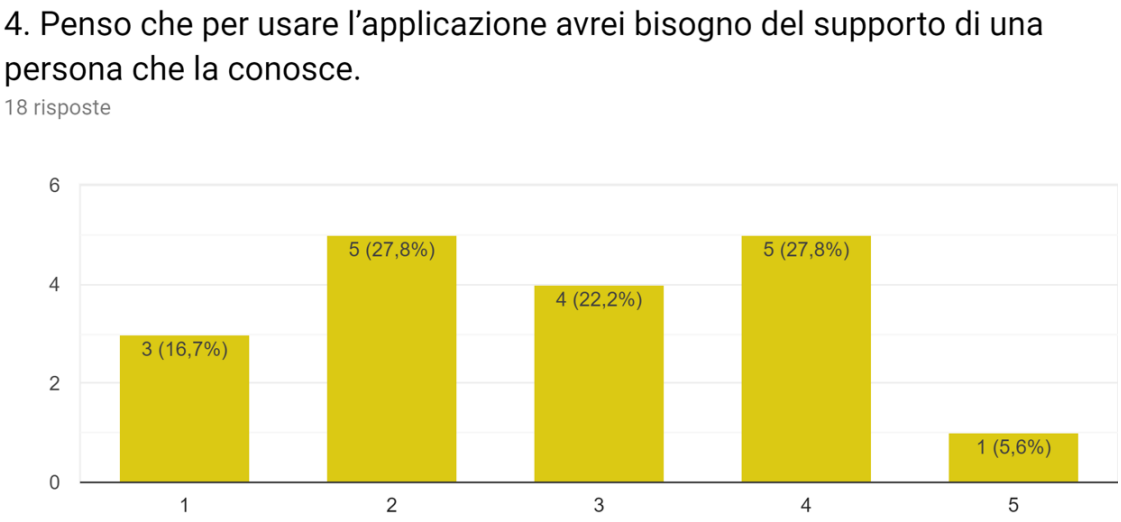

Figura 6.6: Statistiche delle risposte alla domanda 4.

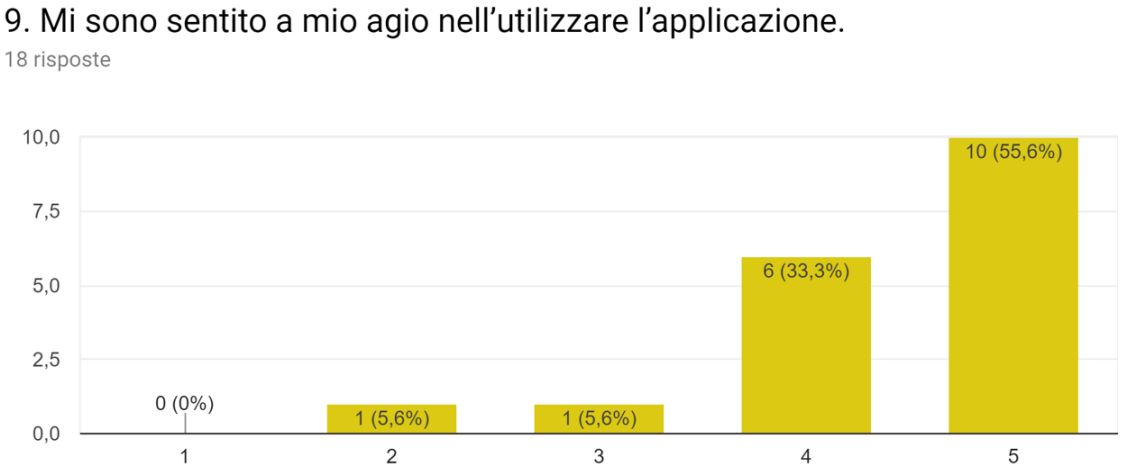

Figura 6.7: Statistiche delle risposte alla domanda 9.

Ciò che è emerso, in generale, è che l'applicazione è stata trovata piacevole nel suo utilizzo, semplice da capire ed usare, priva di incoerenze ed errori, con funzionalità ben integrate, senza la necessità da parte dell'utilizzatore di imparare molte cose prima di poterla utilizzare al meglio. In particolare, è possibile osservare nei grafici delle domande 7 (fig. 6.8) e 9 (fig. 6.7) che tutti gli utenti si sono trovati a loro agio nell'utilizzare l'applicazione e ritengono che sia alla portata di tutti.

