POLITECNICO DI TORINO

Tesi di Laurea Magistrale in
Ingegneria del Cinema e dei Mezzi di Comunicazione

3D Storytelling in Realtà Aumentata

Anno Accademico 2022/2023

Relatore
Prof. Andrea Sanna

Co-Relatore
Prof. Federico Manuri

Candidata
Sara Piumatti

Sommario

Una delle fasi più importanti nella realizzazione di un prodotto cinematografico è la fase di preproduzione: questo stadio permette di progettare e definire le caratteristiche del prodotto finale ed è essenziale per la realizzazione di un film. Sebbene gli strumenti a disposizione siano molteplici uno dei più potenti e importanti risulta essere lo storyboard. Gli storyboard descrivono tramite disegni o immagini ogni inquadratura di un film e la possibilità di previsualizzare un prodotto, prima della sua effettiva produzione, permette di identificarne punti di forza e criticità dello stesso: ciò ha reso lo storyboard uno strumento essenziale per la preproduzione. L'utilizzo degli storyboard nel cinema risale fino ai primi anni del 1900 e, al giorno d'oggi, oltre ad una rappresentazione delle idee cinematografiche, esso permette la pianificazione visiva di altri prodotti digitali come applicazioni e videogiochi.

Lo storyboard tradizionale, realizzato con carta e penna, richiede capacità tecniche e artistiche avanzate e per questo motivo negli anni si sono sviluppati software e sistemi alternativi per semplificare il processo. Tra questi si possono trovare applicazioni di disegno 2D ma anche sistemi 3D desktop, in realtà virtuale o in realtà mista.

L'obiettivo di questo lavoro di tesi è perciò quello di descrivere la progettazione e lo sviluppo di un'applicazione che fornisca uno strumento alternativo alla creazione tradizionale di uno storyboard avvalendosi della realtà aumentata. Il progetto mira alla realizzazione di un applicativo che permetta, tramite l'utilizzo di un caschetto di realtà aumentata, di creare un ambiente 3D con i gesti delle mani, di animare, muovere e far interagire i personaggi tra loro e con gli oggetti che costituiscono lo spazio di scena, di realizzare le vignette dello storyboard con delle camere virtuali e di ottenere un file finale che contenga tutte le componenti essenziali di uno storyboard.

Si è quindi scelto di arricchire la percezione sensoriale mediante la realtà aumentata. Essa è una tecnologia che abilita la sovrapposizione di oggetti virtuali ad oggetti e luoghi reali, permettendo di aggiungere delle informazioni generate al computer al mondo circostante. Per fare ciò questa tecnologia si avvale di una serie di sensori che analizzano il mondo reale dell'utente per poi aggiungere al suo interno oggetti virtuali. La realtà aumentata è ormai utilizzata in molteplici ambiti e settori come quelli scientifici, tecnici ma anche per marketing e intrattenimento.

Per capire come realizzare un sistema per la creazione di storyboard in realtà aumentata il primo passo è stato analizzare le soluzioni già esistenti in questo ambito. Sebbene non esista in letteratura un sistema che permetta di raggiungere tutti gli obiettivi preposti, sono stati analizzati diversi applicativi per approfondire funzionalità differenti dell'applicazione. Sono stati esaminati

aspetti quali la manipolazione di oggetti con delle interfacce AR tangibili, l'animazione dei personaggi in realtà aumentata con l'utilizzo di dispositivi mobile e la creazione di sistemi di previsualizzazione in realtà mista per l'industria cinematografica.

Per sviluppare un applicazione 3D per la realtà mista è stato necessario l'utilizzo di un motore grafico: a questo proposito è stato scelto Unity, un game engine multipiattaforma, che permette la creazione di applicazioni e giochi in 2D e 3D per una vasta gamma di dispositivi e piattaforme. Per quanto riguarda la distribuzione dell'applicazione, invece, il dispositivo scelto è stato il Microsoft Hololens 2: un Head Mounted Display, senza fili, di tipo see through per la realtà aumentata e mista.

Una volta definito l'obiettivo del progetto e le tecnologie da utilizzare per realizzarlo è stato necessario capire quali fossero i requisiti fondamentali dell'applicazione: per fare ciò è stato posto un questionario ad alcuni studenti del Politecnico di Torino. Gli studenti intervistati possiedono competenze specifiche riguardanti il mondo del cinema e la realtà aumentata in quanto iscritti ai corsi di laurea in Ingegneria del Cinema e Ingegneria Informatica orientamento Grafica e Multimedia. Il questionario ha utilizzato per la classificazione dei requisiti il metodo MoSCoW e i risultati ottenuti hanno permesso di identificare quali fossero le funzionalità che l'applicazione doveva supportare.

Lo step successivo è stata la progettazione e realizzazione dell'applicativo. In primis bisogna evidenziare che ciò che ha permesso di sviluppare l'applicazione con il motore grafico Unity, per il dispositivo scelto, è stato il progetto MRTK: un insieme di librerie open source ideate da Microsoft per facilitare lo sviluppo di applicazioni in realtà mista. L'applicazione si compone principalmente di tre scene: la prima contiene un menù iniziale, che permette di accedere alle altre sezioni dell'applicazione, la seconda contiene gli strumenti per la vera e propria realizzazione dello storyboard e, infine, la terza scena contiene un tutorial, ideato per spiegare agli utenti quali sono i principali metodi di interazione del sistema.

La scena che permette la realizzazione dello storyboard a sua volta si può suddividere in due sezioni: la prima sezione permette di creare con le proprie mani l'ambiente virtuale, aggiungendo gli oggetti da una libreria e manipolandoli tramite operazioni di traslazione, scalamento e rotazione. Inoltre, è possibile salvare e rinominare il set creato. La seconda sezione permette invece di muovere, animare e far interagire i personaggi: ogni avatar, in base allo stato in cui si trova, può compiere delle determinate azioni che lo portano in altri stati. Inoltre, le animazioni, associate ad ogni azione che il personaggio compie, possono essere bloccate in modo da mettere arbitrariamente in posa i personaggi, permettendo, così, di realizzare le vignette dello storyboard acquisite da camere virtuali. Infine, è possibile

salvare lo storyboard in formato HTML: esso è composto dalle vignette realizzate, le descrizioni di ogni azione, generate automaticamente dal sistema, e da tutti i parametri scelti per le inquadrature come durata delle azioni o lunghezza focale della camera.

Infine, una volta realizzato il sistema, è stato necessario effettuare una valutazione delle sue caratteristiche, dell'usabilità e della robustezza dell'applicazione. Per fare ciò sono stati scelti due casi studio per permettere agli utenti di esplorare e analizzare il sistema: il primo pone dei vincoli più stringenti, chiedendo a chi valuta di riprodurre uno storyboard tradizionale, mentre il secondo permette di esplorare liberamente l'applicazione.

I test che sono stati effettuati hanno coinvolto tre esperti di dominio e di usabilità che stanno svolgendo un dottorato o un postdoc in Extended Reality e Computer Graphics. È stato chiesto ad ogni esperto di completare il tutorial dell'applicazione e provare le funzionalità di storyboarding per un caso d'uso a scelta e per il tempo necessario a capire il funzionamento dell'applicazione, così da poterne dare un giudizio. I test hanno permesso di evidenziare i punti di forza e di debolezza dell'applicazione e di mettere in risalto gli aspetti da migliorare di quest'ultima. Per esaminare il sistema è stato richiesto ad ogni esperto di valutare l'applicazione secondo le 10 Euristiche di Nielsen dando una valutazione a ogni parametro mediante una scala Likert.

Dai test effettuati sono emersi alcuni importanti ambiti di miglioramento soprattutto per quanto riguarda il feedback fornito agli utenti e l'interazione con il dispositivo. Un aspetto molto importante da considerare è la presenza di un fattore di soggettività legato all'esperienza con il Microsoft HoloLens 2, sono presenti infatti delle difficoltà legate all'esperienza dell'utente, anche date dall'ergonomia del dispositivo. È stata perciò molto importante questa fase che ha permesso, grazie al giudizio degli esperti, di definire gli aspetti chiave da migliorare nell'ottica di condurre uno studio estensivo con gli utenti finali. L'intenzione futura è infatti quella di perfezionare ulteriormente il sistema, migliorandone l'usabilità anche per utenti non esperti, così da poter condurre dei test con gli utilizzatori finali dell'applicazione.

Infine, è molto importante mettere in risalto tutte le possibilità che il sistema ha di evolvere e migliorare in futuro. L'applicativo, infatti, presenta molteplici sviluppi futuri come la possibilità di integrare all'interno dell'applicazione una rielaborazione testuale di un intelligenza artificiale, per rendere più naturale la descrizione dello storyboard, oppure la possibilità di creare dei video con le camere virtuali per la realizzazione di un animatic. Gli sviluppi più interessanti però si concentrano sullo sfruttare a pieno le peculiarità della realtà aumentata: si trovano tra questi lo sviluppo di un interfaccia tangibile che permetta la creazione della scena con modellini e marker reali e la possibilità di visualizzare l'ambiente AR in scala 1:1 a dimensione reale, per immedesimarsi a pieno nel set creato.

Nel capitolo 1 dell'elaborato viene presentato il contesto dell'applicazione, nel capitolo 2 è presente lo stato dell'arte in cui si descrivono le soluzioni già presenti in letteratura, nel capitolo 3 sono descritte le tecnologie utilizzate per la realizzazione del progetto mentre nel capitolo 4 è presente la descrizione del sistema implementato. Nel capitolo 5 sono presentati i test svolti per valutare l'applicazione e i risultati ottenuti. Infine, nel capitolo 6 sono presentate le conclusioni e gli sviluppi futuri.

Indice

Indice delle figure

Indice delle tabelle

Glossario

API: application programming interface o interfaccia di programmazione delle applicazioni. Consente a due applicazioni di comunicare tra loro. .40

AR: Augmented Reality o Realtà aumentata 21

GUI: Graphical User Interface o Interfaccia Grafica Utente consente all'utente di interagire con il computer attraverso icone, finestre e bottoni. 33

HDM: Head-mounted display. Schermo montato direttamente davanti agli occhi dell'utente attraverso l'utilizzo di un caschetto 23; 29; 30; 53

IDE: Ambiente di sviluppo integrato 51

IMU: Inertial Measurement Unit.....22

MRTK: Mixed Reality Toolkit, insieme di librerie open-source per lo sviluppo in realtà mista 55

SPFX: Special Effects 15

TUI: Tangible User Interface o Interfaccia Utente Tangibile. Consente all'utente di interagire con le informazioni digtali tramite degli oggetti fisici 29; 34

UWP: Universal Windows Platform.....56

VR: Virtual Reality o Realtà Virtuale 30

Capitolo 1

1. Introduzione

Nel seguente elaborato viene presentata un'applicazione in realtà aumentata che permette la creazione di storyboard in modalità tabletop. Essa è stata realizzata con il motore grafico Unity ed è stato utilizzato il Microsoft HoloLens, un dispositivo wearable di tipo optical see through.

Questo primo capitolo, oltre agli obiettivi del lavoro di tesi e ad alcuni cenni sulle fasi di preproduzione di un prodotto cinematografico, espone le caratteristiche dei due fattori essenziali che compongono il progetto: l'elemento principale, lo storyboard e la tecnologia di interazione, la realtà aumentata. Nei capitoli successivi dell'elaborato saranno presentati l'analisi dello stato dell'arte, le tecnologie utilizzate per sviluppare l'applicativo, la descrizione del sistema implementato e i risultati ottenuti nella fase di test dell'applicazione. Infine, nell'ultimo capitolo si potranno trovare conclusioni e sviluppi futuri.

1.1 Obiettivo

Lo storyboard è un importante strumento di visualizzazione e progettazione che viene utilizzato sin dalle prime fasi di produzione di un prodotto audiovisivo. Esso permette di rappresentare tramite dei disegni o delle immagini ogni inquadratura di un film. Questa tecnologia richiede competenze tecniche e artistiche professionali e il processo di previsualizzazione può essere un processo molto dispendioso in termini di tempo e denaro [1].

Questo lavoro di tesi ha come obiettivo la progettazione e lo sviluppo di un'applicazione che si avvale di una tecnologia molto potente e all'avanguardia, la realtà aumentata [2], per semplificare il processo di storyboarding.

Per permettere di semplificare il processo, l'applicazione si avvale di funzionalità che permettono all'utente di creare un ambiente 3D AR interattivo, utilizzando gesti, e di visualizzare questo ambiente nel mondo reale. Inoltre, il sistema permette di controllare e animare i personaggi e di utilizzare l'ambiente creato per realizzare le singole vignette dello storyboard in modo naturale, direttamente dal proprio punto di vista nel mondo reale. Questo processo è reso possibile grazie all'utilizzo del Microsoft HoloLens 2,

un caschetto per la realtà aumentata che offre un'interazione immediata con il mondo virtuale. Infine, per semplificare ulteriormente la creazione dello storyboard l'applicazione deve generare un unico file con le vignette, una descrizione testuale, generata automaticamente dal sistema, delle azioni rappresentate in ogni vignetta e tutte le informazioni riguardanti le impostazioni utilizzate per realizzare le inquadrature.

1.2 Preproduzione di un film

Nella realizzazione di un prodotto audiovisivo si delineano principalmente tre fasi fondamentali: la preproduzione, che comprende la progettazione del film, la produzione in cui si procede alla realizzazione delle riprese, e la post-produzione in cui si effettuano il montaggio e l'editing per ottenere il prodotto finale e distribuirlo [3].

La preproduzione è una fase molto complessa e può essere molto dispendiosa in termini di tempo e denaro. Per semplicità si può schematizzare la preproduzione in 4 fasi principali [4]:

- Sviluppo della storia: Nella fase iniziale della preproduzione ci si dedica alla scrittura del concept e del trattamento che permettono la seguente stesura della sceneggiatura del film. Una volta scritta avviene lo spoglio della stessa: si analizza lo script identificando tutto ciò che sarà necessario nelle fasi successive (ad esempio props, cast, SPFX, animali, veicoli…).
- Production design: in questa fase si cerca di ideare quale sarà il look del prodotto. Per prima cosa si procede con la ricerca di reference e la creazione di moodboard che aiutano nella successiva creazione di artwork, caratterizzazione e design dei personaggi e dell'ambiente, scelta dei costumi e tutto ciò che riguarda lo stile dell'universo narrativo. In questa fase si decidono anche quali sono gli effettivi visivi necessari.
- Budgeting: è una fase molto importante per capire quali saranno i costi della produzione del film e per confrontarli con il budget a disposizione.
- Preparazione alle riprese: una volta scelti il cast e la location la crew di preproduzione si dedica ad un operazione molto importante per le riprese ovvero la creazione dello storyboard. Questo strumento permette ad ognuno di capire cosa verrà concretamente realizzato, offrendo una visione di insieme del prodotto. In questa fase inoltre viene creata la shot list, l'elenco delle inquadrature, solitamente in ordine di ripresa. La pre-visualizzazione del film può essere infine completata dalla creazione di un animatic, uno storyboard animato, che contiene informazioni aggiuntive come il timing di ogni azione.

Sebbene ogni strumento abbia la propria importanza lo storyboard risulta essere uno dei più rilevanti perché permette di visualizzare un'anteprima del prodotto prima della sua effettiva realizzazione. Questa tecnica di visualizzazione, in aggiunta, è applicabile a una vasta gamma di ambiti: alcuni esempi sono l'animazione, la pubblicità, l'istruzione, la prototipazione, lo sviluppo di videogiochi.

1.3 Storyboard

Nella realizzazione di un prodotto cinematografico risulta essenziale la fase di preproduzione e di pre-visualizzazione: uno degli strumenti fondamentali in questa fase risulta essere lo storyboard. Lo storyboard è una sequenza di immagini o disegni, posti in ordine cronologico, che rappresentano frame per frame, inquadratura per inquadratura quello che sarà realizzato sul set. Lo storyboard risulta essere essenziale per capire come girare e realizzare ogni singola scena: è cruciale per comprendere che tipo di lenti utilizzare, quale dovrebbe essere la posizione della camera e degli attori durante le riprese e inoltre permette di comprendere quale sarà l'aspetto visivo di un film prima della sua concreta realizzazione.

1.3.1 Cenni storici

Sebbene il concetto di raccontare una storia attraverso l'utilizzo di immagini sequenziali abbia origini ben più antiche, nell'ambito cinematografico possiamo attribuire la nascita dello storyboard a George Méliès. George Méliès era un famoso illusionista, regista e scenografo teatrale francese del diciannovesimo secolo. Egli si può considerare il pioniere nell'utilizzo degli effetti speciali ed è stato uno dei primi registi ad applicare delle strategie di previsualizzazione: Méliès, per la realizzazione di effetti speciali, aveva bisogno di una pianificazione efficiente e perciò produceva numerosi disegni per stabilire come effetti e personaggi si sarebbero integrati all'interno delle scene [5].

I disegni di Méliès, seppure legati alla storia dello storyboard, non possono però considerarsi come gli strumenti che conosciamo noi oggi.

Il processo di storyboarding come è inteso ai giorni nostri è stato sviluppato da Walt Disney Studio nel 1930 [6], si può vedere un esempio in Figura 1.1. Le pratiche di storyboarding sviluppate dalla Disney rappresentano un momento chiave nella storia del cinema: esse hanno ridefinito il processo produttivo dei film di animazione ma anche dei film in live action [5].

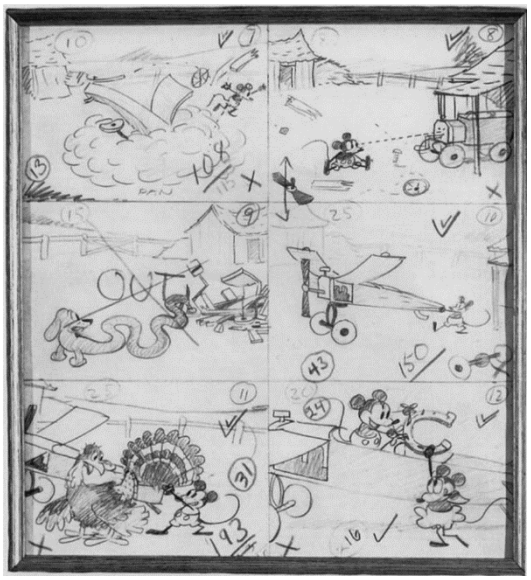

Figura 1.1: Storyboard Walt Disney Studios

1.3.2 Caratteristiche ed elementi di uno storyboard

Le singole vignette di uno storyboard rispettano specifiche caratteristiche e ogni vignetta è accompagnata da alcuni elementi essenziali.

In prima battuta il riquadro che si utilizza per la creazione dello storyboard deve rispettare le proporzioni dell'immagine finale, per questo motivo si scelgono solitamente dei riquadri rettangolari con *aspect ratio* di 16:9 o 4:3 che riprendono i formati standard per la realizzazione di prodotti audiovisivi.

Nella composizione di un inquadratura, inoltre, si cerca di rispettare un principio molto importante in ambito cinematografico: la regola dei terzi. Dividendo l'inquadratura sia orizzontalmente, sia verticalmente in tre sezioni, nei punti intersezione delle 4 linee si vengono a formare dei centri di interesse: in questi punti devono essere posti gli elementi principali dell'inquadratura; si può osservare un esempio in Figura 1.2. La regola viene sempre utilizzata dagli storyboard artist per il corretto posizionamento di oggetti e personaggi [7].

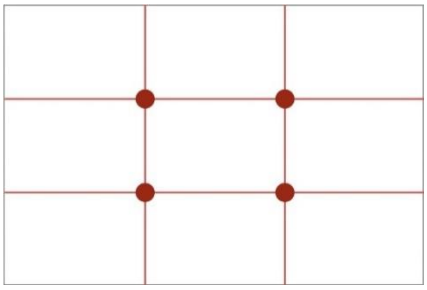

Figura 1.2: Regola dei terzi. Si può osservare la divisione dello schermo secondo la regola dei terzi: i cerchi rossi rappresentano i centri di interesse in cui solitamente si pongono i personaggi nell'inquadratura

È molto importante chiarire che le vignette non sono l'unico elemento presente all'interno dello storyboard ma esso è spesso accompagnato da una serie di informazioni utili per le fasi successive della produzione. Di seguito vengono riportate le principali:

- Vignetta: il disegno è l'elemento principale di uno storyboard, esso permette di capire la posizione dei personaggi, degli elementi in scena e della camera.
- Numero della scena e numero dell'inquadratura: Accanto alla vignetta solitamente è posto il numero della scena e il numero dello shot. Solitamente si indicano separando i due numeri con un punto (1.1: scena 1, inquadratura 1).
- Tipo di obiettivo: viene indicata la lunghezza focale che definisce l'angolo di campo, la porzione della scena che viene ripresa. Si utilizzano obiettivi con lunghezza focale piccola per avere ampi angoli di ripresa (obiettivi grandangolari) e focali più lunghe per angoli più stretti (teleobiettivi)
- Tipo di inquadratura: si parla di campo (lunghissimo, lungo, medio, totale, dettaglio) nel caso in cui siano inquadrati oggetti o paesaggi e si voglia dare più importanza all'ambiente mentre si parla di piano (figura intera, piano americano, primo piano, primissimo piano) nel caso in cui siano inquadrati esseri umani e si voglia dare più importanza ai personaggi.
- Durata dell'inquadratura: tempo stimato per la durata dell'inquadratura rappresentata
- Descrizione della scena: viene descritto sottoforma di testo quello che accade all'interno della vignetta. Si possono riprendere delle parti di testo presenti all'interno della sceneggiatura.
- Movimenti della macchina da presa: vengono solitamente indicati con delle frecce vuote bianche. Il movimento può anche essere descritto a parole, alcuni esempi sono pan, tilt, zoom in, zoom out, dolly, camera roll. I movimenti di camera possono essere anche essere indicati tra una vignetta e l'altra.
- Movimenti dei personaggi: per rappresentare i movimenti dei personaggi si possono utilizzare delle frecce piene che indicano la direzione in cui si muove un personaggio.

- Cambio di inquadratura: possono essere indicati nella stessa vignetta, ad esempio utilizzando più rinquadri, oppure tra una vignetta e l'altra. Un esempio può essere una mezza figura che diventa un primo piano.
- Location e atmosfera: dove è ambientata la scena e che luce e ambiente si vogliono ricreare.
- Dialoghi: eventuali dialoghi presenti tra i personaggi
- Suoni: possono essere indicati suoni presenti all'interno della scena
- VFX: nel caso in cui nella scena siano presenti degli effetti visivi è buona norma indicarli anche all'interno dello storyboard. Questo permette di capire l'integrazione di elementi reali ed elementi grafici.
- Costo di un'inquadratura: nel caso di inquadrature particolari.

La Figura 1.3 mostra un esempio di storyboard in cui vengono indicate alcune informazioni come i movimenti di camera, numero della scena e dell'inquadratura e descrizione di ciò che è rappresentato nella vignetta.

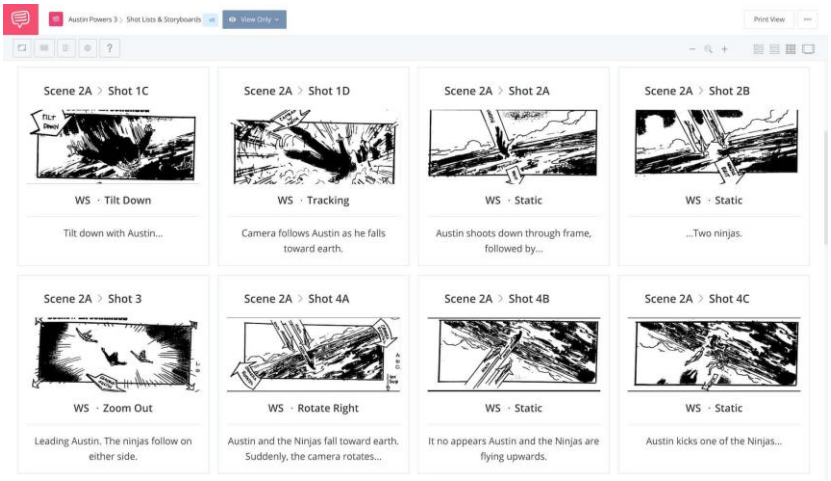

Figura 1.3: Esempio di storyboard con alcune informazioni essenziali

In base alle necessità uno storyboard può anche abbandonare lo stile tradizionale: si possono utilizzare pannelli sproporzionati o pannelli non rettangolari per rappresentare inquadrature o movimenti particolari, ad esempio, per la realizzazione di un piano sequenza si può utilizzare un unico lungo pannello in cui si rappresentano movimenti di camera e movimenti dei personaggi.

Inoltre, le informazioni che completano lo storyboard possono cambiare in base alle esigenze e in base al tipo di prodotto che viene realizzato: gli storyboard, infatti, non vengono solo utilizzati nell'industria cinematografica ma possono essere impiegati per facilitare la creazione di applicazioni, videogiochi o fumetti.

1.4 La realtà aumentata

La realtà aumentata è una tecnologia che sovrappone informazioni digitali e virtuali ad oggetti o luoghi reali allo scopo di migliorare l'esperienza dell'utente [8].

1.4.1 Rapporto tra realtà aumentata e virtuale: il Reality-Virtuality Continuum

Realtà virtuale e aumentata sono senza ombra di dubbio collegate tra loro, si possono rappresentare infatti come due elementi del Reality-Virtuality Continuum [9]. Questo continuum pone ad un estremo il mondo reale, costituito esclusivamente da oggetti reali e che comprende tutto ciò che si può osservare nella realtà, mentre all'altro estremo si definisce l'ambiente virtuale composto esclusivamente da oggetti virtuali, in cui tutto il mondo reale viene escluso dalla simulazione.

È facile a questo punto definire tra il mondo reale e la realtà virtuale la realtà mista (mixed reality MR): al suo interno troviamo la realtà aumentata (AR) che è costituita da oggetti virtuali all'interno del mondo reale e la virtualità aumentata (AV) che pone gli oggetti reali all'interno del mondo sintetico. Una rappresentazione del virtuality-reality continuum si può vedere in Figura 1.4.

Figura 1.4: Rappresentazione del reality-virtuality continuum

1.4.2 Cenni storici

Sebbene il termine *realtà aumentata* risalga alla fine degli anni 90, il concetto di AR è molto più antico e si possono trovare tracce del suo utilizzo fino a

partire dalla Seconda guerra mondiale. La realtà aumentata permette infatti di *aumentare* ciò che è reale fornendo informazioni aggiuntive agli utenti che ne fanno uso.

L'integrazione di virtualità e realtà non è però un operazione semplice: è necessaria grande potenza di calcolo e l'elaborazione di un grande quantitativo di informazioni. Per questo motivo la realtà aumentata si sviluppa maggiormente a partire dai primi anni duemila: gli enormi passi della tecnologia permettono di ridurre il divario tra mondo fisico e dell'informazione e mettere insieme gli elementi dei due mondi, questo permette di sviluppare le potenzialità della realtà aumentata e mixata.

1.4.3 Architettura di un sistema AR

La realtà aumentata è un sistema che combina percezioni dirette del sistema umano e percezioni mediate, cioè elementi digitali che si sovrappongono alla realtà. Il compito di un AR designer è quello di creare degli oggetti dinamici con un proprio comportamento e che dipendono dalla logica di gestione delle interazioni dell'utente. Allo stesso tempo, l'utente in questo sistema non è solo fruitore delle informazioni ma determina in che modo evolverà l'applicazione.

La connessione tra mondo virtuale e mondo fisico avviene attraverso una serie di sensori come fotocamere, microfoni, sistemi IMU, accelerometri e tracker di rilevamento che permettono di estrarre informazioni per il mondo circostante.

L'elemento che si occupa dell'estrazione dei dati viene chiamato *analizzatore di contesto*: esso si occupa di raccogliere ed estrarre le informazioni dei sensori ed elaborarle per fornirle agli altri componenti del sistema AR. Le informazioni sono fondamentali: esse servono per garantire la corretta interazione tra elementi reali e virtuali e per permettere all'utente di agire sul sistema.

L'output dell'analizzatore di contesto viene inviato all'*AR* engine, il motore di gestione dell'ambiente AR: esso si occupa di gestire i dati e gli eventi, seguendo le regole di gestione della logica dell'applicazione. Per gestire l'evoluzione dell'applicazione il motore AR deve riuscire a interpretare le interazioni dell'utente: esse possono essere diverse in base ai tipi di azioni eseguite e posso dipendere dai paradigmi di gestione dell'applicazione AR. In Figura 1.5 si può osservare una rappresentazione grafica dell'architettura del sistema con tutti gli elementi descritti.

Infine, dal momento che l'interazione creata deve essere efficace per permettere di fondere perfettamente mondo virtuale e reale, i dispositivi utilizzati dovranno essere in grado di gestire la percezione mista: questo

comporta la generazione di un output multicanale che comprenda non solo l'aspetto visivo ma anche gli aspetti uditivi e tattili. Tutto ciò permette di fornire all'utente un sistema coerente favorendo l'immersività [10].

