UNIVERSITÀ DI PISA

DIPARTIMENTO DI

FILOLOGIA, LETTERATURA E LINGUISTICA

CORSO DI LAUREA MAGISTRALE IN INFORMATICA UMANISTICA

TESI DI LAUREA

Progettazione e sviluppo di uno strumento di Realtà Virtuale Immersiva per sensibilizzare sull'inclusività e per favorire l'abbattimento delle barriere architettoniche

CANDIDATO

Chiara Zamberti

RELATORE

Prof. Marcello Carrozzino

CORRELATORE

Dott. Riccardo Galdieri

CONTRORELATORE

Prof.ssa Enrica Salvatori

ANNO ACCADEMICO 2020/2021

Indice

«The fact that all of this was happening in virtual space made no difference. Being virtually killed by virtual laser in virtual space is just as effective as the real thing, because you are as dead as you think you are.»

Douglas Adams, *Mostly Harmless*

Introduzione

Il compito dell'elaborato è illustrare e proporre uno strumento destinato a persone normodotate, per sensibilizzare sul tema delle barriere architettoniche e sui problemi ad esse relativi, influenti in particolare sulla classe di soggetti portatori di disabilità. Come *focus* del progetto sono state individuate le disabilità motorie, in particolare quelle che interessano gli arti inferiori e costringono i portatori all'ausilio di carrozzine e supporti additivi per poter permettere loro di godere di un minimo di autonomia, almeno nello svolgimento delle funzioni vitali essenziali. Innanzitutto, è necessario dare una definizione di disabilità: la maggior parte delle persone conosce a grandi linee l'argomento ed è pronta a enunciarne alcuni tipi e a stilarne le caratteristiche, ma non ne comprende tutti i dettagli e, in particolare, non riesce ad immergersi nell'esperienza che tali soggetti provano quotidianamente, soprattutto se messi di fronti ad azioni che per loro possono costituire una vera e propria sfida. Fino a pochi decenni fa[1], essa era considerata solo nel suo aspetto di "limitazione insita" dell'individuo ed era posta sullo stesso piano di altre patologie cliniche, su cui intervenire medicalmente. Ad oggi, viene applicato il *Modello Sociale della Disabilità*[2] come paradigma (quasi) universale per guardare al mondo della disabilità, creato dagli stessi portatori; secondo quest'ultimo, il termine preso in esame si può sintetizzare come "il risultato di un'interazione tra il livello di limitazione individuale fisica o sensoriale o cognitiva o mentale e il contesto di vita". Tale archetipo contiene le informazioni necessarie per l'integrazione di tali soggetti all'interno della società, a partire dai comportamenti che i normodotati sono tenuti ad adottare nei loro confronti, al fine di evitare ogni forma di discriminazione. Un problema trattato nel modello è quello delle barriere architettoniche, da sempre fonte di disagi per la maggior parte delle categorie di disabili, in particolare per quelle dello spettro motorio. Infatti, tali soggetti, sono costretti ad un'autonomia limitata e spesso le infrastrutture tendono a rendere la loro vita più complessa: si pensi ad un ufficio postale privo di rampe per l'accesso facilitato o ad una scuola senza sanitari per persone con disabilità motorie. Prendendo in esame il problema più da vicino, è possibile stimare in Italia – al momento – 288.591 portatori di disabilità motoria di diverso tipo, di cui 240.948 maschi

[1] *I numeri delle disabilità in Italia*. 27 marzo 2020. Lenus.it. Redazione di Maria Chiara Paolini. Link all'articolo: https://www.lenius.it/disabilita-in-italia/.
[2] *Social model of disability*, 2020. Scope – Equality for Disabled People. Link al sito: https://www.scope.org.uk/about-us/social-model-of-disability/.

e 47.643 femmine, secondo i dati relativi ai titolari di rendita INAIL[3]: un numero alto, che non considera le persone non censite (ovvero quelle che non percepiscono alcun reddito di invalidità). Tale valore invita a riflettere sulla necessità di proporre più soluzioni possibili per favorire l'integrazione di queste persone nel mondo circostante, creato ad immagine di soggetti normodotati. Fortunatamente, esistono numerosi espedienti che, attualmente, mirano a rendere più semplice la vita di chi presenta delle disabilità motorie: basti pensare alle opere architettoniche create su misura, all'introduzione delle più moderne tecnologie assistive negli più svariati campi e all'avvento della domotica all'interno delle abitazioni. Nei capitoli a seguire verranno illustrati alcuni studi e strumenti attualmente esistenti, che si pongono l'obiettivo di sensibilizzare il pubblico su determinate problematiche sollevate, con l'obiettivo di migliorare la vita di alcune delle categorie più svantaggiate. Nel presente elaborato, è presa in considerazione la disabilità motoria agli arti inferiori e, a tal proposito, vengono proposte delle soluzioni atte a sensibilizzare sull'integrazione, sociale e fisica, dei soggetti che ne sono portatori. Lo strumento ideato è stato costruito in seguito alla provata necessità di ottenere un prodotto che utilizzasse le più moderne tecnologie, come quella della Realtà Virtuale, per dare rilievo alla problematica, facendola conoscere e scoprire in modo quanto più realistico. Il prodotto ottenuto è un simulatore di Realtà Virtuale (d'ora in poi, anche chiamata RV) immersiva, fruibile tramite un visore di RV, collegato ad un PC con buone prestazioni. Tramite questo tipo di esperienza, l'utente viene posto di fronte alle principali problematiche inerenti il mondo di chi vive con supporti per la deambulazione e, alternandosi tra il ruolo di un giocatore normodotato e di uno con disabilità motorie, costretto su una sedia a rotelle, potrà essere in grado di percepire il livello di difficoltà di ogni azione compiuta. Il prodotto è proposto come uno strumento costruito seguendo una chiave videoludica: il meccanismo è basato sul superamento di alcune missioni incentrate su normali azioni quotidiane, semplicissime per un soggetto senza alcuna limitazione, ma che possono essere davvero articolate e complesse da svolgere in caso della mancanza di un'adeguata idoneità fisica. Delle specifiche del prodotto presentato e di eventuali soluzioni se ne tratterà nei prossimi capitoli, ponendo l'accento anche su eventuali ausili e strumenti per l'inclusività attualmente oggetto di studio.

[3] *Banca dati disabili. Disabilità motoria. Disabili titolari di rendita INAIL*, 31 dicembre 2020. INAIL. Link al sito: http://apponline.inail.it/DisabiliApp/Italia.do?disabilita=M&sesso=T.

1 Le barriere architettoniche: contesto e casi di studio

Negli anni, il mondo delle infrastrutture ha subito molte variazioni e, in linea con l'evoluzione umana, sono state introdotte novità che hanno puntato a migliorare la fruibilità di determinati luoghi da parte degli utenti. Nonostante ciò, il tema dell'accessibilità resta sempre attuale poiché, nonostante vengano implementati numerosi elementi atti ad arricchire e modernizzare le strutture pubbliche e private, viene spesso posta in risalto la mancata adattività di tali prodotti e, di conseguenza, i luoghi dove vengono installati diventano poco o del tutto inaccessibili a determinate categorie di persone. È necessario fare una distinzione tra accessibilità e adattività: non tutto ciò che è accessibile può essere adattivo e viceversa. Per *accessibilità*[4] si intende la facoltà di accedere ad un luogo o ad una risorsa; il termine *adattività*[5], invece, indica, in psicologia e in etologia, la capacità di favorire il processo di adattamento. Dunque, proponendo un esempio, un ufficio postale può essere ritenuto un luogo accessibile, poiché dotato di personale esperto, preparato a soddisfare le richieste dei clienti; tuttavia, tale luogo potrebbe non essere adattivo, in quanto non a favore dell'integrazione dell'utenza non italiana, per la mancanza di cartellonistica nelle principali lingue straniere e forza lavoro con buona conoscenza dell'inglese. Nonostante le due espressioni possano non sempre essere legate tra loro, esse riescono a delineare al meglio i parametri di un certo luogo, infrastruttura o interfaccia e aiutare la figura incaricata al suo *design* e/o realizzazione a stilarne le caratteristiche per renderli fruibili da più utenti possibili. Nel presente elaborato verrà trattato soprattutto il concetto di accessibilità, legato in particolare alle barriere architettoniche, spesso invalicabili da alcune categorie di fruitori e oggetto di numerose problematiche e discussioni. Esse, infatti, possono costituire un problema non solo per i portatori di disabilità dello spettro motorio (di cui si tratterà nei prossimi capitoli), ma anche per bambini, anziani e coloro che presentano determinate patologie cliniche[6]; ogni individuo è singolare per aspetto e forma e, in quanto tale, uno strumento standard deve essere "pronto" ad aprirsi ad ogni possibilità di utenza.

[4] Treccani, voce *Accessibilità*. Link alla pagina: treccani.it/enciclopedia/accessibilita/.
[5] Treccani, voce *Adattivo*. Link alla pagina: https://www.treccani.it/vocabolario/adattivo/.
[6] *Disabili e Barriere Architettoniche.* Fondazione Cesare Serono. Link alla pagina: https://www.fondazioneserono.org/disabilita/disabilita-diritti-e-normativa/disabili-e-barriere-architettoniche/disabilita-e-diritti-disabili-e-barriere-architettoniche/.

In Italia esistono alcuni emendamenti che regolamentano la fruizione delle infrastrutture e trattano il problema delle barriere architettoniche. La legge 13/1989[7], tra le più importanti nel contesto citato, affronta il problema delle barriere architettoniche negli edifici privati. Sebbene ogni essere umano sia libero di costruire un'abitazione seguendo delle regole standard e il proprio gusto personale, un edificio che deve essere facilmente raggiunto da chi potrebbe non averne la facoltà fisica deve rispondere, secondo la normativa, di tre requisiti fondamentali: accessibilità, visibilità e adattabilità. Il concetto di accessibilità è quello enunciato in precedenza: un essere umano deve essere in grado di raggiungere un determinato luogo. Per visibilità si intende, invece, la possibilità per la persona bisognosa di ausili speciali di accedere agli spazi di relazione e ad almeno un servizio igienico. Infine, ma non meno importante, è citato il termine "adattabilità": il significato è analogo a quello di adattività visto precedentemente ma, in questo caso, vi è un riferimento più esplicito al fattore tempo, poiché indica la facoltà dell'infrastruttura di modificare le caratteristiche attuali in un secondo momento, in seguito ad una richiesta di adattamento. Un esempio di struttura che segue il concetto di *adattabilità* (in questo caso, riferito ad un'infrastruttura pubblica) potrebbe essere una scuola che, inizialmente sprovvista di rampe per l'accesso facilitato e di montascale, esegue dei lavori di ristrutturazione in seguito all'iscrizione di un alunno portatore di paraplegia agli arti inferiori, per l'aggiunta di strumenti per permettere al soggetto di accedere agli ambienti in una condizione di autonomia o semiautonomia. La legge di cui sopra prevede, inoltre, la concessione di contributi per l'eliminazione di barriere architettoniche dalla propria abitazione o da spazi condominiali, nel caso specifico di utenza disabile; è inoltre prevista, a tal proposito, una detrazione *IRPEF* per la ristrutturazione edilizia, della quale questi ultimi possono beneficiare.

La normativa che disciplina, invece, le caratteristiche per rendere accessibili e adattivi spazi ed edifici pubblici è contenuta nel D.P.R. 503/96[8], nel quale sono trattati i requisiti da rispettare nella costruzione di parcheggi, arredo urbano, scale e rampe, segnaletica stradale, marciapiedi; al suo interno, sono inoltre presenti disposizioni per la definizione di spazi

[7] *Legge 13 1989: superamento delle barriere architettoniche*. Contact SRL. Link alla pagina: https://www.contactsrl.it/normative/legge-13-89/.
[8] *Regolamento recante norme per l'eliminazione delle barriere architettoniche negli edifici, spazi e servizi pubblici*. DPR 24 luglio 1996, n. 503. Normattiva. Link alla pagina: https://www.normattiva.it/uri-res/N2Ls?urn:nir:presidente.repubblica:decreto:1996;503.

riservati a determinate categorie di utenti e le istruzioni sulla relativa fruizione. Nel medesimo decreto, nell'articolo 23, sono contenuti i requisiti necessari al processo di abbattimento delle suddette barriere negli istituti scolastici, in riferimento, oltre che all'organizzazione delle strutture interne ed esterne, anche all'arredamento, ai sussidi e alle attrezzature necessarie ad assicurare lo svolgimento delle attività didattiche nel miglior comfort ottenibile. Uno dei punti cruciali è quello in cui si trattano le specifiche che devono possedere tali ambienti e strumenti in relazione a differenti tipi di invalidità, fisica e/o mentale. In presenza di uno studente non vedente o ipovedente, ad esempio, le aule devono essere attrezzate, secondo regolamento, con dispositivi di lettura e punzonatura in linguaggio Braille; con il progresso tecnologico, come si tratterà anche più avanti, le più moderne tecnologie assistive per la didattica[9] hanno rimpiazzato parte degli strumenti analogici indicati nella normativa presa in esame. Allo scopo di favorire il processo di eliminazione delle barriere, sono state emanate anche numerose leggi regionali, nel cui campo di applicazione rientrano anche edifici destinati ad attività commerciali e produttive, dei settori industriale, agricolo, artigianale e del terziario. Ad esempio, relativamente a quest'ultimo, esistono delle linee guide specifiche della regione Emilia Romagna, relative ai luoghi di interesse culturale[10] atte a rendere accessibili i principali luoghi di interesse turistico e non, con un occhio di riguardo al rispetto del territorio e degli ambienti naturali.

Durante gli anni Novanta, in particolar modo dalla seconda metà, è stato possibile osservare una progressiva informatizzazione di innumerevoli strumenti, conseguenza della così chiamata "rivoluzione digitale"[11], durante la quale anche soltanto l'avvento dei PC in edifici pubblici e privati ha apportato considerevoli miglioramenti alla vita quotidiana. La crescita, tuttora continua e divenuta esponenziale durante i primi anni del Duemila, per poi appiattirsi progressivamente in una sorta di *plateau*, ha determinato il progressivo distacco dall'analogicità nei campi più disparati. È impossibile stilare un elenco organico di tutti i cambiamenti avvenuti, ma al giorno d'oggi è possibile ritrovare uno strumento che fino a

[9] *Tecnologie assistive*. Abili a proteggere. Link alla pagina: https://www.abiliaproteggere.net/informazioni-utili/tecnologie-assistive/.
[10] Commissione per l'analisi delle problematiche relative alla disabilità nello specifico settore dei beni e delle attività culturali. 16 maggio 2008. *Linee guida per il superamento delle barriere architettoniche nei luoghi di interesse culturale*. Supplemento ordinario alla Gazzetta Ufficiale. Allegato A. Link alla versione in PDF: https://www.gazzettaufficiale.it/eli/gu/2008/05/16/114/so/127/sg/pdf.
[11] Cellini, Paolo. 2018. *La rivoluzione digitale. Economia di internet dallo Sputnik al machine learning*. Introduzione e Capitolo 1, pp 3-22.

dieci anni fa era all'avanguardia e che oggi i nativi digitali potrebbero non conoscerne nemmeno l'impiego: basta prendere come esempio il *VHS* o il *floppy disk*, supporti utilizzati fino ai primi anni Duemila e caduti in disuso parallelamente all'avvento di tecnologie più efficienti in termini di tempo e risorse. Più in generale, negli anni si è assistito ad un progressivo e "morbido" cambiamento degli strumenti atti a facilitare la vita quotidiana e, giorno dopo giorno, non se ne è più riusciti a fare a meno. Allo stesso tempo, molti hanno demonizzato il cambiamento, criticando l'avanzamento tecnologico come Giosuè Carducci fece, in riferimento all'invenzione del treno, nel suo celebre *A Satana*[12]. Nonostante l'esistenza di coloro che apostrofano la digitalizzazione come fautrice di distruzione della naturale manualità umana, il progresso tecnologico ha modificato drasticamente e positivamente la vita di svariate persone; il suo impatto ha infatti determinato una vera e propria rinascita di alcune fasce di popolazione che sono rimaste per millenni intrappolate in una situazione che non permetteva loro di vivere in una pseudo-normalità. Con l'avvento delle più moderne innovazioni, la maggior parte dei mezzi impiegati su larga scala si è evoluta, diventando sempre più fruibile e alla portata di ogni fascia: si è potuto assistere ad una progressiva conversione di supporti analogici in equivalenti strumenti elettronici ed è nato il concetto di interfaccia[13] digitale. In particolare, sono state progettate numerose versioni *software* di oggetti fisici che, progressivamente, hanno sostituito il mezzo originario, in particolare per la loro maggiore fruibilità da più supporti. Mettere a disposizione una versione *software* di un certo strumento è stato un passo importantissimo nel mondo dell'accessibilità e dell'usabilità[14]: in particolare, è possibile citare il forte impatto nel mondo dell'istruzione e dell'educazione: digitalizzare un apparato fisico, non solo ha permesso di ridurre il numero di risorse da impiegare durante, ad esempio, lo svolgimento di una lezione frontale, ma ha anche facilitato la fruizione dei contenuti da parte di fasce d'utenza più fragili. Riguardo queste ultime, l'impiego di determinate tecnologie assistive ne ha favorito l'integrazione in innumerevoli ambienti e situazioni: queste hanno permesso, infatti, l'abbattimento ulteriore delle suddette barriere, architettoniche e non solo. Il concetto di "barriera", infatti, può essere ritenuto sinonimo di "limite": non solo ciò che impedisce uno spostamento fisico – ed è dunque d'intralcio – è etichettabile come tale, ma anche tutto ciò

[12] Carducci, Giosuè. *A Satana*. 1865. Riproduzione e parafrasi a cura del progetto OperaOmnia. Link alla pagina: http://carducci.letteraturaoperaomnia.org/carducci_inno_a_satana.html.
[13] Treccani, voce *Interfaccia*. Link alla pagina: https://www.treccani.it/vocabolario/interfaccia/.
[14] Treccani, voce *Usabilità*. Link alla pagina: https://www.treccani.it/enciclopedia/usabilita/.

che mette in difficoltà una persona nello svolgimento di un compito, sia esso creativo o intellettuale. Valicando tali ostacoli, più esseri umani sono riusciti ad ottenere le stesse opportunità: dal bambino che ha potuto apprendere meglio un concetto troppo complesso, tramite l'estensione multimediale in allegato al proprio libro scolastico, al portatore di *Disturbi dello Spettro Autistico* (DSA) che ha potuto, finalmente, usare la *Comunicazione Aumentativa e Alternativa* (*CAA*)[15] sul proprio tablet per esprimersi, allo studente non in grado di deambulare, al quale è stato dedicato un montascale al fine di donargli autonomia.