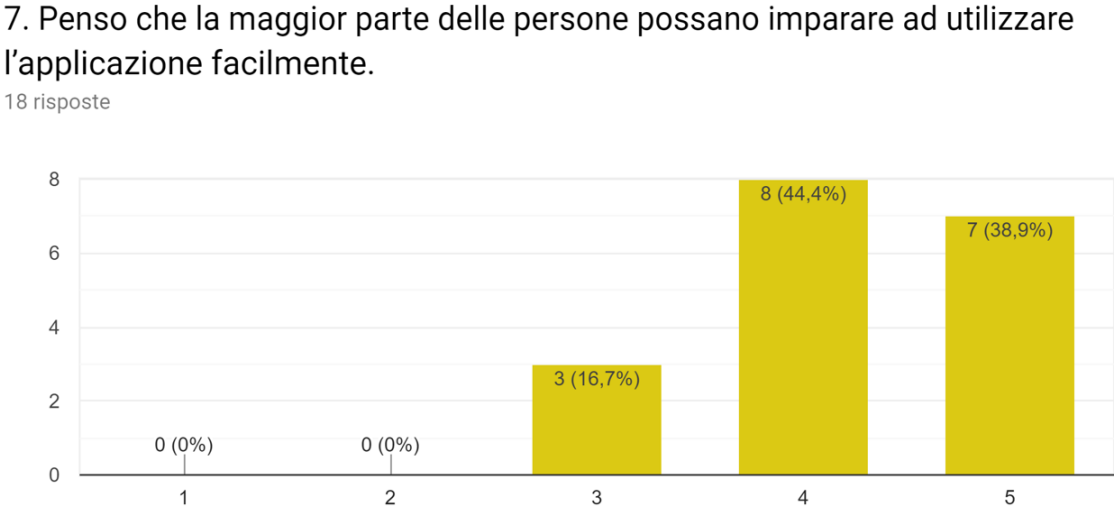

Figura 6.8: Statistiche delle risposte alla domanda 7.

È stato quindi calcolato il SUS score medio tra tutte le risposte, secondo il metodo illustrato nell'app. C.2, e l'applicazione ha totalizzato un punteggio medio complessivo di 82.1/100, facendole ottenere il grado più alto di usabilità.

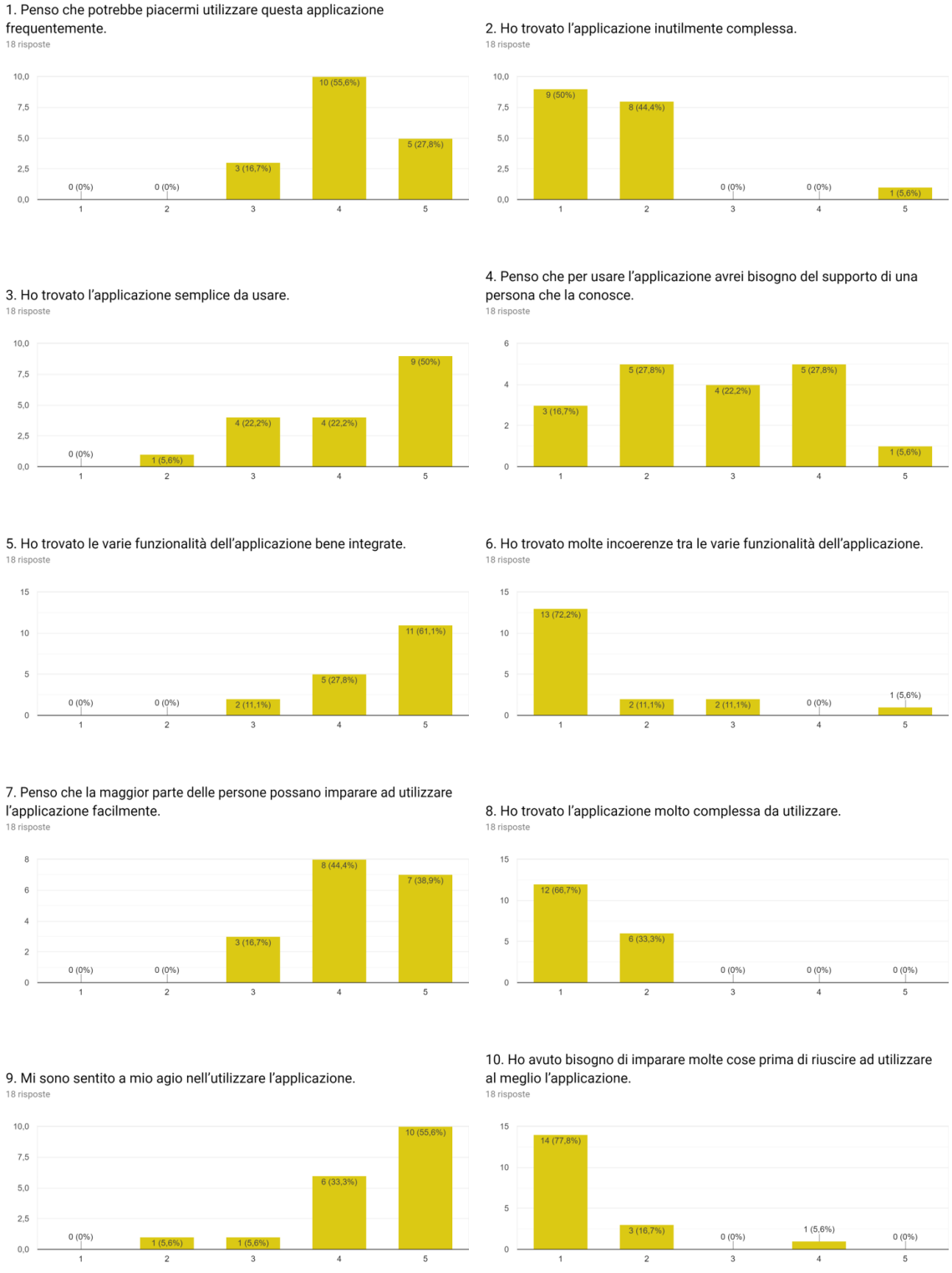

Figura 6.9: Statistiche delle risposte alle domande del SUS.