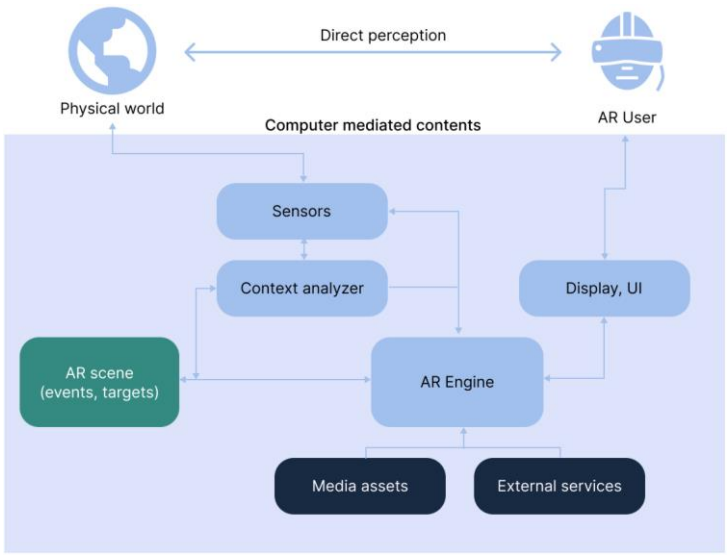

Figura 1.5 : Rappresentazione dell'architettura di un sistema AR

1.4.4 Dispositivi AR

I sistemi che permettono agli utenti di interfacciarsi con la realtà aumentata sono molteplici. Si possono identificare i paradigmi di interazione della realtà aumentata in 3 macrocategorie:

- HDM: utilizzano caschetti per la realtà aumentata, posti direttamente davanti agli occhi dell'utente. Si dividono a loro volta in dispositivi ottici, che fanno utilizzo di lenti trasparenti in cui viene proiettato il mondo virtuale (detti *optical see through*) come l'HoloLens 2 e dispositivi che utilizzano lenti opache con all'interno display che riproducono le immagini aumentate (detti anche *video see-through*). Si può vedere un esempio di HDM in Figura 1.6.

Figura 1.6: Leap Motion, esempio di headset per la realtà aumentata

- Display-based: in questo caso si parla di un paradigma che utilizza un display non posto direttamente davanti agli occhi dell'osservatore. Anche in questo caso il paradigma si divide in due categorie: i sistemi che utilizzano un sistema di visualizzazione fisso con l'ausilio di monitor e una camera mobile che riprende l'ambiente reale e i sistemi mobile, come smartphone e tablet, in cui sia la camera che il display di visualizzazione sono mobili, di cui si può vedere un esempio in Figura 1.7.

Figura 1.7: Esempio di realtà aumentata display-based. In questo caso si parla di sistema mobile che utilizza un tablet.

- Projector based: in questo caso i contenuti virtuali sono proiettati sulla geometria dello spazio fisico e si parla di *spatial AR*. Si può vedere un esempio di questa tecnologia in Figura 1.8.

Figura 1.8: Esempio di applicazione AR Projector Based

1.4.5 Ambiti di applicazione

Al giorno d'oggi l'utilizzo della realtà aumentata trova spazio in moltissimi settori: essa trova applicazione sia in settori scientifici e tecnici sia nel mondo del marketing e dell'intrattenimento. Di seguito sono riportati i maggiori ambiti di applicazione e alcuni esempi:

- Medicina: questo risulta essere uno dei campi di applicazione più interessanti per l'utilizzo della realtà aumentata. L'AR, infatti, oltre a permettere di sovrapporre elementi virtuali al mondo reale, permette di stimare la posizione tridimensionale degli elementi virtuali nell'ambiente circostante. Ciò permette di migliorare le procedure mediche all'interno del mondo reale consentendo di visualizzare regioni di interesse e fornire informazioni altrimenti invisibili o oscurate nella visione reale [11]
- Turismo: la realtà aumentata permette di migliorare l'interazione degli utenti con il mondo circostante, è per questo motivo che il suo utilizzo trova molte opportunità nell'industria del turismo. La realtà aumentata permette di esplorare in modo alternativo luoghi sconosciuti, offrendo informazioni per migliorare l'esperienza, guidando l'utente in esperienze più creative e coinvolgenti [12]. Si cita come esempio l'applicazione TabUi [13], un applicazione mobile che permette l'esplorazione del territorio.
- Marketing: un numero sempre più crescente di aziende utilizza la realtà aumentata per scopi commerciali: l'AR viene utilizzata per mostrare i prodotti, dare maggiori informazioni agli utenti e rendere l'esperienza di acquisto più divertente e coinvolgente. È doveroso citare come applicazione in questo ambito IKEA Place [14].

- Istruzione: le caratteristiche della realtà aumentata la rendono ideale nel campo dell'insegnamento. L'AR ha il potenziale per coinvolgere, stimolare gli studenti, aiutarli ad osservare i concetti da diversi punti di vista e permette di fare un'esperienza diretta del mondo reale. Inoltre, un sistema AR permette di migliorare la collaborazione e promuovere la creatività [15].
- Intrattenimento: l'industria videoludica è quella che ha maggiormente impattato sullo sviluppo delle nuove tecnologie. Nell'ambito AR si sono sviluppati in modo notevole simulazioni e videogiochi, in particolare nell'ambito mobile; a questo proposito è doveroso citare l'applicazione AR Pokémon GO [16].

È doveroso ricordare che, oltre a quelli citati sopra, la realtà aumentata si è espansa in moltissimi altri ambiti, ad esempio, l'ambito militare e di addestramento, le applicazioni per le operazioni di manutenzione, i musei, l'architettura, il design, l'industria e molto altro ancora.

Capitolo 2

2. Stato dell'arte

2.1 Generazione di Storyboard

Nelle prime fasi della produzione, gli storyboard vengono utilizzati per descrivere visivamente la sceneggiatura tramite l'utilizzo di disegni e immagini per capire in che modo organizzare e filmare le scene. Gli storyboard principalmente si concentrano sulla composizione della scena in generale, ponendo attenzione sulle inquadrature, i movimenti di camera e le azioni, la posizioni e i movimenti dei personaggi.

Lo storyboard tradizionale viene creato con carta e matita ma esistono anche diversi strumenti digitali: per primi si possono citare gli strumenti per il disegno digitale come tablet e tavolette grafiche che possono semplificare il processo fornendo funzioni come, ad esempio, il copia e incolla o l'utilizzo di griglie.

Esistono inoltre diversi strumenti digitali che permettono agli utenti di creare gli storyboard senza vere e proprie conoscenze artistiche, essi permettono di creare una vignetta 2D inserendo disegni e immagini, presenti all'interno di librerie. Si può citare a questo proposito StoryboardThat [17], uno storyboard d'esempio realizzato con questo tool si può vedere in Figura 2.1.

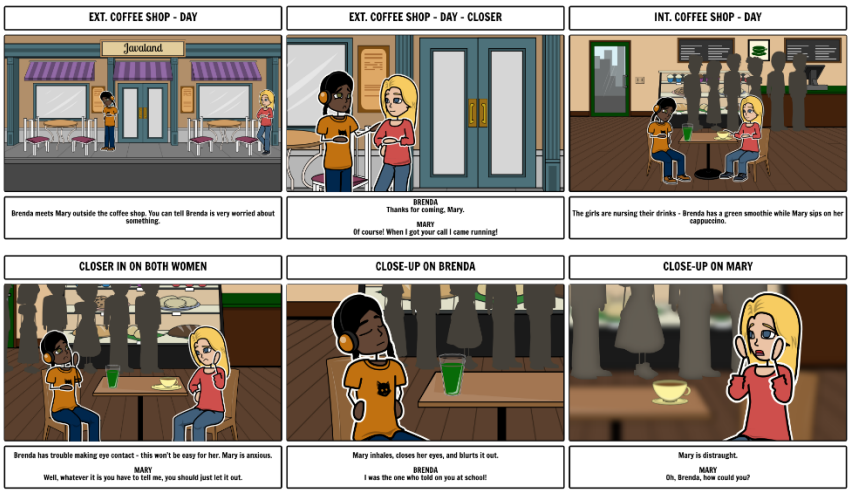

Figura 2.1: Esempio di storyboard creato con StoryboardThat per una scena di un film.

Nella creazione degli storyboard il posizionamento e l'orientamento dei personaggi e della macchina da presa risultano essere dei fattori fondamentali per la creazione di un progetto efficiente, si sono perciò sviluppati strumenti più complessi per la manipolazione di ambienti e oggetti 3D sia utilizzando dispositivi desktop, sia con l'utilizzo di realtà aumentata e virtuale.

2.2 Storyboarding in VR e AR

Il crescente sviluppo delle tecnologie di visualizzazione come la realtà aumentata o mista ha portato a considerare l'utilizzo di queste tecnologie in una grande varietà di domini. Realtà virtuale e aumentata supportano un'esperienza interattiva e immersiva 3D e possono portare a considerare a svariati nuovi elementi e strumenti utili alla creazione di uno storyboard.

Tramite l'utilizzo di queste tecnologie l'utente può cambiare posizione ed orientamento, ciò gli permette di visualizzare eventi e storia da molteplici punti di vista, inoltre l'utente vive un'esperienza molto più immersiva e interattiva, nella quale può interagire con gli oggetti e con lo spazio circostante, ovviamente sempre tenendo conto della libertà di movimento che ha nello spazio.

Lavorare in realtà virtuale può migliorare lo storyboard dando un maggiore controllo della telecamera e degli asset animati: questi strumenti possono essere utilizzati per manipolare lo spazio e creare immagini appiattite da inserire nello storyboard tradizionale. L'utilità maggiore di questi strumenti, perciò, risulta essere lo sviluppo e la progettazione dell'ambiente della scena.

Per quanto riguarda le applicazioni in realtà aumentata o più in generale in realtà mista, si presuppongono degli elementi fissi su cui basare l'interazione, come un tavolo o un pavimento. Su questa superficie si possono sovrapporre oggetti reali e virtuali e creare uno storyboard in cui si distinguono, tramite diverse notazioni, oggetti reali, virtuali e simbologie per rappresentare i movimenti, che guidano gli utenti nella lettura dello storyboard [18]

2.2.1 AR Storyboard: Generazione di Storyboard in AR

AR Storyboard [19] è uno strumento per la creazione di storyboard interattivi in realtà aumentata. Lo strumento utilizza una tecnologia di realtà aumentata basata su dei marcatori che permette la composizione di scene 3D utilizzando delle interfacce poste nell'ambiente reale. Questo progetto utilizza la camera di un computer e una serie di *"Item blocks":* questi ultimi sono dei blocchi di carta sui quali sono stampate delle informazioni visive, quando posizionati nella

visuale della camera la scena, composta dai modelli 3D, viene renderizzata nella vista in realtà aumentata.

I blocchi, rappresentati in Figura 2.2, sono caratterizzati in due macrocategorie: i blocchi statici che comprendono personaggi, sfondi e oggetti e blocchi dinamici che sono costituiti da azioni ed espressioni facciali. Mentre i blocchi statici sono utilizzati per posizionare nella scena personaggi, costruzioni e proprietà dello spazio, i blocchi dinamici permettono la creazione di animazioni per i personaggi. Una scena viene composta preparando gli *"item blocks"*, una volta scelti essi vengono posizionati nella vista della telecamera che renderizza i modelli 3D in base alla posizione e all'orientamento dei blocchi. I blocchi dinamici, invece, possono essere posizionati ovunque nella vista della camera e permettono la creazione di scene e comportamenti complessi. L'utente può catturare le immagini fisse desiderate, controllando contemporaneamente la posa della telecamera, utilizzando la camera del computer.

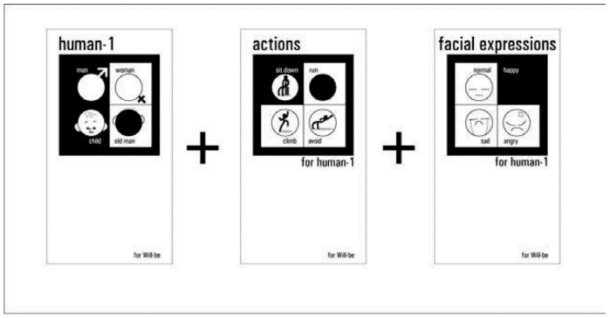

Figura 2.2: Rappresentazione dei blocchi per la creazione dello storyboard

È interessante notare che lo strumento permette la visualizzazione della scena virtuale all'interno di un ambiente reale e permette di modificare l'ambiente e i personaggi tramite la manipolazione di oggetti reali. Da sottolineare sono però anche i problemi quali l'utilizzo ARToolkit[4] che ha un campo di tracciamento limitato, inoltre sono presenti dei problemi di libertà dei movimenti della camera che necessiterebbero di una tecnologia di tracciamento avanzata.

Un esempio del risultato ottenuto con il sistema è visibile in Figura 2.3.

L'applicazione sviluppata per questo progetto di tesi, sebbene utilizzi la tecnologia AR, si differenzia da AR Storyboard in quanto utilizza un HDM, il Microsoft Hololens 2, che permette il tracciamento dello spazio tramite una camera integrata nel dispositivo. Inoltre, la tecnologia avanzata del dispositivo permette la creazione della scena senza l'utilizzo di marker o oggetti fisici ma direttamente con oggetti virtuali: il mondo aumentato viene visualizzato direttamente con il dispositivo *optical see through*.

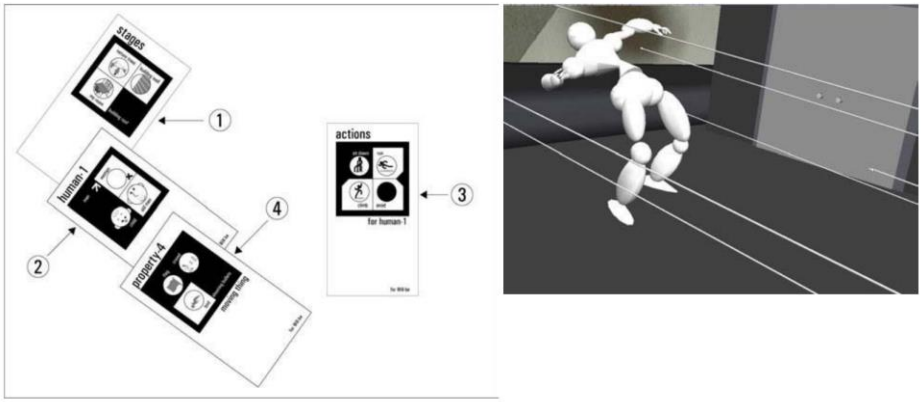

Figura 2.3: Risultato di una scena ottenuta con Storyboard AR

2.2.2 Augmented Reality Storytelling: Story CreatAR

I dispositivi AR HDM (Head-mounted display) permettono un esperienza immersiva e interattiva che comprende l'utilizzo delle mani e quelli senza fili, come HoloLens [20], permettono di spostarsi in ambienti diversi, posizionando ovunque elementi della storia, integrando elementi virtuali con il mondo fisico.

Quando non è disponibile una location fisica un ambiente può essere simulato in un HMD VR che riproduce l'ambiente e gli elementi previsti. Quando l'ambiente è molto complesso questo processo può richiedere molto tempo e può essere complesso integrare perfettamente gli elementi della storia con l'ambiente. È ancora più difficile quando la storia è ambientata in luoghi sconosciuti, a causa della limitata conoscenza dell'ambiente e la difficoltà nel posizionare gli oggetti.

Story CreatAR [21] è uno strumento di authoring locativo che integra tecniche di analisi spaziale, è pensato per permette agli autori di sperimentare e riflettere sul rapporto tra gli elementi di una storia e l'ambiente in cui si trovano, indipendentemente però dall'ambiente in cui la storia sarà vissuta. Tramite questo tool gli utenti possono definire:

- regole di posizionamento degli oggetti in base alle caratteristiche dello spazio
- regole di attraversamento, assegnando comportamenti ai personaggi
- raggruppamento di elementi, in modo tale da poterli inserire insieme e per definire una sequenza di eventi

- regole di formazione, regolando il posizionamento dei gruppi di elementi.

Story CreatAR è fornito di librerie di personaggi, animazioni, oggetti e audio spazializzati ma i contenuti possono anche essere integrati con risorse di terze parti o modelli importati dall'utente. Essi possono essere etichettati e raggruppati per sviluppare la storia, inoltre il posizionamento di essi dipende dall'analisi spaziale dell'ambiente e, una volta definite le regole, essi vengono collocati in posizioni appropriate.

Regole di posizionamento di livello superiore dette *attributi* possono essere associate agli elementi della storia: ognuno di essi ha un nome e delle caratteristiche spaziali, in questo caso quelle supportate sono apertura, complessità visiva e integrazione visiva a cui sono attribuiti dei valori in base a come gli autori hanno descritto gli elementi nelle bozze di storia. Gli attributi inoltre possono essere modificati e personalizzati. Dopo che l'autore è soddisfatto del posizionamento degli elementi, può salvare le posizioni degli elementi della storia in un file Unity [22] in cui può scegliere di effettuare delle regolazioni manuali rispetto al posizionamento di ogni risorsa. Una volta conclusa, inoltre, la scena potrà essere visualizzata in VR (Oculus Quest [23]) e in AR (Hololens 2 [24]).

Il tool Story CreatAR può supportare differenti paradigmi narrativi e permette agli eventi di seguire l'ordine cronologico attraverso una sequenza di avvenimenti legati al tempo. Ogni evento può essere personalizzabile: si possono definire sequenza, durata e ripetizione di un azione. Inoltre, è possibile visualizzare la trama utilizzando un grafo in cui ad ogni nodo corrisponde un evento.

Lo strumento supporta le interazioni dell'utente e gli permette di entrare a far parte della storia, inoltre il paradigma *avatar based* favorisce l'osservazione e l'interazione dello spettatore con gli avatar nell'ambientazione della storia. L'autore può creare dei nodi di conversazione con diverse proprietà e una volta che la conversazione è stata specificata può modificarne i parametri principali. In aggiunta, grazie all'integrazione con MoveBox [24], Story CreatAR fornisce ai personaggi animazioni e stili vocali che permettono agli avatar di svolgere le conversazioni.

Infine, Story CreatAR dispone di opzioni per specificare due tipi di eventi globali: eventi di attraversamento ed eventi timer. Nell'evento di attraversamento l'autore specifica l'avatar che deve muoversi, in che punto si deve spostare e la velocità del movimento: in questo modo il personaggio si

muove verso la destinazione trovando il percorso migliore. Nell'evento timer l'autore invece specifica un tempo di attesa. Questi eventi aiutano la progressione della storia. In Figura 2.4 si può osservare un riassunto delle funzionalità del sistema.

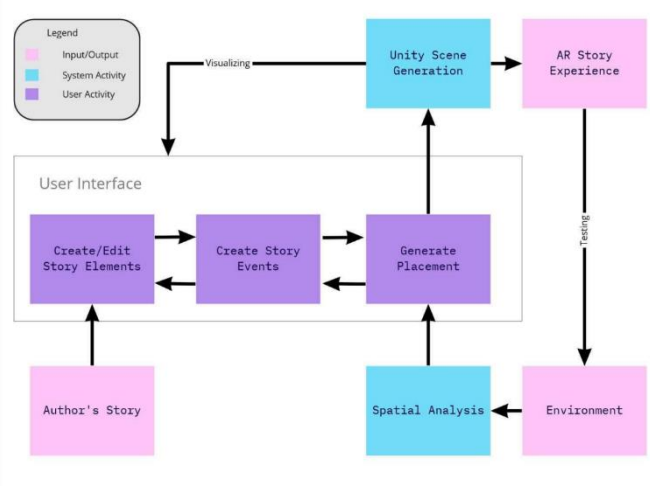

Figura 2.4: Panoramica di tutte le funzionalità e del flusso di lavoro del sistema Story CreatAR

Story CreatAR è stato sviluppato come plugin di Unity e diventerà un software open source dopo la pubblicazione. Gli strumenti di analisi spaziale depthMapX [25] e AFPlan [26] vengono utilizzati per importare gli attributi spaziali in ambiente di distribuzione o di test e l'autore, perciò, può visualizzare l'ambiente in Unity o testare l'esperienza della storia in VR e AR.

Questo progetto offre degli interessanti spunti riguardanti la creazione della scena, le animazioni dei personaggi e le tecniche di storytelling, elementi alla base della creazione di uno storyboard. Bisogna tener conto però della complessità con cui viene definito il sistema che potrebbe impattare sui tempi per la creazione dello storyboard, uno strumento tipicamente più veloce e semplice.

Nell'applicativo di cui parla questo lavoro di tesi si è infatti deciso di concentrarsi principalmente sulla creazione dello storyboard. Per fare ciò non sono presenti vincoli sulla creazione dello spazio o vincoli narrativi: l'applicazione permette all'utente di creare la scena aggiungendo gli oggetti virtuali da una libreria, non sono presenti attributi di posizionamento o regole. Inoltre, nella realizzazione dello storyboard si utilizzano azioni semplici che permettono ai personaggi di interagire tra loro e con l'ambiente, per realizzare le vignette dello storyboard.

2.3 Interfacce tangibili

2.3.1 A tangible tabletop for industry storyboarding

Le interfacce utente tangibili (TUI, Tangible User Interfaces) sono delle interfacce che permettono di controllare lo spazio virtuale attraverso la manipolazione di oggetti fisici, che rappresentano informazioni digitali. Un'interfaccia utente tangibile permette all'utente l'acquisizione di competenze in modo intuitivo e semplice, tramite la manipolazione di oggetti fisici [27].

Previz 2015 [28] è il frutto di una ricerca sperimentale svolta dell'Università di Città del Capo con lo scopo di indagare l'usabilità di un sistema tangibile per lo storyboarding in un contesto professionale. Per realizzare il progetto è stata quindi creato un sistema tabletop per la creazione di uno storyboard tramite l'utilizzo di personaggi stampati in 3D.

Il sistema è composto dai modellini fisici, una scrivania, un sensore di profondità e un monitor fornito di un gamepad per il controllo del sistema. La posizione e l'orientamento dei modelli sono catturati dal sensore di profondità e la scena del mondo reale viene ricreata nella GUI. L'utente posiziona i modelli fisici dei personaggi e della telecamera della scena su un tavolo e il sistema interpreta il loro posizionamento componendo un'inquadratura dello storyboard dal punto di vista della camera. Il fotogramma creato viene visualizzato su un monitor: questo processo può essere ripetuto per ogni inquadratura dello storyboard.

Infine, il sistema mette a disposizione dell'utente un gamepad per manipolare e navigare lungo la sequenza di fotogrammi. In Figura 2.5 si può vedere un fumetto che spiega l'utilizzo del sistema.

Figura 2.5: Illustrazione che rappresenta le funzionalità e l'utilizzo di Previz

Bisogna sottolineare che nel sistema descritto sono state identificate delle limitazioni date dalla tecnologia: per prima cosa i modelli potevano essere spostati con soli 3 gradi di libertà, non era consentita l'occlusione del sensore e la costruzione della scena presentava dei problemi di precisione del riconoscimento dei modelli, infine il feedback presentato era ritardato e perciò non permetteva l'aggiornamento in tempo reale della scena.

Nello studio l'usabilità del sistema è stata verificata da 20 partecipanti, tutti appartenenti all'industria dell'animazione: è stato dimostrato che le TUI (Tangible User Intefaces) sono generalmente adatte alla generazione di storyboard anche se sono presenti svariate limitazioni.

La coordinazione mani-occhi degli utenti può non soddisfare i requisiti di accuratezza e gli oggetti fisici 3D non sono flessibili in quanto non possono cambiare posizione o svolgere azioni, cosa che invece è semplice e immediata nello storyboard tradizionale. Gli oggetti tangibili non possono tornare a stati precedenti e perciò non sono presenti strumenti di annullamento delle azioni come nei software di manipolazione di oggetti 3D come Maya [29] o Blender [30].

L'applicativo Unity si differenza da Previz 2015 in quanto non fa utilizzo di oggetti fisici. Grazie all'utilizzo del dispositivo Microsoft HoloLens 2 è possibile interagire con gli oggetti virtuali direttamente con l'utilizzo delle mani, fornendo un'interfaccia più naturale. Allo stesso tempo ciò permette di scalare gli oggetti a piacimento, permettendo maggior libertà all'utente. Infine, l'utilizzo di un dispositivo di questo tipo permette di visualizzare il mondo *aumentato* direttamente attraverso lo schermo posto davanti agli occhi.

2.3.2 Manipolazione di oggetti in realtà aumentata: realtà aumentata tangibile

Le interfacce AR tangibili [31] permettono di combinare le possibilità avanzate di visualizzazione della realtà aumentata con la manipolazione intuitiva e interattiva di oggetti fisici. La metafora tangible AR sostiene infatti la continua l'interazione tra il mondo reale e quello virtuale, fornendo un'ampia gamma di interazioni naturali. Le interfacce AR tangibili sono quelle in cui ogni oggetto virtuale corrisponde ad un oggetto fisico e in cui l'utente interagisce con gli oggetti virtuali manipolando gli oggetti tangibili corrispondenti.

Nella Realtà Aumentata c'è un'intima relazione tra i modelli virtuali 3D e gli oggetti fisici a cui questi modelli sono collegati. Ciò suggerisce che una promettente direzione di ricerca può derivare dallo sfruttamento

dell'immediatezza e della familiarità di oggetti fisici quotidiani per una manipolazione efficace di oggetti virtuali.

Anche se intuitive le interfacce tangibili possono portare a delle complicazioni per quanto riguarda la visualizzazione: è infatti complesso modificare dinamicamente le proprietà fisiche di un oggetto e perciò spesso la visualizzazione si limita alla proiezione di immagini su piani o superfici aumentate. Inoltre, quando si parla di visualizzazione 3D, spesso si assiste ad una disconnessione tra lo spazio delle attività e lo spazio di visualizzazione. Le attuali interfacce tangibili, quindi, forniscono una manipolazione molto intuitiva dei dati digitali, ma possiedono un supporto limitato per la visualizzazione di oggetti virtuali 3D. Al contrario, le interfacce AR forniscono una buona interfaccia per la visualizzazione di modelli virtuali, ma possono avere un limitato supporto all'interazione.

La maggior parte delle interfacce AR fornisce una visualizzazione spaziale senza soluzione di continuità, che sovrappone il mondo dell'interazione al mondo reale, tuttavia, spesso l'utente deve apprendere nuove tecniche e strumenti per manipolare un contenuto virtuale e perciò le interfacce AR possono introdurre un'interruzione cognitiva. Un'interfaccia tangible AR consente invece agli utenti di interagire con questo contenuto virtuale utilizzando le stesse tecniche che utilizzerebbero con un oggetto fisico reale, facilitando la visualizzazione e l'interazione senza soluzione di continuità. Per ottenere un'interfaccia di questo tipo si utilizzano i concetti appresi nello studio delle interfacce utente tangibili tra cui:

- L'uso di controller fisici per la manipolazione di contenuti virtuali;
- Supporto per tecniche di interazione spaziale 3D;
- Supporto all'interazione nel tempo e nello spazio;
- Supporto per l'interazione a più mani:
- Supporto per la corrispondenza dei vincoli fisici dell'oggetto ai requisiti dell'attività da svolgere;
- Capacità di supportare attività parallele con più oggetti;
- Collaborazione tra più partecipanti.

Le interfacce AR che seguono questi principi di progettazione forniscono un'interazione senza soluzione di continuità e perciò risultano estremamente intuitive da usare.

Per studiare l'efficacia delle interfacce AR tangibili sono stati sviluppati dei prototipi tra cui *Magic Paddle* [32], un'applicazione per la creazione di scene

virtuali. Il progetto ha come obbiettivo la creazione di un'interfaccia tangibile AR che utilizza un singolo dispositivo di input per l'assemblaggio di scene virtuali.

I componenti fisici dell'interfaccia comprendono un vero e proprio libro, una paletta di cartone, un grande foglio di carta e un HMD. Ciascuno degli oggetti riflette la propria funzione: il libro funge da contenitore per tutti i modelli virtuali, la paletta è il principale dispositivo di interazione e il foglio di carta lo spazio di lavoro. Aprendo il libro su ciascuna delle sue pagine l'utente può osservare set differenti di oggetti virtuali, esattamente sovrapposti alle pagine del libro reale. Il foglio di carta permette di visualizzare una stanza virtuale vuota in cui si possono trasferire gli oggetti dal libro utilizzando la paletta, il principale oggetto di interazione, dotato di un marker per il tracciamento. L'applicazione è progettata per essere utilizzata con entrambe le mani e consente all'utente di effettuare gesti statici e dinamici per interagire con gli oggetti virtuali, è presente una rappresentazione dell'interfaccia in Figura 2.6.

Figura 2.6: Interfaccia del prototipo Magic Paddle

Per spostare gli oggetti l'utente posiziona la paletta accanto all'oggetto desiderato presente sul libro ed esso viene copiato sulla paletta: a questo punto il modello può essere spostato e visualizzato da tutti i punti di vista, inoltre si possono utilizzare movimenti della paletta per posarlo, spostarlo nella stanza o rimuoverlo. Le interazioni sono molto naturali e in poco tempo l'utente può assemblare la scena in modo veloce e intuitivo.

Per la registrazione degli oggetti virtuali e reali nello spazio in questo progetto sono stati utilizzati dei marcatori e il sistema di tracciamento ARToolkit. I marker sono semplici quadrati neri con un motivo unico al loro interno che permette loro di essere estratti e identificati. L'orientamento e la posizione dei marcatori vengono stimate dalle coordinate dei 4 vertici e le immagini virtuali vengono disegnate sull'immagine di input.