Nei prossimi capitoli sarà, tuttavia, trattato esclusivamente il tema delle tecnologie e metodologie che permettono di abbattere le barriere architettoniche; ci si occuperà dunque delle limitazioni che interessano l'integrazione fisica di un soggetto nel mondo circostante. A tal proposito, verranno esaminate alcune delle strategie esistenti e saranno proposte ulteriori possibili soluzioni a favore dell'integrazione di determinate fasce di utenza in situazioni poco accessibili. Per l'analisi, verrà considerata la categoria più colpita dal problema della mancanza di strutture facilmente accessibili: i portatori di disabilità motoria, in particolare coloro che presentano difficoltà – dovute a fattori congeniti, patologie, anzianità o ad infortuni – che interessano la mobilità delle gambe e li costringono all'ausilio di dispositivi mobili su ruote, conosciuti come carrozzine o, più semplicemente, sedie a rotelle. Tali soggetti, di fronte ad un ambiente creato esclusivamente su misura di persone normodotate, non possono godere di autonomia: devono, almeno nella maggior parte dei casi, essere continuamente assistiti da un *caregiver* che li aiuti a portare a termine anche le più semplici mansioni. Per tali categorie, anche un semplice gradino o dislivello può essere un importante ostacolo e, per tale ragione, possono risentirne gravemente sul lato psicologico; è infatti noto che una cattiva esperienza, sfocia quasi sempre in un sentimento di frustrazione. Un individuo che presenta invalidità simili può avere, dunque, come sogno, ciò che per un normodotato può essere scontato: godere di autonomia. Ma come è possibile realizzare tale desiderio, se il mondo è ancora colmo di ostacoli, malgrado alcuni progressi (in particolare, nelle infrastrutture pubbliche)? Bisogna sensibilizzare, allo stesso modo delle iniziative di prevenzione delle malattie: la divulgazione di informazioni costituisce un'efficace arma, che

[15] *Cosa e' la CAA - Comunicazione Aumentativa Alternativa*. *Un modello da diffondere*. Ottobre 2018. A cura di Maria Caterina Minardi. FareLeggereTutti. Link alla pagina: https://www.fareleggeretutti.it/cosa-e-la-caa-comunicazione-aumentativa-alternativa.

può ampliare lo spettro della conoscenza e aiutare l'essere umano a tamponare i problemi evidenziati e, nei casi più fortunati, può condurre alla loro definitiva eliminazione.

La domanda che può fungere da *starter* per affrontare l'argomento è: quanto è davvero preparato l'uomo su questa tematica? Spesso, infatti, la mancanza di informazione è causa di problemi: se l'individuo non ha un certo *background* su una determinata disciplina o fatto, non riesce a notarne i problemi. Dunque, è possibile porre una rampa all'ingresso di un centro commerciale, installare un montascale in azienda o affittare un'abitazione a piano terra o constatare l'esistenza di tali sussidi e strategie, ma non conoscere quanto questi possano essere, effettivamente, efficaci a migliorare la vita di una determinata tipologia di individuo. E soprattutto, è difficile determinare se i concetti di accessibilità e adattività siano stati rispettati e se tali strutture risultino in linea con le norme vigenti: non è infatti raro trovarne di incomplete o malfatte, come nel caso di alcuni scivoli di accesso a locali di pubblico interesse[16]. Tali mancanze sono però, purtroppo, denunciate spesso dal personale incaricato ai controlli o, ancora più gravemente, dagli stessi fruitori che possono scoprirne le debolezze in seguito a dolorosi infortuni[17]. In casi analoghi, sono i soggetti normodotati a subirne le conseguenze; non è raro, infatti, che gli stessi strumenti pensati per facilitare la vita degli altri, diventino d'intralcio per altre categorie di utenti, come riportato in un vecchio articolo di Quindici, quotidiano del comune di Molfetta[18]. In altre parole, è sempre necessario bilanciare le regole che disciplinano la realizzazione di determinate opere architettoniche con l'effettiva esperienza utente: è importare testare a lungo e "vivere" quanto più possibile tutto ciò che viene installato nell'ambiente circostante, in modo da marcarne eventuali mancanze ed evitare che si verifichino eventi spiacevoli. In altre parole, è necessario informare su quanto è importante inserire specifici ausili in ambienti pubblici o privati, ma è altrettanto fondamentale istruire gli utenti sul corretto utilizzo degli stessi, marcando anche quanto, effettivamente, questi possano essere omogeneizzati idoneamente al territorio.

[16] *Lo scivolo per disabili senza corrimano è a norma?* 6 Gennaio 2017. A cura di Carlos Arija Garcia. La Legge per tutti. Link alla pagina: https://www.laleggepertutti.it/143687_lo-scivolo-per-disabili-senza-corrimano-e-a-norma.
[17] *Rampa troppo ripida, disabile in carrozzella cade.* 28 aprile 2017. Il Resto del Carlino, edizione per la città di Pesaro. Link alla pagina: https://www.ilrestodelcarlino.it/pesaro/cronaca/disabile-ferito-san-lazzaro-1.3073457.
[18] *Molfetta, un lettore scrive a "Quindici": il marciapiede per i disabili di Corso Dante fa inciampare i pedoni*. 24 agosto 2010. Quindici, edizione per la città di Molfetta. Link alla pagina: http://www.quindici-molfetta.it/molfetta-un-lettore-scrive-a-quindici-il-marciapiede-per-i-disabili-di-corso-dante-fa-inciampare-i-pedoni_20140.aspx.

Tornando alle domande poste in precedenza, è dunque necessario creare informazione per accedere al primo passo verso un ipotetico miglioramento: se l'uomo non conosce bene un problema, non può che continuare a trascurarlo.

Fortunatamente, come accennato, esistono delle campagne di sensibilizzazione che possono venire in soccorso; è importante destinarle ad ogni fascia d'utenza e raccogliere tanti *feedback*, per comprendere gli eventuali disagi manifestati. Tali informazioni, infatti, magari frutto di disagi provati in seguito alla fruizione di determinate strutture del territorio, possono essere il trampolino di lancio per la definizione di nuove soluzioni; ascoltare la voce del popolo è sempre importante e solo coinvolgendolo attivamente e istruendolo è possibile giungere alla definizione di strategie risolutive. Nella prossima sezione saranno illustrati alcuni progetti attualmente in sviluppo che trattano la sensibilizzazione sul problema delle barriere architettoniche e verranno analizzati i relativi risultati ai quali si è giunti tramite le campagne ad essi correlate, in modo da decantarne i punti di forza e marcarne le debolezze in modo quanto più possibile oggettivo.

1.1 Iniziative di sensibilizzazione

Nella presente sezione verranno analizzati alcuni progetti promossi da enti o associazioni che hanno tra gli obiettivi la sensibilizzazione sull'abbattimento delle barriere, tra cui quelle architettoniche, studiati in itinere alla ricerca effettuata a priori della realizzazione del simulatore che verrà presentato, il quale assolve allo scopo in esame. Di ognuno di essi, verranno decantati i punti a favore e, al contempo, saranno sottolineate eventuali lacune presenti o insuccessi dovuti ad una determinata strategia impiegata.

1.1.1 FIABA Onlus

La prima associazione di cui è importante scrivere è FIABA (Fondo Italiano per l'Abbattimento delle Barriere Architettoniche)[19], una fondazione ONLUS (Organizzazione Non Lucrativa di Utilità Sociale) no profit, che nasce per sensibilizzare sull'importanza dell'abbattimento delle barriere, sia architettoniche che culturali. L'organizzazione, nata nel 2000 per opera dell'attuale presidente Giuseppe Trieste, si pone come obiettivo la creazione delle opportune condizioni che possano permettere, in un futuro non troppo distante, la libertà di movimento, in ogni luogo, di persone con disabilità, assicurando loro i dovuti *comfort* e

[19] Fiaba ONLUS. Link al sito web: https://www.fiaba.org/.

autonomia. Come cita l'associazione stessa, infatti, spesso spazi e ambienti risultano ancora inaccessibili e non è facile deambularvi e, in particolar modo, accedervi se in possesso di ausili, come sedie a rotelle, deambulatori o tutori ortopedici. "È l'ambiente a creare gli ostacoli", ed è per tale ragione che è necessario eliminare tutto ciò che possa costituire una barriera per raggiungere un determinato obiettivo, sia esso anche piuttosto scontato o trascurabile. L'attenzione di FIABA si pone in particolare al mondo delle disabilità dello spettro motorio e propone la progettazione di ambienti fisici accessibili nella loro totalità, secondo i principi dell'*Universal Design*[20], le linee guida che regolamentano la progettazione dedicata a tutte le fasce di utenza e pongono l'accento sulle esigenze che ciascuna di questa può manifestare, interfacciandosi con un determinato ambiente o una specifica circostanza. I valori chiave dell'associazione possono concretizzarsi nei termini di accessibilità, attenzione alle diversità, competenza ed esperienza, sui quali FIABA pone le fondamenta per il raggiungimento dei suoi obiettivi. È importante ricapitolare le principali pietre miliari raggiunte da questa Onlus, gli importanti traguardi dell'associazione che hanno lasciato traccia e hanno favorito la sensibilizzazione sulla calda tematica dell'importanza dell'abbattimento di barriere di ogni genere. Tra le più significative, è importante citare l'istituzione del FIABADAY, la Giornata Nazionale per l'Abbattimento delle Barriere Architettoniche, una collezione di eventi che, dal 2003, si svolge ogni prima domenica di ottobre e prevede visite guidate riservate a persone con disabilità - e ai relativi accompagnatori - a Palazzo Chigi. In questa occasione vengono organizzati comizi relativi all'inclusione e al superamento di pregiudizi di ogni genere; dal 2009 lo svolgimento di questi ultimi ha luogo su un palco costruito in Piazza Colonna. A partire dal 2012, FIABA organizza anche concorsi a livello nazionale per educare sull'accessibilità e, dal 2014, collabora attivamente con le aziende per assicurare l'accessibilità di spazi e servizi a favore delle persone a ridotta mobilità. Inoltre, dal 2016, collabora con il MIUR[21] come ente formatore per i docenti di ogni ordine e grado; tale iniziativa è ritenuta di fondamentale importanza poiché è riuscita a dare solide basi comportamentali e legislative a chi ogni giorno si trova di fronte a coloro che possono richiedere bisogni educativi speciali. Nelle scuole, è infatti importante istruire attivamente sul tema dell'inclusione, poiché è un luogo di aggregazione

[20] *Che cos'è l'Universal Design e a chi i rivolge.* Architutti. Link alla pagina: https://www.architutti.it/che-cose-luniversal-design/.
[21] Ministero dell'Istruzione, dell'Università e della Ricerca. Link al sito web: https://www.miur.gov.it/.

dove non è insolito trovare discriminazione e, non conoscendo come poterla evitare, vengono spesso a crearsi episodi spiacevoli, dettati dall'intolleranza verso la diversità. Oltre al tema dell'inclusione sociale, l'associazione collabora con le scuole per abbattere le barriere fisiche e coinvolge aziende che possono rendere perfettamente accessibili gli istituti, con il pieno sostegno di docenti e alunni con particolari esigenze. A proposito del tema della progettazione, nel 2017 FIABA ha presentato una normativa dedicata: la Uni/PdR24:2016[22], che contiene le *Linee guida per la riprogettazione del costruito in ottica Universal Design*, uno statuto fine alla resa accessibile di infrastrutture pubbliche e private.

Al giorno d'oggi, l'associazione si occupa di migliorare la vita delle persone con disabilità o che presentano, in generale, disagi nell'interfacciarsi sia con la società che con l'ambiente e, per sensibilizzare sul tema, non solo organizza incontri con i responsabili dell'abbattimento delle barriere, ma si fa promotrice di iniziative a sostegno dell'inclusività, usando mezzi come la musica e l'attività motoria e coinvolgendo attivamente le relative categorie. In particolare, FIABA pone l'accento sullo sport, ritenuto da sempre un esempio imponente pro-socializzazione, dove uguaglianza e spirito di squadra sono i pilastri fondanti. A tal proposito, è interessante riportare l'intervista di Samuele Bosco, messaggero di FIABA, ad Arturo Mariani[23], calciatore nato con una sola gamba, originario di Roma. Il colloquio, tenutosi il 6 aprile 2021 durante la Giornata Internazionale dello Sport per lo Sviluppo e la Pace, ha fatto emergere un messaggio di positività che incoraggiasse a praticare sport come forma di riscatto personale e sociale, in particolare rivolto a quelle categorie di persone che, per un'incompatibilità più o meno fisica, sono titubanti ad affacciarsi a tale mondo. Durante l'intervista, l'atleta, *coach* e scrittore ha raccontato le sue esperienze sportive e i processi mentale e fisico che l'hanno condotto ad affermarsi nel campo, fino a permettergli di debuttare nella Nazionale Italiana per soggetti amputati e disputare un Mondiale e un Europeo. Arturo non si è però solo limitato a stilare una serie di eventi personali e relativi successi, ma ha voluto lanciare un messaggio di solidarietà verso le categorie deboli costituite da persone con disabilità dello spettro motorio più o meno simili a quella di cui è portatore;

[22] Ente Italiano di Normazione. *Abbattimento barriere architettoniche - Linee guida per la riprogettazione del costruito in ottica universal design*. 29 novembre 2016. Link alla versione in PDF: http://www.cng.it/CMSContent/Consiglio-Nazionale/Repository/files_news/uni_pdr_24_2016.pdf

[23] *Sport e disabilità: il moncalvese Samuele Bosco intervista Arturo Mariani*. 16 aprile 2021. A cura della redazione di ATNews. Link alla pagina: https://www.atnews.it/2021/04/sport-e-disabilita-il-moncalvese-samuele-bosco-intervista-arturo-mariani-142441/

ha inoltre parlato del suo lavoro e delle opere di sensibilizzazione di cui è fautore, che coinvolgono individui e aziende attraverso percorsi e progetti sociali, con lo scopo di incentivare a osare anche nei campi in cui si potrebbero incontrare difficoltà. Ha inoltre marcato come l'attuale situazione sanitaria[24] abbia impedito la socialità e, di conseguenza, sia riuscita a distruggere anche la più sottile forma di fiducia personale creatasi in soggetti portatori di determinate patologie, che ritrovavano in comunità tramite attività ricreative e sportive con persone a loro simili. Nonostante lo spiacevole periodo storico che sta caratterizzando questi anni, l'atleta si ritiene fiducioso nella ripresa della socialità e delle consuete attività e sostiene fortemente la necessità di sensibilizzare sul tema poiché, solo donando voce ai problemi, si può creare una prospettiva di futuro migliore, in primis sul piano dell'inclusività. L'intervista è una delle ultime ad opera di FIABA e, in poco tempo, ha contribuito alla diffusione delle parole di incoraggiamento dell'atleta, che sono diventate virali sui social e hanno spronato a mettersi in gioco nei campi più temuti. Mariani è riuscito, grazie anche al contributo dell'organizzazione, a raggiungere soprattutto i più giovani tramite il social cinese TikTok, dove si autodefinisce "ragazzo in gamba"; l'aver coinvolto attivamente gli adolescenti tramite i suoi video, dove ha impugnato l'autoironia come arma per promuovere l'inclusività in un contesto dove discriminazione, il bullismo e la sfiducia nelle proprie abilità rappresentano spesso un importante problema. Anche solo tramite un'intervista, l'associazione ha contribuito alla diffusione della problematica sulle barriere architettoniche e ha incentivato la parte di popolazione raggiunta al superamento delle stesse e all'inclusività. Iniziative simili, dove hanno preso parte anche esperti di settore, hanno condotto (e conducono) alla formazione di figure professionali che, tramite le campagne di sensibilizzazione, sono state coinvolte e hanno potuto mettersi alla prova, per contribuire al miglioramento del tenore di vita di coloro che sono costretti, purtroppo, ad interfacciarsi con un territorio poco accessibile.

FIABA, inoltre, dà largo spazio alle nuove tecnologie e, come si può leggere in articoli dedicati, dona grande importanza agli ausili che le più moderne innovazioni possono offrire a determinate fasce d'utenza; un campo supportato dall'associazione è quello della domotica[25], definito dalla stessa come un enorme aiuto contro le barriere domestiche.

[24] Il riferimento è alla pandemia da COVID-19.
[25] *Domotica: un aiuto contro le barriere domestiche.* 21 aprile 2021. A cura di Silvia di Fiaba.org. Link alla pagina: https://www.fiaba.org/domotica-e-barriere-domestiche/.

L'organizzazione, infatti, non si occupa solo di campagne relative a strutture pubbliche, ma pone in risalto anche le difficoltà che possono subentrare in quelle private, come le abitazioni, e propone delle soluzioni a proposito. Per quanto riguarda la sopra citata domotica, FIABA ne decanta i benefici successivi alla sua introduzione nella quotidianità, citando il vantaggio che tali tecnologie offrono a coloro che ne usufruiscono e ponendo in rilievo quanto le stesse possano cambiare significativamente la vita di persone con particolari patologie che non permettono loro di svolgere anche le azioni più scontate. Un impianto domotico, infatti, è importante quasi quanto un aiutante fisico in alcune situazioni: con un semplice comando (vocale o tramite un'altra tipologia di interfaccia), ad esempio, è possibile accendere le luci o controllare dispositivi tecnologici da remoto e, come si può facilmente immaginare, può essere una preziosa risorsa per chi, per motivi legati all'età o alla disabilità, ha difficoltà di movimento. La domotica – cita la stessa organizzazione – può essere di grande ausilio per coloro che presentano disabilità dello spettro uditivo e visivo: grazie alle tecnologie *TTS* (*Text to Speech*) e *STT* (*Speech to Text*)[26], è possibile anche convertire il testo in audio e viceversa e impartire comandi ai dispositivi da controllare, utilizzando il mezzo più accessibile, sia esso l'interazione visiva o la voce. Dunque, un grosso aiuto per coloro che necessitano di ausili speciali e, soprattutto, molto versatile sul lato dell'interazione. Inoltre, i dispositivi domotici possono aggiungere un enorme supporto sul lato della sicurezza domestica: tramite l'impiego di sensori, è possibile rilevare fughe di gas nocivi, fumi o fuochi e permettere agli impianti collegati di chiamare i soccorsi in caso di immediato bisogno, senza avvalersi di un assistente umano; in sintesi, la domotica è un mezzo importante per donare autonomia alle persone con disabilità e, per tale ragione, gli strumenti che si avvalgono di tali tecnologie sono anche detraibili fiscalmente, tramite la formula del Superbonus 110%, come recita l'associazione. Tale misura, sostenuta da FIABA e introdotta dal Decreto Rilancio del 19 maggio 2020, permette ai beneficiari che ne possono usufruire di ottenere un incentivo alla realizzazione di interventi che permettono di migliorare il benessere quotidiano, tra i quali rientrano anche quelli relativi all'abbattimento di barriere fisiche con le relative opere di rinnovamento strutturale. Tramite la promozione di tali incentivi per mano di associazioni come quella in oggetto, sono state raggiunte più fasce di popolazione possibili e rese accessibili sul piano economico anche quelle più svantaggiate, anche grazie alle opere di *crowdfunding*, delle

[26] Tavosanis, Mirko. 2018. *Lingue e intelligenza artificiale*. Il riconoscimento del parlato. La trascrizione delle conversazioni. Capp. 3-4. Carocci Editore.

quali molte Onlus si avvalgono. Un punto a favore delle iniziative sostenute da FIABA sono l'elevata produzione di contenuti in formato digitale: come già anticipato, le nuove tecnologie permettono di organizzare al meglio i contenuti e raggiungere tutti coloro siano in possesso di un dispositivo multimediale; gli eventi organizzati sono infatti documentati in articoli disponibili sul Web e presentano quasi sempre filmati a corredo, che testimoniano visivamente gli stessi. L'associazione, al momento, non conta di un numero elevato di eventi ma si occupa, come già citato, della realizzazione di protocolli di intesa, cabine di regia e opere di sensibilizzazione, soprattutto sul piano burocratico e legislativo; il punto di forza più grande è, pertanto, dato dalla grande rete di partner e fornitori qualificati che collaborano con la stessa, al momento tredici, tra i quali enti di rilievo come Confassociazioni e INAIL. A proposito di quest'ultima, è importare citare la sua rivista *SuperAbile*[27], con la quale FIABA collabora, che affronta la disabilità sotto una rosa di aspetti molto ampia e favorisce il dibattito e il confronto tra diversi punti di vista. Il periodico, in uscite mensili, riporta notizie, interviste e approfondimenti di attualità, sport, cultura e viaggi e pone l'accento sui temi di previdenza, assistenza, barriere, legislazione e diritti; inoltre, si avvale dell'omonimo *contact center* per la disabilità, che da oltre dieci anni offre supporto a coloro ne sono portatori (e non solo).