Capitolo 7

Conclusioni

In quest'ultimo capitolo verranno dapprima analizzati gli obiettivi raggiunti dalla tesi (7.1) e in seguito gli sviluppi futuri (7.2), quindi nel contesto di **Rob-O-Cod** (7.2.1) e del **5G-TOURS** (7.2.2). Infine, verrà descritta l'esperienza personale di tirocinio (7.3).

7.1 Obiettivi raggiunti

Le due applicazioni trattate in questa tesi, ancora in fase prototipale, sono riuscite ad inserirsi all'interno del workflow della produzione televisiva, in particolare nella pre-produzione, ovvero nell'allestimento di un set virtuale. A tal scopo, sono stati raggiunti diversi obiettivi:

- Sono stati raggiunti tutti i requisiti dettati dai registi e dagli scenografi, tra cui quelli sulla ricerca dei modelli, sugli approcci per l'allestimento del set, sulle modalità di interazione e sui diversi casi d'uso dell'applicazione.
- Viene offerta ai registi un'anteprima visiva del set in tempo reale, con la possibilità di ricercare modelli su un database remoto e di manipolare oggetti virtuali a proprio piacimento tramite le tre trasformazioni lineari di base (traslazione, rotazione, scalamento). È anche possibile salvare copie modificate di un modello, creando modelli derivati.
- Viene data la possibilità di inserire nel database modelli realizzati sia tramite software di modellazione che tramite scansione 3D, mostrando un'icona di anteprima durante la ricerca del modello e dando la possibilità di aggiungere alcuni campi facoltativi, come la descrizione del modello e i tag, ovvero tipo di oggetto, colore e materiale, in modo da facilitarne la ricerca.
- La ricerca di un modello avviene tramite parametri quali il nome, la descrizione e i tag del modello.
- Il salvataggio e il caricamento delle scene virtuali consente la messa in pausa dell'allestimento del set virtuale, per una futura ripresa del lavoro.
- Il test, prima su un set reale, poi per misurarne l'utilizzabilità, è stato in grado di evidenziare punti forti e criticità dell'applicazione e la forza e i limiti della realtà aumentata.

7.2 Sviluppi futuri

L'applicazione è stata pensata per essere inserita nel contesto dell'allestimento dei tavoli su cui si basa la trasmissione RAI *Rob-O-Cod* (par. 7.2.1), ma per poterlo fare deve essere integrata insieme ad una seconda applicazione esterna a questa tesi.

A seguito dell'incontro con i registi, si è evidenziata l'esigenza di voler lavorare in scenari che tengano conto dell'illuminazione, delle ombre, delle inquadrature delle telecamere e delle occlusioni, caratteristiche che saranno implementate nelle future versioni dell'applicazione.

Inoltre, allo stato attuale si tratta di un prototipo altamente flessibile per essere inserito in contesti più grandi e diversi da quello per cui è stata inizialmente concepita, come quello del *5G-TOURS* (par. 7.2.2).

7.2.1 Un'applicazione per Rob-O-Cod

Come già accennato nel paragrafo 1.1.3, l'applicazione è stata concepita per essere inserita nell'allestimento dei set dei tavoli del programma televisivo **Rob-O-Cod**. Questa tesi, però, tratta una sola parte dell'applicazione. La seconda applicazione presenta nuove caratteristiche:

- **Lo sharing per il multiplayer** permette di poter lavorare in contemporanea sulla stessa scena. L'uso di world anchor lega in modo univoco l'oggetto virtuale all'ambiente reale. In questo modo, è possibile condividere su più dispositivi lo stesso scenario, per fare in modo che registi e scenografi possano cooperare per la realizzazione del set virtuale in modo pratico e veloce.

- **Lo spatial mapping** scansiona l'ambiente reale, in modo da individuare superfici piane, come ad esempio i tavoli, su cui poter posizionare gli oggetti virtuali.

Il merge tra le due diverse applicazioni ne genererà una terza, più specifica e potente, in grado di venire incontro alle esigenze dei registi.

7.2.2 5G-TOURS

L'applicazione è stata realizzata nel contesto del progetto **5G-TOURS** [41], un progetto a livello europeo con l'obiettivo di avvicinare la visione europea del "5G che abilita le industrie verticali" (5GPPP16) all'implementazione commerciale con casi d'uso altamente innovativi che coinvolgono partnership intersettoriali.

5G-TOURS mira a dimostrare l'abilità del 5G di supportare più casi d'uso verticali contemporaneamente sulla stessa infrastruttura. La visione di 5G-TOURS è migliorare la vita in città per cittadini e turisti, rendendo le città più attraenti da visitare, più efficienti in termini di mobilità e più sicure per tutti.

Il progetto è incentrato su tre temi principali:

- **La città turistica:** i visitatori di musei e attrazioni all'aperto sono dotati di applicazioni basate su 5G per migliorare la loro esperienza durante la visita della città. Ciò include applicazioni VR/AR per integrare la visita fisica con contenuti aggiuntivi, che coinvolgono comunicazioni interattive. L'esperienza dei visitatori è inoltre migliorata con servizi robotizzati, telepresenza

per consentire visite a distanza, nonché eventi dal vivo, come ad esempio concerti seguiti da più persone contemporaneamente, permessi da comunicazioni mobili.