Le interfacce AR tangibili sono interfacce trasparenti che forniscono una perfetta interazione 3D a due mani con oggetti virtuali e fisici e non richiedono ai partecipanti di utilizzare dispositivi particolari di interazione. Gli utenti possono manipolare gli oggetti virtuali utilizzando le proprie mani su qualsiasi superficie di lavoro e questo porta ad un'interazione senza soluzione di continuità con il mondo digitale e fisico.

L'applicazione sviluppata in questo progetto di tesi non fa utilizzo di interfacce tangibili: sebbene gli oggetti fisici forniscano un approccio più naturale si è scelto di non interagire con oggetti reali, ad eccezione del piano di appoggio per la creazione della scena. L'interazione dell'applicazione però risulta comunque essere abbastanza naturale per l'utente: grazie al dispositivo utilizzato infatti è possibile interagire con gli oggetti e con il sistema attraverso le mani, la testa e la voce.

2.4 Animazione in realtà aumentata

2.4.1 Animazione con dispositivi mobile

Una volte indagate le maggiori tecniche di posizionamento e manipolazione di oggetti nello spazio 3D bisogna soffermarsi su un altro argomento molto importante: l'animazione.

In uno studio di *Manabu Eitsuka e Masahito Hirakawa* [33] viene mostrato lo sviluppo di un sistema per la creazione di animazioni su oggetti virtuali che utilizza dispositivi mobile in realtà aumentata.

Nel sistema l'utente può definire determinate posture dell'oggetto virtuale nei fotogrammi chiave per creare una sequenza che specifica l'animazione. La manipolazione dell'oggetto virtuale viene eseguita con le dita, poste davanti a una telecamera, sul retro del telefono cellulare. Ciò fornisce all'utente la sensazione di gestire direttamente l'oggetto virtuale all'interno dello spazio 3D aumentato.

Come si può osservare in Figura 2.7 l'oggetto da manipolare viene presentato su un marker nello spazio AR, l'utente può definire le sue pose con le mani: il movimento delle dita permette la manipolazione sulle tre dimensioni dello spazio e l'utente può variare il punto di vista semplicemente spostando il cellulare nell'ambiente.

Figura 2.7: Illustrazione del funzionamento del sistema di animazione in AR.

Nel progetto è stato utilizzato come piattaforma hardware un iPhone 4 mentre l'architettura di sistema è divisa in una parte di elaborazione delle immagini e una parte di elaborazione grafica. Inoltre, è stato utilizzato Vuforia [34] come piattaforma software AR che permette il tracciamento dei marcatori di alto livello.

Il software permette di rilevare un marker davanti alla fotocamera ed un modello 3D ad esso associato viene presentato sul display con OpenGL ES [35], creando una scena aumentata nel mondo reale. La parte di elaborazione delle immagini invece si occupa di analizzare i fotogrammi catturati dalla fotocamera e di riconoscere e seguire il dito dell'utente: i dati di posizione del dito vengono inviati alla parte di elaborazione grafica per determinare il modello 3D da modificare.

Il framework presentato consente agli utenti di interagire direttamente con i modelli 3D utilizzando le dita per la creazione di animazioni, questo risulta essere un modello molto potente ed intuitivo. I principali problemi sono legati all'analisi dei movimenti delle dita nello spazio 3D e in particolare i miglioramenti possono essere studiati per quanto riguarda la precisione e la velocità di esecuzione del riconoscimento del movimento.

Il sistema si differenza da quello sopra citato in quanto le animazioni sono scelte da una libreria, non è possibile infatti mettere in posa il personaggio manualmente. Questo risulterebbe molto complesso dato che l'applicativo si pone l'obiettivo di permettere la creazione di uno storyboard in modo rapido e intuitivo.

2.4.2 ARAnimator

Un altro progetto che si pone come obiettivo la creazione di animazioni in realtà aumentata è *ARAnimator* [36]. Il sistema si pone l'obiettivo di sviluppare uno strumento intuitivo per la creazione di animazioni di

personaggi in ambienti reali complessi, tramite l'utilizzo della realtà aumentata. Il sistema viene proposto per utenti occasionali e prevede la creazione dell'animazione tramite i movimenti stessi del dispositivo, come rappresentato in Figura 2.8, seguiti dall'editing interattivo attraverso un'interfaccia video.

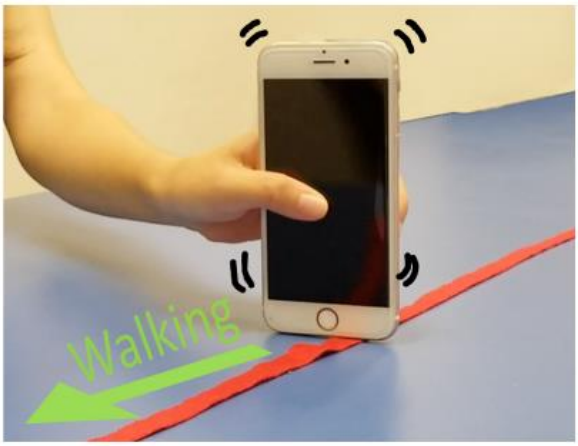

Figura 2.8: Rappresentazione del movimento del dispositivo per la creazione di animazioni con AR Animator

Per il progetto viene utilizzato un telefono cellulare abilitato all'AR come controller diretto per l'animazione di personaggi: il telefono viene mosso nello spazio e l'animazione viene creata in base ai movimenti che vengono eseguiti. Per realizzare un sistema di questo tipo sono stati eseguiti principalmente due studi: nel primo sono stati valutati i movimenti dei personaggi maggiormente richiesti dagli utenti per la creazione di animazioni mentre nel secondo sono stati mappati i movimenti del dispositivo sulle animazioni corrispondenti (rappresentate con Mixamo [37]).

ARAnimator consente agli utenti di utilizzare il telefono cellulare come controller a sei gradi di libertà: esso permette di rappresentare un personaggio virtuale animato sul luogo utilizzando gesti di movimento nell'ambiente reale, alcuni gesti sono rappresentati in Figura 2.9.

Per mappare e tracciare la posa 3D del telefono e mapparla sul personaggio virtuale è stata utilizzata la piattaforma Apple ARKit [38].

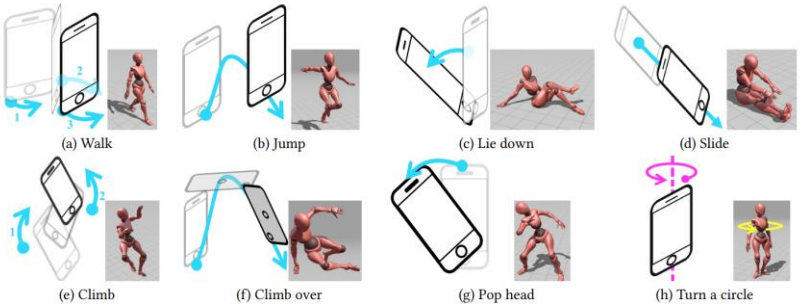

Figura 2.9: Esempio di mappatura dei gesti sulle animazioni

Il sistema analizza automaticamente i gesti e classifica il movimento di ogni segmento della traiettoria. I risultati dell'animazione vengono visualizzati sullo schermo del cellulare, consentendo agli utenti di visualizzare un'anteprima dell'animazione e di riconoscere eventuali gesti non corretti. Tramite la modalità di modifica è possibile modificare l'animazione tramite una timeline: essa permette il cambio di movimento, unione e divisione dell'animazione, estensione e riduzione, modifica della durata di un segmento selezionato e aggiunta o cancellazione. Inoltre, tramite questa interfaccia è possibile anche affinare la traiettoria, come si può vedere in Figura 2.10.

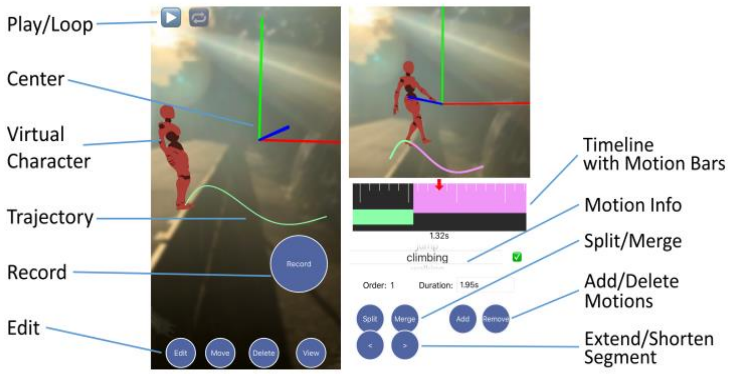

Figura 2.10: Interfaccia per l'editing dell'animazione di AR Animator

Per ogni tipo di movimento supportato il sistema riproduce direttamente le clip di animazione corrispondenti, utilizzando le API di rendering native di iOS. Il sistema inoltre supporta il controllo automatico dei parametri di movimento, consentendo agli utenti di variare le animazioni in base alle informazioni sul movimento, come ad esempio la velocità. Infine, grazie alla portabilità dell'AR mobile, gli utenti possono muovere nello spazio il dispositivo per vedere in anteprima le animazioni create da diversi punti di vista: questo permette di creare facilmente movimenti di camera simili a quelli cinematografici.

La feature principale del sistema consiste nell'utilizzare i gesti di movimento definiti dall'utente come metodo di input diretto. A tal fine si è utilizzato un approccio di apprendimento basato su SVM (Support Vector Machine) per riconoscere i gesti supportati.

ARAnimator risulta essere un sistema molto intuitivo e veloce da utilizzare, esso permette una rapida prototipazione per le animazioni che potrebbe essere utile per la creazione di uno storyboard ma il progetto presenta alcune limitazioni: per prima cosa lo strumento è applicabile solo ad ambienti di piccole dimensioni, perciò, sarebbe interessante lo studio all'interno di scenari su larga scala. In secondo luogo, il sistema dipende fortemente dalla qualità del tracciamento ARKit e gli utenti devono muovere un dispositivo mobile ed eseguire gesti di movimento contemporaneamente: ciò comporta un certo grado di distorsione delle traiettorie desiderate per un personaggio. Infine, si potrebbero apportare miglioramenti nella corrispondenza tra personaggi e ambienti, aggiungendo anche l'interazione tra oggetti virtuali e reali.

Nel progetto di cui si parla in questo elaborato le animazioni sono state gestite in modo molto più semplice rispetto al sistema AR Animator. È presente una libreria di azioni che un personaggio può compiere: quando un'azione viene scelta il personaggio esegue l'animazione corrispondente. Un personaggio inoltre può essere bloccato in una determinata posa, mettendo l'animazione in pausa, per la creazione delle catture per lo storyboard.

2.5 Previsualizzazione in realtà mista

2.5.1 MR-PreViz

Sebbene gli storyboard siano utilizzati tradizionalmente nell'industria cinematografica non è difficile immaginare quali possano essere le limitazioni per la creazione di una previsualizzazione fluida. Per questo motivo un altro strumento spesso utilizzato è l'animatic: esso è basato sull'utilizzo di immagini e video per la visualizzazione delle scene, delle angolazioni, dei movimenti di camera, dell'illuminazione e delle condizioni del set prima delle riprese.

Un progetto che si concentra sull'utilizzo di questa tecnica tramite l'utilizzo della realtà mista è MR-PreViz [39]. Lo strumento consente di utilizzare sfondi reali e umani e creature creati al computer all'interno di un set in studio o all'aperto.

Lo scopo del progetto è stato quello di sviluppare degli strumenti da applicare all'industria cinematografica, che prevedono l'utilizzo delle tecniche di realtà mista: l'obiettivo è quello di fondere mondi virtuali e reali per creare un

sistema che fornisca delle immagini previsualizzate in cui elementi costruiti in CGI siano sovrapposti alla scena reale nel set. Inoltre, MR-Previz si pone anche l'obiettivo di sostenere le azioni vere e proprie che si svolgono all'interno delle scene, animando e muovendo i personaggi nello spazio.

Il sistema MR-PreViz è costituito da tre strumenti:

- un editor di azioni utilizzato per creare le sequenza di azioni dei personaggi e fonderle con lo sfondo reale, esso è in grado di gestire dati di animazioni CG, dati acquisiti tramite *motion capture* e dati video 3D. Ogni azione può essere modificata sia in termini di posizionamento sia in termini temporali.
- uno strumento di layout dello spazio 3D per progettare gli spazi e creare il luogo della ripresa, disponendo gli oggetti tridimensionali. Si può trattare di uno spazio VR, utilizzato per la progettazione approssimativa del set o di uno spazio AR per la regolazione fine nel luogo scelto.
- uno strumento per la creazione di videocamere per esaminare i movimenti e le inquadrature che si vogliono realizzare. I dati delle camere sono descritti in Camera-Work Markup Language (CWML), un linguaggio che descrive il lavoro della fotocamera in linguaggio XML. I dati CWML registrati vengono utilizzati nelle riprese effettive interpretandoli con un software specifico.

Il processo per l'utilizzo di MR-Previz si può riassumere principalmente in quattro fasi. Nella prima si selezionano le scene che vogliono essere visualizzate attraverso lo strumento, ad esempio quelle più critiche per la ripresa. Nella seconda fase si creano le sequenze di azioni e si costruisce lo spazio di visualizzazione 3D: le azioni vengono successivamente poste nello spazio per confermare le relazioni spaziali tra i personaggi e l'ambiente. A questo punto inizia la terza fase: gli spazi virtuali vengono adattati al set reale e vengono creati i filmati MR-Previz tramite il sistema di telecamere, le informazioni sulle inquadrature e i movimenti vengono registrati sottoforma di dati e si possono testare vari modelli di lavoro, visualizzando i filmati ottenuti. Nell'ultima fase si effettuano le riprese vere e proprie utilizzando i dati ricavati dalla fase precedente. Una rappresentazione grafica delle fasi per la creazione del progetto si può vedere in Figura 2.11.

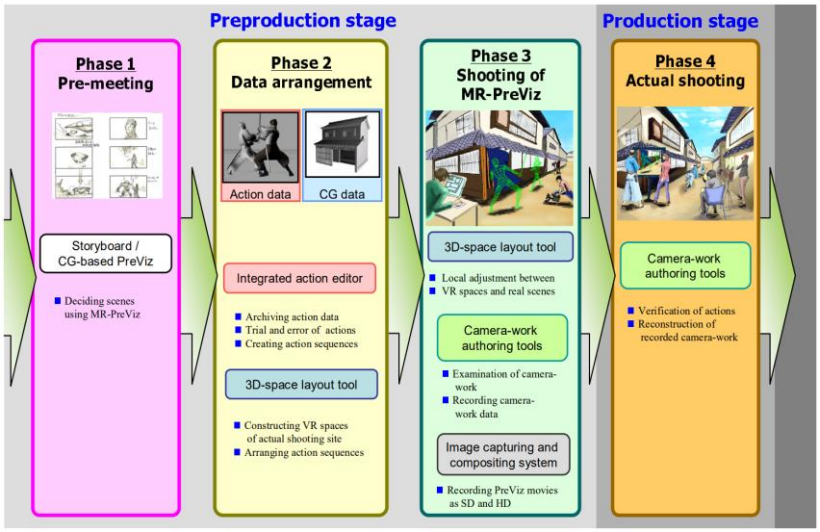

Figura 2.11: Fasi di creazione del progetto con MR Previz

Sebbene il sistema sia innovativo e presenti un'interessante applicazione delle tecniche di realtà mista all'industria cinematografica sono presenti alcuni problemi tecnici soprattutto legati ai sensori meccanici utilizzati per le camere.

Alcuni sviluppi possibili comprendono il miglioramento delle riprese con tecniche di sfocatura e illuminazione. Inoltre, il sistema potrebbe risultare molto utile, oltre che nell'industria cinematografica, anche nel mondo dello spettacolo.

Il sistema realizzato per questo progetto di tesi si differenzia da MR-Previz in quanto permette di realizzare l'ambiente e la scena direttamente nel mondo reale. MR-Previz infatti utilizza elementi reali, virtuali, combinazioni di azioni e camere per poi adattarle al set reale e visualizzarle attraverso lo strumento di previsualizzazione. Il sistema creato per questo progetto invece prevede di creare la scena, svolgere le azioni e aggiungere le camere direttamente all'interno del set, nel caso si utilizzi una scala 1:1, o su un tavolo, nel caso si utilizzi il paradigma tabletop. Infine, l'obiettivo di questo lavoro è quello di permettere la creazione di uno storyboard, mentre non consente di progettare il vero e proprio processo di ripresa.

2.5.2 Augmented Reality-based Pre-Visualization for Film Making

Il progetto *Augmented Reality-based Pre-Visualization for Film Making* [40] si pone l'obiettivo di creare un'applicazione per la previsualizzazione

cinematografica, integrando elementi virtuali e reali e considerando il punto di vista di attori e registi. Per il progetto è stato utilizzato il Microsoft HoloLens: esso permette agli utenti di interagire con gli oggetti digitali tramite degli ologrammi nel mondo circostante.

Nell'applicazione il regista può selezionare gli oggetti da una libreria in cui sono contenuti oggetti di scena, attori e posizionarli nel mondo reale. Il sistema proposto può essere utilizzato all'interno di uno studio o di una normale stanza, che fornisce una prospettiva senza precedenti ai registi e li aiuta a realizzare un film in modo più efficiente e completo.

Il sistema è costituito da 4 layer e si può vedere schematizzato in Figura 2.12:

- Livello data base: contiene le librerie di oggetti di scena, dei personaggi e delle azioni.
- Livello motore: gestisce i dati e implementa la logica. In questo livello sono stati integrati tutti gli elementi, si sono visualizzati i modelli degli oggetti e le applicazioni sono state distribuite sui dispositivi HoloLens. Il sistema è stato sviluppato utilizzando il framework Unity con script C#.
- Livello di interfaccia: Questo livello integra la scansione spaziale per una ricostruzione di alta qualità e precisione della stanza reale. In questa livello sono stati definiti sguardo e gesti specifici per interagire con gli oggetti virtuali.
- Livello utente: indossando l'HoloLens, gli utenti possono vedere gli oggetti virtuali e disporre la scena in base alle loro esigenze.

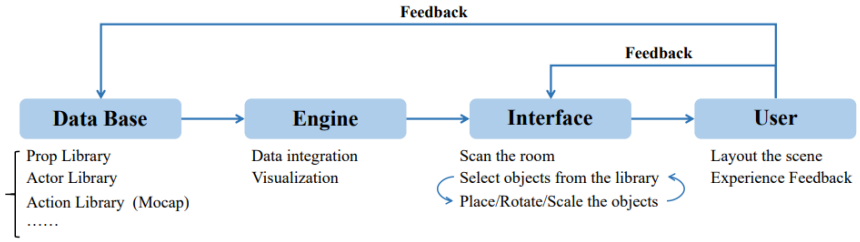

Figura 2.12: Rappresentazione dei 4 livelli di cui è costituito il sistema

Dopo la realizzazione il sistema è stato testato con 45 partecipanti tra studenti e registi che hanno valutato il sistema in termini di funzionalità, immersività e facilità d'uso.

Coerentemente con altri sistemi di previsualizzazione, l'intenzione del progetto è quella di simulare le scene di un film nelle prime fasi della sua

realizzazione: questo strumento consente di aggiungere elementi virtuali allo spazio realistico della recitazione in modo flessibile, i registi possono camminare nella stanza reale e interagire con oggetti virtuali e visibili senza alcuna restrizione. Di conseguenza, gli utenti hanno un senso di immersione più forte, che favorisce la realizzazione di film e gli attori potrebbero sperimentare le prove dalla prospettiva dei registi e avere una migliore comprensione della produzione in anticipo.

Poiché il dispositivo è ancora in fase di sviluppo e si tratta solo di una versione prototipale, alcune limitazioni come il campo visivo relativamente ristretto, il surriscaldamento e il peso un po' elevato possono far sì che gli utenti provino disagio durante i periodi di utilizzo prolungato. Inoltre, il primo passo è la scansione dell'ambiente in cui si trova l'utente, basato su pareti, suolo e soffitti, per cui è difficile usare questo sistema all'aperto.

In questo lavoro viene presentato un prototipo di sistema AR per la previsualizzazione. Un test preliminare mostra che l'applicazione proposta ha una grande usabilità e può migliorare significativamente l'efficienza della realizzazione di film sia per gli studenti registi che per i registi indipendenti.

Il progetto di cui tratta questo elaborato di tesi presenta molte analogie con il sistema Augmented Reality-based Pre-Visualization for Film Making. Sono però presenti alcune differenze: l'obiettivo di questo progetto è infatti creare uno strumento che permetta la realizzazione di storyboard e non la previsualizzazione del set e delle riprese. Il sistema infatti permette di realizzare le vignette che compongono storyboard attraverso l'utilizzo di camere virtuali, oltre che a delle descrizioni della scena. Inoltre, l'obiettivo è quello di fornire uno strumento rapido e intuitivo che permetta di rappresentare visivamente la sceneggiatura, senza però soffermarsi troppo su aspetti come l'immedesimazione degli attori all'interno della scena.

2.6 Idea di partenza: Generazione automatica di Storyboard

Questo applicativo prende ispirazione da un progetto di tesi realizzato precedentemente: "Advanced Storyboard: generazione automatica di storyboard mediante il controllo diretto dei personaggi" [41] di Ing. M. Scarzello. Questo lavoro di tesi ha avuto come obiettivo la creazione di un'applicazione desktop con Unity: l'applicativo permette di creare un ambiente 3D virtuale e di avviare una simulazione, nella quale i personaggi vengono mossi come all'interno di un videogioco e possono compiere della azioni, supportate dalle animazioni corrispondenti. Allo stesso tempo è

possibile muovere, sempre con l'ausilio della tastiera, una camera virtuale che permette di catturare delle immagini che, insieme a una descrizione dell'inquadratura, generata automaticamente, formano lo storyboard finale in formato HTML.

2.7 Requisiti dell'applicazione

Per sviluppare l'applicazione per la generazione di storyboard in realtà aumentata è stato necessario capire, per prima cosa, quali fossero le funzionalità fondamentali da implementare. Le caratteristiche dell'applicativo, infatti, devono soddisfare quelli che sono i principali bisogni degli utilizzatori che a loro volta definiscono quali sono i requisiti essenziali che l'applicazione deve avere per permettere il raggiungimento degli obiettivi.

2.7.1 Questionario

Per meglio comprendere quali fossero i requisiti necessari all'applicazione è stato creato un questionario Google Form [42] ed è stato sottoposto ad alcuni studenti frequentanti i corsi di Ingegneria del Cinema e dei Mezzi di Comunicazione e Ingegneria Informatica (Grafica e Multimedia) del Politecnico di Torino.

All'interno del questionario in primis sono state poste delle domande più generali per comprendere le conoscenze di base degli utenti sulla creazione di storyboard e sull'utilizzo della realtà aumentata. Di seguito è stata presentata una lista di requisiti riguardanti diversi aspetti dell'applicazione: in particolare il questionario è stato diviso in sette sezioni:

1. Requisiti per la creazione/modifica delle scene
2. Requisiti per la gestione dei personaggi
3. Requisiti per la gestione delle camere virtuali
4. Requisiti riguardanti le luci
5. Requisiti riguardanti la creazione dello storyboard o di una previsualizzazione
6. Requisiti riguardanti il sistema di interazione e visualizzazione dell'utente
7. Requisiti riguardanti la libertà dell'utente

Ogni sezione conteneva al suo interno una serie di funzionalità da valutare: si è chiesto ad ogni utente di valutare il singolo requisito utilizzando il metodo MoSCoW [43]. Il metodo MoSCoW è una tecnica di prioritizzatine che

permette ai destinatari dell'applicazione di esprimere l'importanza che essi attribuiscono al raggiungimento di ciascun requisito attraverso una scala di valori. La scala è composta da 4 categorie:

- Must have: requisiti indispensabili e decisivi per il corretto funzionamento dell'applicazione
- Should have: requisiti che sono considerati importanti ma non essenziali
- Could have: requisiti che potrebbero essere utili ma che hanno un impatto meno significativo
- Won't have: requisiti con minima importanza e minimo valore aggiunto. Essi possono essere considerati solo dopo aver sviluppato tutti gli altri.

2.7.2 Risultati

Il questionario è stato completato da 28 studenti e i requisiti sono stati analizzati e ordinati in base alle risposte ottenute.

I dati sono stati analizzati anche escludendo coloro che non avevano mai realizzato uno storyboard ma i risultati non si sono discostati da quelli ottenuti considerando tutte le risposte.

Priorità maggiore è stata data ai requisiti che avevano una più alta percentuale di must e should: in particolare sono state considerate le funzionalità che hanno ottenuto come somma di giudizi must e giudizi should un numero maggiore rispetto alla somma di giudizi could e won't have.

Di seguito una tabella riassuntiva dei risultati ottenuti.

Rendere possibile la visualizzazione dello storyboard una volta completato	MUST
Eliminare oggetti o personaggi da una scena	
Inserire oggetti e personaggi nella scena, scegliendoli da una lista, e definirne posizione e orientamento	
Creare degli screenshot della scena che rappresentano le vignette dello storyboard	
L'applicazione deve permettere l'aggiunta e il movimento di camere virtuali per inquadrare la scena da diversi punti nello spazio ed eseguire gli screenshot che costituiscono le vignette dello storyboard	
Possibilità di poter salvare una scena creata e poterla ricaricare in un secondo momento	

Permettere di visualizzare ciò che è inquadrato da una camera virtuale all'interno dell'applicazione con una preview in tempo reale	
Modificare lo storyboard finale riordinando le immagini e cambiando le descrizioni	
Possibilità di inserire nella scena delle luci virtuali	
Prevedere uno o più livelli di UNDO per tornare alla condizione dello "screenshot precedente"	
Per ogni camera deve essere possibile settare i parametri (fov, lunghezza focale, sensor type…)	SHOULD
Rendere possibile la visualizzazione dello storyboard prima del completamento	
Rinominare i personaggi e gli oggetti in scena	
Il sistema permette massima libertà all'utente di poter scegliere qualsiasi azione tra quelle disponibili da far eseguire ai personaggi, in qualsiasi momento a prescindere dal contesto	
Possibilità di animare un personaggio applicando al modello delle pose predefinite, selezionabili da una libreria	
Dare la possibile di definire la durata di un'azione o di un'inquadratura	
Modificare i parametri delle luci (potenza, area, colore)	
Possibilità per l'utente di aggiungere all'interno dell'inquadratura delle indicazioni che specifichino i movimenti di camera e dei personaggi (es. Frecce che indichino le direzioni del movimento, zoom in/out …)	
Possibilità di definire la durata di un'azione rappresentata da una vignetta dello storyboard	
Possibilità di interazione tra un personaggio e gli altri oggetti presenti nella scena	
Possibilità di definire le pose dei personaggi manualmente, modificando l'armatura del modello	
Possibilità di controllare i personaggi della scena muovendoli nello spazio e animandoli con dei comandi, come all'interno di un videogioco	
Possibilità di avere un preset di illuminazione per ambienti esterni	
L'utente deve avere la possibilità di visualizzare e modificare la scena in scale differenti	
Creare una pre-visualizzazione delle azioni svolte nell'applicazione attraverso un video	COULD
Possibilità di interazione tra personaggio e personaggio	

Il sistema, in base al contesto, permette di selezionare solo determinate azioni che sono coerenti ad esso	WON'T HAVE
L'applicazione deve supportare l'utilizzo di dispositivi fisici o controller aggiuntivi	
L'applicazione deve indicare all'utente la presenza di eventuali oggetti al di fuori del FOV del dispositivo di visualizzazione	
L'applicazione deve supportare un'interfaccia gestuale	
Il sistema deve verificare la coerenza temporale delle azioni tra vignette consecutive	
Avere una descrizione, generata automaticamente, che accompagna ogni inquadratura dello storyboard	
L'applicazione deve supportare un'interfaccia vocale	

Tabella 1: Tabella dei requisiti dell'applicazione classificati secondo lo schema MoSCoW

Capitolo 3

3. Tecnologie utilizzate

3.1 Unity

Unity [44] è un motore grafico multipiattaforma sviluppato da Unity Technologies e rappresenta la principale piattaforma per la creazione e la gestione di videogiochi ed esperienze interattive in tempo reale.

Unity permette di sviluppare progetti su un'ampia gamma di piattaforme del settore e agevola la collaborazione tra artisti, permettendo l'integrazione immediata con software come Maya [29] o Blender [30]. Questo favorisce iterazioni rapide e una sinergia continua tra costruzione del mondo, animazione, cinematica e rendering.

Inoltre, il motore è compatibile con i sistemi operativi principali quali Windows, Mac e Linux.

L'applicazione delle soluzioni di Unity si estende ben oltre il mondo dei giochi, trovando applicazioni nell'architettura, nell'ingegneria, nella cinematografia, nell'automobilismo e in molti altri settori.

Un elemento distintivo di Unity è la sua capacità di permettere la distribuzione di uno stesso prodotto su differenti piattaforme [45]: i contenuti creati possono essere distribuiti su una vasta gamma di dispositivi, tra cui AR, VR, dispositivi mobili, desktop e console. Questo permette di raggiungere una vasta base di giocatori, includendo le principali piattaforme per il gaming.

Lo scripting è un ingrediente essenziale di tutte le applicazioni realizzate in Unity [46]. Gli script possono essere utilizzati per creare effetti grafici, controllare il comportamento fisico degli oggetti e rispondere agli input del giocatore: gli strumenti che Unity fornisce permettono ai programmatori di realizzare dei prodotti su oltre 20 piattaforme e di testare ogni implementazione direttamente nell'editor. Il linguaggio principalmente utilizzato da Unity è C#.