Alla luce di ciò, è possibile affermare che l'associazione si avvale di strumenti solidi per la promozione delle proprie campagne di sensibilizzazione: non conta di molti progetti, ma riesce in egual modo a compensare con la massiccia informazione, anche se probabilmente con strumenti che non riescono a raggiungere ogni fascia e a permettere agli interessati di impattare in maniera immersiva con la problematica. A seguire, verrà trattata un'altra organizzazione Onlus, #Vorreiprendereiltreno, della quale verranno descritti alcuni dei progetti più salienti e saranno illustrati gli obiettivi raggiunti in merito alle opere realizzate, con un occhio di riguardo a quelle relative all'abbattimento delle barriere fisiche.

1.1.2 #Vorreiprendereiltreno Onlus

#Vorreiprendereiltreno[28], è, come la precedente, una ONLUS, che promuove delle iniziative di sensibilizzazione, inclusione e abbattimento delle barriere architettoniche, sociali e culturali, attraverso la realizzazione e il finanziamento di progetti e servizi in tutta Italia.

[27] SuperAbile, Link al sito web: https://www.superabile.it/cs/superabile/home.
[28] #Vorreiprendereiltreno. Link al sito web: https://vorreiprendereiltreno.it/.

L'organizzazione è nata il 20 giugno del 2014 e ha come manifesto un articolo autoironico scritto dall'attivista samminiatese Iacopo Melio – nato nel 1992 con la *sindrome di Escobar*[29], una malattia genetica rara che lo costringe sulla sedia a rotelle – nel quale, mediante il suo umorismo, si rivolge al mondo politico con lo scopo di ricordargli quanto sia complesso vivere la quotidianità, se si è portatori di disabilità motoria. Tali righe sono servite da input ad gruppo di volontari, che hanno usato le parole di Iacopo come trampolino di lancio per una campagna di sensibilizzazione che è, in poco tempo, diventata virale sui principali social network, e ha ottenuto spazio anche su testate internazionali, quali *BBC* e *ALJazeera*. Ad oggi, l'associazione conta quasi quattordicimila sostenitori e ha investito oltre sessantottomila euro per il finanziamento di progetti sociali e il mantenimento in attività di servizi che possono coadiuvare le persone con disabilità e le relative famiglie. Nella sezione archivio del sito web dell'associazione è possibile consultare un elenco dettagliato di tutti i progetti realizzati, riguardanti in particolare gli interventi di eliminazione delle barriere a favore di persone con disabilità, minori in difficoltà e pazienti ospedalizzati. La maggior parte di questi riguardano, tuttavia, i portatori di disabilità dello spettro motorio che, grazie ad interventi mirati sul territorio, sono riusciti finalmente a valicare le tanto ostiche barriere, imposte dal mondo creato ad immagine di persone normodotate. Tra questi, è possibile citarne uno che ha avuto come traguardo il miglioramento del benessere collettivo della popolazione della provincia di Pisa: la resa accessibile ai disabili della stazione ferroviaria di Pontedera[30], ogni giorno frequentata da pendolari della più disparata provenienza. Era infatti noto che la suddetta fosse sprovvista di strumenti atti a favorire l'inclusione di categorie deboli come quelle dei portatori di disabilità motoria, ma purtroppo ogni opera di rinnovo non riusciva a prendere piede. Fortunatamente, in seguito ad una massiccia campagna di sensibilizzazione, mossa anche da un tweet della parlamentare Laura Coccia che, trovandosi in stazione carica di bagagli, ha marcato anche la sola assenza di semplici ascensori, gli attivisti volontari di #Vorreiprendereiltreno sono riusciti a raccogliere i fondi necessari a realizzare le opere di abbattimento delle barriere architettoniche presenti nella stazione e a far compiere i lavori di adattamento delle strutture esistenti tra il 2015 e il 2016. Più nel dettaglio, il loro contributo ha permesso al personale incaricato di effettuare modifiche ai

[29] *La sindrome di Escobar, una malattia quasi sconosciuta*. A cura di Silvia Cargnelutti. Imobility. Link alla pagina: https://www.abbassamentoveicoli.it/la-sindrome-di-escobar-una-malattia-quasi-sconosciuta/.
[30] *Resa accessibile ai disabili la stazione di Pontedera (PI)*. Maggio 2015. A cura dei sostenitori di #Vorreiprendereiltreno. Link alla pagina: https://vorreiprendereiltreno.it/progetti/resa-accessibile-stazione-pontedera-pi/.

binari dei treni e di costruire un sottopassaggio e quattro ascensori, permettendo ai visitatori della struttura di poter godere di un maggior benessere. L'iniziativa promossa, inoltre, ha dato anche un input positivo alla realizzazione di ulteriori interventi nel comune di Pontedera, per i quali l'assessore Franconi ha stanziato trecentonovantamila euro per finanziare l'abbattimento delle barriere architettoniche nel centro urbano. In particolare, i fondi sono stati necessari alla costruzione di cinque percorsi che permettessero ai disabili di raggiungere facilmente i vari punti della città e alla modifica di alcuni marciapiedi, sprovvisti fino al tempo di rampe per l'accesso facilitato.

Sfogliando la pagina dedicata, è possibile consultare molti interventi promossi dall'associazione relativi a luoghi pubblici, come scuole, stazioni e parcheggi, ma la stessa si è occupata anche di iniziative relative a strutture private, come la realizzazione di un appartamento domotico per persone con disabilità dai 18 anni in su[31]. L'iniziativa, nata nel gennaio del 2020 da un protocollo d'intesa tra la Regione Toscana, l'Asl Centro Toscana, la Società della Salute Empolese Valdarno-Valdelsa e l'onlus #Vorreiprendereiltreno, ha permesso di realizzare una *smart home*, ovvero una casa privata adattata alle esigenze di utenti maggiorenni con disabilità motorie, provvista di impianti domotici che permettessero loro di godere una vita in (quasi) totale autonomia. L'ex presidente della Regione Toscana, Enrico Rossi, al momento della firma del documento di intesa tra gli enti sopra citati, ha evidenziato il diritto incontestabile per ogni cittadino di poter vivere in modo indipendente e garantire loro pari opportunità, sfruttando le più moderne tecnologie in modo efficiente. Successivamente, è nata la campagna che, con al timone Iacopo Melio, ha condotto alla creazione di prototipi e esperimenti per la realizzazione di un modello di appartamento domotico esportabile almeno nell'intero territorio regionale. Nell'aprile del 2021[32], il progetto è diventato realtà e per la fase di *testing* sono state coinvolte delle persone con disabilità dello spettro motorio e sensoriale. Per la sperimentazione è stato individuato un appartamento al primo piano, situato appena sopra la Casa della Salute di Empoli, concesso in comodato d'uso dallo stesso comune. Lo spazio messo a disposizione, che ammonta a 70 metri quadri divisi in due grandi camere doppie e provvisto di ampio ascensore al piano per accedervi, verrà occupato da due persone con disabilità dopo il termine dei lavori, previsto

[31] *Domotica, una casa intelligente per persone con disabilità.* 26 gennaio 2020. La Nazione. Link alla pagina: https://www.lanazione.it/empoli/cronaca/domotica-casa-intelligente-persone-disabilita-1.4999994/.
[32] *Una casa domotica per la vita autonoma*. 3 aprile 2021. La Nazione. Link alla pagina: https://www.lanazione.it/empoli/cronaca/una-casa-domotica-per-la-vita-autonoma-1.6204234.

entro la fine del 2021. Le stanze saranno sfruttabili anche da eventuali assistenti ed è precisato che le spese a carico degli inquilini saranno unicamente quelle condominiali e delle utenze domestiche, molto contenute, comunque, poiché la struttura è progettata per essere a impatto zero. Per il progetto, come cita il presidente della Onlus Carlo Tempesti, sono "pronti a spendere centomila euro", poiché se trova un effettivo riscontro positivo, può davvero cambiare la vita delle persone con determinate patologie e, in particolare, potrà essere un'iniziativa importante per il progressivo coinvolgimento di aziende di domotica e impiantistica in opere simili. Se il modello, insomma, riuscirà a confermarsi come pioniere di soluzioni innovative e riuscirà davvero a migliorare la vita di tali individui e, in particolare, troverà un solido sostegno grazie alla relativa campagna di sensibilizzazione, sarà un enorme passo avanti per l'indipendenza di coloro che, purtroppo, sono penalizzati nel compiere le più semplici azioni anche in ambito domestico; per comprendere se le soluzioni proposte si dovrà attendere ma le premesse sono più che promettenti, purché possa godere anche di una buona sponsorizzazione. È infatti spesso difficile, in particolare se si trattano tematiche comprensibili solo settorialmente – come la domotica e le tecnologie assistive, per l'appunto – avere successo in campi simili e va trovato un metodo efficace per raggiungere ogni tipo di utenza; è necessario, dunque, cercare e applicare una strategia alla portata di tutti per poterne favorire la diffusione su una scala più larga.

A sostegno dell'iniziativa e di altre simili sono presenti, almeno al momento, numerosi sostenitori sui *social* più diffusi ma, almeno in seguito ad un'analisi non troppo approfondita, andrebbe coinvolta con altre strategie anche la classe di utenza che non fruisce del mondo del web allo stesso modo, ad esempio, dei nativi digitali. Sarebbero, pertanto, da istituire differenti strategie di sensibilizzazione da dedicare a tali fasce di persone, ad esempio con l'allestimento di più panel nei luoghi più frequentati, come nell'atro di scuole, stazioni e aeroporti. Al momento (2021), a causa della pandemia da COVID-19, che ha modificato radicalmente le abitudini degli italiani da più di un anno, la maggior parte delle iniziative "fisiche" è stata convertita in digitale, fruibile sotto forma di video-tour o workshop online e purtroppo, come non è difficile immaginare, le campagne di sensibilizzazione hanno perso parte dell'utenza appartenente alle fasce di età più alte. Infatti, la progressiva conversione di ogni tipo di evento nella sua forma digitale, alla quale si è assistiti nel periodo corrente, ha alimentato il sentimento di frustrazione negli utenti non abituati alla consultazione di contenuti digitali e li ha ulteriormente scoraggiati dall'iniziativa di prenderne parte.

Parallelamente alla speranza che il mondo digitale diventi perfettamente accessibile per ognuno, indipendentemente da fattori fisici o culturali, è importante anche che vengano, dunque, comunque incentivate le iniziative fruibili di persona, per poter raggiungere anche chi non si è omologato con gli standard dettati dalle più recenti tecnologie.

In queste sezioni sono state illustrati alcuni tipi di opere di sensibilizzazione attuate da due importanti Onlus, operanti in molteplici campi, con differenti tipi di utenza e mediante differenti canali di comunicazione. Esistono, tuttavia, mezzi di incentivo alla risoluzione di problemi e alla loro sponsorizzazione direttamente accessibili al pubblico, senza intermediari, fruibili autonomamente e senza il bisogno di assistere ad eventi promossi da enti o associazioni; il loro vantaggio è quello di poter essere consultati sotto forma di svariate tipologie di contenuto e mediante numerose piattaforme, nello stile della contemporanea narrazione transmediale[33].

Nel prossimo paragrafo verranno introdotti alcuni progetti presentanti strategie atte alla sensibilizzazione sui problemi presentati, in forma alternativa alle convenzionali campagne sopra presentate e ne saranno decantati i punti salienti a favore ed evidenziati eventuali lacune o limiti.

1.1.3 Movidabilia e Speriment'Azione Accessibile: barriere, visual marketing e digital storytelling

Questa sezione è dedicata a Movidabilia[34] – Spazi Senza Barriere, un'associazione di promozione sociale nata nel 2013 a Lecce che si occupa della tutela e dell'incentivo dei diritti delle persone con disabilità motoria, sensoriale e interattiva. È importante citarla, poiché si avvale di tecniche di sensibilizzazione alternative al classico evento con *stand*, volantinistica e conferenze; l'organizzazione, infatti, si serve di spettacoli teatrali, mostre d'arte ed eventi sportivi per coinvolgere l'utente a 360 gradi, talvolta proponendo lo stesso contenuto tramite diversi canali. Con tale strategia, Movidabilia tenta di raggiungere ogni fascia di popolazione, indipendentemente dai fattori che le diversificano, ponendo particolare cura nello studio dell'utente finale, al fine di fornirgli la migliore fonte in termini di accessibilità dei contenuti proposti. Dunque, per ottenere il giusto impatto culturale, l'associazione cura molto l'accessibilità in fase di somministrazione delle proposte di sensibilizzazione sui problemi

[33] Jenkins, Harry. *Cultura Convergente*. 2014. Apogeo.
[34] Movidabilia. Link alla pagina: https://www.movidabilia.it/.

mossi e non solo quella ad essi relativi. Il suo desiderio è, infatti, coinvolgere tutti nelle campagne, anche – ad esempio – il non vedente e il tetraplegico, utilizzando canali differenti per la diffusione dello stesso messaggio, come audiodescrizioni e eventi digitali che non implicano la presenza fisica in un luogo che potrebbe essere inaccessibile a chi non presenta caratteristiche fisiche idonee all'ambiente o non può muoversi in autonomia. Utilizzando la strategia della *narrazione transmediale*, che prevede dunque un'ampia gamma di contenuti (digitali e non) fruibili tramite più mezzi, per la sponsorizzazione dei progetti presentati, l'organizzazione si schiera a sostegno di categorie deboli della popolazione, fornendo loro strumenti per l'integrazione sociale e il miglioramento della vita quotidiana. La principale iniziativa promossa da Movidabilia è Speriment'Azione accessibile, la sperimentazione che le ha dato il "la" e le ha permesso di vincere il bando regionale "Puglia Capitale Sociale 2.0"; il progetto si avvale di una rete di operatori culturali e associazioni dei comuni di Lecce e Taranto e punta a sensibilizzare e proporre soluzioni sul tema delle difficoltà fisiche, proponendo l'introduzione di tecnologie assistive in luoghi simbolo di arte e cultura, come teatri e cinema per il superamento delle barriere (fisiche, percettive e cognitive) che possono limitare o impedire la fruizione degli eventi da parte di categorie di portatori di disabilità. Per il raggiungimento degli obiettivi sono coinvolti anche i partecipanti ad alcune *hackaton*[35], al fine di eseguire un *test* ambientale dei dispositivi convolti e individuare i luoghi chiusi e all'aperto dove poter eseguire la sperimentazione. Parallelamente a questa fase, viene eseguita una mappatura dei principali luoghi culturali e di movida, al fine di individuarne di nuovi, renderli progressivamente accessibili e inserirli in una lista, successivamente consultabile da diversi dispositivi dai diretti interessati. La maggior parte degli interventi, in sperimentazione, sono anche presi in esame durante laboratori formativi sul tema della cultura accessibile, aperti a tutti grazie anche alla collaborazione di numerosi partner di progetto; tali iniziative, fruibili dal vivo o da remoto, sono un'importante risorsa per l'associazione, poiché permettono loro di sensibilizzare in modo coinvolgente e non limitandosi a semplici rassegne, spesso riservate a un pubblico piuttosto settoriale. Inoltre, l'associazione che se ne fa promotrice si avvale di una ricca rete sociale, perlopiù composta da volontari e dai diretti soggetti portatori di disabilità, dove è possibile confrontarsi e creare *topic* di discussione inerenti le problematiche sollevate.

[35] Treccani, voce *Hackaton*. Link alla pagina: https://www.treccani.it/vocabolario/hackathon/.

È dunque possibile affermare che le metodologie di realizzazione e promozione delle campagne di Movidabilia presentano un enorme punto a favore: la disponibilità di fruirne tramite numerosi mezzi e modalità senza lasciare nessuno in zona d'ombra e, dunque, escluso. Poter raggiungere, infatti, gli utenti tramite differenti strategie e canali è un primo importante step verso l'abbattimento delle barriere: rendendo accessibili i contenuti a tutti, anche a chi presenta disagi nel fruirne, tramite la comunicazione transmediale, significa espandere il campo di azione e, di conseguenza, consentire alle campagne di risultare più efficaci. Organizzare, inoltre, laboratori e dare, dunque, spazio all'interattività, permette di rendere accattivanti gli incontri e, talvolta utilizzando uno sfondo ludico, di non lasciare alcuno escluso, nemmeno i più giovani o chi presenta un deficit dello spettro cognitivo. In sintesi, le iniziative promosse sono altamente inclusive e coinvolgenti: due caratteristiche che è possibile definire il *gold standard* delle migliori pratiche per impattare positivamente su problematiche che riguardano proprio l'eliminazione delle barriere.

A seguire, non verranno esaminate campagne e relative iniziative, ma due strumenti che, sfruttando le più moderne tecnologie, possono tentare di donare all'utente finale un'esperienza ancora più immersiva riguardante le problematiche evidenziate, che una tradizionale campagna di sensibilizzazione potrebbe far emergere, ma non riuscire a rappresentare in maniera pseudo-realistica.