- **La città efficiente in termini di mobilità:** la mobilità per raggiungere e spostarsi all'interno della città è resa più efficiente e confortevole. Ciò comporta città più intelligenti, la raccolta di informazioni sulla città e il suo utilizzo per migliorare i sistemi di navigazione così come anche il parcheggio. Il viaggio è anche reso più piacevole, fornendo servizi VR/AR ai passeggeri, e gli aeroporti diventano logisticamente più efficienti facendo affidamento sul 5G per il loro funzionamento.

- **La città sicura:** la tecnologia 5G migliora notevolmente la sicurezza in città, fornendo mezzi per seguire meglio l'assistenza sanitaria in tutte le fasi di un incidente, che vanno dal monitoraggio sanitario per la prevenzione e la diagnosi precoce, alla diagnosi e all'intervento in ambulanza, e un intervento chirurgico nella sala operatoria wireless dell'ospedale. Inoltre, il 5G consente il monitoraggio e l'assistenza remota ai turisti dal loro paese di origine.

Il progetto fornirà servizi vicini al commercio per turisti, cittadini e pazienti in tre diversi tipi di città:

- Rennes, la città sicura dove saranno dimostrati i casi di utilizzo della sanità elettronica.

- Torino, la città turistica focalizzata sui media e sui casi d'uso della trasmissione.

- Atene, la città efficiente in termini di mobilità che porta il 5G agli utenti in movimento e ai fornitori di servizi relativi ai trasporti.

Questi servizi non solo miglioreranno la qualità della vita di cittadini e turisti, ma rappresenteranno anche un'importante opportunità commerciale. La caratteristica fondamentale del concetto 5G-TOURS è l'uso dinamico della rete per fornire senza soluzione di continuità diversi tipi di servizi adattati alle esigenze specifiche dei singoli casi d'uso. 5G-TOURS consentirà diverse funzionalità come il network slicing, la virtualizzazione, l'orchestrazione o la trasmissione della rete, nonché funzionalità aggiuntive sviluppate dal progetto per offrire maggiore flessibilità e prestazioni migliori. L'ambizione è di dimostrare pienamente le tecnologie pre-commerciali 5G su larga scala, mostrando la capacità della rete 5G di soddisfare ICP (Indicatori Chiave di Prestazione) estremi e contrastanti, supportando allo stesso tempo requisiti molto diversi sulla stessa infrastruttura.

Il sistema di rete mobile 5G-TOURS integrerà componenti strategici dell'ecosistema, tra cui l'infrastruttura di rete, i terminali e i dispositivi finali, le soluzioni verticali abilitate da 5G e i clienti verticali che ricevono i servizi. 5G-TOURS ha elaborato un piano di valutazione approfondito per esaminare la fattibilità dei casi d'uso, affrontando le prestazioni tecniche analizzando gli indicatori ICP del servizio di rete e a livello di applicazione, l'impatto economico, analizzando i ricavi generati stimati, e, in definitiva, la soddisfazione dei clienti verticali.

7.3 L'esperienza di tirocinio

Il progetto è stato realizzato nel contesto di un tirocinio presso il RAI CRITS a Torino della durata di 3 mesi. L'intero progetto è stato suddiviso in due parti, il lato backend, di cui mi sono personalmente occupato, e quello frontend.

Durante la prima fase sullo stato dell'arte, sono stati cercati metodi di importazione dei modelli da remoto mentre l'applicazione è in esecuzione, in quanto non è una funzione implementata direttamente da Unity. È stato utilizzato il plugin TriLib, acquistato dall'azienda dall'Asset store di Unity, e ne è stato studiato il funzionamento a livello di codice in modo tale da adattarlo alle caratteristiche e ai requisiti del progetto.

La difficoltà più grande è stata quella di importare i modelli da remoto in tempi accettabili ed evitando i problemi del freeze dell'applicazione e dell'improvvisa scomparsa e ricomparsa della User Interface (UI), dovuta ad operazioni computazionalmente complesse, come la creazione della mesh e il caricamento delle textures, eseguibili solo dal thread principale, che però si occupa anche del refresh della UI. Si tratta di limitazioni principalmente dovute alla natura dell'hardware. Queste difficoltà sono state risolte riscrivendo parte del codice di TriLib grazie ad una combinazione di coroutines e thread asincroni che riducono l'impatto di operazioni onerose sul thread principale.

L'utilizzo del Mixed Reality Toolkit e l'aiuto dei miei tutor sono stati fondamentali per comprendere la programmazione su un dispositivo così recente e specifico come Microsoft HoloLens, su cui non esiste ancora un così grande supporto su Internet.

Lo scambio di idee con i registi e gli scenografi mi hanno permesso di focalizzarmi su aspetti legati ad un caso d'uso reale. Essi hanno espresso impressioni positive ed il centro ricerche, in collaborazione con gli scenografi ed i registi RAI, continueranno sicuramente la sperimentazione.

Il progetto, nel contesto del 5G-TOURS, nasce con l'intenzione di sfruttare le innovazioni tecnologiche per agevolare la vita delle persone e si era prefissato degli obiettivi, già descritti all'inizio di questo capitolo, che sono stati praticamente raggiunti tutti. Inoltre, l'esperienza di lavoro presso il RAI CRITS è stata senz'altro positiva, ed è stato per me un onore partecipare ad una piccola parte di un progetto così grande che potrebbe alterare e migliorare la percezione del nostro presente in un futuro ormai non troppo remoto.