3.1.1 L'interfaccia

L'interfaccia di Unity [47], rappresentata in Figura 3.1, si divide principalmente in 8 sezioni:

1. Toolbar: si trova nella parte alta dell'editor Unity e sono presenti alcuni controlli di base per la manipolazione degli oggetti.

2. Hierarchy: sono indicati in modo gerarchico tutti gli oggetti presenti nella scena selezionata. In questa sezione viene mostrata la struttura e i rapporti di parentela tra gli elementi presenti nella scena.

3. Game: simula l'aspetto del gioco finale dal punto di vista della telecamera in scena. Questa modalità viene visualizzata automaticamente una volta cliccato sul Play. È possibile, inoltre, settare alcuni parametri di visualizzazione come la grandezza della finestra.

4. Scene: consente di navigare all'interno della scena e modificare gli elementi contenuti al suo interno. La finestra può permettere viste prospettico o isometriche, 3D o 2D.

5. Inspector: consente di visualizzare e modificare i parametri di un oggetto selezionato. Poiché ogni oggetto ha una serie di diverse proprietà, il contenuto della finestra cambia dinamicamente in base all'elemento selezionato nella scena.

6. Project: consente di visualizzare tutte le cartelle e le librerie di cui il progetto è costituito. Quando vengono importate delle risorse all'interno del progetto appaiono in questa sezione.

7. Console: fornisce notifiche sui vari processi di Unity e sull'eventuale presenza di errori all'interno del progetto.

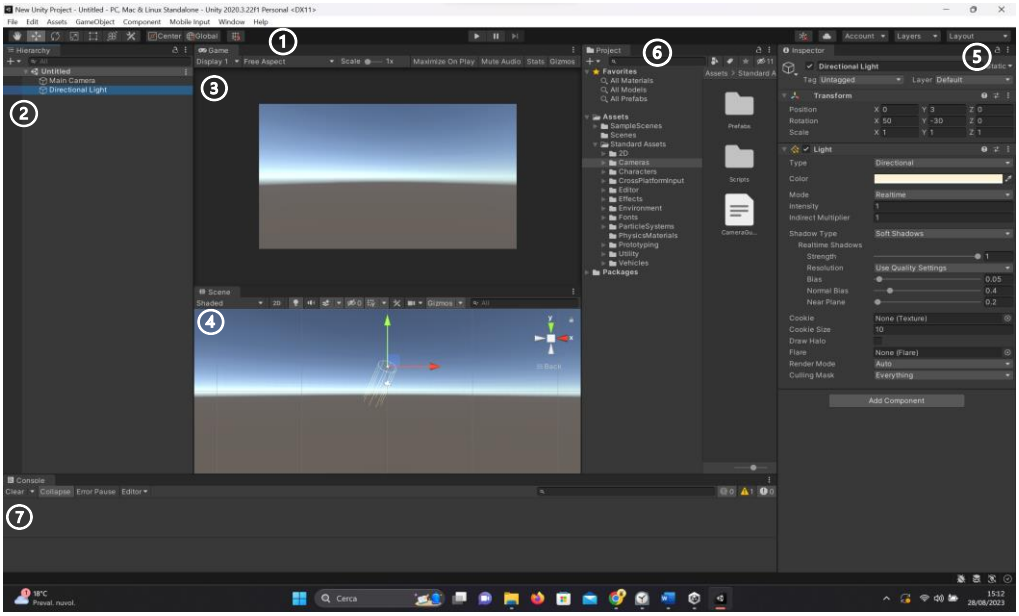

Figura 3.1: Interfaccia Unity. I numeri rappresentano le componenti sopra elencate

Per questo progetto è stata utilizzata la versione di Unity 2021.3.14f1 [22].

3.2 Visual Studio

Per scrivere e compilare il codice C# dell'applicazione realizzata è stato utilizzato Microsoft Visual Studio [48] .

Visual Studio è un ambiente di sviluppo integrato, IDE, sviluppato da Microsoft: esso permette di scrivere, eseguire e debuggare codice in 36 diversi linguaggi di programmazione.

Visual Studio include un editor di codice che supporta il completamento del codice oltre al refactoring dello stesso. Accetta inoltre dei plug-in che ne espandono la funzionalità come, ad esempio, l'aggiunta di nuovi set di strumenti per linguaggi specifici.

Unity si integra con Microsoft Visual Studio attraverso il pacchetto *Code Editor* per Visual Studio, questo pacchetto è preinstallato nel momento in cui viene installato Unity. È inoltre possibile scaricare Visual Studio insieme a Unity tramite Unity Hub: se ciò avviene Visual Studio diventa automaticamente l'editor di script predefinito all'interno del motore grafico.

L'integrazione di Visual Studio per Unity [49] include un set completo di funzionalità che migliorano scrittura e debug degli script C# all'interno dei progetti Unity. Per citare alcune funzionalità il sistema permette di scrivere rapidamente script Unity con il completamento del codice, permette di accedere rapidamente alla documentazione di Unity nelle descrizioni, supporta il miglioramento del codice con opzioni di refactoring specifiche per gli script Unity, consente di identificare la posizione in cui viene chiamato uno specifico frammento di codice all'interno dello script e permette l'aggiornamento automatico degli asset Unity una volta salvati i file.

All'interno dell'editor Unity è possibile creare degli script che, una volta cliccati, vengono automaticamente aperti in Microsoft Visual Studio: al momento della creazione allo script vengono aggiunte le librerie necessarie al suo funzionamento ed esso eredita le proprietà della classe *MonoBehaviuor* [50], appartenente alla libreria Unity Engine.

Allo script inoltre vengono aggiunte, in maniera automatica, due funzioni: la funzione `Start()` che viene chiamata nel momento in cui uno script viene abilitato, prima che venga chiamato qualsiasi metodo Update(), questa funzione viene chiamata esattamente una volta nella vita dello script. La seconda funzione è la funzione `Update()` che viene chiamata in ogni frame in cui il *MonoBehaviuor* è abilitato.

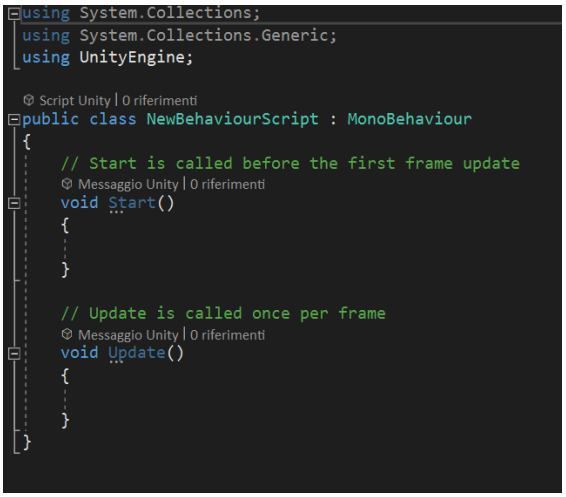

Figura 3.2: Metodi Start() e Update()

Per questo progetto è stato utilizzato Microsoft Visual Studio Community 2019, versione 16.11.6.

3.3 GitHub

Per il salvataggio delle modifiche del progetto e per tenere traccia dell'avanzamento del lavoro è stato utilizzato un repository GitHub. GitHub è un servizio che permette agli sviluppatori di memorizzare, gestire e tenere traccia di un progetto, in questo caso un progetto Unity, nel tempo.

In particolare, un repository GitHub permette di caricare un progetto su un cloud personale e di tenere traccia di tutti i file e delle modifiche effettuate. È interessante notare che tramite il *version control* il servizio permette di tenere traccia di tutta la cronologia del lavoro, permettendo allo sviluppatore di recuperare versioni precedenti ed elementi non più presenti nella copia locale del progetto. Inoltre, ogni qual volta si effettua una modifica è possibile salvarla sul proprio repository attraverso un'operazione chiamata *commit* tramite la quale le modifiche vengono salvate sul cloud insieme ad una descrizione.

Infine, GitHub permette anche la collaborazione tra più utenti che possono caricare le proprie modifiche su un unico progetto o accedere e scaricare il codice di altri utenti.

3.4 Microsoft HoloLens 2

Il dispositivo su cui si è scelto di realizzare l'applicazione è il Microsoft Hololens 2 [24], rappresentato in Figura 3.3. Questo dispositivo è un HDM senza fili per la realtà aumentata e la realtà mista. L'Hololens 2 è stato realizzato da Microsoft [51] e rilasciato nel 2019 e risulta essere il successore del Microsoft Hololens [52]. Rispetto al modello di prima generazione il dispositivo presenta diversi miglioramenti quali maggior ergonomia, miglioramento degli input (vocali e gestuali), maggiore campo visivo e potenziamento del processore.

Gli ambiti di applicazione dell'Hololens 2 sono molteplici, può essere utilizzato per la produzione e l'acquisizione di nuove competenze, in ambito medico e sanitario ma anche per quanto riguarda l'istruzione.

3.4.1 Specifiche hardware

Alcune specifiche hardware [24] che hanno reso interessante l'utilizzo del Microsoft Hololens 2 sono:

- Campo visivo esteso rispetto alla prima versione (42 x 28.5) con densità olografica di 47 pixel per grado di visione, in grado di garantire una maggiore immersività
- Tracciamento della mano completamente articolato per toccare, afferrare e manipolare gli ologrammi. Questo è reso possibile grazie all'utilizzo di un sensore di profondità abbinato all'integrazione dell'intelligenza artificiale.
- Funzionalità vocali grazie alle quali è possibile gestire il dispositivo anche con le mani occupate: è presente un gruppo di microfoni a 5 canali e degli autoparlanti che permettono l'audio spazializzato.
- Tracciamento oculare in tempo reale per adattare gli ologrammi in base alla direzione in cui si sta osservando. Sono presenti, infatti, quattro telecamere a luce visibile per il tracciamento della testa e 2 telecamere a raggi infrarossi per il tracciamento oculare.
- Mappatura spaziale che permette di mappare perfettamente l'ambiente fisico e ancorare i contenuti digitali al mondo reale.

- Diversi tipi di connettività tra cui Wi-Fi 5 802.11 ac, Bluetooth 5.0 e USB Type-C

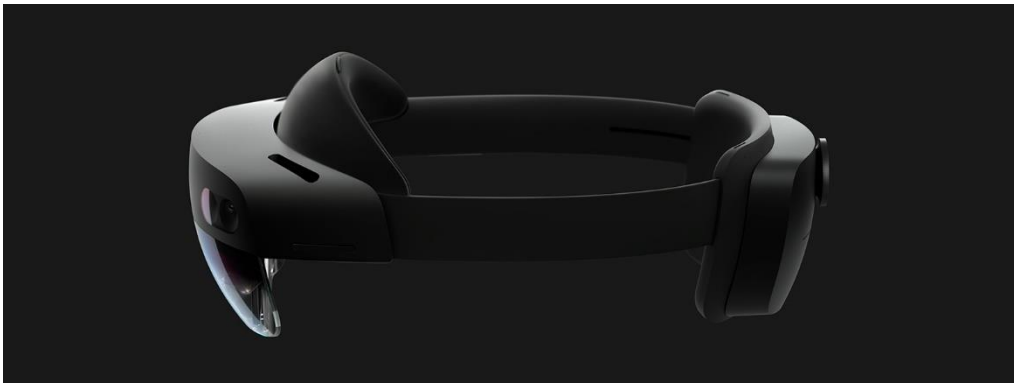

Figura 3.3: Microsoft Hololens 2

Infine, per lo sviluppo di applicazioni per Hololens 2 su Unity, Micorsoft mette a disposizione MRTK [53], un progetto che fornisce una serie di componenti e funzionalità per accelerare lo sviluppo di applicazioni multipiattaforma in realtà mista su Unity [54].

3.4.2 Modalità di interazione

Le interazioni con HoloLens 2 [55] sono molteplici e includono il tracciamento delle mani, il tracciamento degli occhi e il riconoscimento vocale. Inoltre, i modelli di interazione sviluppati da Microsoft funzionano su tutti i dispositivi di realtà mista: ciò permette lo sviluppo di applicazioni cross-device che utilizzano un'interfaccia naturale.

Per utilizzare un dispositivo Hololens 2 esistono principalmente tre modelli di interazione:

1. Interazioni con mani e motion controller [56]: questa interazione permette di interagire con gli oggetti virtuali con l'utilizzo delle mani o dei controller. Ci sono principalmente tre scenari possibili: la manipolazione diretta con le mani, la manipolazione attraverso un puntatore che dalla mano raggiunge elementi più lontani attraverso un raggio e la manipolazione con i controller di movimento, dei dispositivi hardware che forniscono delle scorciatoie per l'interazione.
2. Interazioni hands-free [57]: questa modalità di interazione è pensata per gli utenti che hanno bisogno di utilizzare le mani nell'interazione con gli oggetti reali e perciò le hanno occupate. Si hanno principalmente due modalità: l'interazione vocale, che usa la voce per controllare l'interfaccia, e l'interazione con il modello *gaze dwell* che permette di

interagire con l'interfaccia puntando lo sguardo e soffermandosi, utilizzata soprattutto per ambienti rumorosi.

3. Interazioni gaze and commit [58]: questo scenario è simile all'utilizzo del computer. Sono presenti due tipi di input: lo sguardo e il commit, che può essere eseguito con diversi comandi. Per lo sguardo può essere utilizzato il tracciamento della testa o dello sguardo mentre per l'operazione di commit si possono usare comandi manuali come l'air tap, rappresentato in Figura 3.4, un comando vocale o l'utilizzo di dispositivi hardware come clicker o controller.

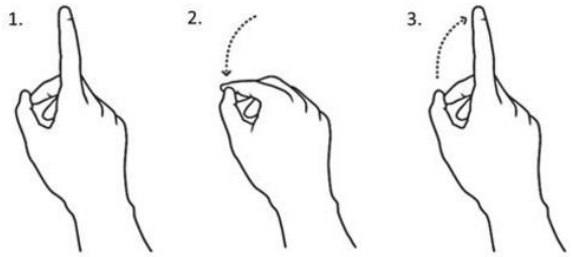

Figura 3.4: Rappresentazione del gesto air tap

3.5 MRTK

Per la realizzazione e lo sviluppo dell'applicazione Unity sul dispositivo Microsoft Hololens 2 [24] è stato necessario l'utilizzo della libreria MRTK [59]. MRTK è un progetto sviluppato da Microsoft composto da una serie di librerie open-source che forniscono componenti, risorse e linee guida per semplificare e accelerare lo sviluppo di applicazioni in realtà mista.

MRTK offre un insieme di elementi pronti all'utilizzo che permettono di semplificare la gestione di alcuni processi di interazione e grafica in ambienti di realtà mista. Questi componenti coprono una vasta gamma di funzionalità come la gestione degli input dell'utente, il tracciamento dei movimenti di testa e mani, la creazione di interfacce e di elementi interattivi.

Alcune caratteristiche chiave di MRTK, che ne hanno reso l'utilizzo essenziale per la realizzazione dell'applicazione, sono:

- MRTK è stato progettato per integrarsi perfettamente con Unity. Vengono forniti componenti, script e funzionalità che si possono integrare direttamente all'interno di un progetto. Inoltre, è possibile

testare e visualizzare direttamente le modifiche nell'editor Unity, tramite la simulazione.

- MRTK è progettato per essere compatibile e supportare un'ampia gamma di dispositivi diversi per realtà virtuale e realtà mista. Nel caso particolare è perfettamente compatibile con Hololens 2.
- Sebbene siano presenti degli elementi pronti all'uso l'approccio modulare permette agli sviluppatori di personalizzare i componenti in base alle proprie necessità, semplificando l'integrazione di funzionalità avanzate.
- La librerie offrono degli strumenti per la creazione immediata di interfacce utente immersive e semplificano la gestione degli input dell'utente.
- MRTK include delle funzionalità per il tracciamento e la comprensione dell'ambiente fisico circostante consentendo agli oggetti virtuali di integrarsi perfettamente con il mondo reale. Questo inoltre permette di semplificare lo sviluppo di applicazioni di realtà aumentata.
- Infine, il progetto MRTK è supportato da una vasta documentazione, tutorial e scene di esempio.

Per questo progetto è stata utilizzata la versione Mixed Reality Toolkit 2.8.3

3.5.1 Configurare un progetto in realtà mista su Unity con MRTK

Per sviluppare un progetto in realtà aumentata è necessario seguire una configurazione specifica per il progetto [60].

Il primo passaggio necessario alla creazione di un progetto in realtà mista è quello di configurare il progetto Unity con un modello 3D e successivamente modificare la piattaforma per l'esportazione su UWP, attraverso le impostazioni per la *build* dell'applicazione. In questo modo si potrà scegliere come destinazione dell'applicazione un qualsiasi dispositivo VR o AR.

Per sviluppare un'applicazione in realtà mista è fondamentale importare il pacchetto Mixed Reality Toolkit Foundation che contiene al suo interno i componenti necessari alla creazione di un'applicazione in realtà mista: una volta aggiunto il pacchetto al progetto Unity, infatti, vengono aggiunti alla scena due oggetti essenziali: *MixedRealityToolkit* e *MixedRealityPlayspace*. Il primo gameObject contiene il toolkit stesso e permette di configurare il

comportamento dei principali componenti di MRTK, il secondo invece consente la gestione della camera e degli SDK nello spazio di gioco.

Oltre al pacchetto Foundation, MRTK fornisce numerosi altri pacchetti che possono essere utili allo sviluppo di differenti feature all'interno dell'applicazione.

Per importare, aggiornare e gestire i pacchetti di MRTK viene utilizzato il Mixed Reality Feature Tool [61], un'interfaccia che permette agli utenti di scaricare e configurare le funzionalità di realtà mista che si vogliono aggiungere in un determinato progetto Unity.

A questo punto, dopo aver configurato le impostazioni di progetto in base alle necessità è possibile iniziare lo sviluppo dell'applicazione utilizzando script, classi ed elementi forniti da MRTK.

3.5.2 Windows Device Portal

Il Windows Device Portal (WDP) è un server web incluso con i dispositivi Windows che consente, tramite l'utilizzo di una connessione USB o tramite la connessione di rete, di configurare e gestire le impostazioni del device [62].

WDP fornisce degli strumenti di diagnostica e di risoluzione dei problemi, inoltre, permette di visualizzare le prestazioni e le configurazioni del dispositivo in tempo reale. Per utilizzare il portale è necessario abilitare la modalità da sviluppatore sul dispositivo utilizzato e per accedervi è necessaria una connessione di rete o USB.

Il WDP per questo progetto viene utilizzato per lo scambio di dati tra l'applicazione e il computer: una volta creato lo storyboard HTML esso viene salvato all'interno della memoria locale del Microsoft HoloLens 2. Tramite il portale, più precisamente connettendosi all'indirizzo IP del dispositivo Microsoft, è possibile accedere al *File Explorer* in cui sono contenuti tutti i file locali, l'interfaccia è rappresentata in Figura 3.5. Questo processo permette quindi di scaricare lo storyboard su altri dispositivi e accedervi in modo semplice e veloce.

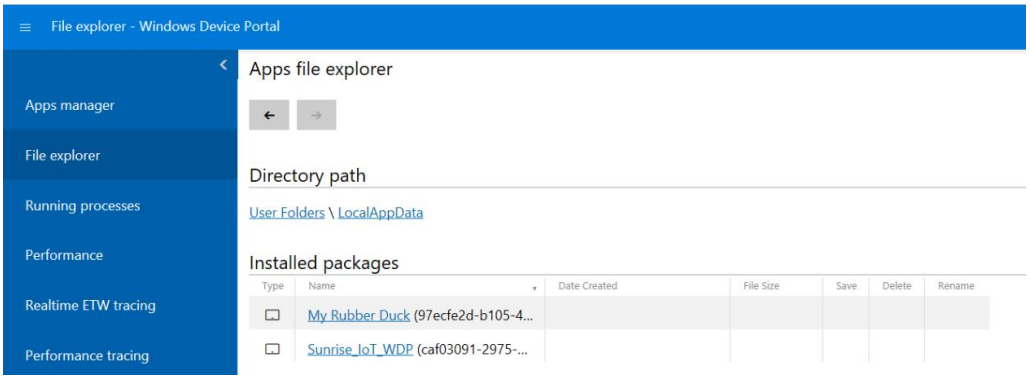

Figura 3.5: Cattura schermo che presenta il Windows Device Portal

3.5.3 Pacchetti Installati da MRTK

Per lo sviluppo dell'applicazione oltre al pacchetto obbligatorio Mixed Reality Toolkit Foundation sono stati importati nel progetto tramite l'utilizzo di Mixed Reality Feature Tool i seguenti pacchetti [53]:

- *Mixed Reality Toolkit Examples*: contiene al suo interno numerose scene di esempio, script ed elementi che aiutano allo sviluppo di applicazioni con MRTK. Gli esempi sono numerosi e divisi in sezioni in base all'argomento trattato. Questo pacchetto è molto utile per capire come utilizzare le funzionalità fornite da MRTK.
- *Mixed Reality OpenXR Plugin*: permette di semplificare il processo di sviluppo di applicazioni in realtà mista ed è particolarmente indicato quando come dispositivo si utilizza Hololens 2.
- *Mixed Reality Plane Finding:* questo pacchetto permette di riconoscere all'interno di un ambiente reale delle superfici piane. È stato particolarmente utile per lo sviluppo dello storyboard tabletop.
- *Mixed Reality Toolkit Tools:* contiene al suo interno degli strumenti che migliorano l'esperienza d'utilizzo del tool per la realtà mista. Nel progetto è stato molto utilizzato lo strumento Toolbox [63] che permette di inserire in maniera immediata dei componenti della UX prefabbricati.
- *Mixed Reality Toolkit Extensions*: contiene degli elementi per espandere le funzionalità del pacchetto MRTK.
- *Mixed Reality Toolkit Standard Assets:* è una raccolta di componenti consigliati per la creazione di applicazioni in realtà mista: contiene al suo interno elementi come shader, materiali, audio, fonts, texture e icone.

3.6 Componenti aggiuntivi

Oltre ai pacchetti MRTK per la gestione della realtà aumentata sono stati installati dei pacchetti aggiuntivi per gestire determinati aspetti dell'applicazione. I pacchetti sono stati principalmente scaricati tramite lo strumento *Package Manager* di Unity.

3.6.1 Pacchetti installati

Vengono sottoelencati i principali package utilizzati all'interno dell'applicazione:

- *Text Mesh Pro* [64]: è il componente di testo migliore per Unity. Utilizza delle tecniche avanzate di rendering del testo e una serie di shader che migliorano la qualità visiva delle UI. Inoltre, rende più semplice il controllo e la formattazione del testo: l'utilizzo di questo pacchetto risulta essere essenziale se si utilizzano asset e prefab MRTK perché permette la miglior lettura degli ologrammi
- *AI Navigation* [65]: il pacchetto fornisce un sistema di navigazione che permette ai personaggi di muoversi in modo intelligente nello spazio di gioco. In particolare, il pacchetto è stato utilizzato per la creazione dinamica delle *NavMesh* all'interno della scena: in base agli oggetti che vengono posti nell'ambiente viene calcolato il percorso migliore che può eseguire il personaggio evitando gli ostacoli.
- *Input System* [66]: il sistema consente di controllare gli input dell'utente all'interno dell'applicazione, in questo caso particolare si occupa di gestire i comandi gestuali.
- *NuGetForUnity*: è un sistema di gestione di pacchetti per installare, aggiornare e creare pacchetti che vengono distribuiti su un server e utilizzati dagli utenti.
- Sono infine stati scaricati dei pacchetti per lo sviluppo in realtà aumentata. Essi sono stati aggiunti automaticamente una volta scaricato il package *Mixed Reality Open XR Plugin*, tra questi troviamo XR Core Utilities, XR Legacy Input Helpers e XR Plugin Management.

3.6.2 Assets esterni

Alcuni oggetti 3D che sono presenti nella libreria per la creazione della scena sono stati scaricati gratuitamente da delle librerie online. In particolare, sono stati utilizzati modelli presi da Sketchfab [67], Turbosquid [68] e Free3D [69].

In aggiunta è stato utilizzato il servizio Mixamo [70] di Adobe [71]. Mixamo è una libreria che permette di accedere a una vasta gamma di animazioni, realizzate con la motion capture, e di personaggi che possiedono già un armatura. Grazie a questo tool è stato possibile scaricare gratuitamente i personaggi da aggiungere alla scena ed è stato possibile trovare tutte le animazioni corrispondenti alle azioni scelte per il sistema. L'interfaccia del servizio è presentata in Figura 3.6.

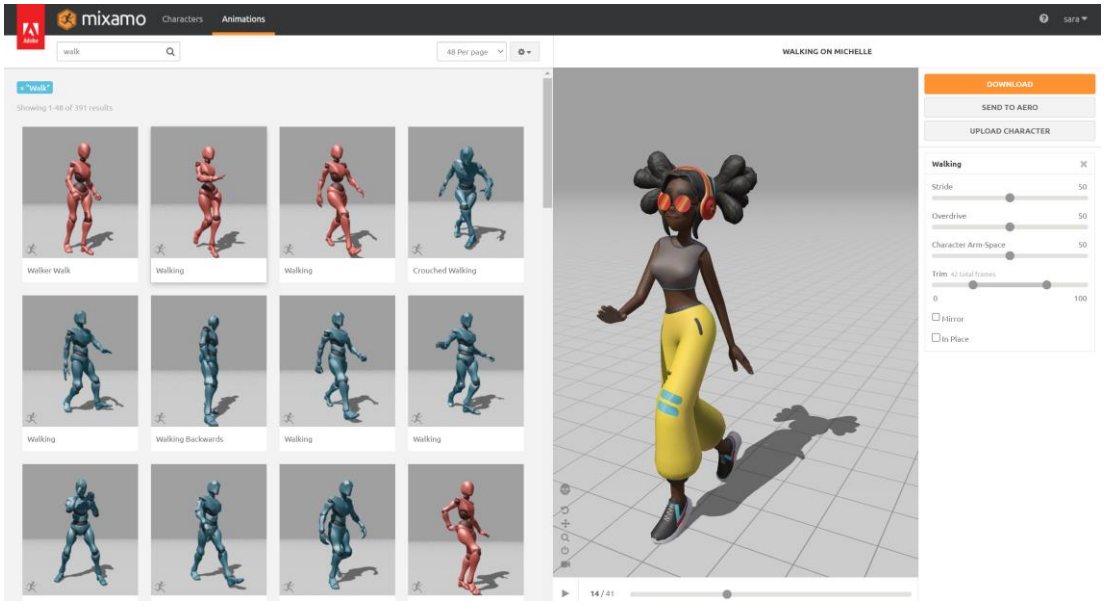

Figura 3.6: Interfaccia di Mixamo

Capitolo 4

4. Progettazione e realizzazione dell'applicazione

Per la realizzazione dell'applicazione in realtà aumentata è stato necessario impostare il progetto Unity con piattaforma di distribuzione "Universal Windows Platform". Inoltre, per semplificare lo sviluppo è stata installata e utilizzata la libreria MRTK, fornita da Microsoft per la creazione di applicazione in realtà mista.

4.1 Configurazione delle scene

Per configurare le scene, una volta aggiunta la libreria MRTK all'interno del progetto Unity, è necessario aggiungere due GameObject essenziali per il suo funzionamento: *MixedRealityToolkit* e *MixedRealityPlayspace*. Il primo contiene tutte le impostazioni e i profili di interazione mentre il secondo permette di gestire la camera, che riprende il mondo virtuale, all'interno della scena. Questi due oggetti si possono aggiungere automaticamente nella sezione *Mixed Reality > Toolkit > Add to Scene and Configure*.

L'aggiunta della libreria MRTK permette di previsualizzare l'esperienza in realtà aumentata anche nella modalità *Game* di Unity. È possibile spostarsi all'interno dello spazio con i tasti WASDEQ, visualizzare le mani virtuali rappresentate in Figura 4.1, che si possono muovere e ruotare, con SHIFT e SPACE, visualizzare la direzione di osservazione indicata da un cursore e la direzione del raggio delle mani.

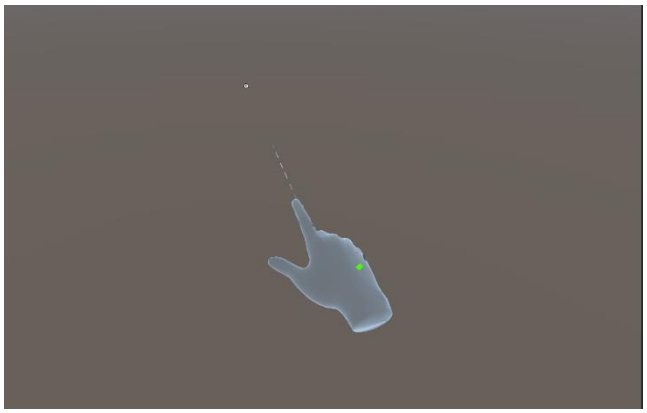

Figura 4.1: Mani virtuali all'interno dell'interfaccia Unity

4.2 Struttura

L'applicazione è composta da tre scene. La scena iniziale permette all'utente di creare un nuovo ambiente per creare lo storyboard, caricarne uno esistente o accedere al tutorial per l'utilizzo dell'applicazione.