1.1.4 Simulare per sensibilizzare: Wheelchair Simulator VR

In questa sezione verrà presentato un efficace strumento per promuovere la sensibilizzazione sui problemi che derivano dall'interfacciamento di specifiche fasce d'utenza con la società e l'ambiente circostante; più nel dettaglio, si tratterà di un simulatore di Realtà Virtuale, Wheelchair Simulator[36], sviluppato da ViRa Games e presente sullo store della piattaforma Steam e acquistabile per meno di tredici euro, presentato come un videogioco di avventura in prima persona. Il gioco incide sul problema delle disabilità motorie e delle barriere architettoniche e mette a disposizione dell'utente un'esperienza ludica immersiva, fruibile tramite un visore di Realtà Virtuale – prodotto dalle case Oculus, HTC e Valve – e rappresenta un ottimo strumento per provare in modo realistico esperienze di vita anche scontate, nel ruolo di un portatore di disabilità motoria agli arti inferiori, costretto all'utilizzo

[36] Wheelchair Simulator VR. Link alla pagina dedicata sul portale Steam: https://store.steampowered.com/app/841120/Wheelchair_Simulator_VR/.

di una sedia a rotelle. Durante la simulazione, il giocatore (nel ruolo del protagonista), muovendosi nello spazio tramite la trazione delle ruote, è chiamato a esplorare una città ricca di ostacoli e insidie fisiche. Nell'esplorazione, il protagonista segue un tracciato in stile *run-platform* 3D e raccoglie monete, monta su rampe e sfreccia sulle strade a bordo della sua carrozzina, ma non senza imprevisti: per egli, è infatti è semplice cadere, essere spinto giù dalla sedia, finire investito da un'auto in corsa o, persino, perdere la vita. Il tutto, condito da una chiave comica e autoironica e presentato come un normale gioco, all'apparenza anche poco accattivante poiché privo di particolari missioni o *boss* finali da eliminare, rappresenta un eccellente modo per lanciare un importante messaggio all'utente normodotato che decide di provarlo. Di livello in livello, il giocatore infatti incontra difficoltà sempre crescenti che lo mettono a dura prova, pur non essendo un'esperienza troppo condita da tranelli o trappole; la sola dinamicità urbana, caratterizzata da strade con mezzi in movimento, lavori in corso, pedoni e un'infinità di dislivelli, bastano a rendere il *gameplay* frustrante e complesso nell'avanzamento. Dunque, la vera impresa alla base della simulazione è affrontare, nei panni di un soggetto con un *deficit* fisico, il mondo creato ad immagine delle persone normodotate, all'apparenza scontato e con problemi minimizzabili, ma nella realtà una vera avventura per chi non ha il vantaggio di potersi muovere autonomamente. La non accessibilità degli ambienti, quindi, costituisce la vera sfida da affrontare. Il narratore del gioco è Dmytro Schebetyuk, un ragazzo e atleta paralimpico che in seguito ad un trauma vertebrale avvenuto nel 2011, è costretto su una sedia a rotelle; proprio dalla sua esperienza nel panorama urbano ucraino, poco accessibile – racconta – è nata l'idea di un prodotto che potesse sensibilizzare sul tema dell'abbattimento delle barriere architettoniche, in un modo estremamente realistico. In seguito alla collaborazione con un team di sviluppatori esperti in RV e disabilità dello spettro motorio, l'atleta ha accettato di rendere pubblica la sua storia e di utilizzarla per incentivare non solo l'abbattimento delle suddette barriere nella realtà, ma per promuovere la realizzazione di prodotti simili, ottimi per sensibilizzare su temi delicati, giungendo direttamente al fruitore, senza bisogno di mediazione. Dalla sua pubblicazione, databile al maggio del 2018, il simulatore è stato acquistato e scaricato migliaia di volte e gli incassi sono stati devoluti all'organizzazione volontaria Dostupno UA (in cirillico: *Доступно UA*), che si occupa di aiutare persone con disabilità in Ucraina. Il prodotto è efficace in particolare per raggiungere la fascia dei più giovani, spesso disinteressata ad argomenti pesanti quale il mondo delle disabilità, e più in generale il videogiocatore: presenta tuttavia dei limiti, tra i

quali spicca quello del costo dei supporti necessari alla fruizione. Infatti, lo strumento, come la maggior parte i *software* di Realtà Virtuale, richiede delle specifiche tecniche della macchina ospitante elevate, poiché esoso di risorse e, naturalmente, necessita di un apposito visore, acquistabile quasi unicamente *online* e riservato ad una fascia di utenza di settore. Inoltre, la fruizione potrebbe non essere immediata e alla portata di tutti: le interfacce e i programmi di questo tipo necessitano di una certa conoscenza, seppur limitata, della RV e dei pericoli connessi, compresi il pericolo di ferirsi mentre si è immersi nella simulazione o di presentare effetti collaterali fisici, relativi alla cattiva calibrazione del casco di RV e al movimento in-game (*motion sickness*). Una soluzione simile, tuttavia, risulterebbe efficace in occasione di campagne di sensibilizzazione promosse da istituzioni pubbliche: potrebbe essere un ottimo modo per coinvolgere gli alunni di scuole superiori, ad esempio (usare strumento di realtà virtuale è sconsigliato al di sotto dei tredici anni), o in tutte le situazioni in cui viene promossa la socialità. In tali occasioni, infatti, basterebbe avere a disposizione un solo set di attrezzatura e pochi supervisori per ogni gruppo di partecipanti e, tale iniziativa, rappresenterebbe un incentivo alla promozione degli eventi e vedrebbe, probabilmente, coinvolte più persone proprio grazie alla curiosità che potrebbe scaturire la possibilità di poter provare un'esperienza immersiva. In sintesi, Wheelchair Simulator VR rappresenta una nuova frontiera degli strumenti dedicati a normodotati per sensibilizzare sul problema delle barriere architettoniche, ma il mondo è ancora troppo "acerbo" per permettere a soluzioni simili di raggiungere una grossa fetta di pubblico; tuttavia, se usato nel modo corretto, costituisce una solida strategia per evidenziare i problemi esaminati e proporli nel modo più coinvolgente all'utente.

Nella sezione a seguire si analizzerà un ulteriore prodotto fruibile in realtà virtuale, simile a quello appena trattato, dove verrà unito, al tema della disabilità e dell'abbattimento delle barriere, il mondo dello sport.

1.1.5 Sport e inclusività nella RV: Wheelchair Basket VR

Un altro prodotto di cui è interessante scrivere è Wheelchair Basket VR, un videogioco di RV dedicato ad utenti normodotati, frutto di un progetto accademico degli studenti Lukas Hort, Louis Vogt, Marius Kröger e Juliana Kral del Mediencampus dell'Università di

Darmstadt, scaricabile gratuitamente dal sito web dedicato[37] e fruibile tramite i principali visori di Realtà Virtuale connessi ad un PC (non è disponibile, al momento, una modalità *standalone* avviabile direttamente dai visori che supportano questo tipo di applicazioni, come l'Oculus Quest e Quest 2). Lo scopo del simulatore è far immergere il giocatore nel personaggio di un veterano degli anni '50, mutilato di guerra e impossibilitato a deambulare autonomamente, e permettergli di vivere l'esperienza di tornare a giocare a basket dal "basso" della sua sedia a rotelle, affrontando tutti gli ostacoli che la vita gli pone. L'ambiente proposto è un centro amministrativo degli anni '50 che include una palestra da basket, dove si svolgono le partite che vedono protagonista il veterano; l'esperienza di gioco non è ottimale a livello grafico, ma la presenza di diverse modalità di partita rendono il simulatore comunque godibile sul lato dell'interattività. Nello specifico, è possibile giocare individualmente a basket, ma anche in modalità *multiplayer*, dove sono coinvolti più giocatori – comandati dalla CPU e non da altri utenti – ed è consentito sollevare, lanciare e utilizzare alcuni attrezzi ginnici messi a disposizione. Il movimento del protagonista è, come nel caso precedente, a traino: impugnando i *controller* del sistema di RV, è possibile trascinare la sedia a rotelle e spostarsi nello spazio all'interno del gioco. Sebbene il simulatore sia frutto di un progetto universitario, è piuttosto curato nei dettagli e il messaggio che vi è alle spalle è molto forte, soprattutto a livello di coinvolgimento emotivo: come nell'esempio di Wheelchair Simulator VR, ci sono situazioni che diventano frustranti e obiettivi che risultano quasi irraggiungibili, non a causa di imperfezioni tecniche, ma della realisticità delle situazioni proposte, da affrontare in un contesto di ridotta mobilità. Inoltre, il tema dello sport, utilizzato spesso per promuovere l'inclusività dei portatori di disabilità motoria, e del basket in particolare, attività paralimpica per eccellenza, sono utili per porre in risalto non solo ciò che non è possibile fare con determinati limiti fisici, ma anche quali sono i punti di forza e cosa, dunque, è possibile svolgere. Rispetto al prodotto presentato precedentemente, che metteva in risalto solo le difficoltà che il giocatore avvertiva nei panni del protagonista portatore di disabilità alle gambe, dunque, Wheelchair Basket VR ha un fine ancora più educativo: rappresentare le possibilità e non solo i limiti. Come il precedente, però, non può avvalersi – per il momento – di un vasto pubblico, a causa dell'inaccessibilità a tutti della Realtà Virtuale e degli strumenti ad essa connessi, ma rappresenta un modo eccellente per

[37] Wheelchair Basketball VR. Link al sito web: https://ag.mediencampus.h-da.de/projekt/wheelchair-basketball-vr/

promuovere tematiche calde come quella dell'inclusività e della sensibilizzazione sull'abbattimento di pregiudizi e barriere di ogni genere. Trovare una soluzione a quello che viene imposto da certe tipologie di limiti, dunque, può essere una sfida, ma è indispensabile per guadagnarsi platea nel campo: dare possibilità agli utenti svantaggiati e coinvolgere nelle problematiche il resto della popolazione, proponendo soluzioni, costituisce una buona strategia se si vuole giungere ad un compromesso efficace.

L'analisi di progetti come i due di cui si è appena discusso, seppur non molto approfondita, ha portato allo studio e alla realizzazione del simulatore che verrà presentato nei prossimi capitoli, riguardante l'esperienza in un'abitazione privata da parte di diverse tipologie di utente-giocatore (normodotato e non), con annesse soluzioni ai limiti evidenziati.

Prima di introdurre quanto appena citato è necessario, tuttavia, conoscere meglio il problema in esame, per poter proporre, successivamente, delle strategie risolutive efficaci.

2 Le barriere architettoniche: approccio al problema

In questo capitolo si conoscerà meglio il problema già trattato in precedenza, quello delle barriere architettoniche, e verranno presentate alcune strategie di superamento delle stesse nel mondo reale e diversi metodi di sensibilizzazione per attuarlo. Innanzitutto, è necessario dare una definizione scientifica delle due espressioni chiave: *disabilità motoria* e *barriera architettonica*. Il primo termine, più precisamente "disabilità dello spettro motorio", indica dei deficit relativi ad una persona fisica riguardanti la motricità e l'efficienza degli organi delle parti del corpo deputati al movimento. Tali limiti possono essere congeniti o acquisiti, dunque l'individuo di cui ne è portatore può esservi affetto fin dalla nascita o può svilupparli durante la sua vita, in seguito ad eventi traumatici, malattie neurodegenerative o anzianità. Solitamente, quando si fa riferimento a tali persone, viene lampante l'esempio del soggetto impossibilitato alla deambulazione e costretto su una carrozzina, ma i tipi di disabilità motoria si dividono in svariate tipologie e possono interessare gli arti superiori, la prima porzione del busto e ogni parte mobile del corpo. Questa, infatti, comprende un'ampia varietà di condizioni in cui il movimento può essere danneggiato in uno degli aspetti che lo caratterizzano e, più nel dettaglio, il tono muscolare, la postura, la coordinazione e la prassia. Gli ultimi due concetti, in particolare, sono l'oggetto chiave di discussione di questi capitoli: indicano rispettivamente la capacità di eseguire un movimento e l'abilità di compiere correttamente gesti coordinati e finalizzati al perseguimento di un obiettivo. Per citare alcuni esempi di disabilità riguardanti lo spettro analizzato è possibile citare quelle causate da paralisi celebrali infantili e da encefalopatie. Il primo, che interessa circa un bambino su cinquecento, rappresenta un danno irreversibile del sistema nervoso centrale avvenuto in fase prenatale, perinatale o neonatale e può causare un grado di infermità più o meno grave, che solitamente si manifesta con spasticità degli arti o con la completa immobilità dei suddetti. Il secondo, invece, sta ad indicare un gruppo di affezioni di origine genetica (ad esempio, di natura ereditaria) o di acquisizione più tardiva, rilevabili rispettivamente nelle fasi embrionale e fetale; i sintomi motori tipici di tutte le encefalopatie sono identificabili con tremori, atrofia muscolare, e mioclonia (contrazioni involontarie dei muscoli) [38]. I due citati sono solo due dei molteplici esempi di cause che possono indurre a presentare deficit che

[38] Bortolotti, Elena. 2018. *Le disabilità motorie e sensoriali*. Università degli Studi di Trieste. Link alla presentazione in PDF:
https://moodle2.units.it/pluginfile.php/306908/mod_resource/content/1/lez%203a%20Disabilit%C3%A0%20mot.pdf.

incidono, più o meno significativamente, sulla motilità; possono anche incorrere, come già citato, in seguito a incidenti (su strada, sul lavoro, in ambito militare, ecc) o all'anzianità (artrite reumatoide, artrosi e altri disturbi dell'apparato muscolo-scheletrico[39]). Inoltre, possono essere consequenziali ad altri tipi di patologie che colpiscono il sistema nervoso e si caratterizzano perlopiù come alterazioni dello spettro sensoriale, a causa – ad esempio – di una cattiva postura ed ad alterazioni della percezione dello spazio circostante.

La seconda espressione, *barriera architettonica*, su cui è importante fare luce, è possibile sintetizzarla in un qualunque elemento costruttivo che impedisce o limita gli spostamenti o la fruizione di servizi, spesso a svantaggio di categorie deboli, quali i portatori di disabilità degli spettri motorio e sensoriale. Un esempio pratico potrebbe essere costituito da una scala, un gradino o anche una rampa troppo ripida; qualunque elemento caratterizzante un'infrastruttura può trasformarsi in un ostacolo invalicabile per determinati soggetti e costituire, anche implicitamente, un segno di mancata inclusione e discriminazione[40]. Come già scritto in fase di presentazione del contesto, è possibile rendere accessibili determinati luoghi, privati e pubblici, adattando quanto già disponibile ai requisiti richiesti o integrarvi strutture e ausili, quali rampe o montascale; per realizzare tali opere architettoniche è fondamentale la collaborazione di enti, è richiesta una buona disponibilità economica ed è necessario realizzarle seguendo rigide normative, affinché siano efficaci e non creino intralcio ad altre categorie di utenti. Fortunatamente, a supporto di tali interventi esistono numerose associazioni di natura volontaria (come già trattato), che non solo provvedono a sanare i problemi evidenziati, ma si occupano di sensibilizzare su tematiche affini per ampliare il pubblico dei sostenitori e, dunque, poter trovare maggiore forza nella successiva realizzazione delle opere di cui si fanno promotori. Esistono, inoltre, altri mezzi volti all'incentivo dell'abbattimento delle barriere architettoniche che, tramite l'impiego delle nuove tecnologie, possono rendere una campagna più accattivante o essere fruibili autonomamente o in piccoli gruppi, anche esternamente ad un'iniziativa. Proporre uno strumento meno convenzionale ed altamente interattivo può essere utile ad accanire il pubblico e a raggiungere fasce solitamente poco inclini all'informazione su problematiche di

[39] *Disturbi della deambulazione negli anziani*. Manuale MSD. Settembre 2019. A cura di James O. Judge. Link alla pagina: https://www.msdmanuals.com/it-it/professionale/geriatria/disturbi-della-deambulazione-negli-anziani/disturbi-della-deambulazione-negli-anziani.

[40] *Barriere architettoniche e accessibilità*. Disabili.com. Link alla pagina: https://www.disabili.com/mobilita-auto/speciali-mobilita-a-auto/barriere-architettoniche-e-disabilita.

un certo peso sociale, come i giovani. Tra i prodotti più interessanti, sono da citare quelli a scopo didascalico che, se presentati in chiave ludica, possono conquistare porzioni di pubblico più ampie; un tipo di strumento efficace può consistere, dunque, in un sistema in grado di avvicinare il soggetto in modo più rasente possibile al problema sollevato, rendendolo partecipe in modo quanto più immersivo possibile ad esso. Un sistema capace di coinvolgere appieno l'utente, anche sul profilo sensoriale, è la Realtà Virtuale, come già accennato; di tale concetto, in relazione anche al progetto presentato, se ne tratterà in seguito.

Dopo aver compreso quale strategia seguire per raggiungere un *target* di utenti, è fondamentale conoscere quali operazioni compiere per migliorare, effettivamente, una soluzione dove vi è bisogno di intervenire; è necessario, dunque, non solo capire cosa attuare nel mondo reale, ma anche quali tecnologie utilizzare per porre rimedio ad alcune comuni problematiche relative alla presenza di barriere architettoniche. Porre in un piano preliminare gli strumenti fisici da impiegare in un progetto non è solo utile ai fini del miglioramento della vita di chi ne fruirà, una volta completato, ma è altresì importante per la creazione di prototipi che possono essere di ausilio a personale più o meno di settore, per comprendere in anticipo come potrebbe essere il risultato finale. Sono particolarmente utili, ad esempio nel campo del design, le tecnologie che permettono di realizzare dei *rendering* tridimensionali fotorealistici[41] degli ambienti e la loro trasposizione in una realtà virtuale immersiva, dove è possibile esplorare il prodotto finale prima di realizzarlo.

Nel prossimo paragrafo verranno trattate alcune delle soluzioni attuabili in ambito architettonico per l'abbattimento delle relative barriere e, successivamente, si potrà conoscere come queste siano utilizzabili nella creazione di prodotti virtuali con lo scopo di sensibilizzare gli utenti, ma anche, nei casi di lavori più tecnici, di aiutare il personale incaricato nella successiva messa all'opera.

2.1 Possibili soluzioni nella vita reale

Come accennato sopra, esistono numerose soluzioni atte al miglioramento del tenore di vita di quelle categorie che, a causa di limiti o caratteristiche fisiche, possono risultare a disagio nello svolgere anche le azioni più basilari della vita quotidiana, se si trovano in ambienti non

[41] *A cosa serve il rendering 3D?* NativeDesign, sezione Servizi. Link al sito: https://www.nativedesign.it/servizi/rendering-3d.

a loro idonei. In questa sezione non si tratterà di specifiche situazioni, come già trattato in relazione ad opere infrastrutturali oggetto di progetti citati nel cap. 1, né ci si dilungherà sul lato normativo, ma verranno illustrate alcune strategie attuabili per poter tentare di donare un'esperienza fisica e sensoriale idonea a coloro presentino disturbi a svantaggio dello spettro motorio.