Appendice A

XAMPP e PhpMyAdmin

A.1 XAMPP

XAMPP [42] è una distribuzione di Apache gratuita contenente MySQL, PHP e Perl. È stato utilizzato per la realizzazione del **server in locale**. La schermata di XAMPP si mostra come in fig. A.1: sarà sufficiente abilitare, cliccando su *start*, i moduli di Apache e di MySQL, per consentire al PC di funzionare come un server locale. L'indirizzo IP del server locale, quindi, sarà identico all'indirizzo IP del PC su cui gira XAMPP.

Per far comunicare le due applicazioni con il database si fa uso di una serie di script, scritti in **PHP**, che effettuano delle query specifiche sul database e restituiscono i risultati alle applicazioni sotto forma di oggetti di tipo **JSON**. Le applicazioni potranno accedere al contenuto della cartella */xampp/htdocs/*, in particolare, tutti i contenuti inerenti alle due appliazioni sono contenuti nella cartella */xampp/htdocs/robocodserver/*, pertanto, per accedere, basterà collegarsi all'indirizzo */(indirizzo IP del PC)/robocodserver/*[1].

A.2 PhpMyAdmin

PhpMyAdmin [43] è un software gratuito scritto in PHP, con lo scopo di gestire l'amministrazione di MySQL sul Web. Consente di eseguire un grande numero di operazioni su MySQL e su MariaDB, tra cui le più frequenti (come la gestione di database, tabelle, colonne, relazioni, etc.) che sono eseguibili sull'interfaccia utente. In fig. A.2 è mostrata l'interfaccia utente di PhpMyAdmin per il database delle due applicazioni trattate in questa tesi.

Per gestire il database con PhpMyAdmin, ci si collega all'indirizzo *localhost/phpmyadmin* direttamente da un browser Web.

[1] Le applicazioni non riusciranno a collegarsi se XAMPP non avrà avviato i moduli Apache e MySQL.