La seconda scena contiene tutte le funzionalità dell'applicazione e a sua volta si divide in due sezioni. La prima permette all'utente di creare l'ambiente in cui si muovono i personaggi scegliendo da una lista gli oggetti: ogni oggetto può essere rinominato, eliminato, scalato, ruotato e spostato sul piano di lavoro. Una volta conclusa la fase di costruzione del set l'utente può decidere di passare alla fase di creazione dello storyboard: all'interno di questa sezione può muovere, animare e far interagire i personaggi, aggiungere e spostare le camere e creare le vignette dello storyboard. Queste due sezioni sono tra loro collegate: si può passare da una all'altra in qualsiasi momento tramite un apposito menù.

Infine, la terza scena contiene un tutorial: al suo interno vengono spiegati e fatti provare i principali metodi di interazione presenti all'interno dell'applicazione.

4.3 Tutorial

La scena Tutorial ha l'obiettivo di spiegare e far provare agli utenti i metodi di interazione. È diviso principalmente in 3 parti:

- Parte 1: Vengono spiegate le interazioni con gli oggetti virtuali. In particolare, vengono fatte provare all'utente le due modalità di manipolazione degli oggetti: l'interazione con il controller ray, un raggio che esce dalla mano dell'utente e permette di operare con oggetti lontani, e l'interazione con le mani dell'utente che permette una manipolazione vicina. Inoltre, in questa sezione viene spiegato l'utilizzo dei menù che sono presenti nella fase di creazione di una scena.
- Parte 2: In questa parte vengono presentate le principali feature che possono essere trovate nella fase di creazione dello storyboard: in particolare viene spiegato come interagire con i personaggi, come muoverli, animarli e come effettuare le catture che rappresentano le vignette dello storyboard.
- Parte 3: Questa sezione della scena tutorial spiega un altro paradigma di interazione: la modalità di tracciamento della testa e dell'interazione vocale. Viene proposto all'utente di riprodurre tutte le operazioni effettuate nelle prime due parti del tutorial utilizzando i comandi vocali e il puntatore che rappresenta la direzione di vista.

Il tutorial viene gestito attraverso lo script `Tutorial` che permette di gestire tutte le feature elencate.

4.4 Scena iniziale

La scena iniziale è composta da un menù all'interno del quale sono presenti tre bottoni che permettono all'utente di creare una nuova scena, caricarne una già esistente o passare al tutorial dell'applicazione. Il menù viene presentato in Figura 4.2. È possibile interagire con il menù sia con le mani sia con il Controller Ray. È presente un bottone *pin* in alto a destra che permette di abilitare e disabilitare lo script di MRTK `FollowMeToggle`: questo script permette al menù di seguire la camera principale e perciò l'utente che indossa il caschetto.

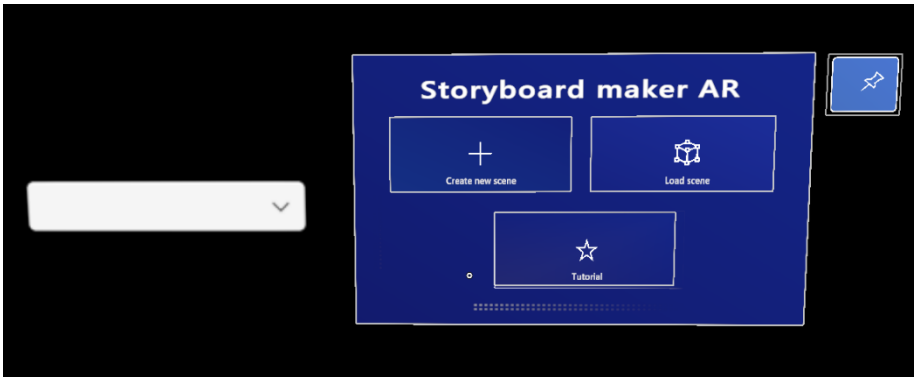

Figura 4.2: Menù presente nella scena iniziale.

Sono presenti, inoltre, all'interno della scena altri tre elementi: *LoadingManager*, *SaveManager* e *LoadingBar*.

Il *SaveManager* è un componente su cui è presente lo script *SaveLoadStage*. Lo script svolge diverse operazioni: in primis si occupa del salvataggio degli ambienti creati per la realizzazione dello storyboard; infatti, il componente *SaveManager* viene mantenuto tra le scene con il metodo `DontDestroyOnLoad()`. Per il salvataggio dei set creati lo script utilizza un file .csv nel quale vengono memorizzati il nome della scena, le caratteristiche del piano di lavoro utilizzato e le informazioni di posizione e orientamento di tutti gli oggetti presenti.

Sebbene l'operazione di salvataggio non avvenga nella scena iniziale il componente viene inizializzato in quest'ultima per permettere la creazione del menù a tendina: quest'ultimo serve a selezionare quale tra gli ambienti creati in precedenza si vuole caricare.

Infine, visto che questo è l'unico script che viene utilizzato in entrambe le scene, al suo interno sono presenti delle variabili che permettono di salvare la scelta dell'utente e perciò capire se bisogna caricare un ambiente già esistente e nel caso quale tra quelli salvati.

Il componente *LoadingManager* contiene invece lo script *StartingScene*: esso si occupa di caricare la scena successiva una volta che l'utente ha premuto i pulsanti nel menù. Lo script carica la scena in modo asincrono e permette la comparsa tra una scena e l'altra del componente *LoadingBar*, in Figura 4.3, che segnala l'avanzamento del caricamento della scena.

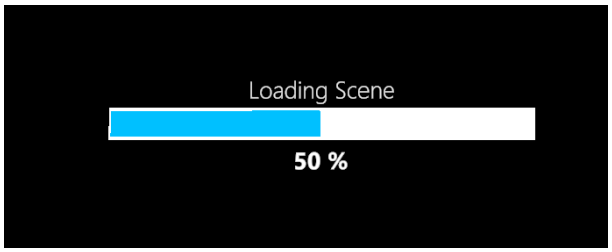

Figura 4.3: Barra di caricamento della scena

4.5 Seconda scena: Creazione dell'ambiente

All'interno della seconda scena Unity sono presenti le principali funzionalità dell'applicazione. In particolare, quest'ultima si divide in due sezioni: la prima, che sarà descritta in questo capitolo, permette di creare l'ambiente con personaggi e oggetti, esso servirà per la successiva creazione dello storyboard.

4.5.1 Spatial Awareness

Per la creazione dell'ambiente è necessario, visto che si lavora in un sistema tabletop, scegliere un piano di lavoro al di sopra del quale posizionare tutti gli elementi. Dato che l'applicazione è realizzata in realtà aumentata e il piano di lavoro è un piano nel mondo reale si utilizza una delle funzionalità di MRTK: la *spatial awareness* [72]. Questo strumento permette di creare una raccolta di mesh dell'ambiente circostante permettendo l'interazione tra mondo reale e mondo virtuale: in questo modo è possibile utilizzare una superficie del mondo reale come piano di creazione dell'ambiente.

È possibile attivare la spatial awareness tramite il componente *MixedRealityPlayspace*, inoltre esso permette di creare e definire gli *spatial observer*, dei componenti specifici che permettono di raccogliere e mostrare i dati delle mesh rispetto ad una posizione specifica.

Dato che è un processo dispendioso, in termini di prestazioni e memoria, la mappatura dello spazio non è utilizzata per tutta la fase di creazione dell'ambiente ma viene utilizzata per individuare i piani disponibili all'interno dello spazio a disposizione. Per compiere questa operazione è stato utilizzato un pacchetto sperimentale della libreria Microsoft, il *Mixed Reality Plane Finding*. Esso permette di individuare tra le mesh dello spazio delle superfici piatte e di generare al loro posto dei piani: in particolare è stato utilizzato il metodo `SurfaceMeshesToPlanes.MakePlanes()`.

4.5.2 Piano di lavoro

Per la selezione del piano è stato creato lo script *SurfaceCreation*: esso si occupa di aggiungere a tutti i piani individuati nello spazio un componente di tipo *Interactable* che, come suggerisce il nome, permette di interagire con un piano e di attivare una risposta al click dell'utente.

L'utente può selezionare una superficie per volta all'interno dello spazio e può confermare la scelta di un piano tramite un bottone: esso richiama il metodo `setChosenPlane()` che si occupa di configurare il piano per le fasi successive, di eliminare i piani scartati dalla scena e di disattivare la spatial awareness.

Come si può notare anche nella Figura 4.4, oltre al menù per la configurazione automatica del piano di lavoro, è possibile effettuare una configurazione manuale: essa permette di creare una superficie quando questa non viene identificata correttamente dal sistema (superfici occluse, trasparenti…).

Figura 4.4: Si può osservare la scena con il caricamento di una mesh d'esempio, rappresentata dalle linee bianche e i poligoni neri, i menù che permettono la configurazione del piano in modo manuale e automatico, i piani ottenuti tramite il metodo MakePlanes() in rosa e il piano scelto di colore marrone.

In questo caso un piano 3D viene istanziato e può essere spostato, scalato e ruotato all'interno dello spazio, per essere posto su una superficie reale. Il piano contiene diversi script che permettono di compiere queste operazioni, in particolare:

- *Tap To Place*: è un componente di interazione utilizzato per posizionare un oggetto su una superficie e in particolare su una mesh spaziale [73]. Esso permette di cliccare l'oggetto per selezionarlo e di cliccare un punto nello spazio per posizionarlo. Utilizzando questo componente l'oggetto si posiziona automaticamente su una superficie con un orientamento corretto, è possibile impostarne i parametri e i tipi di superfici abilitate.
- *Near Interaction Grabbable*: questo script permette di rendere afferrabile qualsiasi oggetto che possiede un collider [74].
- *Object Manipulator*: questo componente MRTK permette di scalare, ruotare o muovere un oggetto con una o entrambe le mani. È possibile configurare come un oggetto può essere manipolato in base a come esso viene afferrato dall'utente [75].
- *Solver Handler:* è un componente che facilita il calcolo della posizione e dell'orientamento di un oggetto secondo un algoritmo predefinito. Una delle sue proprietà, *Tracked Target Type*, permette di definire il punto di riferimento che il solutore utilizza per il calcolo dell'algoritmo: in questo caso si utilizza il Controller Ray che serve per posizionare il piano con la modalità tap to place [76].
- *Bounds Control*: lo script permette di visualizzare un riquadro intorno all'ologramma per indicare come è possibile interagire con esso: le maniglie agli angoli e ai bordi consentono di scalare, ruotare o traslare l'oggetto. Inoltre, le maniglie appaino quando l'utente è in prossimità, fornendo un feedback visivo che aiuta a percepire la distanza dall'oggetto [77]. Si può vedere una rappresentazione delle maniglie in Figura 4.5.

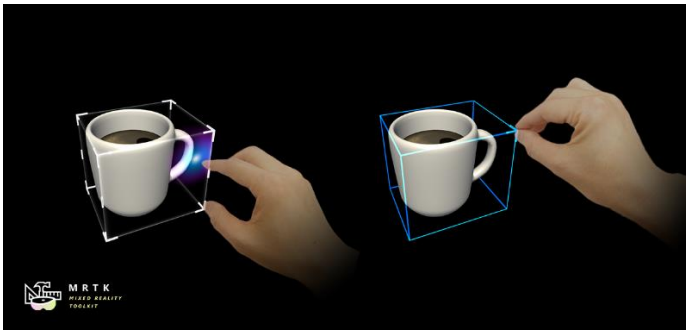

Figura 4.5: Manipolazione degli oggetti virtuali con le mani con bounds control

4.5.3 Posizionamento e manipolazione degli oggetti

Una volta definito il piano su cui si intende realizzare lo storyboard si possono iniziare ad aggiungere gli oggetti al set.

L'interfaccia utente per l'aggiunta degli ologrammi è composta da una collezione di oggetti: questo componente, fornito tra gli assets MRTK, permette di inserire al suo interno gli oggetti e di distribuirli su una griglia. Nel caso particolare di questa applicazione è stato utilizzato il componente *Scrolling Object Collection*, in Figura 4.6, che permette di esplorare la lista tramite lo scorrimento dei contenuti [78]. Sono inoltre presenti due pulsanti al di sotto del componente che permettono il movimento a destra e a sinistra per esplorare la collezione.

Figura 4.6: Collezione di oggetti

Per posizionare un ologramma sul piano è necessario cliccare uno degli elementi della collezione: quando questo accade viene chiamato il metodo `CreateObject()` della classe *ObjectManager*. Il metodo permette, una volta cliccato su un oggetto, di istanziarne un clone: a quest'ultimo vengono aggiunti tutti gli script per la manipolazione e per lo spostamento e in particolare *TapToPlace*, *NearInteractionGrabbable*, *ObjectManipulator*, *SolverHandler* e *BoundsControl*, descritti precedentemente. Inoltre, ad ogni oggetto sono aggiunti altri due script, *State* e *CharacterManager*, il cui utilizzo sarà spiegato nei capitoli successivi.

Una volta che l'oggetto viene istanziato si attiva automaticamente lo script *TapToPlace*: esso permette di cliccare su un punto del tavolo per posizionare

l'oggetto. In aggiunta, esso si riattiva ogni volta che si clicca sull'oggetto per cambiarne la posizione nello spazio. Per segnalare all'utente che questo componente è attivo è presente un menù all'interno della scena: quest'ultimo permette di passare alla modalità *Edit Mode* nella quale il *TapToPlace* è disattivato.

Per effettuare il passaggio da una modalità all'altra sono presenti i metodi `ActivateEditMode()` e `DeactivateEditMode()` nella classe *Object Manager*.

Nell'opzione *Edit Mode* è possibile manipolare gli oggetti con le proprie mani e con il controller ray: gli oggetti si possono afferrare, ruotare, scalare e traslare e anche in questo caso sono presenti delle maniglie per semplificare l'interazione. Inoltre, in questa modalità, è possibile rinominare o eliminare gli oggetti presenti sul tavolo: ogni volta che viene cliccato un oggetto il suo nome compare sul menù, quando viene schiacciato il tasto elimina viene chiamato il metodo `DeleteObject()`, che si occupa della distruzione dell'oggetto, mentre premendo il tasto rinomina viene chiamato il metodo `OpenKeyboard()`, che consente di visualizzare la tastiera MRTK. Infine, una volta premuto il tasto di conferma viene chiamato il metodo `Rename()` che permette di salvare le modifiche effettuate all'oggetto. Tutti questi metodi appartengono alla classe *ObjectManager*.

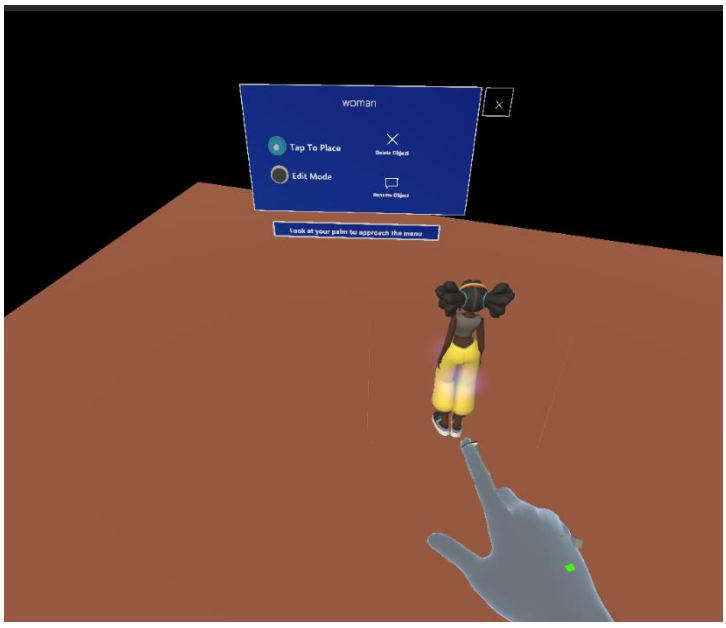

Figura 4.7: Menù che permette di passare da Tap To Place a Edit Mode, rinominare ed eliminare il personaggio.

Il menù che permette di modificare, rinominare ed eliminare gli oggetti è un particolare tipo di componente fornito dalla libreria Microsoft che prende il

nome di *Hand Menu*, in Figura 4.7. I menu manuali consentono agli utenti di aprire rapidamente un'interfaccia collegata alla mano e vengono utilizzati per operazioni frequenti. Per evitare false attivazioni durante l'interazione il componente richiede che la mano sia piatta con il palmo rivolto verso l'alto e che lo sguardo sia su di essa [79].

Il componente usato nell'applicativo appare vicino alla mano dell'utente quando sono presenti le condizioni appena descritte mentre si ferma nello spazio nel momento in cui la mano viene abbassata. Come tutti i menù presenti nel progetto può essere spostato con le mani e con il controller ray per essere posto in un punto specifico dell'ambiente.

4.5.4 Salvataggio e caricamento della scena

In questa sezione dell'applicazione, oltre al menù per la manipolazione degli oggetti, è presente un altro componente molto importante che permette, oltre al passaggio alla modalità di storyboarding, di rinominare e salvare gli ambienti creati e di ritornare al menù principale.

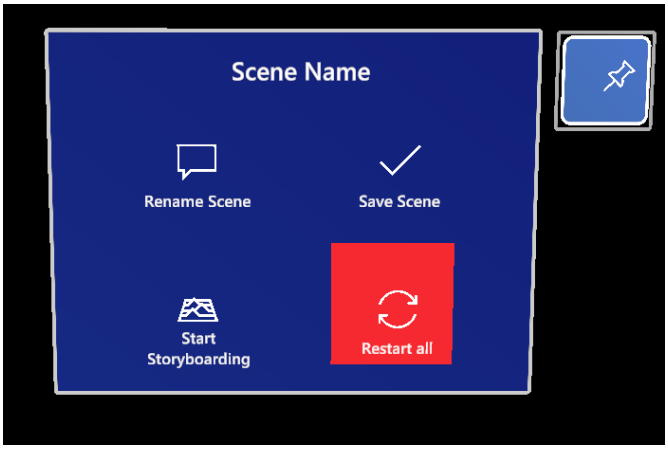

Figura 4.8: Menù per salvare e rinominare la scena, passare alla creazione dello storyboard e ritornare al menù iniziale.

Come si può osservare anche in Figura 4.8 il menù è composto da 4 bottoni al suo interno e un pulsante in alto a destra. Quest'ultimo si occupa di settare il comportamento del componente *FollowMeToggle*, uno script che permette al menù di seguire l'utente o di fermarsi in un punto dello spazio scelto.

Gli altri tre bottoni invece si occupano di eseguire delle funzioni molto importanti per l'applicazione. Il pulsante *Rename Scene* permette di attivare la tastiera MRTK e richiamare il metodo `renameScene()` della classe *ObjectManager*.

Il pulsante Save Scene chiama il metodo `SaveScene()` della classe *ObjectManager* che a sua volta richiama il metodo `SaveData()` della classe *SaveLoadStage*.

Il componente *SaveLoadStage*, di cui si è già parlato nella sezione Scena iniziale, si occupa di salvare nel file *saved_stages.csv* tutte le informazioni della scena corrente, in particolare nome, tipo, posizione, fattore di scala e orientamento di tutti gli oggetti presenti nella scena e fattore di scala del piano utilizzato.

Quando una scena deve essere ricaricata invece è responsabilità del componente *ObjectManager* di riprodurla: il metodo `LoadScene()` di questa classe legge il file *saved_stages.csv* e identificando l'attributo *type* di ogni oggetto è in grado di istanziarlo e rinominarlo per poi posizionarlo, ruotarlo e scalarlo correttamente. Inoltre, il componente si occupa di aggiungere ad ogni oggetto tutti gli script necessari per permettere la manipolazione e lo spostamento dello stesso, descritti nel capitolo Posizionamento e manipolazione degli oggetti.

Il tasto *Start Storyboarding* una volta cliccato permette il passaggio alla sezione che permette la realizzazione dello storyboard, di cui verrà parlato nel capitolo successivo.

Infine, il tasto *Restart all* è stato ideato per permettere agli utenti di tornare al menù principale, per ricominciare da capo con la creazione o il caricamento di una scena. Inoltre, il metodo può essere utilizzato anche nel caso in cui il piano scelto non sia corretto e si voglia ricominciare la configurazione dell'ambiente. Una volta premuto, il tasto richiama il metodo `Restart()` del componente *RestartScene*: il metodo utilizza il componente Unity *SceneManager* che permette tramite il metodo `LoadScene()` di ricaricare la scena iniziale.

4.6 Seconda scena: Realizzazione dello storyboard

Una volta premuto il tasto *Start Storyboarding* presente nella prima sezione della seconda scena del progetto si accede alla sezione che permette la realizzazione dello storyboard. Il tasto richiama il metodo `StartSimulation()` della classe *SimulationManager*: questo metodo si occupa per prima cosa di disattivare tutti i menù per la manipolazione, contenuti nella sezione precedente, e di attivare invece i menù necessari alla creazione delle vignette e all'animazione dei personaggi. Questa funzione inoltre richiama il metodo `StopManipulation()` della classe *ObjectManager:* esso si occupa di disattivare tutti gli script che permettono di muovere e manipolare gli oggetti e permette di aggiungere i componenti che permettono il movimento dei personaggi sul piano.

4.6.1 Movimento dei personaggi

Per far muovere un personaggio all'interno di uno spazio 3D solitamente si utilizza una *NavMesh*, o navigation mesh. Essa è una struttura di dati utilizzata nelle applicazioni di intelligenza artificiale per permettere ai personaggi, chiamati *NavMeshAgent*, di trovare il percorso migliore per raggiungere un punto nello spazio disponibile [80].

Per permettere la creazione della *NavMeshSurface,* e quindi il movimento dei personaggi, è stato necessario scaricare il pacchetto *AI Navigation*: questo componente permette di creare delle superfici di navigazione in modo dinamico e quindi di considerare nel percorso che deve compiere il personaggio anche gli ostacoli che vengono aggiunti durante la fase di creazione della scena.

Quando si passa alla modalità di storyboarding a tutti i personaggi presenti nella scena viene aggiunto il componente *NavMeshAgent* mentre al piano viene aggiunto il componente *NavMeshSurface* e viene chiamato il metodo `BuildNavMesh()` di questa classe.

Per muovere un personaggio l'utente deve eseguire tre azioni principali:

1. Selezionare con il raggio o con le mani l'agente che vuole far camminare.
2. Premere il tasto *Move* presente nel menù delle azioni. Il pulsante è visibile solo nel momento in cui viene selezionato un personaggio, identificato su Unity con il tag *player*. Inoltre, esso permette di settare una variabile booleana a *true* per indicare al sistema che il personaggio deve camminare.
3. Indicare e cliccare con il controller ray un punto del piano dove vuole che il personaggio si sposti. Per permettere questa azione al piano viene aggiunto in precedenza lo script *PointerHandler* che permette di gestire l'evento click e di richiamare il metodo `FindDestination()` della classe *ObjectsAnimator*.

Il metodo `FindDestination()` accetta come parametro l'evento click: questo permette di capire in che posizione si trova il punto in cui l'utente ha puntato il raggio e permette di chiamare il metodo `SetDestination()` della classe *NavMeshAgent*. Questo metodo permette automaticamente di muovere il personaggio attraverso il percorso migliore per arrivare alla sua destinazione. Inoltre, quando il personaggio si muove, viene attivata l'animazione corrispondente alla camminata.

4.6.2 Azioni dei personaggi

Nella modalità di storyboarding è possibile far compiere ai personaggi delle azioni: le azioni possono essere di diverso tipo. Per prima cosa si possono classificare in azioni che il personaggio compie su altri oggetti o personaggi, come l'azione "suona il pianoforte" o "saluta" e azioni che il personaggio compie su sé stesso come "salta" o "balla".

Per far compiere un'azione ad un personaggio si utilizza il menù in Figura 4.9. Questo menù è del tipo *Hand Menu* e viene mostrato in corrispondenza della mano ogni volta che l'utente rivolge il palmo verso l'alto e lo osserva.

Per svolgere un'azione per prima cosa bisogna selezionare il personaggio che deve compierla, una volta selezionato il suo nome appare nella scritta in alto sul menù. Se nessun personaggio è stato selezionato in precedenza automaticamente appaiono le azioni che il personaggio può compiere su sé stesso.

Figura 4.9: Menù per la gestione della animazioni posto all'interno della scena

Se ad essere cliccato è invece un oggetto, e non un personaggio, non appare nessuna azione ma solo il nome dell'oggetto attivo al momento.

Per gestire le azioni che i personaggi fanno su sé stessi e le azioni che compiono su altri personaggi o oggetti si utilizzano le classi *SimulationManager* e *ObjectsAnimator*. All'interno della prima classe è presente un parametro che indica il personaggio attivo *activeCharacter*, colui che può compiere le azioni. Nella seconda classe invece si tiene traccia dell'ultimo oggetto o personaggio che è stato cliccato, chiamato *character*: esso può essere l'oggetto o il personaggio su cui viene compiuta l'azione oppure può essere il personaggio che diventerà il prossimo personaggio attivo.

Ogni volta che un personaggio o un oggetto sono cliccati vengono chiamati rispettivamente due metodi della classe *SimulationManager:*

`setActiveCharacter()` e `setActiveObject()`. Entrambi i metodi si occupano di chiamare il metodo `setCharacter()` della classe *ObjectsAnimator*. In questo modo viene definito qual è l'ultimo gameObject selezionato, il nome appare nel menù di gestione delle animazioni.

Nel caso in cui l'oggetto che è stato cliccato sia un personaggio, quindi sia identificato con il tag "player", e non sia già presente un personaggio attivo l'oggetto diventa a sua volta l'*activeCharacter* e all'interno del menù vengono presentate le azioni che esso può compiere.

Nel caso in cui sia già presente un personaggio attivo le operazioni sono differenti:

- Se esiste un personaggio attivo e l'oggetto cliccato *character* non è un personaggio, all'interno del menù delle azioni che si possono compiere appariranno le azioni che il soggetto *activeCharacter* può compiere sul complemento oggetto *character*.
- Se l'oggetto *character* è un personaggio appariranno le azioni che il soggetto *activeCharacter* può compiere sul complemento oggetto *character*. Allo stesso tempo però sul menù delle azioni apparirà il tasto "Get Control": questo tasto permette di far diventare il *character* il personaggio attivo *activeCharacter*, cambiando il soggetto delle azioni.

Per far comparire le azioni si utilizzano due metodi della classe *ObjectsAnimator*: `ShowAloneActions()` e `ShowActions()`. Il primo metodo viene chiamato nel caso in cui il personaggio attivo è uguale all'ultimo oggetto cliccato, vengono quindi mostrate le azioni che il personaggio compie su sé stesso. Il secondo metodo invece mostra le azioni che il personaggio attivo può compiere su altri oggetti o personaggi. Entrambi i metodi vengono chiamati nel metodo `setCharacter()`.

4.6.3 Database di azioni

Per ottenere tutte le azioni che un personaggio può compiere su sé stesso o su altri oggetti si utilizza la classe *ActionsDataBase*. I metodi `ShowAloneActions()` e `ShowActions()` si occupano infatti di chiamare il metodo `ReturnActions()` della classe. Lo script si appoggia sul file actions.csv che contiene al suo interno le azioni che un personaggio può fare nel formato:

Soggetto-Complemento Oggetto, Azione1, Azione2 …

Nel caso in cui le azioni sono svolte del personaggio su sé stesso il complemento oggetto viene identificato dalla parola *self*.

Il metodo `ReturnActions()` si occupa quindi di leggere il file e restituisce un vettore di stringhe che rappresentano le azioni che il personaggio può

compiere. All'interno dei metodi `ShowAloneActions()` e `ShowActions()` il vettore di azioni viene utilizzato per istanziare i bottoni che, se cliccati, permettono al personaggio di compiere le azioni. Inoltre, viene sempre istanziato il pulsante *otherAction* che permette di aprire una casella di testo in cui inserire un azione che non è tra quelle presenti.

Ogni pulsante che viene creato per rappresentare un'azione ha al suo interno il componente *ButtonConfigHelper* che permette di impostare l'interazione `onClick()`. Quando viene cliccato il bottone richiama il metodo `ActionClick()` della classe *ObjectsAnimator*: esso ha il compito di gestire i personaggi e gli oggetti nel momento in cui viene selezionata un'azione da compiere.

4.6.4 Azioni transitorie

Prima di comprendere come viene gestita la singola azione bisogna precisare che si possono classificare le azioni in due categorie: sono presenti, infatti, le azioni *transitorie* che hanno una breve durata e le azioni che invece *cambiano lo stato* del personaggio e che devono essere stoppate dall'utente.

Quando viene scelta un azione transitoria per prima cosa viene eseguita l'animazione corrispondente tramite il metodo `Play()` della classe *Animator* di Unity. L'Animator [81] è un componente che permette di rappresentare le animazioni che un personaggio può fare attraverso una macchina a stati, rappresentata in Figura 4.10. Quando viene chiamato il metodo `play()`, che vuole come parametro il nome dell'azione, il componente entra nel blocco corrispondente e fa in modo che venga riprodotta l'azione. Poi procede attraverso le frecce fino a ritornare nel blocco *idle*.