Purtroppo, come già trattato in precedenza, esistono innumerevoli tipi di disabilità motoria e ognuna di queste può presentare caratteristiche più o meno differenti, a seconda del soggetto interessato; ne deriva, dunque, una difficoltà di base nel trovare una soluzione univoca e oggettiva, in grado di accontentare ogni categoria. Per tentare di ovviare a questo tipo di limite, è necessario proporre strategie risolutive quanto più vicine ai soggetti alle quali sono dedicate, ma anche introdurne di ausiliarie, in modo da permettere loro di scegliere, in caso una fallisse nell'intento di poterlo aiutare efficacemente. Ad esempio, risulterebbe più efficace affiancare ad una rampa mobile, apribile elettronicamente tramite un pulsante, un sistema in grado di azionarla tramite un semplice comando vocale. Soluzioni simili, ancora purtroppo poco impiegate nella vita quotidiana ma fortemente sostenute dalla comunità, sono attualmente oggetto di sperimentazione, sia in ambito di strutture pubbliche che ambienti privati, come già visto nel caso della casa smart della Onlus #Vorreiprendereiltreno, che si avvale di un sistema di domotica per donare autonomia ai soggetti a mobilità limitata.

A seguire verranno illustrati i principali sistemi per l'accesso facilitato, indirizzati ai portatori di disabilità motoria; a causa dell'ampio spettro di queste ultime, verranno considerate unicamente quelle relative a limitazioni alla deambulazione, che richiedono l'ausilio di una sedia a rotelle (carrozzina) classica o motorizzata.

2.1.1 Caratteristiche strutturali e ausili per l'accessibilità

In questa sezione saranno illustrate le principali strategie atte a migliorare ambienti e contesti, al fine di renderli accessibili ad utenti portatori di disabilità motoria che utilizzano un mezzo a ruote; saranno trattati sia i requisiti strutturali ai quali gli edifici e i loro elementi devono rispondere, sia alcuni strumenti ausiliari, di natura digitale, per semplificare ulteriormente la vita di tali individui. Ogni concetto affrontato sarà legato ai campi all'edilizia e all'impiantistica relativi a luoghi pubblici e privati; in seguito, in riferimento al progetto oggetto del presente elaborato, si farà riferimento quasi esclusivamente alle opere attuabili in campo domestico.

È importante citare tali caratteristiche strutturali poiché, proprio a partire dalla raccolta di informazioni ad esse inerenti, è nata l'idea del simulatore proposto nell'elaborato, il quale è stato costruito considerando, almeno in parte, le vere proprietà che gli ambienti dedicati a tali categorie di utenza devono possedere.

2.1.1.1 Pavimenti

In apertura, è necessario illustrare le caratteristiche delle quali deve disporre la pavimentazione, interna ed esterna, per essere confortevole all'accesso di persone su sedia a rotelle e ugualmente efficiente nel caso di trasporto di bambini in passeggino, ad esempio. Innanzitutto, è necessario che il rivestimento sia liscio e non presenti fessure troppo larghe o dislivelli che possono costituire intralcio; più nel dettaglio, i giunti non devono superare i 5mm, i risalti non devono essere più spessi di 2mm ed è assolutamente sconsigliato l'impiego di moquette nel rivestimento degli interni, poiché può creare grinze in caso di perdita di aderenza del collante e aggrovigliarsi alle ruote del supporto per la deambulazione. Sulle rampe e in tutte le situazioni ove risulta necessario garantire un'aderenza ottimale, anche in presenza di acqua o umidità, è consigliato utilizzare un tipo di pavimentazione antiscivolo; in caso di piattaforme mobili, è inoltre buona regola assicurarsi che queste siano saldamente agganciate e non presentino stacchi percepibili con le superfici fisse.

2.1.1.2 Infissi e serramenti

In ogni situazione, come ripetutamente citato, deve essere necessario disporre di un'ottima accessibilità quando si progetta un ambiente dedicato a soggetti che presentano deficit dello spettro motorio: per un individuo normodotato potrebbe sembrare quasi scontato doversi soffermare sulle caratteristiche strutturali, ad esempio, che porte e finestre devono avere, ma nel caso in esame è necessario rispondere a requisiti ben precisi, non solo agli standard edili. Per quanto riguarda le finestre, queste devono, innanzitutto, garantire sicurezza: non devono presentare ante apribili verso l'interno e necessitano di essere facilmente raggiungibili in altezza, da chi è costretto a restare in posizione seduta o, se poste – ad esempio – in prossimità del letto, anche da sdraiati. In quanto al posizionamento, è necessario che queste garantiscano una buona visibilità a chi, come già detto, si trova permanentemente in una posizione più svantaggiata rispetto ad un normodotato e, per soddisfare tali esigenze, devono rispondere a rigidi requisiti. Innanzitutto, gli infissi devono essere dotati di *sottofinestratura* fissa o a filo pavimento ed avere le maniglie a non più di 100-130 cm di altezza; è necessario, inoltre, che

queste ultime siano ergonomiche e che non richiedano l'effettuazione di complessi movimenti per le operazioni di apertura e chiusura. È possibile, se alla portata economica del fruitore, dotare gli infissi di comandi a distanza o di sistemi di apertura automatica; lo stesso discorso si applica a tapparelle, veneziane, scuri e serramenti simili. Ove realizzabile, è preferibile installare tipi di finestra che non occupino spazio utile alla deambulazione e non provochino incidenti dovuti all'impatto con gli spigoli delle ante: nel caso siano da porsi in alto e non presentino grosse dimensioni, è raccomandabile utilizzare un modello *vasistas* (ad oscillobattente), con asta-leva o a comando elettronico; se invece sono destinate agli sbocchi principali dai quali è possibile affacciarsi all'esterno, i migliori formati da considerare sono a scorrimento orizzontale o verticale (a ghigliottina). Nel caso dei balconi, infine, è sconsigliato installarvi parapetti opachi, in quanto possono creare spazi ciechi tra pavimento e davanzale, dove possono incastrarsi le ruote della carrozzina, creando dunque il pretesto per bloccaggi o rovinose cadute.

2.1.1.3 Porte e passaggi

Durante l'installazione di porte, interne ed esterne, è necessario donare molta cura al fattore intralcio: vanno assolutamente evitate strutture troppo ingombranti e, come già scritto a proposito delle finestre, sono da scartare modelli che richiedono troppo sforzo nelle fasi di apertura e chiusura. Il tipo più semplice da manovrare è quello scorrevole: non è complesso da utilizzare e, soprattutto, è la soluzione ideale da adottare dove si ha poco spazio a disposizione o non se ne può dedicare. Porte e cancelli girevoli, a ventola o, semplicemente, a battente non sono, infatti, agevoli per il passaggio di sussidi a ruote, poiché non solo intralciano fisicamente, ma costituiscono anche pericolo. Sono da evitare, inoltre, meccanismi a molla di ritorno automatico, in particolare quelli senza fermo a fine corsa, perché possono impattare con il soggetto in carrozzina e, persino, sbalzarlo; le porte vetrate, in quanto più fragili e, dunque, più pericolose in caso di rottura, se non commutabili con un'alternativa valida, devono presentare una protezione nella parte inferiore, fino a 40 cm di altezza. Le maniglie, di ogni genere, devono essere poste ad un'altezza di 90 cm, per essere facilmente raggiungibili da seduti. È, infine, necessario anche rispettare delle misure specifiche per tipologia di utenza, per quanto riguarda la realizzazione dei passaggi: la larghezza minima deve essere di 70-80 cm, meglio se di 80-90 cm, per permettere anche alle carrozzine più ingombranti di accedervi.

2.1.1.4 Rampe e scivoli

Una caratteristica della maggior parte dei luoghi accessibili è possedere, almeno a monte di un ingresso, rampe e scivoli, indispensabili per poter dare autonomia a coloro che non hanno le caratteristiche fisiche idonee a valicare scale o importanti dislivelli. Per risultare in linea con le normative vigenti, le rampe devono presentare delle proprietà per assicurare ai fruitori un utilizzo sicuro e non costituire intralcio (una struttura troppo lunga o larga diventa essa stessa una barriera); più in dettaglio, devono essere larghe almeno 90 o 150 cm, per assicurare il passaggio, rispettivamente, di una o due carrozzine, a seconda del tipo di ingresso dove devono essere collocate ed è necessario che abbiano una pendenza longitudinale del 5% e una laterale di massimo l'1%, per un dislivello di massimo 150-200 cm. In aggiunta, è necessario dotarle, almeno ogni 10 metri, di ripiani di sosta per eventuali inversioni di marcia o rotazioni (150x150 cm). È importante anche scegliere peculiarmente il materiale con cui rivestirle, poiché devono presentare una superficie antiscivolo e avere, eventualmente, un *pattern*, per creare maggior attrito in caso di umidità; le stesse hanno bisogno, infine, di essere corredate di parapetto e cordolo di protezione. In luoghi dove invece non vi sono dislivelli importanti – di massimo 15-20 cm –, è possibile installare gli scivoli, anche in formato richiudibile (montabili all'occorrenza); anche questi devono, però, rispondere a determinati requisiti strutturali: la loro inclinazione non deve superare il 12%, la larghezza non deve eccedere i 150 cm e la somma tra i parametri di pendenza e contropendenza deve essere al massimo del 22%.

2.1.1.5 Montascale ed elevatori

I montascale, così come gli elevatori, sono dei sussidi utili non solo a coloro che si trovano costretti su una sedia a rotelle, ma anche agli anziani e chi presenta problemi di affaticamento, per esempio chi soffre di patologie cardiache. Questi ausili, come le rampe e gli scivoli, vanno adattati alla struttura dove devono essere montati, ma presentano ulteriori vincoli dovuti alla morfologia delle scale, alla quale bisogna attenersi per una perfetta integrazione. Se, ad esempio, la scalinata è unica e lineare, basterà installare un montascale rettilineo; in caso contrario, si verterà su un altro tipo che presenterà una guida in grado di seguire al meglio i tratti curvi. In caso di scale a più rampe, il montascale va montato sul lato interno, per evitare interruzioni in corrispondenza di pianerottoli o interferenze con i corrimano. Per poter essere installato, è necessario che non ci sia un dislivello di più di 4 m e che la scala

abbia almeno 100-110 cm di spazio in larghezza, dove poter anche ospitare una piattaforma di almeno 70x75 cm, dimensione ottimale per alloggiare una carrozzina; è inoltre necessario avere gli spazi necessari all'imbarco e allo sbarco della stessa. Al fine di garantire la sicurezza della persona trasportata, è altresì necessario predisporre un sistema di sicurezza e accertarsi di avere almeno la visuale libera a 2 metri dal montascale in movimento.

Per quanto riguarda invece gli elevatori, piccole piattaforme mobili simili ad ascensori, spesso installati a ridosso di piccole scalinate o gruppi di gradini, è necessario accertarsi che presentino una cabina di 80x120 cm e abbiano, davanti all'accesso, lo spazio utile ad eventuali manovre della carrozzina (di almeno 150x150 cm). Inoltre, non è possibile montarli – come i montascale – in prossimità di dislivelli superiori a 4 m e devono essere sempre omologati al tipo di scalinata dove vanno installati; se inseriti in un ambiente esterno, vanno dotati di protezioni dagli agenti atmosferici. Infine, bisogna prestare attenzione ad eventuali ostacoli che potrebbero bloccarne o intralciarne il funzionamento: non devono essere, infatti, montati in prossimità di luoghi che possono impedirne l'apertura della porta (meccanica o manuale) ed è necessario integrarli con i dispositivi di sicurezza previsti dalla legge.

2.1.1.6 Maniglioni ausiliari

In ambienti dedicati, come bagni per portatori di disabilità o, in prossimità dei sopra citati montascale e elevatori, è necessario installare dei maniglioni ausiliari, sussidi in grado di fornire un solido supporto in momenti in cui è necessario smontare dalla carrozzina, rimontarvi o aggrapparsi per avere un sostegno mentre si compiono operazioni che necessitano di un appoggio. Anche questi, come gli strumenti sopra citati, devono presentare delle caratteristiche particolari per essere idonei all'utilizzo: vanno posizionati, orizzontalmente, ad un'altezza di 90 cm e, in caso siano inclinati, dai 60 ai 100 cm. Devono essere, inoltre, installati dove vi è sufficiente spazio di manovra e non costituiscono alcun intralcio al movimento.

2.1.1.7 Interruttori e regolatori

Per garantire il giusto compromesso tra *comfort* e sicurezza alla tipologia di utenti a limitato movimento in oggetto, è necessario non solo disporre di strumenti ausiliari, ma anche di posizionarli correttamente, come si è già potuto osservare in riferimento a quelli in grado di valicare le barriere architettoniche più imponenti (quali scale e gradini). Nel caso di attuatori

e regolatori, è fondamentale che questi siano ben raggiungibili e possano essere controllati adeguatamente, nonostante l'ingombro causato dalla carrozzina; in particolar modo, gli impianti e i contatori generali devono essere posizionati scrupolosamente per potervi accedere anche in caso di emergenza. In generale, gli interruttori vanno posizionati tra 60 e 120 cm dal suolo per una fruizione ottimale e tra 40 e 140 cm ove non sia possibile procedere con l'installazione nella posizione più adeguata; eventuali manopole di caloriferi, termostati e altri centri di controllo non vanno montate negli angoli, poiché zone inaccessibili a chi non deambula autonomamente. Le prese elettriche sono, invece, da installare tra i 90 e i 110 cm dal suolo e non vanno mai poste in punti in cui è difficile raggiungerle. È necessario, inoltre, in particolare nei casi di abitazioni private, dotare gli ambienti di sistemi di allerta, tramite l'installazione di pulsanti o cordicelle in punti in cui può essere richiesto un intervento immediato da parte di un assistente, come nei bagni o nella zona notte.[42]

2.1.1.8 Interni domestici

In questa sezione verranno trattate alcune delle strategie e degli strumenti impiegabili in ambiente domestico, dove è importante assicurare, in primo luogo, comodità e sicurezza, ma anche dotare di un notevole livello di autonomia gli occupanti che presentano patologie riguardanti lo spettro motorio. Innanzitutto, è necessario fare luce su un concetto: non esiste una definizione univoca di casa accessibile, ma esistono delle strategie per poter personalizzare le abitazioni private, al fine di renderle più idonee alle caratteristiche del singolo occupante. Sono, tuttavia, disponibili numerose strategie per progettare e arredare un ambiente domestico, per renderlo fruibile da chi presenta dei requisiti speciali: nel caso in analisi, dai portatori di disabilità motoria alle gambe che, dovendo servirsi di una sedia a rotelle per poter deambulare, necessitano di particolari accorgimenti per poter godere di un certo livello di autonomia. È necessario, prima di iniziare a progettare, considerare che gli ambienti devono essere, in generale, quanto più larghi possibili e, soprattutto, sgomberi da ostacoli: vanno infatti predilette soluzioni aperte e arredamenti essenziali. Sia nella zona giorno che in quella dedicata alla notte bisogna garantire spazi adeguati tra mobili e componenti edilizie, per permettere il passaggio agevole di chi si serve di ausili (della carrozzina, in questo caso): lo spazio minimo consigliato per attraversare le strettoie tra gli

[42] *Speciale barriere architettoniche: accessori e impianti*. Disabili.com. Link alla pagina: https://www.disabili.com/mobilita-auto/speciali-mobilita-a-auto/barriere-architettoniche-e-disabilita/barrarch09-accessori-e-impianti.

arredi è di 70-90 cm, mentre quello necessario alla rotazione della carrozzina è individuato nella fascia 150x150 cm e 170x170 cm.

È importante prestare particolare attenzione soprattutto alla zona notte, dove molti soggetti con disabilità trascorrono la maggior parte della giornata; tale luogo può essere organizzato in diversi modi, a seconda del tipo di patologia e delle esigenze presentate. Vi sono, tuttavia, delle regole fondamentali da seguire in fase di progettazione, che servono a dotare di spazio sufficiente ogni sua parte, riguardanti lo spazio a lato del letto (che deve misurare 150 cm per assicurare la rotazione e 90 cm per il percorso per raggiungerlo), il passaggio tra mobili e muro (70-95 cm), l'area per il trasferimento dal letto alla carrozzina e viceversa (120 cm a fianco del letto), lo spazio di rotazione (non meno di 150x150cm), la superficie minima della stanza (da 350x300cm a 550x400 cm) e l'altezza del letto (di circa 40-50 cm). In quanto a quest'ultimo, che può essere standard o di tipologia speciale, è necessario che sia integrato con un sostegno o dei sollevatori corporei – disponibili in versione fissa o mobile –, nel caso di disabilità più gravi che impediscono lo spostamento autonomo del soggetto da e sulla sedia a rotelle e richiedono l'intervento di un assistente per l'igiene quotidiana (che avviene, in tali casi, direttamente dal letto). Riguardo invece la mobilia, è possibile avere abbastanza scelta, ma è sempre preferibile evitare soluzioni ingombranti e con ante a battente; è, inoltre, necessario prediligere armadi a muro con ante scorrevoli, senza ripiani posti in alto e settimini e comodini con cassetti che non richiedano sforzi per la loro apertura e chiusura. Sono da eliminare, come è facilmente deducibile, i pensili posti in luoghi inaccessibili ed è necessario adattare in altezza arredi quali specchi e mensole; vanno inoltre rimossi tende e tappeti dove si è di passaggio con la carrozzina: costituiscono intralcio e possono creare situazioni di pericolo.

La cucina, invece, andrebbe strutturata seguendo una disposizione centrale – o in modo da non creare spazi in cui è difficile muoversi con la sedia a rotelle – e ogni elemento, come piani di lavoro, elettrodomestici e credenze, dovrebbe essere installato in una posizione raggiungibile da seduti. È altresì fondamentale lasciare sufficiente spazio (circa 70 cm) al di sotto dei ripiani e regolare bene l'altezza da terra dei piani (circa 70-80 cm da terra), in modo da potervi far passare le gambe durante i lavori domestici e la preparazione dei pasti. Come in altri ambienti, lo spazio libero per garantire una buona rotazione deve essere di 150x150 cm e bisogna sempre assicurare una continuità tra i piani di lavoro per evitare situazioni di

rischio; è inoltre consigliato utilizzare cestelli o ripiani estraibili in sostituzione dei regolari cassetti e ante *vasistas* (apribili verso l'alto) o scorrevoli al posto di quelle a battente. Il lavello è un'altra fornitura da adattare alle caratteristiche della persona con disabilità, in quanto il rubinetto deve essere posto in una posizione più comoda e la leva del miscelatore deve essere, preferibilmente, più lunga.