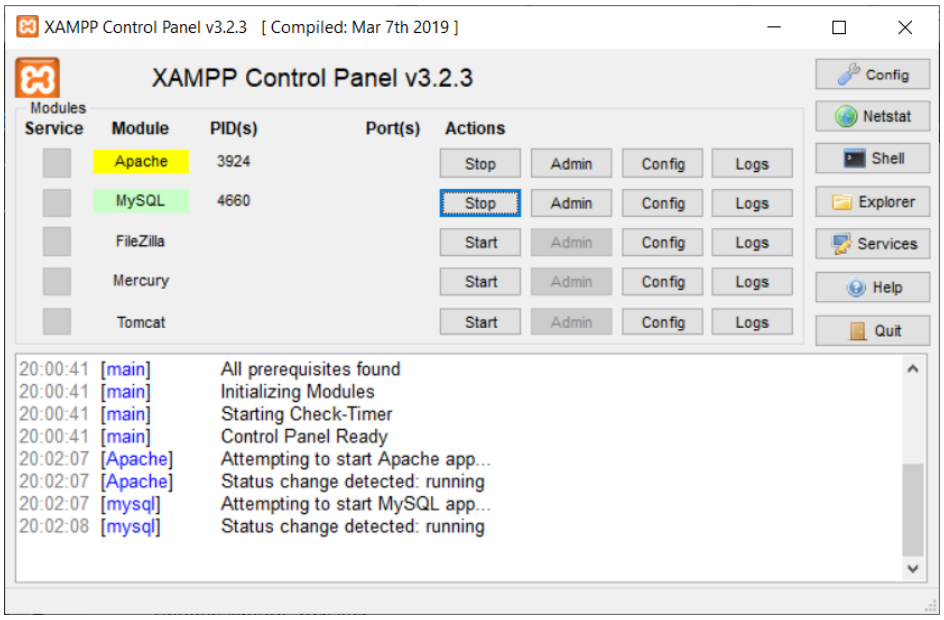

Figura A.1: Interfaccia utente di XAMPP

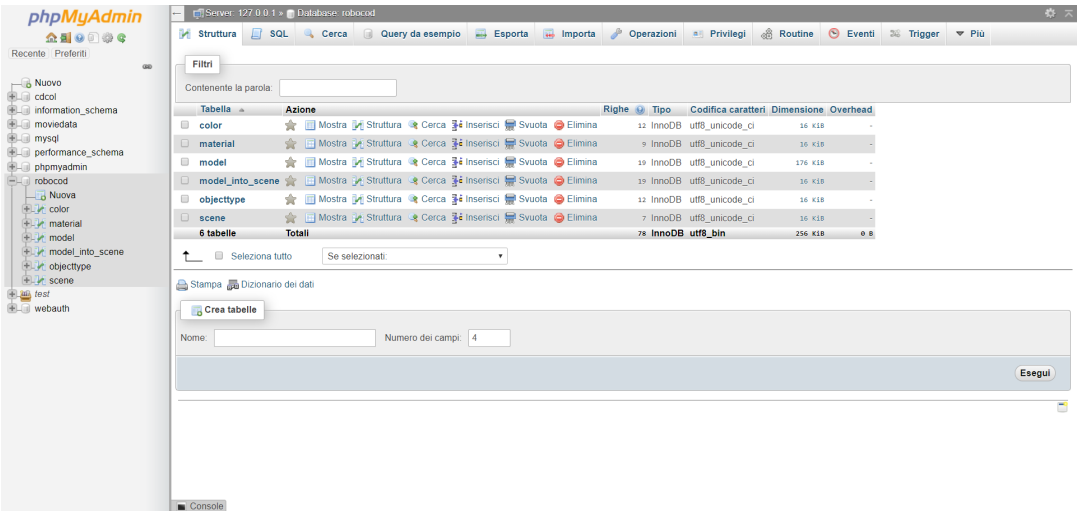

Figura A.2: Interfaccia utente di PhpMyAdmin.

Appendice B

TriLib

B.1 TriLib

TriLib [44] è un plugin, acquistabile nell'Asset Store di Unity [45], in grado di risolvere le limitazioni di Unity e le problematiche riscontrate con gli AssetBundles. Infatti, è in grado di importare, esportare ed istanziare modelli in tempo reale senza alterarli o senza impacchettarli in formati particolari, cosa che invece accadeva con gli AssetBundles. È possibile importare modelli locali, sul PC dell'host, selezionando il path corrispondente, ma è anche importare modelli in remoto, indicando l'indirizzo e il path in cui si trova il modello da istanziare.

I file che vengono importati ed esportati sono prevalentemente in formato **.zip**, contentente la **mesh** (in formato .obj, .fbx, etc.) ed eventuali **textures** e **metadati**[1], ma è anche possibile importare direttamente la mesh senza le textures.

Quando viene importato un modello realtime, il plugin estrae i files dal file .zip, recupera le informazioni della mesh e di eventuali metadati e textures e genera un nuovo gameObject ricostruendo il modello sulla base di queste informazioni, il tutto in modo asincrono.

[1] Mesh con molti poligoni e textures di grandi dimensioni aumentano la complessità computazionale e riducono le prestazioni del plugin. Su HoloLens, ciò comporta la scomparsa della scena per un tempo proporzionale alla complessità computazionale dovuta al modello da istanziare

Appendice C

Test di valutazione soggettivo

C.1 System Usability Scale

Il **System Usability Scale** [46] è stato sviluppato da John Brooke nel 1996 e consiste in un test soggettivo composto da 10 domande le cui risposte variano da una scala da 1 a 5, dove 1 corrisponde a fortemente in disaccordo e 5 a fortemente d'accordo. Il test, in figura C.2, è stato elaborato per essere una strategia universale di valutazione dell'usabilità di un particolare sistema attraverso il calcolo del SUS score.

C.2 SUS score

Una volta compilato il test si calcola il **SUS score** [47], un punteggio che varia da 0 a 100 così calcolato:

- Per ogni domanda dispari: il punteggio della risposta è X-1, dove X è la risposta che il candidato ha fornito per quella domanda.
- Per ogni domanda pari: il punteggio della risposta è 5-X.

Si sommano i punteggi ottenuti per ogni domanda e si moltiplica il risultato per 2.5. Il punteggio totale fornirà indicazioni riguardo l'usabilità dell'applicazione, così come mostrato in figura C.1.

SUS Score	Grade	Adjective Rating
> 80.3	A	Excellent
68 – 80.3	B	Good
68	C	Okay
51 – 68	D	Poor
< 51	F	Awful

Figura C.1: Tabella per l'interpretazione del SUS score.