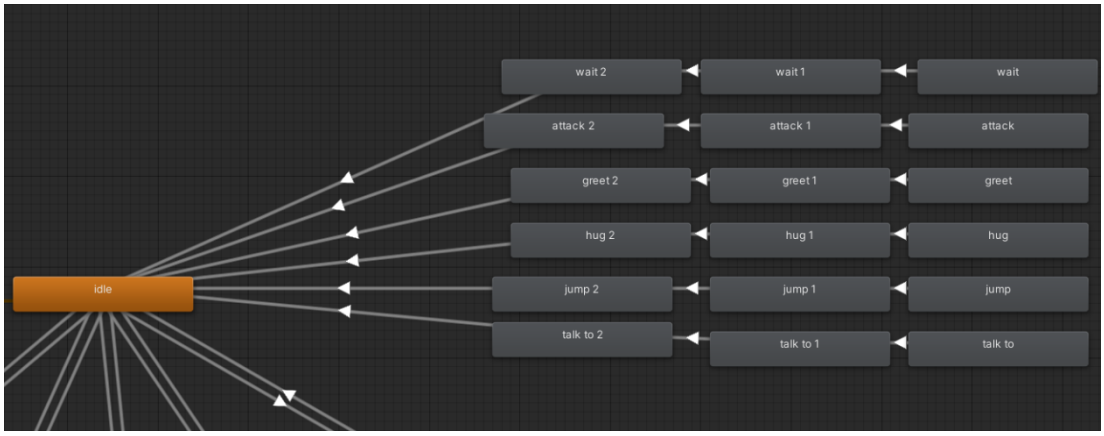

Figura 4.10: Macchina stati delle azioni di un personaggio

Le azioni transitorie, che sono rappresentate nel diagramma sulla destra, vengono ripetute in loop tre volte: questo per permettere all'utente di effettuare una cattura dell'azione nella posa desiderata.

Ogni volta che viene compiuta un'azione da un personaggio viene chiamato il metodo `GenerateSimplePhrase()` che si occupa di generare una descrizione dell'azione che si è svolta. Della descrizione testuale si parlerà nel capitolo Descrizione testuale della scena.

4.6.5 Cambiamenti di stato

Come accennato in precedenza, esistono delle azioni più complesse che comportano il cambiamento dello stato di un personaggio.

Per prima cosa bisogna specificare che ogni oggetto all'interno della scena viene rappresentato come una macchina a stati finiti [82] (o finite state machine FSM). Il personaggio si trova in uno stato iniziale, che ha determinate caratteristiche. Se il personaggio compie un azione specifica, che lo porta al cambiamento di stato, si effettua una transizione: nel nuovo stato il personaggio avrà nuove caratteristiche e attributi e a sua volta potrà raggiungere nuovi stati, in base a specifiche condizioni. Un esempio si può vedere in Figura 4.11.

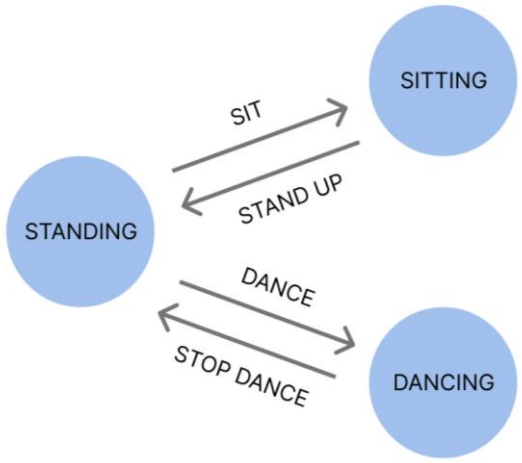

Figura 4.11: Esempio di macchina a stati di un personaggio

Per gestire lo stato ogni personaggio ha come componente lo script *State*. Quando un oggetto viene creato gli viene assegnato automaticamente uno stato iniziale. Quando viene compiuta un'azione sul personaggio che provoca una transizione di stato viene chiamato il metodo `ChangeState()` della classe *State*.

La classe State si appoggia sul componente *ActionDataBase,* che allo stesso tempo si appoggia sul file *states.txt*. Il file contiene al suo interno i dati

riguardanti ogni personaggio in formato JSON. In particolare, per ogni personaggio, identificato dal nome, sono presenti una lista chiamata *transitions,* che riporta le transizioni possibili da uno stato ad un altro, ed una lista chiamata *S_A* in cui sono presenti, per ogni stato, le azioni possibili. Un esempio si può vedere in Figura 4.12.

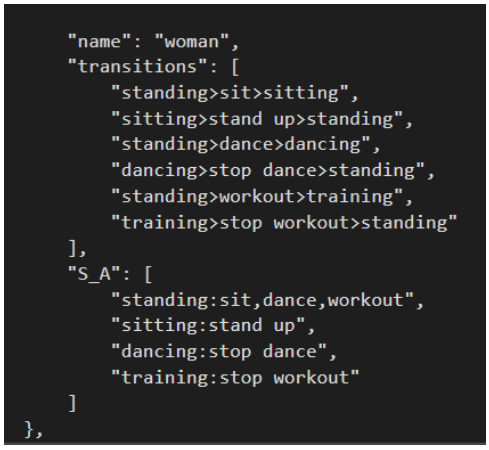

Figura 4.12: Rappresentazione dei dati nel file actions.txt

In particolare, nella Figura 4.12 si può vedere che la lista *transitions* contiene le transizioni nel formato *stato1>azione>stato2*. Il personaggio *woman* si può trovare in differenti stati come *standing*, *sitting*, *dancing* o *training*. Come esempio si prenda la prima riga: se il personaggio si trova nello stato *standing* e l'azione scelta è *sit*, lo stato successivo in cui si troverà il personaggio sarà *sitting*. Inoltre, nella lista *S_A* sono rappresentate le azioni che possono essere compiute quando il personaggio si trova in un determinato stato: se il personaggio è *standing* può compiere le azioni *sit, dance* e *workout* mentre non può compiere l'azione stand up, che è possibile svolgere solo quando si trova nello stato *sitting*.

Questo file permette, per prima cosa, di selezionare e mostrare all'utente solo le azioni che è possibile svolgere quando il personaggio si trova in un determinato stato. Inoltre, quando un'azione viene scelta dall'utente, permette di cambiare lo stato del personaggio in modo tale da aggiornare le successive azioni che potrà compiere.

Per ricapitolare, ogni volta che viene cliccata un'azione che il personaggio può eseguire viene chiamato il metodo `ActionClick()` della classe *ObjectsAnimator*, in questo metodo a sua volta viene chiamato il metodo `ChangeState()` della classe *State*, che gestisce lo stato del personaggio. A sua volta il metodo `ChangeState()`, per capire che transizione effettuare e in che stato portare il personaggio, si appoggia al metodo `GetStateTransition()` della classe *ActionsDataBase*.

Per generare le animazioni legate al cambiamento di stato si utilizza sempre il componente *Animator* di Unity: viene chiamato il metodo `ChangeStateAnimation()` della classe *State,* che permette di settare delle variabili booleane del componente Animator. Queste variabili permettono il passaggio da un animazione all'altra e, differentemente dal caso precedente, le animazioni vengono riprodotte ripetutamente, fino al momento in cui lo stato non varia di nuovo e la variabile booleana viene posta a *false*. Il componente *Animator Controller* con le animazioni che portano al cambiamento di stato è rappresentato in Figura 4.13.

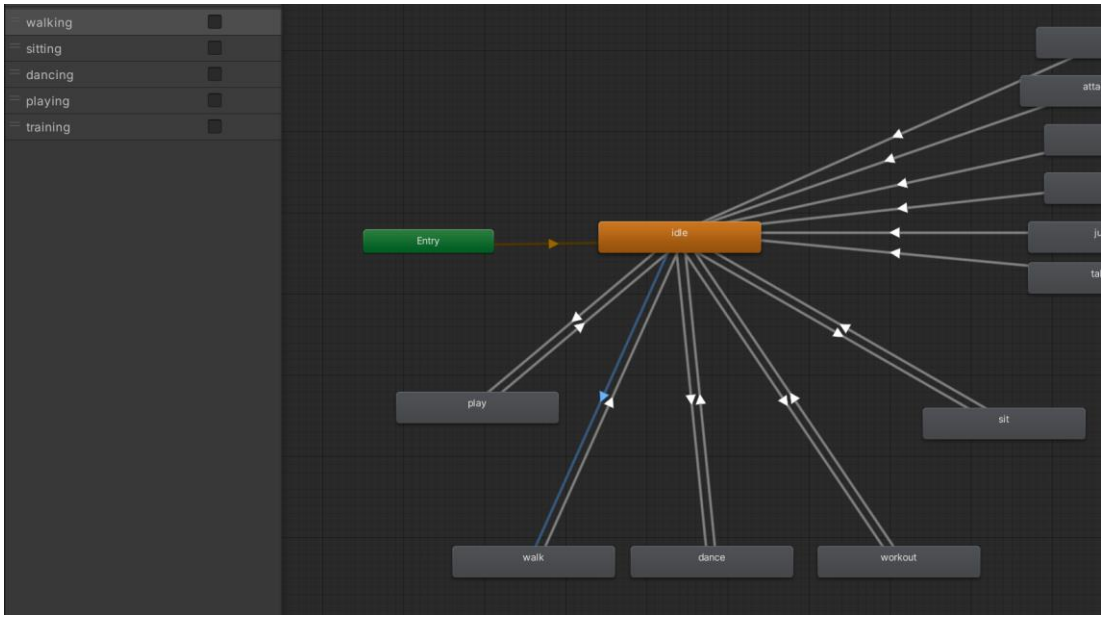

Figura 4.13: In figura viene mostrato il componente Animator Controller e in particolare le azioni che portano al cambiamento di stato. Ogni transizione è provocata dal cambiamento delle variabili booleane presentate nell'immagine in alto a sinistra.

4.6.6 Start and Stop

È molto importante per la realizzazione dello storyboard generare delle vignette che rappresentino le azioni svolte. Ogni vignetta può essere generata con una camera virtuale, come verrà spiegato nel capitolo 4.6.8.

Lo scopo della cattura è quello di immortalare il personaggio in una determinata posa che ne rappresenti l'azione. Risulta quindi necessario un sistema che permette di bloccare il personaggio in una determinata posa, soprattutto quando le azioni sono transitorie.

Per questo motivo nel sistema è presente il tasto *Stop and Play*, posto nel menù per la gestione delle animazioni, mostrato nella Figura 4.9. È stato scelto di posizionare il tasto all'interno del menù in modo tale che fosse sempre vicino

alle mani dell'utente e che fosse quindi semplice stoppare l'animazione del personaggio in qualsiasi momento.

Quando il tasto viene premuto viene chiamato il metodo `startStop()` della classe *ObjectsAnimator*: all'interno del metodo viene utilizzato il componente *Animator* del personaggio attivo e viene settato il parametro *speed* a zero. Viceversa, quando il pulsante viene premuto nuovamente, il valore del parametro che esprime la velocità dell'animazione viene portato al valore che aveva precedentemente. Inoltre, se il personaggio è in movimento, quindi è nello stato *walking*, è necessario chiamare i metodi `Start()` e `Stop()` della classe *NavMeshAgent* rispettivamente per cominciare e stoppare, oltre all'animazione, anche il movimento di camminata del personaggio.

Questi metodi permettono perciò di bloccare ogni personaggio nella posa desiderata ed effettuare lo screenshot. È stato inoltre implementato il comando vocale "Stop" per permettere lo stesso comportamento con un input vocale, dei comandi vocali si parlerà nel capitolo 4.7.

È infine presente, sempre nel menù per la gestione delle animazioni, il tasto `Play All()`. Esso permette di sbloccare tutti i personaggi nello stesso momento, in modo che concludano le azioni che hanno iniziato. Esso utilizza, come il metodo precedente, i componenti *Animator* e *NavMeshAgent*.

4.6.7 Menù della simulazione

Per gestire la fase di simulazione, oltre al menù di gestione delle animazioni di cui si è parlato in precedenza, è presente un altro componente, rappresentato in figura Figura 4.14.

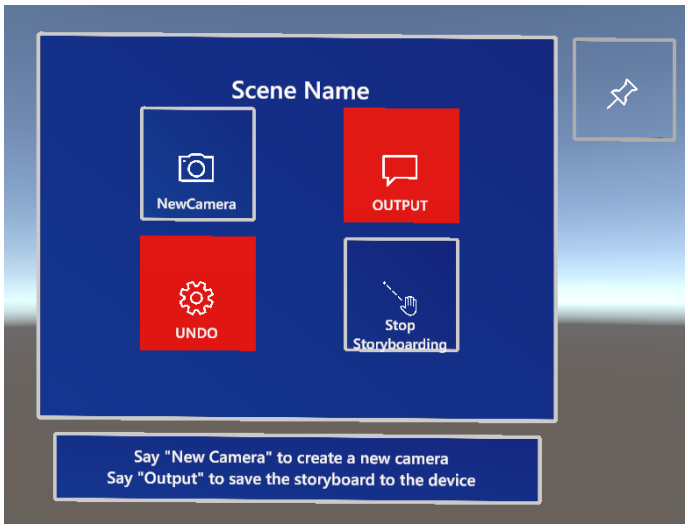

Figura 4.14: Menù della simulazione

Il menù contiene al suo interno quattro pulsanti più un bottone in alto a sinistra che permette di fissare il menù in un punto dello spazio o fare in modo che esso segua l'utente. I bottoni presenti all'interno del menù permettono l'esecuzione di task molto importanti per l'applicazione.

Il tasto *NewCamera* permette la creazione di una nuova camera virtuale all'interno della scena, di cui si parlerà nel capitolo 4.6.8. Il tasto *UNDO* permette di resettare la posizione dei personaggi e delle camere all'ultimo screenshot effettuato, di ciò si parlera nel capitolo 4.6.10. Il bottone *OUTPUT* permette di generare lo storyboard HTML, una volta effettuate tutte le vignette necessarie, e di salvarlo in locale sul dispositivo Microsoft HoloLens 2. Di queste operazioni si parlerà nel capitolo 4.6.12

Infine, il pulsante *Stop Storyboarding* ha il compito di interrompere la fase di storyboarding e di tornare alla fase di creazione della scena. Quando viene premuto questo tasto viene chiamato il metodo `StopSimulation()` della classe *SimulationManager*, al suo interno vengono svolte molteplici operazioni

- Per prima casa vengono riattivati tutti i menù necessari alla creazione della scena e allo stesso vengono disattivati tutti i menù della simulazione.
- Il valore intero *status* che indica se il sistema si trova nella fase di simulazione o di creazione viene posto a zero (che indica la fase di creazione)
- Viene chiamato il metodo `RestartManipulation()` della classe *ObjectManager*. Il metodo si occupa di riattivare, per ogni oggetto presente nella scena, tutti i componenti necessari alla manipolazione. Inoltre, si occupa di disattivare il componente *NavMeshAgent* per i personaggi, in modo tale che sia possibile spostarli sul piano.

4.6.8 Camera virtuale

Per creare lo storyboard risulta essenziale, una volta messi i personaggi in posa, catturare un'immagine che rappresenterà una vignetta dello storyboard. Per fare ciò all'interno dell'applicazione si utilizzano una o più camere virtuali. Nel caso più semplice, quando si accede alla modalità di storyboarding, viene automaticamente aggiunta alla scena una camera virtuale. L'oggetto *SecondCamera* è un oggetto complesso: esso è infatti l'unione di più gameobject e inoltre possiede al suo interno diversi script. L'oggetto è composto da:

- L'oggetto 3D che rappresenta la camera, rappresentato in Figura 4.15
- Piano: il piano presente all'interno della camera possiede una texture particolare chiamata *Display Render Texture* che permette di

rappresentare al suo interno ciò che viene renderizzato dalla camera virtuale.

- Slider: Questo componente è uno slider fornito dalla libreria MRTK e permette all'utente di cambiare il suo valore con l'utilizzo delle mani. All'interno dell'oggetto è presente lo script `PinchSlider` che permette di chiamare delle funzioni quando si interagisce con l'oggetto o quando il valore dello slider viene cambiato. Quando il valore dello slider varia viene chiamato il metodo `OnSliderUpdated()` della classe `ShowFocalLengthValue` che verrà spiegato nel paragrafo successivo.
- Bottone Screenshot: il bottone si occupa di permettere all'utente di effettuare lo screenshot e può essere cliccato con le mani o con il raggio. Al suo interno è presente lo script `Interactable` che permette di chiamare il metodo `CamCapture()` della classe `SecondCameraManager`, di cui verrà spiegato l'utilizzo in seguito. Il bottone della camera è identificato con il tag "*cameraButton*".

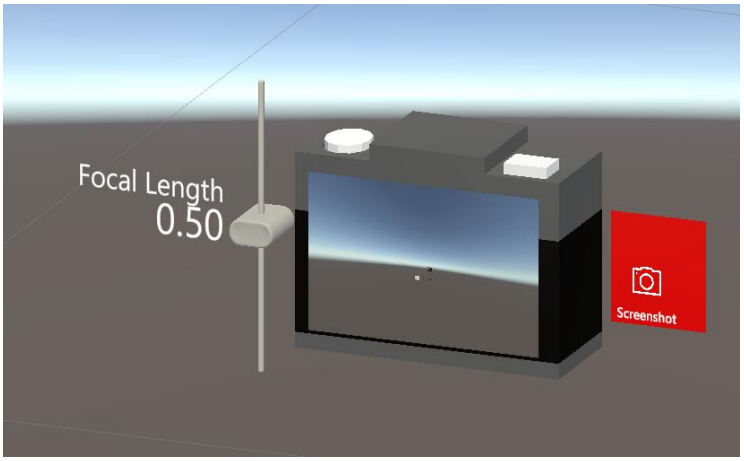

Figura 4.15: Camera Virtuale

Gli script che sono legati all'oggetto SecondCamera sono invece:

- *NearInteractionGrabbable* e *ObjectManipulator*: questi script sono già stati spiegati in precedenza nel capitolo 4.5.2. Essi permettono la manipolazione della camera che può essere spostata, ruotata e scalata con le mani e con il raggio.
- Componente *Camera* [83]: questo componente Unity permette di renderizzare il mondo virtuale da un determinato punto di vista, scelto dall'utente. Del componente si possono settare parametri come lunghezza focale o tipo di proiezione. Quello che viene renderizzato da questo oggetto è la texture visibile all'interno del piano dell'oggetto camera.

- *ShowFocalLengthValue*: Per effettuare una cattura schermo, di ogni camera si vuole settare la lunghezza focale. La lunghezza focale viene scelta tramite l'utilizzo dello slider descritto in precedenza. Per fare in modo che i cambiamenti che avvengono all'interno dello slider si ripercuotano sull'effettiva lunghezza focale della camera viene utilizzato questo script che permette di mappare i valori dello slider sul parametro *focal lenght* del componente *Camera*.
- *SecondCameraManager*: questo componente si occupa di tutta la gestione della camera virtuale e comunica con il componente *CameraManager* che si occupa della gestione di tutte le camere presenti nella scena.

4.6.9 Aggiunta e gestione di camere virtuali

Una volta capito da quali elementi è composta una singola camera si può procedere a capire il funzionamento delle camere all'interno della scena. Il componente che si occupa della gestione prende il nome di *CameraManager*.

Uno dei metodi più importanti è il metodo `NewCamera()`: questo metodo permette di creare una nuova camera davanti allo sguardo dell'utente. Esso, infatti, prende come riferimento la posizione e la direzione di vista dell'utente nel momento in cui il metodo viene chiamato e istanzia un oggetto *SecondCamera*, clone di quello descritto nel capitolo 4.6.8. Inoltre, durante la creazione, il metodo si occupa di assegnare al bottone della camera creata i componenti necessari all'interazione.

Il secondo metodo presente all'interno della classe *CameraManager,* che risulta essere essenziale per l'applicazione, è il metodo `CamCapture()`. Il metodo viene chiamato quando, da una qualsiasi camera, viene premuto il bottone screenshot e permette di creare la cattura immagine e salvarla, per la successiva creazione dello storyboard. Quando il metodo viene chiamato vengono passati come parametri la camera da cui è avvenuta la cattura e la lunghezza focale: il primo parametro serve per effettuare un render della scena dal punto di vista della camera, l'immagine che viene ottenuta viene convertita in Byte e viene salvata sia all'interno di una lista di immagini, sia all'interno del dispositivo. Infine, per ogni screenshot vengono salvati la durata dell'azione corrispondente, il numero dello screenshot e la lunghezza focale, anch'essa ottenuta come parametro del metodo. Questi parametri serviranno per la successiva fase di creazione dello storyboard.

Infine, ogni volta che viene effettuato uno screenshot all'interno della classe vengono chiamati i metodi `CompletedAction()` e `IncrementScreenshotCount()` della classe *SimulationManager* che permettono rispettivamente di salvare lo stato della scena nel momento in cui

è stato effettuato uno screenshot, per l'operazione di UNDO, e di incrementare il contatore che tiene traccia del numero delle catture effettuate.

4.6.10 UNDO

Con il tasto UNDO presente nel Menù della simulazione è possibile tornare alla configurazione dell'ultimo screenshot effettuato. Le operazioni che permettono di gestire quest'operazione sono contenute all'interno della classe *SimulationManager*. Ogni volta che viene effettuata una cattura viene chiamato il metodo `CompletedAction()`: questo metodo si occupa di salvare all'interno di un file .csv le condizioni di ogni oggetto presente all'interno della scena. Di ogni camera vengono salvati posizione, orientamento e lunghezza focale, di ogni oggetto posizione e orientamento e di ogni personaggio vengono salvati posizione, orientamento e stato in cui si trova. Ogni riga della tabella è identificata dal numero dello screenshot che è stato effettuato.

Quando viene premuto il tasto UNDO presente nel menù viene invece chiamato il metodo `UndoLastAction()` della classe *SimulationManager*. Il metodo si occupa di recuperare tutte le informazioni all'interno del file .csv e di riportare la scena alle condizioni precedenti: gli oggetti, le camere e i personaggi vengono riportati alle condizioni dello screenshot precedente e le camere aggiunte vengono eliminate.

4.6.11 Descrizione testuale della scena

Ogni volta che un'azione viene compiuta all'interno del sistema un testo automatico viene generato come descrizione di ciò che accade. Questa descrizione, insieme alle catture generate con le macchine virtuali, permette la generazione dello storyboard.

Il componente che si occupa di ciò prende il nome di *PhraseGenerator*. Questo componente è composto da svariati metodi che permettono la creazione delle frasi dello storyboard.

Frasi semplici

Uno dei più importanti è il metodo `GenerateSimplePhrase()`: esso viene chiamato ogni volta che un'azione è compiuta e riceve in input sei parametri: il tipo di soggetto (woman, man), il nome del soggetto (Anna, Luca), il tipo e il nome del complemento oggetto, il nome dell'azione svolta e una variabile booleana che indica se l'azione è compiuta dal soggetto su un complemento oggetto o su sé stesso. Il metodo costruisce la frase elaborando le informazioni

precedenti, unendo le stringhe di testo tra loro. La generazione delle frasi in lingua inglese prevede alcuni accorgimenti:

- L'azione che viene utilizzata per costruire la frase si riferisce a una terza persona singolare, perciò, viene aggiunta una *s* alla fine della prima parola che compone l'azione. Ad esempio, se l'azione è *stand up* viene la frase avrà il verbo *stands up*.
- Nel caso in cui l'azione preveda come prima parola *stop* si dovrà aggiungere alla seconda parola dell'azione il suffisso *-ing*. Ad esempio per indicare che il personaggio smette di ballare si userà il verbo *stops dancing*.
- Per quanto riguarda il verbo *sit* viene aggiunto nella generazione della frase la scritta *on,* per indicare dove il personaggio si sta sedendo.
- Infine, nel caso in cui l'azione scelta sia *talk,* si utilizza il metodo `GenerateDialogue()` che permette di includere nella descrizione dell'azione anche il dialogo tra i due personaggi. Di ciò se ne parlerà in seguito nel capitolo.

Infine, per rendere la generazione delle frasi più naturale viene utilizzata un API che permette di utilizzare su Unity Wordnet [84], un database lessicale inglese che permette di raggruppare parole affini tra loro. Per gestire questo database viene utilizzato un componente chiamato *WordnetManager* che tramite il metodo `GetSyn()` fornisce al componente *PhraseGenerator* una serie di sinonimi da utilizzare per la generazione delle frasi.

È inoltre importante segnalare che all'inizio di ogni frase creata con il metodo `GenerateSimplePhrase()` viene inserito un avverbio di tempo. Esso viene generato attraverso il metodo `GenerateTimeAdverb()` della classe *PhraseGenerator*. Questo metodo aggiunge un avverbio di tempo in base alle condizioni della scena. In particolare, se il metodo viene chiamato per la prima azione della scena si aggiunge all'inizio della frase la stringa "First,". Nel caso in cui l'azione sia preceduta da un'azione precedente invece, l'avverbio che viene posto davanti alla frase dipende dal fatto che le due azioni siano contemporanee o consecutive.

L'utente può definire se due o più azioni sono contemporanee o consecutive sempre utilizzando il menù per la gestione delle azioni: in particolare sono presenti due radio button che permettono di definire la temporaneità degli eventi, come si può vedere in Figura 4.16. È infatti presente un parametro booleano chiamato *contemporaryAction,* all'interno della classe *SimulationManager,* che definisce di che tipo di azione si tratta, esso cambia in base a quale pulsante sul menù viene premuto.

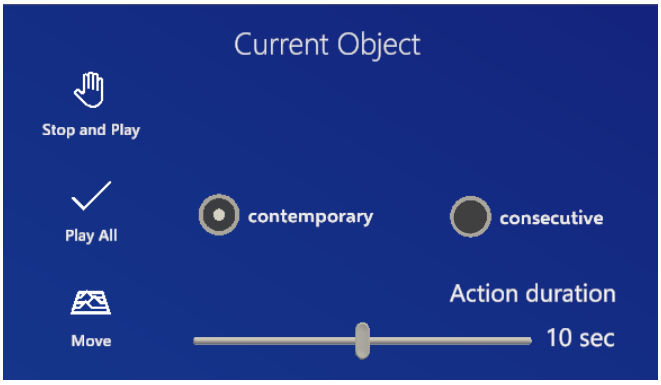

Figura 4.16: Menù per la gestione delle azioni

Nel momento in cui il metodo `GenerateTimeAdverb()` viene chiamato il valore contenuto all'interno del parametro *contemporaryAction* viene letto e viene generato un avverbio di tempo corrispondente al tipo di azione. Gli avverbi sono contenuti all'interno di due vettori di stringhe.

Dialoghi

Il secondo metodo presente all'interno della classe *PhraseGenerator* viene utilizzato per i dialoghi tra i personaggi. Per prima cosa bisogna specificare che nel momento in cui si sceglie l'azione *talk* tra due personaggi viene aperto un menù che permette l'inserimento di un dialogo: esso può essere inserito tramite tastiera oppure utilizzando l'interfaccia vocale. In particolare, quest'ultima contiene il componente MRTK *DictationHandler* che permette il riconoscimento e la trascrizione delle frasi. Il pannello che permette l'inserimento del testo è rappresentato in Figura 4.17.

Per la generazione delle frasi si utilizza il metodo `GenerateDialogue()` che si occupa di leggere il valore presente all'interno del menù e di inserire il dialogo all'interno della descrizione.

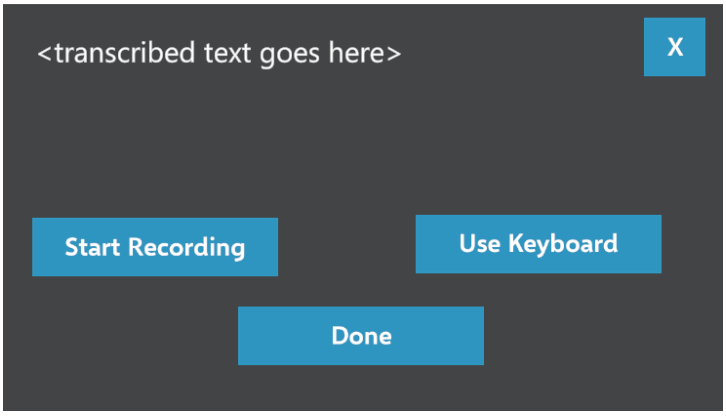

Figura 4.17: Pannello per l'inserimento dei dialoghi

Frasi di movimento

L'ultimo metodo della classe *PhraseGenerator* che viene utilizzato per la generazione di frasi è il metodo `GenerateMovementPhrase()`: questo metodo permette di generare delle frasi quando il personaggio cammina nell'ambiente. Il metodo accetta in input come parametri nome e tipo del personaggio che cammina e il nome e il tipo dell'oggetto verso il quale si sta muovendo. La funzione viene chiamata all'interno del metodo `FindDestination()` della classe *ObjectsAnimator* di cui si è spiegato l'utilizzo nel capitolo 4.6.1 Movimento dei personaggi. Per capire verso quale oggetto si sta muovendo il personaggio si utilizza il metodo Unity `Physic.Raycast()` [85] che permette di tracciare un raggio da un'origine, il personaggio, verso una determinata direzione, la destinazione della camminata. Il metodo restituisce un valore booleano che indica se il raggio ha colpito un oggetto nella scena e attraverso la struttura RaycastHit si può accedere all'oggetto che ha intersecato il raggio. Il metodo `GenerateMovementPhrase()` prende quindi in input i parametri per generare la frase di movimento. Nel caso in cui nessun oggetto venga colpito la frase viene semplificata descrivendo la camminata del personaggio nella stanza.