Per il bagno, invece, sono necessari accorgimenti relativi soprattutto al fatto che si tratta di un ambiente soggetto all'umidità, dove è possibile scivolare facilmente o trovarsi in circostanze dove si perde totalmente l'autonomia. Per preservare quest'ultima, almeno in parte, è buona regola collocare i sanitari in modo idoneo, in modo da garantire un giusto spazio per le manovre del soggetto con disabilità (si applica sempre la regola dei 150x150 cm per la rotazione della sedia) e, soprattutto, dotare la stanza di numerosi maniglioni, dove egli può aggrapparsi per trasferirsi dalla sedia ai servizi. Esistono, inoltre, sanitari realizzati su misura per questa tipologia di utenti, come WC – con supporti laterali, pulsante dello sciacquone dislocato in una posizione più facile da raggiungere, spruzzino per igienizzarsi e struttura della tazza idonea all'aggancio della carrozzina – e docce – che dispongono di un *box* più ampio, hanno il saliscendi del doccino posizionato più in basso, presentano una seduta e sono fornite di rampe per superare il dislivello del piatto –. È altresì importante dotare il bagno di interruttori a filo e bottoni di allarme, con i quali poter richiamare all'attenzione eventuali assistenti o familiari in caso di bisogno, e montare le porte in modo che queste possano aprirsi verso l'esterno, in modo da evitare spiacevoli situazioni in cui la persona, in caso di caduta, ne impedisca l'apertura per i soccorsi. [43]

Esistono, infine, numerose strategie atte al miglioramento della vita di questo tipo di individui anche negli ambienti esterni all'abitazione e riguardano, ad esempio, l'impiego di saracinesche automatiche al posto di porte basculanti nei garage, l'acquisto o affitto di parcheggi auto riservati, o l'impiego di rampe o passerelle per facilitare l'accesso in casa; tuttavia, non ci si soffermerà su questa tipologia di luoghi, poiché il focus dell'elaborato verte principalmente sugli interni domestici.

[43] *Speciale barriere architettoniche: spazi interni di un'abitazione accessibile*. Disabili.com. Link alla pagina: https://www.disabili.com/mobilita-auto/speciali-mobilita-a-auto/barriere-architettoniche-e-disabilita/barrarch09-spazi-interni.

2.1.1.9 Dispositivi di controllo ambientale e domotica

È possibile, grazie anche a contributi dedicati alla fascia di popolazione in esame – come si è accennato nel cap. 1 –, arricchire gli spazi, domestici o meno, con dispositivi di controllo ambientale e impianti domotici. Nel caso specifico di persone con disabilità dello spettro motorio piuttosto gravi, è fondamentale dotare, ove possibile, le strutture di un sistema di controllo ambientale, ovvero di una serie di interfacce che permettono il controllo remoto di determinati elementi. Tra questi, solitamente compaiono i dispositivi di illuminazione, gli elettrodomestici e l'impiantistica di base, come i termostati ambientali e, soprattutto negli ultimi anni, anche porte, finestre e tapparelle. Automatizzare costituisce una comodità in più per le persone normodotate ma, nel caso di portatori di disabilità, consente loro di guadagnarsi una maggiore autonomia; poter controllare a distanza un dispositivo, tramite un telecomando, un'*app* sullo *smartphone* o un'interfaccia vocale, significa semplificare l'adempimento a molte attività. Nel caso specifico degli ambienti domestici è possibile riferirsi all'insieme di tali sistemi con il termine di *domotica* (dal latino, *domus* e dal suffisso greco *-ticos*, che indica "l'appartenenza ad una disciplina") ed è necessario, se dedicati a portatori di disabilità, effettuare una valutazione preliminare sulle potenzialità residue dell'utente prima di procedere all'installazione di qualsiasi ausilio. Non esiste, infatti, una strumentazione standard adattabile ad ogni caso, ma è necessario comprendere le necessità soggettive dell'utente e osservarlo nei suoi movimenti, al fine di conoscerne le reazioni mentre si interfaccia con lo spazio circostante e, solo successivamente, procedere nella scelta dei sussidi più efficienti nel favorirgli maggiori comfort e autonomia. È inoltre fondamentale analizzare i bisogni della persona, oltre che le potenzialità, poiché solo egli può conoscere come poter migliorare, anche emotivamente, la propria vita: un dispositivo troppo invasivo, ad esempio, potrebbe finire per farlo sentire ulteriormente a disagio e provocargli, dunque, blocchi psicologici oltre che fisici. Solo nel momento in cui l'utente è stato scrupolosamente osservato e se ne sono comprese le effettive necessità, è possibile passare alla scelta del sistema di automazione più idoneo, anche in riferimento al tipo di interfaccia di comando tramite la quale egli potrà azionarne i singoli elementi, come *touch screen*, dispositivo mobile (*smartphone*), *joystick* o controllo vocale[44]. Riguardo queste ultime, è possibile trovarne di molti tipi e di adattabili ad ogni esigenza manifestata e a ciascun tipo di disabilità, anche in

[44] *Speciale domotica: automazione domestica*. Disabili.com. Link alla pagina: https://www.disabili.com/prodotti/speciali-prodotti-a-ausili/domotica/17040-domotica-automazione.

relazione alla gravità della stessa; per una persona con un lieve impaccio motorio e una leggera difficoltà cognitiva sarà, ad esempio, consigliato un semplice telecomando con tasti grandi e retroilluminati mentre, ad un individuo con grave *deficit* motorio, ne verrà dedicato uno con bottoni facilmente azionabili e con a corredo una serie di sensori esterni, in grado di sfruttare qualunque tipo di movimento residuo di arti, capo o palpebre (a seconda del caso). Ove non sia, invece, possibile utilizzare telecomandi che interagiscono tramite controllo fisico, è possibile valutare l'impiego di dispositivi speciali in grado di funzionare mediante *input* vocale, utili in particolare nei casi in cui è presente anche una disabilità dello spettro sensoriale, come nel caso degli ipovedenti. In tali casi è molto importante che il sistema presenti una buona funzione di feedback, in grado di ripetere il comando impartito, donando così all'utente la consapevolezza della sua correttezza e donandogli la possibilità, in caso di cattiva comprensione o errori di formulazione, di ripeterlo. I telecomandi rappresentano l'interfaccia di controllo più efficace poiché, in particolare quelli di ultima generazione, permettono di eseguire molte operazioni tramite il tocco di un solo bottone e, dunque, sono di fruizione più immediata per persone presentanti *deficit* cognitivi oltre che dello spettro motorio; non sono tuttavia consigliati a coloro che presentano difficoltà nell'interazione con apparati fisici. Un altro tipo di dispositivo da citare è la tastiera evoluta: allo stesso modo del telecomando, permette di effettuare azioni da remoto e tramite l'interazione con bottoni fisici ma permettendo, in questo caso, l'eliminazione delle tradizionali pulsantiere, consentendo di controllare luci, audio e impianti senza doversi muovere nello spazio. Nei casi in cui non sia possibile premere sui bottoni meccanici dei quali le interfacce citate sono dotate, è possibile impiegare soluzioni sensibili al tocco, dove non è necessaria alcuna forza fisica per impartire i comandi ma è sufficiente il solo sfioramento con la pelle, in particolare nei moderni modelli a touch capacitivo (dove è sfruttata la conducibilità elettrica corporea e non la pressione, come nel caso di quelli resistivi). I sistemi *touchscreen* sono disponibili sia fissi (posizionabili a parete), che portatili e consentono un semplice accesso a tutte le funzioni del sistema di controllo; le relative interfacce sono inoltre pienamente personalizzabili a livello grafico e, dunque, adattabili ad ogni esigenza personale. I dispositivi *touch* di tipo portatile sono preferibili nei casi di disabilità grave o di persone costrette a stazionare in posizione supina per molto tempo, in quanto permettono di impartire i comandi a distanza e senza la necessità di spostarsi nello spazio. Tale tecnologia è, inoltre, ormai impiegata su larga scala e costituisce uno dei metodi di interazione più comuni, grazie al suo avvento su dispositivi

di consumo, come *smartphone* e *tablet* e, per tale motivo, in molti casi non sorprende o spaventa l'utente, poiché già conosce come interagirvi. I due tipi di interfaccia appena citati, oggigiorno, rappresentano un'importante risorsa per quanto concerne l'interazione con la domotica: sono alla portata della maggior parte delle persone e, supportando una grande rosa di applicazioni pilota dedicate al controllo remoto dei sistemi in oggetto, sono facilmente collocabili in una rete di dispositivi connessi. Attraverso queste *app*, solitamente disponibili per le principali piattaforme (iOS e Android) è possibile, infatti, tenere sotto controllo i vari sistemi e commutarne le funzionalità anche da remoto; a tale scopo assolvono anche i computer – fissi e portatili – dove, tramite una semplice connessione a Internet, è possibile gestire in ogni momento gli strumenti connessi alla rete. Un'altra tecnologia, utilizzata in particolare per attivare o disattivare dispositivi per la sicurezza (come le centraline di allarme e di sorveglianza), è rappresentata dalla classica connettività GSM di cui tutti i telefoni cellulari[45] sono dotati, tramite la quale è possibile modificare lo stato via telefonata o messaggio SMS. In questo caso, è sufficiente che gli strumenti supportino l'inserimento di una scheda SIM in grado di scambiare anche solo pochi *kilobytes* di dati, come accade per quelle appositamente dedicate al mondo dell'IoT, ad esempio quelle del *provider* ThingsMobile[46]. Infine, è possibile trattare le interfacce vocali, che permettono all'utente di interagire con gli strumenti connessi semplicemente parlando e offrono un feedback in tempo reale in seguito al comando impartito; è possibile trovarle integrate in altri sistemi, come *smartphone* e altoparlanti intelligenti (ad esempio, Amazon Echo e Google Home) o direttamente nel dispositivo di destinazione, che può contenere una centralina in grado di comprendere il messaggio ed eventualmente, di sintetizzarne uno di risposta o di avvenuto completamento dell'operazione. L'interfaccia vocale rappresenta, dunque, il mezzo tramite il quale l'utente interagisce, tramite parole isolate o parlato continuo, al fine di eseguire un'operazione relativa ad uno strumento ad esso connesso; costituisce un importante elemento della *user experience*[47] per applicazioni mobili, nelle quali l'elemento prevalente è l'esperienza dell'utente nell'utilizzo del cosiddetto *Internet delle Cose* (finora citato come *Internet of Things* o, in forma abbreviata, IoT). Questo tipo di strumento richiede una

[45] *Speciale domotica: interfacce di comando*. Disabili.com. Link alla pagina: https://www.disabili.com/prodotti/speciali-prodotti-a-ausili/domotica/17042-domotica-interfacce-di-comando.
[46] ThingsMobile. Link al sito: https://www.thingsmobile.com/it/business.
[47] *Interfaccia vocale: una realtà sempre più presente.* Geeks Academy. Link alla pagina: https://www.geeksacademy.it/articolo-28/l_interfaccia-vocale-e-la-figura-dello-ux-designer/.

particolare cura nelle fasi di realizzazione, implementazione e testing e, nei casi in cui il prodotto finale sia da destinarsi proprio ad individui affetti da uno o più tipi di disabilità, è necessario che vi sia una cooperazione attiva tra *UX Designers*, personale esperto nelle patologie *target* e professionisti operanti nel campo delle tecnologie assistive, per non rischiare di creare prodotti inefficaci. Come già accennato, le interfacce vocali costituiscono un potente metodo per interagire con dispositivi elettronici e sistemi ambientali ove risultino efficaci ma presentano, tuttavia, delle lacune, tra le quali spicca il fattore della loro sensibilità ai disturbi ambientali. Non sono, infatti, rari i casi nei quali risulta pressoché impossibile "svegliare" l'interfaccia o permetterle di captare determinati comandi, ad esempio quando questa presenta i microfoni già impegnati da voci di terzi o si trova nelle vicinanze di oggetti rumorosi o che emettono suoni (come la TV). Il canale, ove si propaga la voce, richiede dunque di essere sgombero da disturbi, per minimizzare le occasioni di fallimento delle operazioni: in caso di disabilità grave, sarebbe opportuno avvalersi – ove possibile – di strumentazioni più affidabili per, ad esempio, richieste di emergenza e disporre, in ogni caso, di un'alternativa a carico di un altro tipo di interfaccia per poter impartire i comandi, preferibilmente indipendente dalla rete Internet.

Quanto appena descritto rappresenta solo uno scorcio della miriade di soluzioni esistenti sul mercato dei prodotti atti a migliorare la qualità della vita di specifiche tipologie di utenti ma vuole illustrare, in linea generale, i principali tipi di sistemi che saranno citati successivamente a proposito del progetto che verrà presentato. Alcuni degli ausili trattati verranno inseriti, infatti, nel simulatore di cui si tratterà nei prossimi capitoli, al fine di renderne l'esperienza maggiormente immersiva per l'utente finale, al quale sarà data la possibilità di provarli, anche se in una realtà fittizia, allo scopo di sensibilizzarlo sulle problematiche mosse. Dopo aver analizzato le strategie che, dunque, permettono effettivamente di porre rimedio – in parte o totalmente – al problema delle barriere architettoniche, in particolare in ambiente domestico, nel prossimo capitolo verrà analizzato come queste possano essere riprodotte all'interno di un prodotto di RV (dedicato a persone normodotate) e riescano ad assolvere all'obiettivo di far meditare l'utente sulla difficoltà del compimento di alcune semplici azioni, permettendogli di immergersi nel ruolo di un portatore di disabilità motoria alle gambe.

Prima di addentrarsi nel vivo della presentazione dell'idea, di seguito verrà introdotto il concetto di Realtà Virtuale più nel dettaglio e saranno citati alcuni impieghi pratici, dove la stessa ha permesso un incentivo agli utenti e ha fornito un ausilio alla risoluzione di alcune problematiche.

3 La Realtà Virtuale: cenni storici e impieghi pratici

La *Realtà Virtuale* (RV o VR, dall'inglese *Virtual Reality*) rappresenta un tipo di esperienza dove l'utente percepisce se stesso immerso in un mondo simulato, tramite l'impiego di un'interfaccia *hardware* – come un casco di Realtà Virtuale con eventuali *controller* annessi, sensibili al movimento – e di un *software* che elabora i contenuti e li rende disponibili alla fruizione. Le radici della Realtà Virtuale è possibile trovarle già in un *visualizzatore stereoscopico* della fine degli anni Trenta, il View-Master (oggi rilanciato sul mercato da Mattel e Google[48]) e nel teatro multisensoriale Sensorama del 1959, ad opera del regista Morton Heilig; successivamente, nel 1968, viene creato il primo casco munito di *display* interni, simile agli attuali *HMD* (*head-mounted display*), per poi subire ulteriori evoluzioni a cavallo degli anni Settanta e Ottanta, periodo che coincide con l'avvio della rivoluzione digitale, il cui progresso diverrà poi esponenziale a ridosso dei Novanta. A partire proprio da questi anni, infatti, la RV diventa nota al pubblico e non è più mero oggetto di ricerca: anche se non ancora aperta al consumo di massa, a partire dal 1990 inizia a coinvolgere sempre più interessati, in particolare *videogamers*, grazie anche all'abbassamento dei costi necessari all'acquisto della strumentazione. In seguito, la Realtà Virtuale ha integrato numerose innovazioni, che consentono all'utente non solo di vedere gli ambienti proposti durante la sessione di gioco, ma anche di esplorarli, muovendosi fisicamente nello spazio e di potervi interagire, grazie all'implementazione di sensori impugnabili o delle ultime tecnologie in grado di rilevare il moto, come il nuovo *hand tracking* (supportata, ad esempio, negli ultimi visori di casa Oculus, Quest e Quest 2) che permette di registrare ogni singolo spostamento delle dita e di trasporlo nella simulazione. L'identificazione e la rilevazione dei movimenti corporei fa parte delle ultime frontiere che hanno permesso alla RV di diventare ancora più immersiva; una dei pionieri di tale tecnologia è la *startup* Leap Motion che, a partire dal 2010, si occupa di proporre innovazioni in questo campo e che, dal 2013, ha immesso sul mercato la sua periferica più conosciuta: il Leap. Questa, collegabile al PC via USB, utilizza due telecamere e tre LED a infrarossi per identificare dita e altri oggetti con una precisione di 0,01 mm, supporta una serie di *gestures* per l'interazione e risulta più idonea per il rilevamento di piccoli movimenti. È importante citare il suo contributo per quanto concerne

[48] *Google and Mattel Relaunch View-Master as Virtual Reality Headset*. 2015. eTeknik. Link alla pagina: https://www.eteknix.com/google-mattel-relaunch-view-master-virtual-reality-headset/.

il mondo della RV: da circa cinque anni Leap ha introdotto Orion[49], una tecnologia che si avvale di librerie software che permettono di poter sfruttare la periferica per progettare, tramite il *software* Unity[50], applicativi di Realtà Virtuale immersiva, dove è possibile utilizzare le mani per creare o interagire con gli oggetti nel simulatore. Un prodotto analogo che, invece, punta su aree di funzionamento più ampie e permette di registrare movimenti di più parti del corpo, anche se con meno accuratezza, è il Kinect di Microsoft[51], presentato con tale nome il 13 giugno 2013 all'Electronic Entertainment Expo; tale periferica, collegabile alla *console* XBOX della stessa casa o al PC via USB tramite opportuno supporto *software* e *hardware*, consentiva, nella sua prima versione, di rilevare tramite una telecamera a zone termiche i movimenti dell'intero corpo (fino a sei simultaneamente) e di seguirlo nello spazio mediante l'impiego di motori. L'accessorio, supportato fino al 2017, ha permesso lo sviluppo di numerosi videogiochi che ne presentassero il supporto e ha coinvolto anche un pubblico di ricercatori e amatori che hanno tentato di sfruttarne le caratteristiche, ai fini di applicarle in altri campi e applicativi, come proprio quello della RV, trovando il modo di serializzarne i dati rilevati e di trasmetterli al visore tramite la mediazione di *software* ad hoc ed emulatori[52]. A differenza del prodotto precedente, però, il Kinect si è rivelato essere di difficile integrazione con il mondo della RV e, data la dismissione del progetto, è possibile trovare solo esperimenti ormai obsoleti.