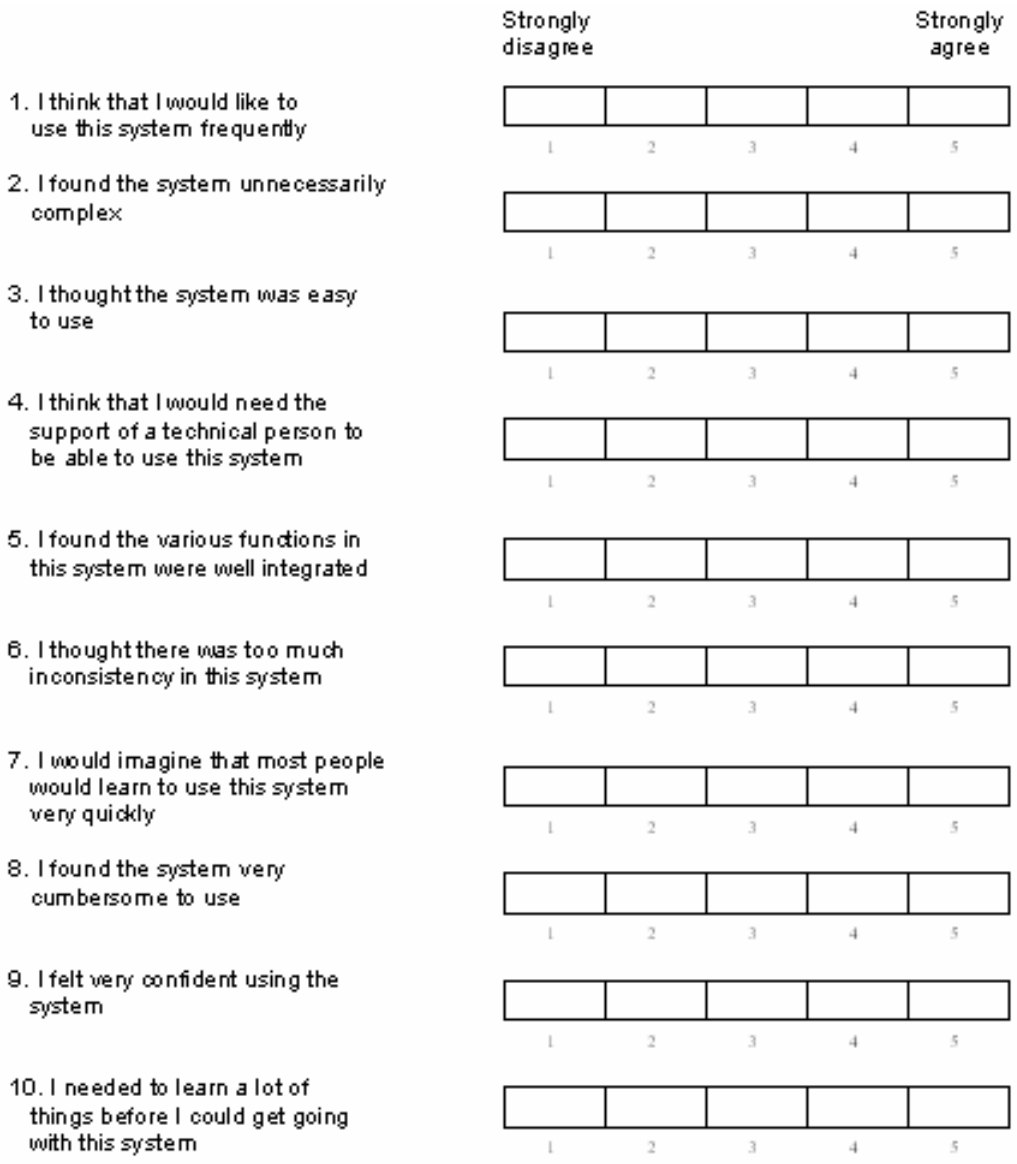

Figura C.2: Il System Usability Scale.

Bibliografia

[1] Rai Radiotelevisione Italiana Spa. *Rob-O-Cod, Sfida all'ultimo cod.* 2019. URL: https://www.raiplay.it/programmi/rob-o-cod/.

[2] Gartner. *Gartner Hype Cycle.* 2019. URL: https://www.gartner.com/en/research/methodologies/gartner-hype-cycle.

[3] Zugara. *Augmented Reality Startups and Venture Capital.* 2014. URL: http://zugara.com/augmented-reality-startups-and-venture-capital.

[4] Gartner. *Gartner Hype Cycle for Emerging Technologies, 2018.* 2018. URL: https://www.fourquadrant.com/gartner-hype-cycles-magic-quadrants/.

[5] A. Vovk, F. Wild, W. Guest e T. Kuula. «Simulator sickness in augmented reality training using the microsoft holoLens». In: (2018).

[6] P. Misteli, S. Poulakos, M. Kapadia e R. Sumner. «Towards Emergent Play in Mixed Reality». In: *International SERIES on Information Systems and Management in Creative eMedia (CreMedia)* (2018). ISSN: 2341-5576.

[7] Bianca-Cerasela-Zelia Blaga D. «DAR: Implementation of a Drone Augmented Reality Video Game». In: (2019).

[8] M. Hanna, I. Ahmed, J. Nine, S. Prajapati e L. Pantanowitz. «Augmented reality technology using Microsoft HoloLens in anatomic pathology». In: *Archives of pathology and laboratory medicine* (2018). ISSN: 1543-2165.

[9] S. Wang, M. Parson, J. Stone-McLean, P. Rogers, S. Boyd, K. Hoover, O. Meruvia-Pastor, M. Gong e A. Smith. «Augmented reality as a telemedicine platform for remote procedural training». In: *Sensors* (2017).

[10] A. Norberg e E. Rask. «3D visualisation of breast reconstruction using Microsoft HoloLens». In: (2018).

[11] Bel Lang D. «Augmented reality phobia treatment including biofeedback». In: (2018).

[12] D. Mehta, M. Siddiqui e A. Javaid. «Facial emotion recognition: A survey and real-world user experiences in mixed reality». In: *Sensors* (2018).

[13] M. Hoffman e J. Provance. «Visualization of molecular structures using HoloLens-based augmented reality». In: *AMIA Summits on Translational Science Proceedings* (2017).

[14] A. Karambakhsh, A. Kamel, B. Sheng, P. Li, P. Yang e D. Feng. «Deep gesture interaction for augmented anatomy learning». In: *International Journal of Information Management* (2019). ISSN: 0268-4012.

[15] A. Orsini, G. Venkatesan, G. Huang, G. Shah e N. Shah. «Augmented Reality Enhanced Cooking With Microsoft HoloLens». In: *Rutgers senior project, 4th place winner. Retrieved on Mar* (2017).

[16] F. De Pace, F. Manuri e A. Sanna. «Augmented reality in industry 4.0». In: *American Journal of Computer Science and Information Technology, Turin* (2018).