Salvataggio

Tutte le frasi che vengono generate dalla classe *PhraseGenerator* vengono salvate all'interno di una lista chiamata *buffer*: ogni frase ha come indice della lista il numero dello screenshot a cui essa è associata. La lista di stringhe viene utilizzata nella successiva fase di generazione dello storyboard HTML.

ChatGPT

Nonostante tutti gli accorgimenti utilizzati per la generazione delle frasi spesso risultano ripetitive e poco naturali. Per questo motivo si è deciso, dopo aver generato lo storyboard, di rielaborare le descrizioni tramite l'utilizzo di un'intelligenza artificiale come ChatGPT [86]. Si può trovare un esempio del risultato ottenuto in Figura 4.18.

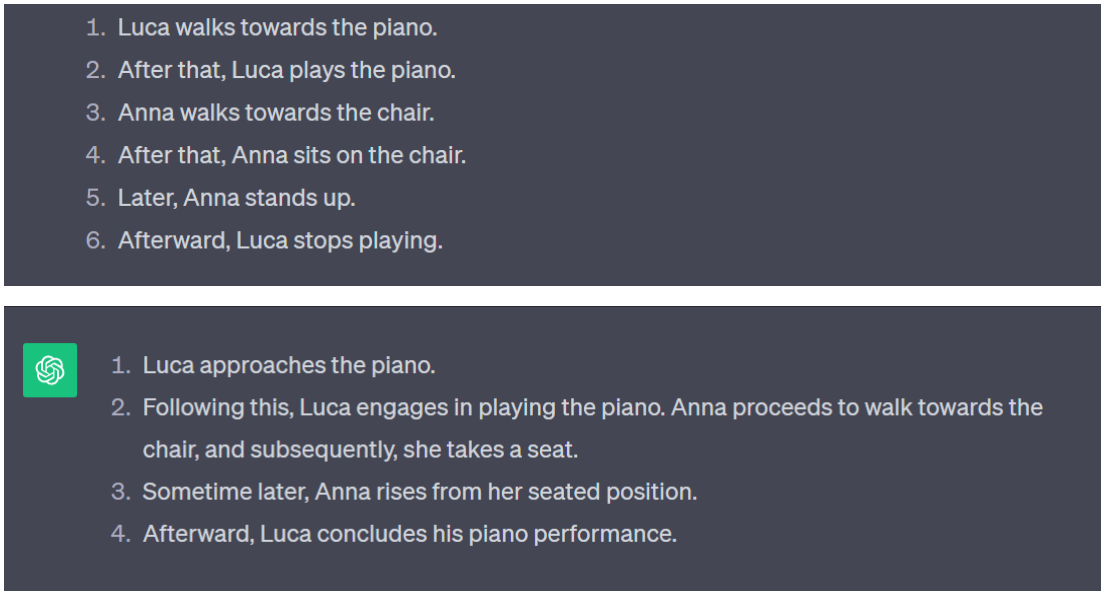

Figura 4.18: Esempio di rielaborazione della descrizione della scena eseguita con ChatGPT. Nella figura in alto si possono vedere le frase generate con l'applicazione Unity prima della rielaborazione. Nella parte sottostante invece si trovano le frasi rielaborate con l'utilizzo dell'intelligenza artificiale.

4.6.12 Generazione dello storyboard

Una volta generate tutte le catture della scena tramite l'utilizzo delle camere virtuali si può procedere alla generazione dello storyboard vero e proprio. La classe che si occupa di gestire il file in output prende il nome di *OutputGenerator*. La classe contiene principalmente due metodi: il metodo `GenerateFile()` che permette la vera e propria creazione dello storyboard HTML e il metodo `SaveStoryboard()` che consente di salvare sul dispositivo Microsoft HoloLens 2 i file realizzati.

Il metodo `GenerateFile()` viene chiamato nel momento in cui l'utente preme sul tasto OUTPUT presente sul menù della simulazione, di cui si è parlato nel capitolo 4.6.7. Questo metodo crea un file HTML utilizzando tutti i dati salvati nelle fasi precedenti: in particolare, per ogni screenshot che è stato salvato, crea una sezione all'interno del documento. Ogni sezione del documento contiene al suo interno il numero dello shot, lo screenshot realizzato con la camera virtuale, la durata dell'azione, la lunghezza focale utilizzata e la descrizione generata automaticamente dal sistema. Quest'ultimo parametro risulta essere editabile all'interno del documento HTML e permette perciò la rielaborazione del testo.

Tutte le informazioni necessarie alla creazione dello storyboard sono contenute all'interno delle liste di dati create nel componente

CameraManager di cui si è parlato nel capitolo “4.6.9 Aggiunta e gestione di camere virtuali ”.

Inoltre, in alto nel file è presente il titolo dello storyboard nonché il nome con cui è stata salvata la scena creata. Si può vedere un esempio di file generato con il sistema in Figura 4.19.

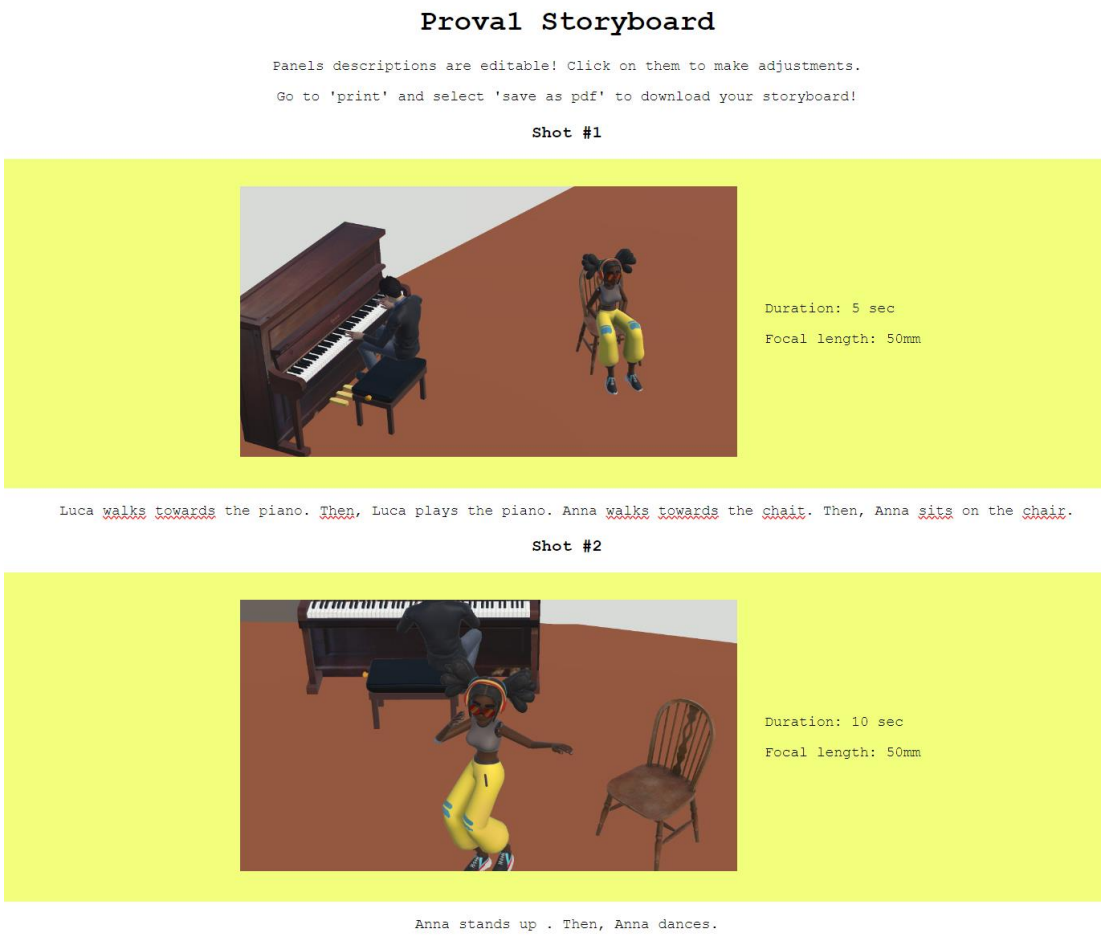

Figura 4.19: Esempio delle prime due vignette di uno storyboard realizzato con il sistema

Dato che il dispositivo utilizzato per la creazione dello storyboard è il Microsoft HoloLens 2 che, dato il suo field of view ridotto, non permetterebbe una corretta visualizzazione dello storyboard, si è scelto di trovare un metodo per permettere la visualizzazione del file non direttamente all’interno dell’applicazione ma su altri dispositivi. Per fare ciò è necessario salvare il file generato all’interno dell’applicativo e condividerlo con altri dispositivi. Dopo svariate ricerche si è concluso che il metodo più intuitivo e veloce per la condivisione dello storyboard fosse l’utilizzo del Windows Device Portal di cui si è discusso nel capitolo 3.5.2.

Per condividere il file tramite l'utilizzo del portale è però necessario salvarlo in locale all'interno del dispositivo. Per fare ciò si utilizza il metodo `SaveStoryboard()` della classe *OutputGenerator*. Il metodo, che viene chiamato all'interno del metodo `GenerateFile()`, si occupa di creare delle cartelle all'interno del dispositivo attraverso i metodi della classe Directory [87] di Unity e successivamente si occupa si salvare il file generato, e tutti gli screenshot realizzati, all'interno delle cartelle create. I file sono convertiti in Bytes e salvati con il metodo `WriteAllBytes()` della classe File [88] di Unity.

Una volta salvato, si può accedere facilmente al file tramite il portale Windows, per scaricarlo su altri dispositivi.

4.7 Comandi vocali

Dopo una prima versione dell'applicazione che prevedeva l'interazione con l'utilizzo delle mani e del controller ray ci si è accorti di alcune difficoltà legate all'usabilità dell'applicazione. Per questo motivo è stata aggiunta un'altra modalità di interazione al sistema che prevede l'utilizzo dei comandi vocali e del tracciamento della testa. Sebbene head tracking e utilizzo del controller ray siano mutuamente esclusivi, i comandi vocali possono invece essere utilizzati anche in combinazione con il raggio che si basa sul tracciamento delle mani.

4.7.1 Configurazione

Per utilizzare i comandi vocali all'interno di un applicazione Unity MRTK per prima cosa bisogna configurare il componente *MixedRealityToolkit* all'interno della scena. In particolare, nel percorso *MixedRealityToolkit* > *Input* > *Speech* bisogna creare un nuovo profilo per la gestione dei comandi, il menù viene presentato in Figura 4.20. In questa sezione è possibile definire tutte le parole che devono essere riconosciute dal sistema, a che pulsante della tastiera sono associate e che azione possono innescare.

Una volta effettuata la fase di configurazione delle parole si procede a programmare le risposte dell'applicazione all'input vocale. Per fare ciò si utilizzano prevalentemente due approcci: il primo prevede l'utilizzo di comandi vocali globali, non legati in particolare al nessun oggetto della scena ma che permettono di richiamare dei metodi dell'applicazione, l'altro approccio invece prevede delle risposte ai comandi vocali legate allo specifico oggetto che si indica con un puntatore che può essere quello della testa o delle mani.

In entrambi i casi si utilizza il componente MRTK *SpeechInputHandler*.

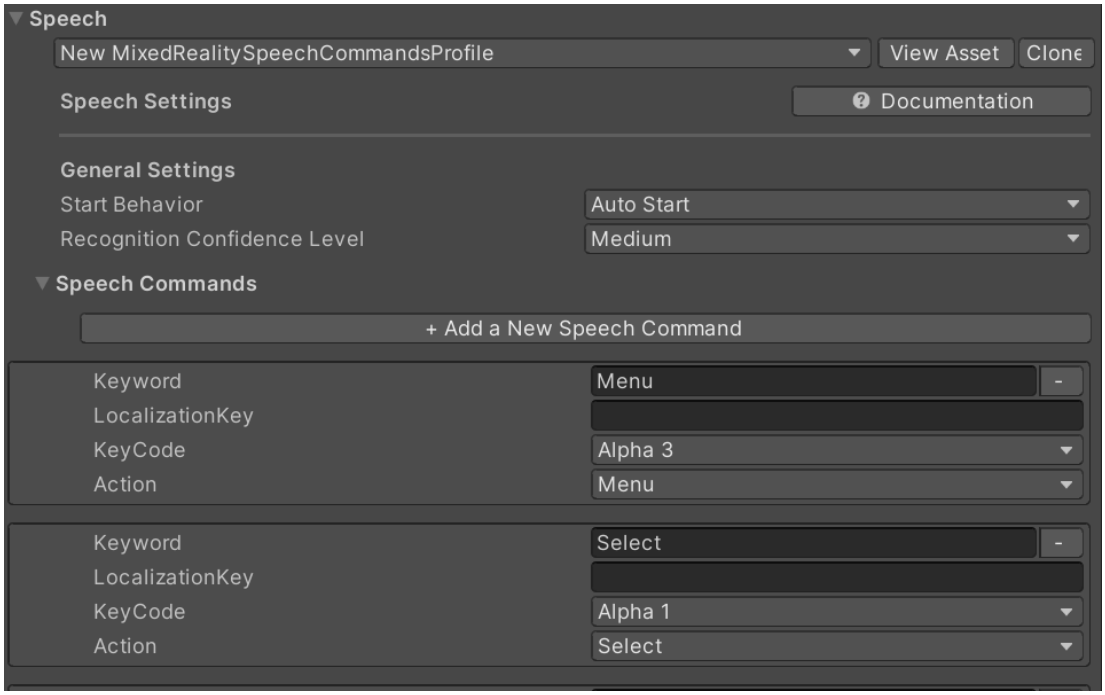

Figura 4.20: Profilo di configurazione dei comandi vocali nel componente MixedRealityToolkit

4.7.2 Comandi globali

I comandi vocali globali permettono di richiamare dei metodi senza che l'utente si focalizzi su un determinato oggetto della scena. Nell'applicativo per la gestione di questi comandi è presente un game object chiamato *GlobalSpeechHandler*. Al suo interno è presente il componente *SpeechInputHandler* che consente di associare una chiamata a un metodo a ogni input vocale. In particolare, nel componente in questione viene settata la proprietà *isFocusRequired* a *false*, ciò permette di gestire gli input dell'utente senza bisogno che l'utente punti l'oggetto con lo sguardo o con la mano.

I comandi vocali che vengono gestiti in modo globale sono:

- "New Camera": il comando vocale comporta la creazione di una nuova camera davanti all'utente con la sua stessa direzione di vista. In questo caso il metodo che viene chiamato quando si riceve l'input vocale è il metodo `NewCamera()` della classe *CameraManager* di cui si è parlato nel capitolo 4.6.9.
- "Stop": questo comando consente di stoppare il personaggio in una specifica posa. Il suo effetto è equivalente a quello che si ottiene a premere il pulsante Stop And Play nel menù delle azioni. Il metodo che viene chiamato è `startStop()` della classe *ObjectsAnimator*.

- “Output”: Il comando vocale in questione permette di generare lo storyboard, chiamando il metodo `GenerateFile()` della classe *OutputGenerator*
- “Place Here”: questo comando viene utilizzato per posizionare gli oggetti sul piano scelto nel momento in cui viene utilizzata la modalità Tap To Place. Il metodo associato a questo comando è stato creato per gestire il posizionamento tramite l’input vocale: prende il nome di `PlaceObject()` ed è contenuto all’interno della classe *ObjectManager*. Il metodo si occupa di identificare l’oggetto selezionato nel momento in cui si esegue il comando vocale e di chiamare il metodo `StopPlacement()` della classe *TapToPlace*.

4.7.3 Comandi che richiedono il focus dell’utente

La seconda categoria di comandi sono quelli che richiedono l’attenzione dell’utente perché l’input vocale venga ricevuto. In particolare, all’interno dell’applicativo, sono presenti due comandi vocali di questo tipo: il comando “Screenshot” e il comando “Select”.

Il comando “Screenshot” viene utilizzato per effettuare una cattura attraverso la camera virtuale: è necessario che il puntatore si soffermi sul pulsante screenshot della camera virtuale che si sta utilizzando per fare il modo che l’input sia ricevuto. Questo input vocale equivale a premere il bottone screenshot della camera e richiama il metodo `CamCapture()` della classe *SecondCameraManager*.

Il comando “Select” risulta essere molto importante, soprattutto per quanto riguarda la modalità di interazione che prevede il tracciamento della testa. Il comando vocale sostituisce il gesto *tap air* o il click effettuato con le mani. Grazie a questo comando vocale è quindi possibile interagire con ogni bottone o oggetto all’interno della scena: tutti i metodi spiegati nei capitoli precedenti che permettevano di essere chiamati con l’ausilio delle mani funzionano allo stesso modo se l’interazione avviene con l’input vocale.

Si rende quindi evidente la necessità di impostare la variabile *isFocusRequired* a *true* per tutti i componenti *SpeechInputHandler* presenti negli oggetti della scena.

4.8 Head Tracking

Come già spiegato nei capitoli precedenti dopo una prima versione dell’applicazione dove era prevista esclusivamente l’interazione con le mani si è deciso di permettere agli utenti di interagire anche con il movimento della testa e gli input vocali.

4.8.1 Configurazione

Per permettere l'interazione attraverso lo sguardo è, per prima cosa, necessario avere un puntatore che rappresenti la direzione di vista. Per fare ciò è necessario accedere alle proprietà del componente *MixedRealityToolkit* e aggiungere nella sezione *Input>Pointers>Pointer Options* un puntatore del tipo GGV Pointer (Gaze Gesture Voice). Questo elemento è in grado di gestire l'interazione con lo sguardo, la sua posizione e la sua direzione sono definite da posizione e direzione della testa dell'utente.

Per aggiungere un puntatore è inoltre necessario creare un nuovo profilo per i puntatori nella scena, sempre all'interno del componente *MixedRealityToolkit*.

4.8.2 Gestione delle due modalità nell'applicativo

Una volta abilitata la modalità di interazione con lo sguardo è stato necessario creare un componente che gestisse il passaggio da una modalità d'interazione all'altra. Per questo è stato creato il componente *ModalityManager*.

Inoltre, per permettere all'utente di cambiare in qualsiasi momento la modalità utilizzata è stato creato un menù, rappresentato in Figura 4.21, che permette attraverso l'utilizzo di due radio button di settare la modalità desiderata.

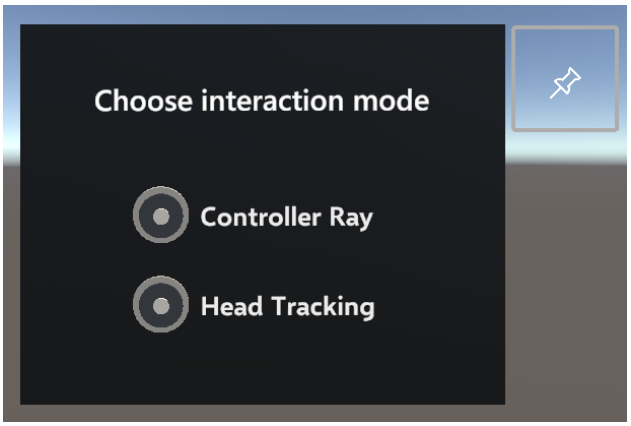

Figura 4.21: Menù che permette il passaggio da una modalità di interazione ad un'altra

Quando si passa da una modalità all'altra il componente *ModalityManager* si occupa di svolgere tre operazioni principali:

- Per prima cosa se si passa dalla modalità *Controller Ray* alla modalità *Head Tracking* è necessario nascondere il puntatore che rappresenta la

direzione delle mani e rendere visibile il puntatore che permette di mostrare la direzione dello sguardo. Questa operazione viene svolta con i metodi `SetGazePointerBehavior()` e `SetHandRayPointerBehavior()` della classe *PointerUtils*.

- In secondo luogo, è necessario passare da un profilo per la gestione dei puntatori ad un altro, in modo tale da settare come puntatore principale quello della testa o quello delle mani. In questo caso si accede attraverso lo script *ModalityManager* alla proprietà *PointerProfile* appartenente al componente *MixedRealityToolkit*.
- Infine, risulta necessario per tutti i gli oggetti che all'interno della scena hanno il componente *SolverHandler* settare il parametro *TrackedObjectType* in modo corretto. Questa proprietà si occupa di calcolare la posizione e l'orientamento degli oggetti rispetto a un punto specifico e serve per posizionare gli oggetti nella modalità Tap To Place. Nel caso in cui si utilizza come interazione principale quella che prevede l'utilizzo dello sguardo la proprietà *TrackedObjectType* viene settata come *Head*, in caso contrario la proprietà è impostata come *ControllerRay*.

Per passare da una modalità all'altra sono chiamati i metodi `ActivateGazeMode()` e `ActivateControllerRay()` della classe *ModalityManager*.

Per la modalità *Head Tracking* è necessario l'utilizzo dei comandi vocali come spiegato nel capitolo precedente, mentre per quanto riguarda la modalità *Controller Ray* il principale gesto di interazione è l'*air tap*.

4.9 Accorgimenti per la build

Per utilizzare l'applicazione sviluppata su Unity su un dispositivo di realtà aumentata è necessario utilizzare degli accorgimenti per fare il modo che la *build* del progetto funzioni in modo corretto. In particolare, risulta essere necessario l'utilizzo di una cartella chiamata *Streaming Asset* per l'utilizzo di file esterni, che devono essere scritti e letti dall'applicazione.

4.9.1 Streaming Asset Folder

La cartella *Streaming Assets* [89] permette agli utenti inserire dei file nel filesystem del dispositivo e di accederci semplicemente con il nome del percorso della cartella. Per fare ciò si crea una cartella su Unity con il nome StreamingAssets, tutti i file inseriti vengono automaticamente copiati in un progetto Unity, in una particolare cartella del computer di destinazione.

Inoltre, per accedere alla cartella via codice si può utilizzare il comando *Application.streamingAssetsPath* che consente di puntare sempre alla posizione corretta del file, qualsiasi sia la piattaforma su cui l'applicazione è in esecuzione.

L'utilizzo di questa cartella è stato essenziale per l'accesso ai file .csv che contengono, ad esempio, tutte le informazioni per ricaricare una scena creata in precedenza.

Capitolo 5

5. Test e Risultati

Per analizzare l'usabilità del sistema e per valutare se l'utilizzo della realtà aumentata possa semplificare e velocizzare la realizzazione di uno storyboard sono stati svolti dei test di usabilità sull'applicazione sviluppata.

5.1 Casi d'uso

Per l'analisi di un sistema è essenziale definire uno o più casi d'uso. Questa metodologia permette di porre degli obiettivi e consente di creare una serie di task specifici che l'utente deve eseguire per valutare ed esplorare il sistema.

Per valutare l'applicativo è stato scelto di realizzare due casi d'uso differenti: il primo pone dei vincoli più stringenti sulla realizzazione dello storyboard, mentre il secondo concede più libertà all'utente per misurare l'usabilità del sistema.

In entrambi i test sono forniti gli ambienti 3D già costruiti per concentrare la valutazione sulla fase di storyboarding.

5.1.1 Caso d'uso 1

Vengono forniti agli utenti uno script e uno storyboard tradizionale. Lo storyboard rappresenta ciò che gli utenti devono riprodurre utilizzando il sistema fornito. All'interno dello script, invece, sono presenti la descrizione della scena e un elenco dettagliato delle inquadrature da riprodurre, con indicazione sulla durata dell'azione, descrizione della stessa e tipo di campo o piano che deve essere realizzato.

Script

```
Anna e Luca si trovano nella loro casa.
Luca si avvicina al pianoforte e comincia a suonare, nel
mentre Anna si siede su una sedia accanto a lui.
A un certo punto Anna si alza dalla sedia e inizia a
ballare.
Successivamente Luca smette di suonare e si avvicina ad
Anna.
Luca applaude ad Anna. Anna smette di ballare e gli
sorride.
```

Storyboard di riferimento: Figura 5.1

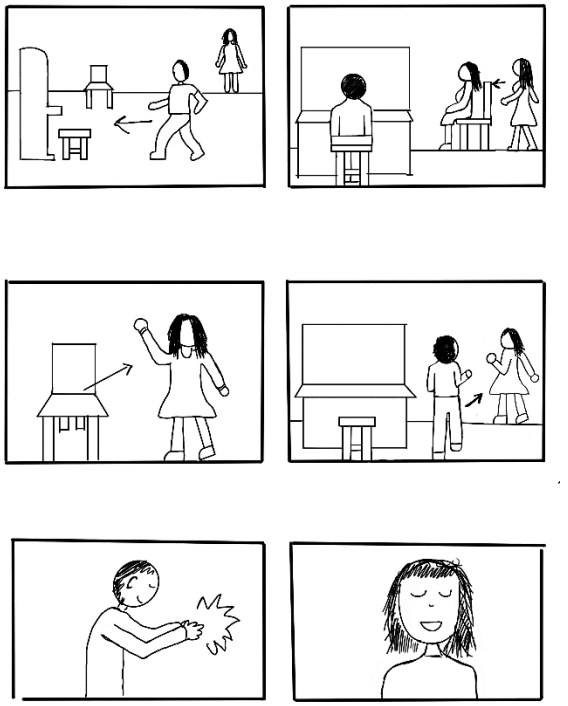

Figura 5.1: Storyboard caso d'uso 1

Elenco delle inquadrature

	Tipo di inquadratura	**Durata**	**Descrizione**
1	Totale	≈ 2 sec	Luca si avvicina al pianoforte.
2	Figura intera	≈ 4 sec	Luca inizia a suonare il pianoforte. Nel mentre Anna cammina verso una sedia vicino a lui e si siede.
3	Figura intera	≈ 3 sec	Ad un certo punto Anna si alza e inizia a ballare.
4	Totale	≈ 5 sec	Luca smette di suonare e cammina verso Anna.

5	Mezza figura	≈ 4 sec	Luca applaudisce ad Anna
6	Primo piano	≈ 2 sec	Anna smette di ballare e sorride.

Tabella 2: Elenco delle inquadrature per il primo caso d'uso

5.1.2 Caso d'uso 2

Per il secondo caso d'uso all'utente viene fornito solamente uno script con la descrizione della scena e degli avvenimenti. Viene data all'utilizzatore la libertà di scegliere quante e quali inquadrature realizzare in modo che possa valutare l'usabilità del sistema.

Script

```
Sofia e Paola sono al parco.
Sofia è intenta a fare degli esercizi fisici di fianco a
una panchina. Nello stesso momento Paola, che si trovava
dall'altra parte del parco, inizia a camminare verso la
panchina.
Una volta arrivata vicino alla panchina Paola saluta
Sofia.
Sofia invita Paola ad unirsi a lei dicendole "Alleniamoci
insieme!"
Paola inizia ad allenarsi con Sofia.
Dopo un po' di tempo Paola si siede sulla panchina.
```

5.2 Fase di Test

I test dell'applicazione sono stati svolti in una sala del Politecnico di Torino fornita di un tavolo e di un ambiente ampio per permettere di verificare le funzionalità del sistema e la creazione di uno storyboard in modalità tabletop.

Per provare l'applicazione sono stati contattati tre esperti di dominio che stanno attualmente svolgendo un dottorato o postdoc in Computer Graphics o Extended Reality.

L'obiettivo è stato quello di testare l'applicazione in termini di usabilità, robustezza e di controllare la presenza di eventuali bug o errori.

5.2.1 Svolgimento dei test

Si può dividere lo svolgimento dei test in principalmente tre fasi differenti.

Nella prima fase è stata fatta una breve introduzione dell'applicazione: sono stati spiegati gli obiettivi e le funzionalità dell'applicativo, le diverse modalità di interazione e come sarebbe avvenuto lo svolgimento del test.

Nella seconda fase è stata provata l'applicazione: si è chiesto ad ogni esperto di svolgere per prima cosa il tutorial introduttivo, che presenta le diverse funzionalità, e successivamente di svolgere un caso d'uso a scelta per creare uno storyboard all'interno dell'applicazione. Non sono stati posti limiti per quanto riguarda la percentuale del caso d'uso da portare a termine o per quanto riguarda le tempistiche. È stato infatti chiesto ad ogni partecipante di utilizzare l'applicazione per il tempo desiderato al fine di capire il funzionamento dell'app e da poter dare un giudizio. Durante questa fase sono stati raccolti i feedback che ogni esperto ha dato a voce durante l'utilizzo dell'applicativo.

Nell'ultima fase è infine stato richiesto ad ogni esperto, tramite un'intervista, di valutare il sistema. Sono state utilizzate per la valutazione le 10 euristiche di Jakob Nielsen [90], 10 parametri che permettono di valutare l'usabilità di un'interfaccia utente. Dopo l'esecuzione dei test è stato perciò chiesto ad ogni esperto di valutare ogni euristica tramite una scala Likert [91] che ha permesso di dare un voto da 1 a 5, dove 1 è la valutazione minima e 5 la massima.

5.2.2 Risultati

Come già accennato nel paragrafo precedente gli esperti hanno valutato a voce il sistema durante il test e sono stati intervistati una volta concluso. Di seguito sono elencati alcuni aspetti importanti da migliorare:

- Migliorare il riconoscimento della voce: spesso nella modalità che utilizza il riconoscimento vocale ci sono stati degli errori. Il sistema fatica a riconoscere le parole e i comandi vocali non vengono eseguiti correttamente
- Suoni di feedback: è molto importante dare un feedback all'utente durante l'utilizzo dell'applicativo. Per alcuni pulsanti manca un feedback del sistema che evidenzia se l'azione è stata eseguita correttamente.