Dopo alcuni cenni storici e una breve postilla su alcuni sistemi inerenti il rilevamento del movimento del corpo ed eventuali, è necessario tornare a trattare le ulteriori caratteristiche e strategie di applicazione che caratterizzano il mondo della Realtà Virtuale, in particolare quelle riguardanti i più recenti sviluppi. In tempi attuali, essa ha coinvolto molteplici campi, da quello videoludico a quello del *design*, passando per ambiti più atipici, quali quelli educativo e della ricerca in campo medico-scientifico[53]; in ogni caso, è fondamentale, per creare una buona esperienza di RV, progettare considerando attivamente la prospettiva di chi

[49] *Leap Motion introduce Orion, tracciamento delle mani in VR di nuova generazione.* 17 febbraio 2016. A cura di Rosario Grasso. Quotidiano Nazionale. Link alla pagina: https://www.hwupgrade.it/news/wearables/leap-motion-introduce-orion-tracciamento-delle-mani-in-vr-di-nuova-generazione_61026.html.
[50] Unity. Link al sito: https://unity.com/.
[51] Kinect. Link al sito: https://developer.microsoft.com/it-it/windows/kinect/.
[52] *Oculus Rift + Kinect = follia psichedelica*. 2014. Vigamusmagazine. Link alla pagina: https://www.vigamusmagazine.com/138243/oculus-rift-kinect-follia-psichedelica/.
[53] Un esempio di impiego della RV nel campo della ricerca medica è quello proposto dall'azienda farmaceutica statunitense Pfizer, a proposito del quale è possibile consultare un filmato esplicativo alla seguente pagina web: https://www.pfizer.com/news/featured_stories/featured_stories_detail/how_virtual_reality_takes_scientists_inside_new_molecules.

la proverà. È necessario, a tal proposito, effettuare uno studio del *target* prima di procedere con ogni fase di sviluppo, perché è importante individuare a priori la fascia d'utenza a cui dedicare gli applicativi di RV, in modo da poterne adattare i contenuti e non correre il rischio di elaborare un prodotto inefficace nel suo obiettivo poiché, ad esempio, non accattivante o inadatto per età e/o requisiti psicologici e fisici. Nella RV, infatti, l'obiettivo è permettere ai fruitori di vivere una sorta di esistenza alternativa, attraverso la quale ciascun senso viene attivamente coinvolto; più il lato della percezione viene progettato con cura, più l'utente si sente isolato dal mondo reale e riesce a godere appieno dell'esperienza proposta[54]. Attualmente, il campo della Realtà Virtuale è spesso ancorato a quello della *Realtà Aumentata*[55] (*RA*), ovvero un insieme di caratteristiche che consentono, tramite la mediazione di un dispositivo multimediale, di arricchire la percezione sensoriale umana dell'ambiente circostante; la differenza sostanziale con la RV sta nella mancata possibilità di sentirsi totalmente immersi e isolati, poiché la RA permette di avere sempre un contatto con il mondo reale. Tuttavia, esistono circostanze in cui le due realtà sono facilmente assimilabili e, in questi casi, si parla di *Realtà Mista* (meglio conosciuta come *Extended Reality* abbreviabile in *XR*). Tornando alla Realtà Virtuale, oggetto di queste pagine, è possibile individuare tre generi di prodotto, ciascuno dei quali è classificabile a seconda dell'obiettivo e alla fascia di utenza da raggiungere. Il primo, riguarda elaborati di RV iper-immersiva, basati sulla percezione fisica e che, talvolta, possono anche prevedere – ad esempio – l'impiego di profumi per coinvolgere attivamente i sensi ed, eventualmente, amplificarne l'uso, tramite l'impiego di sensori e del *feedback* aptico. Il secondo, invece, si riferisce alla rosa di prodotti di Realtà Virtuale che contemplano la fruizione di ambienti realistici in prima persona, nei quali l'utente può muoversi per esplorarli, al fine di conoscerne le caratteristiche; tale tipo riguarda in particolare i simulatori a scopo didascalico di paesaggi naturali o musei virtuali. Il terzo, infine, rappresenta i videogiochi e le esperienze ludiche, dove il fruitore è invitato non solo a osservare, ma anche ad interagire attivamente con l'ambiente in RV e a concludere obiettivi e missioni; questa tipologia di prodotto è quella che si accosta di più al simulatore creato di cui si tratterà.

[54] *Virtual Reality. What is Virtual Reality? Interaction Design Foundation*. Link alla pagina: https://www.interaction-design.org/literature/topics/virtual-reality.
[55] *Realtà aumentata: cos'è, come funziona e ambiti applicativi in Italia.* Digital4. Link alla pagina: https://www.digital4.biz/executive/realta-aumentata-cose-come-funziona-e-ambiti-applicativi-in-italia/.

Inoltre come già anticipato, è necessario comprendere a priori la fisiologia e la psicologia umana, per capire i bisogni che ciascun individuo può manifestare, affinché l'esperienza di RV sia piacevole e non comporti sensazioni di fastidio o risulti inaccessibile. Per assolvere a quanto detto, bisogna concentrarsi su quattro requisiti chiave, da tenere in considerazione durante le varie fasi di sviluppo, riassumibili nei termini di *credibilità* (*believability*), *interattività* (*interactivity*), *esplorabilità* (*explorability*) e *immersività* (*immersiveness*). Un prodotto di RV, per essere *credibile*, deve presentare delle caratteristiche necessarie al coinvolgimento dell'utente, come ambientazioni 3D realistiche e suoni spazializzati; per rispondere, invece, al secondo requisito, quello dell'*interattività* deve, innanzitutto, avvalersi di un design intuitivo, utile ad eliminare la maggior parte delle interferenze esterne, ed è necessario ricreare con cura le azioni della vita reale, in rapporto alle caratteristiche fisiche e all'ambiente simulato. I concetti di *esplorabilità* e *immersività* rispondono, invece, rispettivamente, alle caratteristiche di assicurare agli utenti di potersi muovere senza vincoli all'interno della realtà offerta e di permettere loro di omogeneizzarsi negli spazi messi a disposizione e di provare una sensazione di isolamento dal mondo reale. È importante assicurarsi che il fruitore si senta al sicuro durante l'esperienza di Realtà Virtuale che sta provando: bisogna assicurarsi che non provi nausea, vertigini o capogiri dovuti ad un'alterata percezione sensoriale (come la nota *motion sickness*) dell'ambiente e che abbia spazio sufficiente per muoversi, al fine di evitare cadute o colpi a carico di terzi o di oggetti circostanti; per migliorare il comfort visivo, è altresì necessario regolare correttamente le caratteristiche video del visore, come la luminosità e il *framerate*. L'utente deve essere, altresì, istruito sul corretto utilizzo delle apparecchiature impiegate per entrare in contatto con la Realtà Virtuale e deve poter indossare e regolare al meglio ogni accessorio, al fine di evitare situazioni pericolose dovute ad eccessivo sforzo degli occhi, dei muscoli del collo e di altre parti del corpo; fortunatamente, secondo le attuali normative, ogni strumento deve essere obbligatoriamente corredato di schermate contenenti le istruzioni sulla corretta configurazione degli stessi. Va prestata, naturalmente, particolare cura nel caso di soggetti epilettici, ai quali la Realtà Virtuale è sconsigliata ma, in caso volessero comunque provarla, va donata peculiare importanza al contrasto visivo (che deve essere impostato al minimo) e vanno eliminate situazioni dove sono presenti effetti *stroboscopici*, noti per scatenare reazioni avverse in questi individui. Infine, per godere di una buona esperienza senza subire danni all'udito, è necessario regolare il volume in maniera idonea, senza superare soglie di

pericolo, per evitare di ignorare eventuali situazioni di emergenza nell'ambiente reale e prevenire sensazioni di confusione mentale al termine della simulazione.

In sintesi, esaminando le buone pratiche per ottenere un prodotto di RV efficace, è facilmente deducibile quanto il mancato rispetto di anche solo un requisito possa condurre alla realizzazione di un risultato inefficace che, se magari è associato anche ad una cattiva configurazione della strumentazione, può risultare nocivo all'utente.

Nonostante ciò, la Realtà Virtuale, se correttamente sfruttata, può aiutare davvero l'essere umano nella vita quotidiana: uno dei campi nei quali essa ha preso più piede è quello dell'*interior design,* che fino a pochi anni fa costituiva un mondo caratterizzato esclusivamente dall'artigianalità degli elementi manifatturieri che lo costituivano, nel quale gli unici elementi digitali utilizzati in fase di progettazione erano i computer e i relativi applicativi per la creazione di bozze e prototipi. Attualmente, il mondo del *design* di interni sta cambiando radicalmente e l'introduzione di nuove tecnologie che sfruttano la Realtà Virtuale sta avendo un impatto positivo sia sul lavoro dei tecnici, che sulla soddisfazione dei destinatari del prodotto finito: un utente che può provare in anteprima, ad esempio, come sarà il proprio soggiorno di casa a lavori ultimati, indossando un semplice casco di RV, riesce a comprendere con meno sforzo il risultato finale e, in caso di basso gradimento di alcune soluzioni, può facilitare l'esperto a trovarne di alternative. Per citare un esempio di ditta che si avvale di questa tecnologia e coinvolge attivamente coloro siano interessati ai loro prodotti, è Mondo Camerette[56], azienda italiana specializzata nella realizzazione di spazi dedicati ai bambini, che permette agli utenti di progettare e provare, in poche mosse, la cameretta perfetta, mediante l'utilizzo di un visore di Realtà Virtuale. Inoltre, la RV è largamente impiegata nel contesto di esibizioni museali ed è utile in particolare per ricostruire ambienti andati ormai distrutti nel tempo o per ricreare scenari di vita di altre epoche, come accade ne *La Macchina del Tempo,* il museo di Realtà Virtuale di Bologna[57]. In questo è possibile, infatti, mediante l'impiego di un visore di RV, entrare in contatto con paesaggi appartenenti ad altri tempi, curati maniacalmente nei dettagli e comprensivi di oggetti di arredo urbano, animali e popolazione abbigliata con abiti d'epoca, con la quale è possibile anche interagire.

[56] *Progetta la tua cameretta con la realtà virtuale*. MondoCamerette. Link alla pagina: https://www.mondocamerette.it/progetta-la-tua-cameretta-con-la-realta-virtuale/.
[57] La Macchina del Tempo. Sezione Esperienze. Link alla pagina: https://www.lamacchinadeltempo.eu/#esperienze.

Integrare la RV nell'ambito museale o, come in questo caso, dedicarvi l'intera visita, è sicuramente un modo per raggiungere più pubblico, in quanto l'offerta di vivere un'esperienza immersiva è sicuramente più accattivante di un tradizionale *tour* con operatore e audioguida. Dunque, il mondo dei beni culturali trae i suoi vantaggi dall'impiego della Realtà Virtuale per valorizzare ambienti e ricostruirne di perduti, donando agli utenti la possibilità di entrarvi in contatto; tuttavia, esistono campi nei quali tale tecnologia è anche parte integrante di un percorso che riesce a migliorare il tenore di vita di particolari soggetti. Ad esempio, se utilizzata in ambito medico, la RV può aiutare i pazienti a rilassarsi e a superare momenti stressanti delle terapie alle quali sono sottoposti, se si impiegano particolari contenuti audiovisivi in grado di interagire positivamente con il sistema nervoso; tale ausilio risulta utile in particolare con soggetti difficili da gestire sul lato psicologico, come i bambini, che sono solitamente spaventati dalle procedure mediche, anche se poco invasive, come prelievi di campioni di sangue o visite dentistiche. È altresì possibile utilizzare la RV a scopo didascalico, per mettere al corrente sulle relative operazioni coloro debbano subire particolari trattamenti medici, proponendo loro, ad esempio, simulatori immersivi dove è possibile vederne in anteprima le procedure. Inoltre, la Realtà Virtuale, sotto il nome di *Telepresenza Immersiva Virtuale* (*TIV*)[58], trova impiego nella riabilitazione, supportando il recupero funzionale delle abilità nei pazienti affetti da alcuni tipi di disturbi cognitivi e/o motori o che soffrono di ansia o siano vittima di stress e fobie. Questo tipo di ausilio consiste in sedute durante le quali vengono somministrati ai pazienti contenuti virtuali finalizzati al recupero delle funzionalità compromesse, che prevedono esercizi fisici e mentali suddivisi per grado di difficoltà; le attività vengono proposte tramite un dispositivo di Realtà Virtuale, che propone ambienti realistici ed interattivi, che riproducono situazioni di vita quotidiana. Rispetto alle tradizionali tecniche di riabilitazione, la TIV permette di situare l'esercizio riabilitativo in ambienti che riproducono al meglio le caratteristiche degli ambienti di vita e consente, dunque, di ampliare la risposta sensoriale del paziente; inoltre, la difficoltà delle attività proposte può essere, come già accennato, aumentata o diminuita dinamicamente in base alle abilità acquisite dal soggetto e questo semplifica, senz'altro, l'attività terapeutica.

[58] *Riabilitazione con Telepresenza Immersiva Virtuale*. Istituto Auxologico Italiano. Link alla pagina: https://www.auxologico.it/prestazione/riabilitazione-telepresenza-immersiva-virtuale.

Come si può dedurre dagli esempi riportati, la RV trova il suo impiego in numerosi campi, che possono interessare, tra i più disparati, arte, medicina, storia e natura; tuttavia, essa può costituire un'importante risorsa sul piano della sensibilizzazione, in particolare su problematiche trascurate di sovente dalla popolazione i cui punti critici sono, purtroppo, spesso marcati tramite campagne che utilizzano mezzi poco accattivanti. Nel prossimo paragrafo si scriverà di come la RV possa essere un prezioso ausilio, utile a smuovere gli animi della popolazione e riesca a proporre strategie che, in modo analogo a quanto trattato nel cap. 1, coinvolgano quante più possibili fasce d'utenza e, di conseguenza, favoriscano il successo delle campagne ad esse correlate.

3.1 La RV immersiva come strumento di sensibilizzazione

Come già accennato, la RV – specie se immersiva – può essere di ausilio quando si presenta la necessità di sensibilizzare su una tematica presentante lati sfavorevoli, poiché costituisce un sistema in grado di coinvolgere appieno l'utente, ponendolo di fronte a situazioni e scenari realistici, che possono spronarlo emotivamente e condurlo, dunque, ad assumere comportamenti in grado di eliminare determinati problemi. Una campagna di sensibilizzazione che impiega uno strumento meno convenzionale, quale è la Realtà Virtuale Immersiva, si differenzia dalle altre perché pone lo spettatore in condizione di osservare in prima persona le lacune inerenti un certo tema: se si vuole, ad esempio, porre l'accento sulle conseguenze delle azioni dell'uomo sull'ambiente, come ha di recente attuato WWF Italia[59], si verterà su un'applicazione che mostrerà, tramite scenari anche dai contenuti piuttosto forti, come il fenomeno del bracconaggio visto con gli occhi di un lupo, affinché l'utente sia scosso emotivamente e manifesti più rispetto nei confronti dell'animale. Puntare sull'empatia è, infatti, una strategia molto utilizzata nel campo della RV, poiché questo tipo di tecnologia consente più di altre di coinvolgere i cinque sensi e permette al fruitore di allineare la sua percezione spazio-temporale ai contenuti presentati e, dunque, di distaccarsi dalla realtà e di provarne una nuova, in prima persona e non da spettatore. Un progetto di RV che si avvale di ciò è *EXPERIENCE* (*The "EXtended-PErsonal Reality": augmented recording and transmission of virtual senses through artificial - IntellgENCE*)[60], un sistema del Centro di

[59] *Realtà Virtuale tra sensibilizzazione ed empatia: un progetto di Uqido e WWF.* 13 aprile 2017. A cura di Laura Fasano. Benessere Tecnologico. Link alla pagina: https://benesseretecnologico.it/realta-virtuale-uqido-wwf/.
[60] *Con il progetto EXPERIENCE la realtà virtuale entra nei social network.* 12 gennaio 2021. UNIPI News. Link alla pagina: https://www.unipi.it/index.php/news/item/19862-con-il-progetto-experience-la-realta-virtuale-entra-nei-social-network.

Ricerca E. Piaggio e del Dipartimento di Ingegneria dell'Informazione dell'Università di Pisa che permette agli utenti di provare e condividere emozioni sui social, tramite un visore di RV e un sistema in grado di rilevare e trasmettere segnali neurali e cardiovascolari, coordinati dall'intelligenza artificiale. Il progetto, anche se non punta esplicitamente a sensibilizzare i fruitori, si avvale comunque di un'ottima strategia per porre in rilievo le potenzialità della Realtà Virtuale e metterli al corrente del fatto che è possibile, indossando semplicemente un visore e poco più, anche provare quanto sentono gli altri e viceversa. Questa, se utilizzata per sollevare lacune inerenti ogni campo, può tentare di giungere oltre il confine dove un semplice video illustrativo o una conferenza possono arrivare: l'esperienza è parte integrante dell'apprendimento e, ove questa non sia attuabile concretamente, è possibile assolvere al suo ruolo simulandola tramite applicativi di RV immersiva, con contenuti e grafica quanto più prossimi alla realtà.

Dunque, se la RV permette di coinvolgere così attivamente gli utenti sul lato psicologico e percettivo (e su quello fisico, se si usano opportuni feedback, come quello aptico), perché non impiegarla per far provare loro anche situazioni spiacevoli, al fine di migliorarle?

Tale domanda è servita da sprone per ideare lo strumento che sarà presentato tra poco, in riferimento al problema delle barriere architettoniche, del quale si sono illustrate le caratteristiche e alcune delle metodologie per arginarlo: *è possibile sensibilizzare sulla gravità del problema, proponendo ai normodotati un'esperienza di RV immersiva nella quale possono rivestire il ruolo e provare le stesse sensazioni (talvolta frustranti) di un portatore di disabilità motoria?*

La risposta è sicuramente affermativa e, dopo un'attenta analisi di alcuni prodotti simili, sono state gettate le basi per un progetto – presentato di seguito – che assolvesse al suo compito: sensibilizzare sulle difficoltà che certe categorie di soggetti incontrano quotidianamente, interfacciandosi con un mondo fatto su misura di coloro che sono in grado deambulare sulle proprie gambe e muoversi agevolmente anche tra eventuali ostacoli, proponendo allo stesso tempo delle soluzioni per tamponarle, il tutto utilizzando un'impronta ludica per ottenere maggior coinvolgimento.

3.1.1 L'idea del simulatore di RV Immersiva

Di seguito verranno descritte le motivazioni che hanno condotto alla creazione di un simulatore di RV immersiva, del quale verranno trattate le principali caratteristiche e le fasi di sviluppo che l'hanno condotto all'attuale versione. Innanzitutto, è necessario affermare che l'idea della generazione di un prodotto di Realtà Virtuale di questo tipo è stata lanciata dalla mancanza di strumenti analoghi sul mercato e resa concretizzabile grazie alla collaborazione con Laboratorio di Robotica Percettiva (PercRo) della Scuola Universitaria Superiore Sant'Anna di Pisa[61], che ha fornito gli strumenti necessari – in particolare in fase di avvio – per lo sviluppo delle applicazioni di Realtà Virtuale. Il lavoro in oggetto è il risultato di una serie di sviluppi avvenuti in ambito accademico ed è nato dal bisogno di proporre una metodologia di sensibilizzazione, efficace ed alternativa, al problema delle barriere architettoniche, in grado di raggiungere tutti e che non richiedesse, obbligatoriamente, un intermediario.