[17] F. Lamberti, F. Manuri, A. Sanna, G. Paravati, P. Pezzolla e P. Montuschi. «Challenges, opportunities, and future trends of emerging techniques for augmented reality-based maintenance». In: *IEEE Transactions on Emerging Topics in Computing* (2014). ISSN: 2168-6750.

[18] P. Fraga-Lamas, T. Fernández-Caramés, Ó. Blanco-Novoa e A. Vilar-Montesinos. «A review on industrial augmented reality systems for the industry 4.0 shipyard». In: *IEEEAccess* (2018). ISSN: 2169-3536.

[19] L. Zhang, S. Chen, H. Dong e A. El Saddik. «Visualizing Toronto city data with Hololens: Using augmented reality for a city model». In: *IEEE Consumer Electronics Magazine* (2018). ISSN: 2162-2248.

[20] S. Tsai. «Augmented reality enhancing place satisfaction for heritage tourism marketing». In: *Current Issues in Tourism* (2019). ISSN: 1368-3500.

[21] W Hou. «Augmented Reality Museum Visiting Application based on the Microsoft HoloLens». In: *Journal of Physics: Conference Series* (2019). ISSN: 1742-6596.

[22] European Broadcasting Union (EBU). *Virtual reality: how are public broadcasters using it?* 2017. URL: https://www.ebu.ch/media/virtual-reality.

[23] European Broadcasting Union (EBU). *Augmented Reality - A first guide for broadcasters*. 2018. URL: https://tech.ebu.ch/publications/bpn115.

[24] Pixomondo. *Visual Effects: The True Magic of Game of Thrones*. 2019. URL: https://www.pixomondo.com/feature/visual-effects-the-true-magic-of-game-of-thrones/.

[25] JoBlo Superheroes. *AVENGERS: ENDGAME (2019) Final Battle Behind the Scenes VFX [HD]*. 2019. URL: https://www.youtube.com/watch?v=1mbunStY6XA.

[26] VFX GURU. *Aladdin (2019) - Behind The Scenes*. 2019. URL: https://www.youtube.com/watch?v=uRe3G0x1al8.

[27] Jason Guerrasio. *The actor behind the CGI Tarkin in 'Rogue One' tells us how he created the character*. 2019. URL: https://www.businessinsider.com/cgi-moff-tarkin-rogue-one-guy-henry-2017-1?IR=T.

[28] Scott Hayden. *Filming 'The Lion King' in VR was like a "multiplayer filmmaking game," Says Director*. 2019. URL: https://www.roadtovr.com/lion-king-vr-production/.

[29] Microsoft. *Hololens overview*. 2019. URL: https://docs.microsoft.com/en-us/hololens/.

[30] Microsoft. *HoloLens (1st Gen) hardware*. 2019. URL: https://docs.microsoft.com/it-it/hololens/hololens1-hardware.

[31] Blender community. *Blender*. 2019. URL: https://www.blender.org/.

[32] Artec3D. *Artec Eva - Scansioni 3D rapide per professionisti*. 2019. URL: https://www.artec3d.com/it/portable-3d-scanners/artec-eva.

[33] Microsoft. *Getting started with MRTK v2*. 2019. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-getting-started.

[34] Microsoft. *MixedRealityToolkit-Unity*. 2019. URL: https://github.com/Microsoft/MixedRealityToolkit-Unity/releases.

[35] Microsoft. *Windows 10 SDK*. 2019. URL: https://developer.microsoft.com/en-us/windows/downloads/windows-10-sdk#.

[36] Unity Technologies. *Unity*. 2019. URL: https://unity.com/.

[37] Microsoft. *Visual Studio*. 2019. URL: https://visualstudio.microsoft.com/it/.

[38] Microsoft. *MR Basics 100: Getting started with Unity*. 2018. URL: https://docs.microsoft.com/it-it/windows/mixed-reality/holograms-100.

[39] Unity Technologies. *AssetBundles*. 2019. URL: https://docs.unity3d.com/Manual/AssetBundlesIntro.html.

[40] Unity Technologies. *WMR input and interaction concepts*. 2019. URL: https://docs.unity3d.com/Manual/wmr_input_types.html.

[41] Silvia Provvedi. *5G-TOURS - Horizon 2020 project*. 2019. URL: https://5gtours.eu/.

[42] Apache Friends. *XAMPP Apache + MariaDB + PHP + Perl*. 2019. URL: https://www.apachefriends.org/it/index.html.

[43] PhpMyAdmin development team. *Bringing MySQL to the web*. 2019. URL: https://www.phpmyadmin.net/.

[44] Ricardo Reis. *TRILIB – UNITY 3D MODEL LOADER PACKAGE*. 2019. URL: https://ricardoreis.net/?p=14.

[45] Ricardo Reis. *TriLib - Model loader package*. 2019. URL: https://assetstore.unity.com/packages/tools/modeling/trilib-model-loader-package-91777.

[46] K. Finstad. «The system usability scale and non-native English speakers». In: *Journal of usability studies* (2006). ISSN: 1931-3357.

[47] Hadi Alathas. *How to Measure Product Usability with the System Usability Scale (SUS) Score*. 2018. URL: https://uxplanet.org/how-to-measure-product-usability-with-the-system-usability-scale-sus-score-69f3875b858f.