- Highlight del personaggio: Quando viene selezionato un personaggio manca un elemento vicino che ne evidenzi la selezione, risulta scomodo guardare il menù delle azioni per sapere se il personaggio selezionato è quello corretto.
- Tasto stop and play: Il nome di questo pulsante può risultare ambiguo per gli utenti. Sarebbe meglio utilizzare due pulsanti con nomi differenti.
- Possibilità di vedere gli screenshot eseguiti: gli esperti hanno consigliato di implementare un modo per rivedere gli screenshot eseguiti durante l'utilizzo dell'applicazione, in modo da ricordare all'utente il lavoro che ha svolto nelle fasi precedenti ed evidenziare l'eventuale presenza di errori.
- Possibilità di spostare gli oggetti con il tracciamento della testa: attualmente non è presente un comando vocale che permetta di spostare e ruotare gli oggetti con l'utilizzo della modalità *Head Tracking*. Questo potrebbe essere utile per permettere all'utente di eseguire tutte le operazioni senza l'utilizzo delle mani.
- Disattivare i tasti che non possono essere utilizzati in determinate situazioni. Un esempio può essere il tasto *Move* che dovrebbe essere disattivato nel momento in cui l'utente si trova in uno stato in cui non può camminare.
- Creare un tasto UNDO più intuitivo per l'utente: il tasto UNDO prevede di tornare alla configurazione dell'ultimo screenshot. Sarebbe utile implementare un tasto che permetta di eliminare anche le azioni o i movimenti una volta eseguiti in modo errato.
- Velocità delle animazioni: a volte risulta complesso riuscire a eseguire lo screenshot corretto per un'animazione. Bisognerebbe rallentare determinate animazioni o permettere di vedere l'animazione frame per frame.
- Pannello dei comandi vocali: potrebbe essere utile all'utente, per ricordare i comandi vocali, inserire un pannello riassuntivo che indichi ogni comando e la sua funzione.

Dato che lo scopo dei test è quello di valutare l'usabilità e la robustezza del sistema il miglior modo per farlo è stato utilizzare le Euristiche di Nielsen: di seguito sono riportate in Tabella 3 le valutazione di ogni esperto per le 10 metriche. Inoltre, il grafico in Figura 5.2 confronta tra loro i punteggi per le diverse euristiche.

Euristiche	Esperto 1	Esperto 2	Esperto 3
1) Visibilità dello stato del sistema	3	3	4
2) Corrispondenza tra sistema e mondo reale	4	4	4
3) Controllo e libertà	2	4	4
4) Consistenza e standard	3	4	5
5) Prevenzione dell'errore	2	4	4
6) Riconoscimento anziché ricordo	4	3	5
7) Flessibilità d'uso	4	4	5
8) Design ed estetica minimalista	5	4	5
9) Aiuto all'utente	3	3	4
10) Documentazione	4	4	5

Tabella 3: Punteggi di valutazione del sistema attraverso le Euristiche di Nielsen tramite la scala Likert con punteggio da 1 a 5.

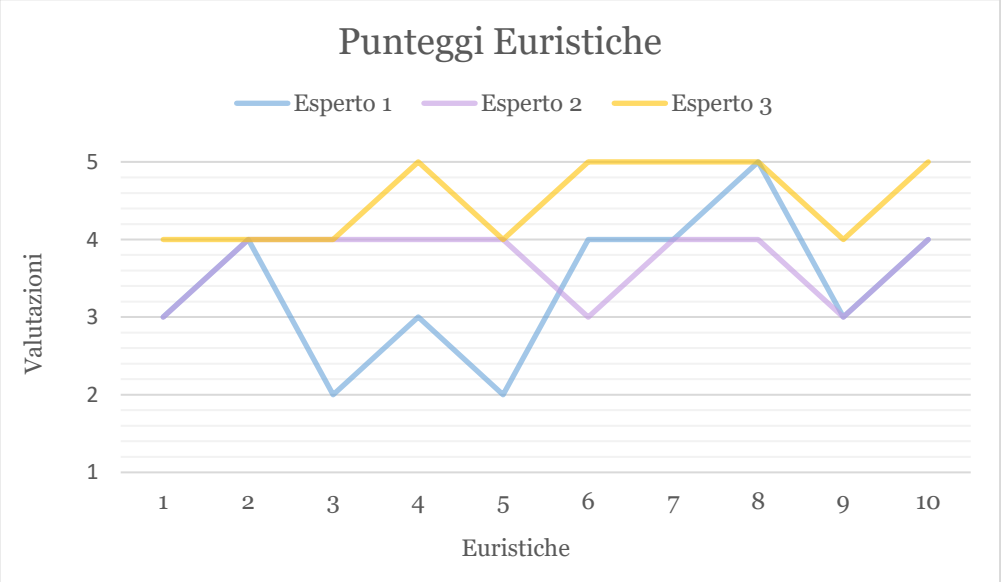

Figura 5.2: Grafico di rappresentazione dei punteggi degli esperti.

5.3 Analisi dei risultati

Dai dati raccolti si può notare che l'aspetto maggiormente gradito dagli esperti è stata la metrica che si riferisce all'estetica dell'applicazione che consente di dare maggior enfasi al contenuto tramite un design minimalista. Inoltre, sono stati molto apprezzati la flessibilità dell'uso dell'applicazione e la documentazione, rappresentata dal tutorial e dai pannelli, per la comprensione del sistema.

Gli aspetti invece che hanno riscontrato maggiori criticità sono stati differenti per i 3 esperti. In particolare, si possono notare due punteggi negativi per quanto riguarda il l'esperto 2 che si riferiscono a controllo e libertà e prevenzione dell'errore. Queste due valutazioni si riferiscono rispettivamente alla mancanza della possibilità di poter effettuare degli screenshot in determinate pose perché le animazioni sono troppo veloci e la mancanza di un UNDO più intuitivo per poter annullare le operazioni sbagliate.

Un altro aspetto che è stato valutato in modo non sempre positivo è la visibilità dello stato del sistema, questo si riferisce in particolare alla mancanza di feedback adatti alle varie azioni che l'utente compie.

In linea generale bisogna tener conto che sono presenti alcune difficoltà nell'utilizzo del Microsoft HoloLens 2: queste difficoltà sono legate sia all'esperienza dell'utente, sia all'ergonomia del dispositivo. A parità di competenze, quindi, è presente un fattore di soggettività che andrà poi considerato nei test futuri.

È stata in ogni modo essenziale questa fase: grazie agli esperti è stato possibile definire quali sono gli aspetti chiave da migliorare nell'ottica di condurre uno studio estensivo con gli utenti finali.

Capitolo 6

6. Conclusioni e sviluppi futuri

6.1 Conclusioni

Questo progetto di tesi si è posto come obbiettivo la creazione di un'applicazione in realtà aumentata per la creazione di storyboard. È stato creato un sistema che permette agli utenti di creare con i gesti delle mani un ambiente virtuale, inserire e animare i personaggi e infine ottenere uno storyboard completo di vignette e descrizioni. Ovviamente alcuni aspetti del sistema vanno ancora migliorati, soprattutto per consentire agli utenti finali, per la maggior parte non esperti, di utilizzare il dispositivo con semplicità e immediatezza.

Nella fase di test finale condotta con gli esperti di dominio e di usabilità sono emersi infatti alcuni aspetti chiave: essi permetteranno di migliorare ulteriormente l'usabilità dell'applicazione per poi, in una fase successiva, portare a condurre uno studio estensivo con gli utenti finali dell'applicazione.

È da tener conto in ogni caso la forte componente soggettiva legata all'utilizzo del dispositivo Microsoft HoloLens 2, ogni utente infatti può avere una percezione diversa, sia dal punto di vista dell'esperienza di utilizzo, sia per quanto riguarda l'ergonomia e la comodità che prova ad indossare il dispositivo.

Per questo motivo sono state implementate diverse modalità di interazione e le implementazioni future punteranno a migliorare l'aspetto dell'esperienza utente.

6.2 Sviluppi futuri

Il sistema presenta numerosi campi per ulteriori sviluppi, sia per quanto riguarda il miglioramento delle feature già presenti, sia per quanto riguarda l'aggiunta di funzionalità.

6.2.1 Integrazione con ChatGPT

Come visto nel capitolo "4.6.11 Descrizione testuale della scena" il testo che viene creato per la descrizione dello storyboard presenta alcuni problemi e si è perciò deciso di fare una successiva rielaborazione del testo con ChatGPT. Sarebbe dunque vantaggioso inserire all'interno dell'applicazione una

funzionalità che permetta, in modo automatico, di integrare la rielaborazione dell'intelligenza artificiale, per consentire la creazione di un testo più naturale e corretto. Questo può essere fatto con l'utilizzo di API apposite che integrano ChatGPT su Unity.

6.2.2 Possibilità di esperire il sistema in scala 1:1

Al momento il sistema è studiato per supportare la visualizzazione delle scene in modalità tabletop. In questo modo la scena viene creata e il sistema viene utilizzato dall'utente su un tavolo o, meglio, un piano.

Date le possibilità che la realtà aumentata fornisce sarebbe interessante implementare un sistema che permetta di visualizzare oggetti e personaggi in scala 1:1 e cioè a dimensione reale. La visualizzazione della scena, in questo modo, potrebbe essere un potente strumento non solo per creatori di storyboard ma anche per scenografi e registi e darebbe la possibilità di visualizzare il set, camminare al suo interno, capire la posizione di attori, oggetti di scena e camere. Consentire di visualizzare la scena a dimensione reale potrebbe risultare utile anche per gli attori stessi, permettendo loro di previsualizzare il set, immedesimarsi nel personaggio e capire prima delle effettive riprese in che modo interagire con l'ambiente.

All'interno dell'applicazione attualmente è possibile creare un piano delle dimensioni di un pavimento ed è possibile scalare a piacimento gli oggetti. Questo però non permette, ad esempio, di creare un'ambiente in scala tabletop e di rivederlo in scala 1:1 nel set reale con le giuste proporzioni della stanza. Per implementare questa feature bisognerebbe procedere con l'implementazione di un sistema che utilizzi i dati raccolti dallo strumento *Spatial Awareness* di MRTK e consenta, tramite uno studio dello spazio, di adattare un ambiente in scala tabletop ad un ambiente reale.

Sicuramente questa feature potrebbe essere molto interessante in ambito cinematografico come strumento di previsualizzazione.

6.2.3 Animatic

Attualmente il sistema in realtà aumentata creato permette di effettuare le catture delle immagini per la creazione dello storyboard. Sarebbe però interessante sfruttare la presenza delle animazioni e delle azioni dei personaggi per creare un altro strumento molto potente per la fase di preproduzione, l'animatic.

L'animatic si può definire come uno storyboard animato, esso consente infatti di previsualizzare graficamente la sceneggiatura attraverso una serie di immagini temporizzate o di video. Esso offre la possibilità di capire anche la

durata della singola scena e la temporaneità delle azioni. Attualmente, all'interno dell'applicativo, è presente uno slider che imposta la durata delle singole azioni e si potrebbe creare un animatic che consente di visualizzare ogni vignetta per la durata settata. Questo sistema però non darebbe l'idea delle azioni che vengono svolte e non sfrutterebbe a pieno la presenza delle animazioni. Sarebbe invece interessante implementare un sistema che permetta, oltre alla creazione delle immagini, anche la creazione di video attraverso le camere virtuali. Inoltre, sarebbe interessante creare uno strumento per editare il video una volta creato, per montare l'animatic secondo le preferenze dell'utente.

La creazione dell'animatic insieme allo storyboard permetterebbe perciò di risparmiare moltissimo tempo nella fase di preproduzione, fornendo degli strumenti importantissimi per la fase di riprese.

6.2.4 Possibilità di utilizzare oggetti reali

Attualmente, all'interno del sistema l'utente può creare la scena con l'utilizzo delle proprie mani, scegliendo da una libreria gli oggetti virtuali da porre all'interno del set. Bisogna però ricordare che la realtà virtuale è uno strumento molto potente e la sua peculiarità è la possibilità di permettere all'utente di interagire, oltre che con l'ambiente virtuale, con quello reale. Uno sviluppo futuro molto interessante sarebbe dato dalla possibilità di integrare all'interno della creazione della scena e dello storyboard anche oggetti reali. Utilizzando degli oggetti reali, come modellini che rappresentano gli elementi di scena, si potrebbe creare un interfaccia tangibile per l'utente, che permetta il posizionamento e il movimento dei personaggi.

Per fare ciò bisognerebbe implementare all'interno dell'applicazione un sistema di riconoscimento degli oggetti, della loro posizione e del loro orientamento sul piano di creazione. Una volta identificato ogni oggetto e le sue proprietà si dovrebbe consentire l'interazione tra gli oggetti virtuali e quelli reali, o altri oggetti virtuali che sostituiscono quelli reali.

Sarebbe interessante, ad esempio, sviluppare un sistema di creazione della scena che utilizzi un'interfaccia tangibile, ponendo gli oggetti reali su un tavolo reale per poi successivamente creare degli oggetti virtuali, che possiedono tutte le caratteristiche di quelli reali. Questo tipo di sistema potrebbe essere implementato, più semplicemente, anche tramite l'utilizzo di marker al posto di modellini veri e propri. Associando ogni marker ad un oggetto virtuale si può creare un interfaccia tangibile che permette la creazione del set tramite l'utilizzo di oggetti reali.

Tra gli sviluppi citati questo risulta senza dubbio uno dei più interessanti e consentirebbe di sfruttare a pieno le funzionalità che offre la realtà aumentata.

Bibliografia

[1] R. Henrikson, B. Araujo, F. Chevalier, K. Singh e R. Balakrishnan, «Multi-Device Storyboards for Cinematic Narratives in VR,» vol. Association for Computing Machinery, p. 787–796, 2016.

[2] «Realtà aumentata: dalla definizione all'uso: tutto quello che devi sapere,» Adobe, [Online]. Available: https://www.adobe.com/it/products/substance3d/discover/what-is-ar.html.

[3] «Le fasi della produzione cinematografica,» Movie Production, [Online]. Available: https://www.moviproduction.com/le-fasi-della-produzione-cinematografica/.

[4] N. F. Institute, «Pre-Production – Everything You Need To Know,» [Online]. Available: https://www.nfi.edu/pre-production/. [Consultato il giorno 09 2023].

[5] C. Pallant, Storyboarding: A critical history, Basingstoke, Hampshire: Palgrave Macmillan, 2018.

[6] «The History of Storyboarding,» [Online]. Available: https://makestoryboard.com/blog/the-history-of-storyboarding.

[7] J. Hart, «Basic Components and Principles of the Storyboard,» in *The Art of the Storyboard, Second Edition: A filmmaker's introduction*, Focal Press, 2007, pp. 37-40.

[8] D. R. Berryman, «Augmented Reality: A Review,» *Medical Reference Services Quarterly,* vol. 31, n. 2, pp. 212-218, 2012.

[9] P. Milgram, H. Takemura, A. Utsumi e F. Kishino, «Augmented reality: A class of displays on the reality-virtuality continuum,» in *Telemanipulator and telepresence technologies*, vol. 2351, Spie, 1995, pp. 282-292.

[10] A. Bottino, *AR System Architecture, Lezioni del corso di Realtà Virtuale anno 2020/2021,* Politecnico di Torino: Dipartimento di Automatica e Informatica, 2021.

[11] H. Ho-Gun e H. Jaesung, «Augmented Reality in Medicine,» *hmr,* vol. 36, n. 4, pp. 242-247, 2016.

[12] Cranmer, Eleanor E., M. Claudia tom Dieck e Paraskevi Fountoulaki, «Exploring the value of augmented reality for tourism,» *Tourism Management Perspectives,* vol. 35, 2020.

[13] «TabUi,» [Online]. Available: https://www.tabui.app/it.

[14] «IKEA Place,» [Online]. Available: https://www.ikea.com/global/en/newsroom/innovation/ikea-launches-ikea-place-a-new-app-that-allows-people-to-virtually-place-furniture-in-their-home-170912/.

[15] S. C.-Y. YUEN, G. YAOYUNEYONG e E. JOHNSON, «Augmented Reality: An Overview and Five Directions for AR in Education.,» *Journal of Educational Technology Development and Exchange (JETDE),,* vol. 4, n. 1.

[16] «Pokemon Go,» [Online]. Available: https://pokemongolive.com/?hl=en.

[17] «StoryboardThat,» [Online]. Available: https://www.storyboardthat.com/.

[18] T. Talbot, K. Thiry e M. Jenkins, in *Storyboarding the Virtuality: Methods and Best Practices to Depict Scenes and Interactive Stories in Virtual and Mixed Reality*, Ahram, T., Falcão, C. (eds) Advances in Usability, User Experience, Wearable and Assistive Technology. AHFE 2020. Advances in Intelligent Systems and Computing, vol 1217. Springer, Cham., 2020, pp. 129-135.

[19] M. SHIN, B.-s. KIM e J. PARK, «AR storyboard: an augmented reality based interactive storyboard authoring tool,» in *Fourth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'05)*, 2005, pp. 198-199.

[20] «Microsoft Hololens 2 – Mixed Reality Technology for Business,» Microsoft, [Online]. Available: https://www.microsoft.com/en-us/hololens/.

[21] A. Singh, R. Kaur, P. Haltner, M. Peachey, M. Gonzalez-Franco, J. Malloch e D. Reilly, «Story CreatAR: a Toolkit for Spatially-Adaptive Augmented Reality Storytelling,» *2021 IEEE Virtual Reality and 3D User Interfaces (VR),* pp. 713-722, 2021.

[22] «Unity 2021.3.14,,» [Online]. Available: https://unity.com/releases/editor/whats-new/2021.3.14 .

[23] Wikipedia Contributors, «Oculus Quest,» Wikipedia, The Free Encyclopedia., 30 Agosto 2023. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Oculus_Quest&oldid=1172944647. [Consultato il giorno 18 Settembre 2023].

[24] «Hololens 2 - Overview, funzionalità e specifiche: Microsoft HoloLens,» [Online]. Available: https://www.microsoft.com/it-it/hololens/hardware.

[25] varoudis, «depthmapX - multi-platform spatial network analyses software,» [Online]. Available: https://github.com/varoudis/depthmapX.

[26] cansik, «AFPlan: Automatic analysis and simplification of architectural floor plans,» [Online]. Available: https://github.com/cansik/architectural-floor-plan.

[27] A. Krestanova, M. Cerny e M. Augustynek, «Review: Development and Technical Design of Tangible User Interfaces in Wide-Field Areas of Application,» *Sensors,* 2021.

[28] K. Rix, «Viability of a tangible tabletop for industrial storyboarding. Honours report, University of Cape Town.,» 2015.

[29] «Autodesk Maya: Create expansive worlds, complex characters, and dazzling effects,» [Online]. Available: https://www.autodesk.com/products/maya/overview?term=1-YEAR&tab=subscription.

[30] «Blender,» [Online]. Available: https://www.blender.org/.

[31] M. Billinghurst, K. Hirokazu e I. Poupyrev, «Tangible augmented reality,» *Acm Siggraph Asia,* pp. 1-10, 2008.

[32] Kawashima, Tagashi e et al, «Magic paddle: A tangible augmented reality interface for object manipulation.,» 2001.

[33] M. EITSUKA e M. HIRAKAWA, «Authoring Animations of Virtual Objects in Augmented Reality-Based 3D Space,» *Proceedings - 2nd IIAI International Conference on Advanced Applied Informatics, IIAI-AAI 2013,* pp. 256-261, 2013.

[34] «Vuforia: Market-Leading Enterprise AR,» [Online]. Available: https://www.ptc.com/en/products/vuforia .

[35] «OpenGL ES, The Standard for Embedded Accelerated 3D Graphics,» [Online]. Available: https://www.khronos.org/opengles/ .

[36] H. Ye, K. Chung Kwan, W. Su e H. Fu, «ARAnimator: In-situ Character Animation in Mobile AR with User-defined Motion Gestures,» *ACM Transactions on Graphics,* vol. 39, 2020.

[37] «Mixamo.com, Animate 3D characters for games, film, and more,» [Online]. Available: https://www.mixamo.com/#/ .

[38] «Apple AR Kit,» [Online]. Available: https://developer.apple.com/augmented-reality/arkit/ .

[39] R. Tenmoku, R. Ichikari, F. Shibata, A. Kimura e H. Tamura, «Mixed reality pre-visualization and camera-work authoring in filmmaking,» pp. 1-7, 2006.

[40] T. Xue, G. Ding e F. Zhang, «Augmented Reality-Based Pre-visualization for Film Making: Proceeding of the Second International Conference on Smart Vehicular Technology, Transportation, Communication and Applications, October 25-28, 2018 Mount Emei, China, Part 2,» pp. 217-222, 2019.

[41] M. Scarzello, «Advanced Storyboard: Generazione automatica di storyboard mediante il controllo diretto dei personaggi,» Politecnico di Torino, Corso di laurea magistrale in Ingegneria Del Cinema E Dei Mezzi Di Comunicazione, Torino, 2022.

[42] «Google Form,» [Online]. Available: https://www.google.com/forms/about/.

[43] «MoSCoW Prioritization,» [Online]. Available: https://www.productplan.com/glossary/moscow-prioritization/.

[44] «Imagine, create, grow, do more, imagine with Unity. Unity Real-Time Development Platform | 3D, 2D, VR & AR Engine.,» [Online]. Available: https://unity.com/.

[45] [Online]. Available: https://unity.com/solutions/multiplatform.

[46] [Online]. Available: https://unity.com/solutions/programming.

[47] [Online]. Available: https://docs.unity3d.com/Manual/UsingTheEditor.html.

[48] «Visual studio, Editor di Codice e IDE per sviluppatori e team software,» [Online]. Available: https://docs.unity3d.com/Manual/UsingTheEditor.html.

[49] [Online]. Available: https://learn.microsoft.com/it-it/visualstudio/gamedev/unity/get-started/visual-studio-tools-for-unity?pivots=windows .

[50] [Online]. Available: https://docs.unity3d.com/ScriptReference/MonoBehaviour.html .

[51] «Microsoft,» [Online]. Available: https://www.microsoft.com/.

[52] lolambean, beelia e S. Paniagua, «HoloLens (1st gen) hardware,» Microsoft, 23 Novembre 2021. [Online]. Available: https://learn.microsoft.com/en-us/hololens/hololens1-hardware.

[53] «MRTK packages — MRTK2,» [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/packages/mrtk-packages?view=mrtkunity-2022-05.

[54] «Documentazione per sviluppatori MRTK2-unity - MRTK 2 | Microsoft Learn,» [Online]. Available: https://learn.microsoft.com/it-it/windows/mixed-reality/mrtk-unity/mrtk2/?view=mrtkunity-2022-05 .

[55] S. Kerawala, alexbuckgit, v-hearya, hferrone e T. Sherer, «Introducing instinctual interactions,» Microsoft, 22 Settembre 2022. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/design/interaction-fundamentals.

[56] S. Kerawala e T. Sherer, «Hands and motion controllers,» 03 Ottobre 2022. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/design/hands-and-tools.

[57] S. Kerawala e T. Sherer, «Hands-free,» Microsoft, 10 Marzo 2022. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/design/hands-free.

[58] Microsoft Learn Contributors, «Gaze and commit,» Microsoft, 2023 Marzo 03. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/design/gaze-and-commit.

[59] «Welcome to MRTK, What is Mixed Reality Toolkit 2?,» [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/?view=mrtkunity-2022-05.

[60] «Configurare un progetto di realtà mista in Unity con Mixed Reality Toolkit,» [Online]. Available: https://learn.microsoft.com/it-it/training/modules/mixed-reality-toolkit-project-unity/.

[61] «Welcome to the Mixed Reality Feature Tool,» [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/develop/unity/welcome-to-mr-feature-tool.

[62] Karl-Bridge-Microsoft, «Windows device portal overview - UWP applications,» UWP applications | Microsoft Learn, [Online]. Available: https://learn.microsoft.com/en-us/windows/uwp/debug-test-perf/device-portal.

[63] «Toolbox — MRTK2,» [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/toolbox?view=mrtkunity-2022-05.

[64] «TextMeshPro,» [Online]. Available: https://docs.unity3d.com/Manual/com.unity.textmeshpro.html.

[65] «AI Navigation,» [Online]. Available: https://docs.unity3d.com/Packages/com.unity.ai.navigation@1.1/manual/index.html.

[66] «Input System,» [Online]. Available: https://docs.unity3d.com/Packages/com.unity.inputsystem@1.7/manual/index.html.

[67] «Sketchfab,» [Online]. Available: https://sketchfab.com/feed.

[68] «Turbosquid,» [Online]. Available: https://www.turbosquid.com/it/.

[69] «Free3D,» [Online]. Available: https://www.turbosquid.com/it/.

[70] «Mixamo,» [Online]. Available: https://www.mixamo.com/#/.

[71] «Adobe,» [Online]. Available: https://www.adobe.com/.

[72] M. Wang, vtieto e qianw211, «Spatial awareness getting started — MRTK2,» 8 Febbraio 2022. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/spatial-awareness/spatial-awareness-getting-started?view=mrtkunity-2022-05.

[73] CDiaz-MS, MaxWang-MS, vtieto e qianw211, «Tap to place — MRTK2,» 8 Febbraio 2022. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/solvers/tap-to-place?view=mrtkunity-2022-05.

[74] «NearInteractionGrabbable Class,» [Online]. Available: https://learn.microsoft.com/en-us/dotnet/api/microsoft.mixedreality.toolkit.input.nearinteractiongrabbable?view=mixed-reality-toolkit-unity-2020-dotnet-2.8.0.

[75] thalbern, thalbern, vtieto e qianw211, «Object manipulator — MRTK2,» 22 Settembre 2022. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/object-manipulator?view=mrtkunity-2022-05.

[76] CDiaz-MS, MaxWang-MS, vtieto e qianw211, «Solver overview — MRTK2,» 2 Agosto 2022. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/solvers/solver?view=mrtkunity-2022-05.

[77] thalbern, MaxWang-MS, vtieto e qianw211, «Bounds control — MRTK2,» 2 Agosto 2022. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/bounds-control?view=mrtkunity-2022-05.

[78] vaoliva, MaxWang-MS, vtieto e qianw211, «Scrolling object collection — MRTK2,» 2 Agosto 2022. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/scrolling-object-collection?view=mrtkunity-2022-05.

[79] cre8ivepark, keveleigh, MaxWang-MS, vtieto e qianw211, «Hand menu — MRTK2,» 4 Novembre 2022. [Online]. Available: https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/hand-menu?view=mrtkunity-2022-05.

[80] W. contributors, «Navigation mesh,» Wikipedia, The Free Encyclopedia., 19 Agosto 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Navigation_mesh&oldid=1105252354.

[81] «Animator,» Unity, [Online]. Available: https://docs.unity3d.com/Manual/class-Animator.html.

[82] TechTargetContrbutor, «Finite state machine,» Settembre 2019. [Online]. Available: https://www.techtarget.com/whatis/definition/finite-state-machine.

[83] «Camera Component,» Unity Documentation, [Online]. Available: https://docs.unity3d.com/2021.3/Documentation/Manual/class-Camera.html.

[84] «WordNet: A Lexical Database for English,» Princeton University, [Online]. Available: https://wordnet.princeton.edu/.

[85] «Physics.Raycast,» Unity Documentation, [Online]. Available: https://docs.unity3d.com/ScriptReference/Physics.Raycast.html.

[86] «ChatGPT,» OpenAI, [Online]. Available: https://openai.com/blog/chatgpt.

[87] «Directory,» Unity Documentation, [Online]. Available: https://docs.unity3d.com/ScriptReference/Windows.Directory.html.

[88] «File,» UnityDocumentation, [Online]. Available: https://docs.unity3d.com/ScriptReference/Windows.File.html.

[89] «Streaming Assets,» Unity Documentation, [Online]. Available: https://docs.unity3d.com/Manual/StreamingAssets.html.

[90] M. Trapella, «Le 10 euristiche di Nielsen per lo UI design,» trapstudio, [Online]. Available: https://www.trapstudio.it/ui-design-le-10-euristiche-di-nielsen/.

[91] Wikipedia Contributors, «Scala Likert,» Ottobre 2013. [Online]. Available: https://it.wikipedia.org/wiki/Scala_Likert.

[92] «Canva.com, Modelli per storyboard,,» [Online]. Available: https://www.canva.com/storyboards/templates/.

[93] «miro.com, Storyboard Template,» [Online]. Available: https://miro.com/templates/storyboard/.

[94] «Microsoft,» [Online]. Available: https://www.microsoft.com/it-it/.

[95] «Meta Quest 2,» [Online]. Available: https://www.meta.com/it/quest/products/quest-2/.

[96] [Online]. Available: https://docs.unity3d.com/Manual/UsingTheEditor.html.

[97] «Hardware HoloLens (prima generazione),» [Online]. Available: https://learn.microsoft.com/it-it/hololens/hololens1-hardware.

[98] M. Gharbharan, StoryTIME: Development of a Tangible Interface for Storytelling, University of Ontario Institute of Technology, 2018.

[99] «DepthMap X,» [Online]. Available: https://github.com/varoudis/depthmapX.

[100] «Automatic analysis and simplification of architectural floor plans,» [Online]. Available: https://github.com/cansik/architectural-floor-plan.

[101] «Mixamo,» [Online]. Available: https://www.adobe.com/.