3.1.1.1 Target e scopo

Prima di iniziare, sono stati stabiliti il target del prodotto e il suo scopo: inizialmente, si era ipotizzato di creare un *tool* in grado di facilitare la realizzazione di ambienti domestici e fosse indirizzato, dunque, ad esperti di *interior design*, idea presto scartata poiché non si possedevano i giusti requisiti per dare istruzioni a professionisti del settore ed esistevano già strategie atte a semplificare il lavoro di progettazione di interni che impiegassero la grafica 3D e la RV. Successivamente, si è consolidata l'idea di vertere su uno strumento di sensibilizzazione, che coinvolgesse il comune utente e lo ponesse di fronte alla spiacevole realtà che caratterizza la vita di coloro che presentano un deficit riguardante lo spettro motorio. Durante scelta del *target*, non è stato impostato un discrimine in grado di scartare nettamente una parte di utenza, poiché il simulatore è nato con l'idea di essere accessibile a chiunque volesse provarlo; tuttavia, lo strumento è progettato per essere destinato a soggetti normodotati o che possano, in ogni caso, deambulare liberi da vincoli o attrezzature ausiliarie. Parallelamente alla scelta della fascia, è stata individuata la tipologia di disabilità sulla quale vertere per far leva sulle problematiche: è stata presa in oggetto quella motoria riguardante la porzione inferiore del corpo, poiché altamente invalidante, in quanto

[61] TECIP - Perceptual Robotics - PERCRO LAB. Scuola Universitaria Superiore Sant'Anna. Link al sito: https://www.santannapisa.it/it/istituto/tecip/perceptual-robotics-percro-lab.

impedisce la deambulazione e costringe i soggetti che ne sono portatori all'impiego di sedie a rotelle o ausili per lo spostamento equivalenti. In seguito, è stato identificato l'ambiente da ricreare, ove sviluppare l'intreccio del simulatore: si è scelto di incentrarsi su quello domestico, in quanto scarsamente trattato nei progetti fruibili online e perfettamente in linea con le soluzioni da integrarvi, riguardanti ad esempio il mondo dell'automazione.

4 Come progettare il simulatore: fase preliminare

Nei prossimi capitoli sarà presentato il simulatore di RV immersiva, proposto come strumento di sensibilizzazione sul problema delle barriere architettoniche e creato partendo dallo studio degli strumenti simili sopra illustrato. Il prodotto è stato creato nel tentativo di proporre una soluzione in grado di utilizzare una strategia alternativa di comunicazione quale è la Realtà Virtuale, con lo scopo di offrire all'utente l'opportunità di porsi direttamente di fronte alle problematiche sollevate, in un modo quanto più naturale e di fornirgli la possibilità di interfacciarsi in una maniera quanto più immersiva con un mondo visto attraverso gli occhi di una persona con caratteristiche fisiche e modi di percepire lo spazio differenti. Tuttavia, prima di addentrarsi nel pieno dell'architettura dello stesso e illustrarne i contenuti e l'implementazione, bisogna trattare ciò che è necessario fare a priori, poiché si tratta di un prodotto di creazione e fruizione non immediati e che richiede dei requisiti di cui disporre in anticipo, tra i quali una strumentazione dedicata.

4.1 Prerequisiti

Al fine di preparare un buon campo di lavoro, è necessario essere al corrente delle conoscenze e dei requisiti dei quali bisogna disporre per poter procedere allo sviluppo di un simulatore di Realtà Virtuale e per poterne fruire al meglio durante le fasi di realizzazione e *testing*. A tal proposito saranno, innanzitutto, affrontate le competenze da possedere al momento della messa all'opera e, successivamente, verranno trattati i requisiti *hardware* minimi che la macchina ospite dovrà possedere e il tipo di strumentazione per l'accesso alla RV da utilizzare; infine, verranno individuate le piattaforme e i software da impiegare per procedere nel lavoro.

4.1.1 Competenze informatiche

In primo luogo, è consigliato possedere qualche conoscenza nel campo della programmazione informatica, specialmente di linguaggi orientati agli oggetti, come il C#[62], linguaggio sviluppato da Microsoft e impiegato in Unity per la creazione degli *script* dedicati alla parte dinamica del simulatore; non è, comunque, obbligatorio aver avuto esperienze pregresse nel campo, poiché esiste un'enorme documentazione a riguardo, ma conoscerne la logica può senz'altro aiutare. Per la realizzazione di videogiochi e simulatori simili a quello

[62] Documentazione di C#. Link alla pagina: https://docs.microsoft.com/it-it/dotnet/csharp/.

in oggetto e per varie piattaforme, tuttavia, per l'implementazione delle funzioni di base è sufficiente conoscere il funzionamento della classe principale dalla quale derivano tutti gli *script* e i componenti di Unity: la *MonoBehaviour*[63], della quale si tratterà successivamente. È, inoltre, necessario avere un'infarinatura generale sul software Unity (e sulla terminologia proprietaria degli elementi dell'*editor*) o affini e sulle principali componenti che costituiscono un ambiente tridimensionale, come materiali, *mesh di poligoni*, *texture* e *shaders.* Oltre quanto detto, non bisogna possedere ulteriori prerequisiti o imparare nuove strategie di programmazione, poiché è sempre possibile consultare, come già accennato, l'ampia documentazione disponibile online, la quale comprende anche un ricco panorama di esempi e, sulla piattaforma Unity Learn[64], sono disponibili numerosi *tutorial* dedicati ai neofiti dello sviluppo di videogiochi e affini.

È indispensabile, tuttavia, disporre di una macchina (PC) con caratteristiche *hardware* superiori alla media e di una strumentazione apposita allo sviluppo di strumenti di grafica 3D e di RV, poiché i *software* da utilizzare sono esosi di risorse e le operazioni di *rendering* richiedono dei requisiti minimi da rispettare per poterli supportare. Inoltre, è fondamentale possedere un visore di Realtà Virtuale per poter effettuare le prova necessarie; per il lavoro in questione è stato impiegato un Oculus Quest, con relativo programma pilota, *driver* e *plugin* per l'integrazione in Unity.

4.1.2 Requisiti hardware e specifiche tecniche

Come accennato, bisogna disporre di una macchina con delle buone caratteristiche hardware e che presenti, in particolare, un ottimo processore grafico, scelta indispensabile per permettere l'esecuzione di contenuti ad alta risoluzione. Il panorama dei PC da poter utilizzare è pressoché infinito: la maggior parte dei computer da *gaming*, come quelli di casa MSI[65] e HP della serie Omen[66], ad esempio, assolvono bene allo scopo, poiché sono dotati di schede video potenti (come quelle della serie GTX o RTX 20 della Nvidia[67]), di CPU performanti (ad esempio, i processori Intel I9 da 5.3 GHz) e dispongono di banchi di RAM di tipo DDR4 da 16-32GB in poi, oltre che possedere ottimi sistemi audio, in grado anche di

[63] MonoBehaviour. Link alla pagina: https://docs.unity3d.com/ScriptReference/MonoBehaviour.html.
[64] Unity Learn. Link al sito web: https://learn.unity.com/.
[65] MSI. Link al sito web: https://it.msi.com/.
[66] HP Omen. Link alla pagina: https://www.hp.com/us-en/shop/slp/omen-gaming.
[67] Nvidia. Link al sito: https://www.nvidia.com/it-it/.

spazializzare il suono; è possibile, inoltre, utilizzare PC *custom*, assemblati in base alle proprie preferenze. In sintesi, è importante che questi dispongano di determinati requisiti minimi, che è possibile stabilire analizzando quelli richiesti dal *software* Oculus[68], indispensabile a stabilire un collegamento tra il visore e computer: sarà necessario usare una macchina con almeno 3 GB di memoria video, un processore a partire da 2.5-3GhZ di *clock* e una RAM da 16 GB di capacità. Inoltre, la macchina dovrà disporre di almeno una porta USB 3.0 o di tipo C per potervi collegare *l'HMD* Oculus (previo acquisto dell'apposito cavo ad alta velocità). Nel particolare caso del simulatore in oggetto, è stato impiegato un MSI Leopard 95E, un *notebook* da gaming della casa taiwanese dalla buona performance, ben rispondente alle caratteristiche sopra citate ma che non desta, tuttavia, di una componentistica tipica degli ultimi top di gamma della stessa tipologia. È preferibile, infine, dotare la postazione di almeno due *display*, per poter programmare gli *script* o consultare la documentazione senza ridurre a icona Unity ed altri *software* in uso per lo sviluppo.

4.1.3 Requisiti software e programmi

In questa sezione verranno specificati i requisiti *software* necessari all'implementazione e alla fruizione di un simulatore di RV come quello in oggetto. In primo luogo, è necessario premettere che bisogna disporre del sistema operativo Windows 10, poiché Oculus, dal 2018, non supporta più le versioni 7, 8 e 8.1; non è possibile, inoltre, utilizzare una macchina con installati MacOS o una distribuzione Linux, poiché non sono disponibili versioni del *software* e dei *driver* dedicati a queste piattaforme.

È inoltre necessario disporre di due *tools* che rappresentano il cuore dello sviluppo del simulatore: Unity e un buon *ambiente di sviluppo integrato* (*IDE*), a scelta tra Visual Studio Code[69], MonoDevelop[70] o prodotti equivalenti; per la scrittura di gran parte degli *script* realizzati è stato impiegato il primo della Microsoft, ma è necessario donare voce in capitolo anche al secondo, poiché, oltre ad essere *open source* e multipiattaforma, è realizzato principalmente per il linguaggio C#, del quale contiene già molti suggerimenti di sintassi dedicati.

[68] Requisiti minimi per il software Oculus per PC. Link alla pagina: https://support.oculus.com/
[69] Visual Studio Code. Link al sito web: https://code.visualstudio.com/
[70] MonoDevelop. Link al sito web: https://www.monodevelop.com/

4.1.3.1 Unity: un motore grafico multipiattaforma

Unity, conosciuto anche con il nome Unity3D, è un ambiente e motore – *engine* – grafico che, a partire dal 2005, anno del suo primo rilascio, ha acquistato grande popolarità grazie alla sua caratteristica di essere non solo disponibile per diverse piattaforme (tra le quali i s. o. *Unix based* e di casa Microsoft), ma anche di fornire la possibilità agli utenti, in modo piuttosto semplice, di creare *videogames* e applicazioni (anche destinate al reparto *mobile*, per sistemi quali Android o iOS e altri, come BlackBerry OS, ormai giunti al capolinea), grazie alla sua interfaccia altamente personalizzabile e alla possibilità di integrarvi numerosi *plugin*, al fine di estenderne le potenzialità e il supporto con strumenti di terze parti. Sebbene esso non costituisca un eccellente sistema per la produzione di elaborati di altissimo livello, poiché non adatto alla creazione di contenuti ad alta performance o qualità di rendering, presenta numerosi vantaggi, tra i quali spiccano la sua gratuità (è disponibile anche una versione "pro", ma non utile al presente scopo), il supporto di più linguaggi di programmazione (C#, JavaScript e Boo), la sua intuitività di utilizzo e, non da meno, l'enorme supporto in termini di documentazione, disponibile online e a cura di canali ufficiali e non. Inoltre, come già accennato, supporta più piattaforme e consente di produrre svariati tipi di contenuto, da semplici *videogames* in stile *platform* o RPG, alle GUI (*Graphic User Interface*, interfaccia grafica) di applicazioni 2D, ai *serious games* a fini educativi, fino ad ambienti virtuali 3D e di RV immersiva. Grazie proprio al vasto supporto relativo a quest'ultima e alla disponibilità di numerosi esempi di progetti dedicati in rete, è stato possibile realizzare il simulatore, essendo disponibili numerose estensioni compatibili con il visore Oculus impiegato, tra le quali spiccano alcuni *plugin* proprietari (e non, come le Oculus Integrations per XR, di cui si tratterà più avanti) che permettono di comunicare con il software Oculus e provare il prodotto durante lo sviluppo e, naturalmente, creare applicazioni dedicate.

Per la realizzazione del simulatore nell'ultima versione, sono state impiegate le *release* LTS del software (*Long Term Support*, ovvero supportate per patch e aggiornamenti per più tempo e, di conseguenza, più solide) relative all'anno 2020 (figura 1), per poter usufruire delle ultime migliorie messe a disposizione dagli sviluppatori e, in particolare, per includere una versione stabile dei *plugin* XR – dei quali si tratterà più avanti –, in supporto *beta* per le precedenti uscite.

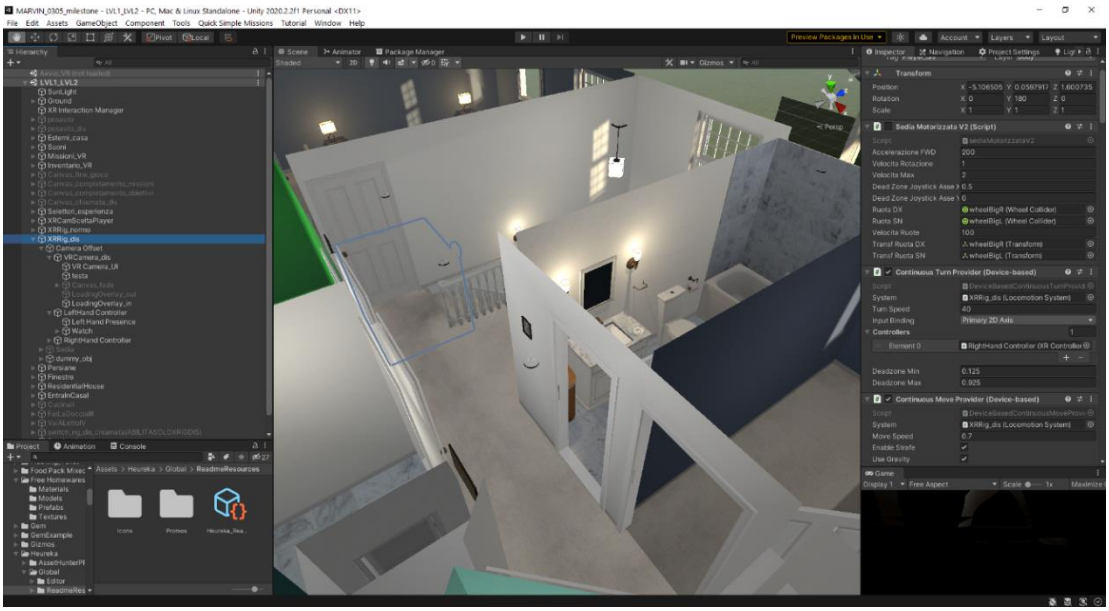

Figura 1. Unity – Versione 2020.2.2f1 in esecuzione su Windows 10.

4.1.3.2 Visual Studio Code e MonoDevelop

In questa sezione saranno trattati due *editor*, dei quali è necessario disporre per lo sviluppo degli *script* per Unity in C#; è possibile, come già accennato, scegliere di utilizzarne soltanto uno, poiché assolvono entrambi allo stesso compito: produrre del codice ben indentato e fornire suggerimenti di completamento automatico durante la digitazione, al fine di velocizzarne la stesura e limitare, dunque, la possibilità di ottenere errori. Entrambi sono selezionabili come *editor* predefinito in Unity e, oltre ad offrire un'ampia rosa di estensioni per l'integrazione con altri linguaggi, scaricabili su richiesta, sono dotati di differenti opzioni di configurazione dell'interfaccia, al fine di offrire un buon *comfort* visivo. MonoDevelop (nella figura a seguire), impiegato solo in una prima fase di sviluppo, è un IDE *open source*

appartenente al progetto Mono[71], quasi interamente dedicato allo sviluppo in C# e altri linguaggi proprietari di Microsoft, appartenenti alla piattaforma .NET.

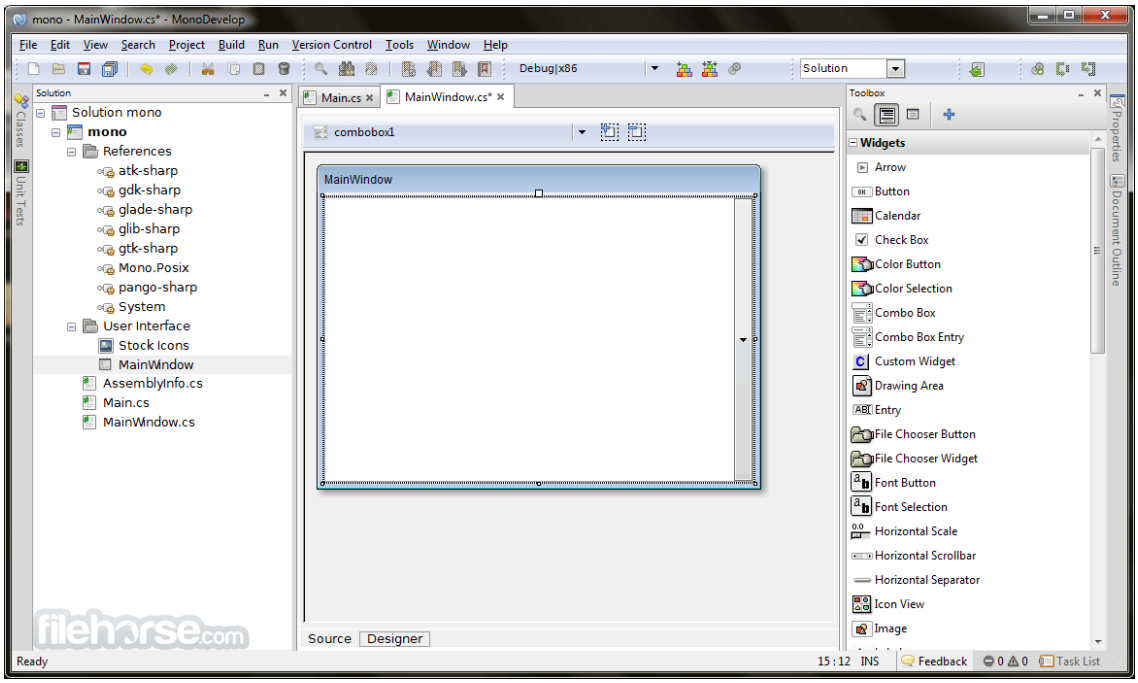

Figura 2. MonoDevelop – Versione 8.4.3.12 su Windows 10.

Uno dei suoi punti di forza è rappresentato dalla disponibilità di versioni per i sistemi operativi MacOS e Linux, oltre che per Windows, e permette, come il suo concorrente, di importare l'intero progetto al suo interno, per avere una visione generale dell'albero gerarchico delle classi e delle sue estensioni sulle quali si sta lavorando. Nonostante i lati positivi del *software*, tra i quali la vasta offerta di plugin, questo presentava alcuni *bug* di compatibilità con il s. o. Windows 10 (alla versione impiegata all'inizio dello sviluppo del simulatore) e, pertanto, è stato presto rimpiazzato da Visual Studio Code (anche detto VSCode), un *editor* sotto licenza proprietaria di Microsoft per i linguaggi .NET, più completo sul lato *debugging* e che dispone di varie estensioni che consentono all'utente di fruire di contenuti aggiuntivi, tra i quali il *software di controllo di versione* Git[72], che permette di gestire al meglio il codice e le relative revisioni. Rispetto a MonoDevelop, VSCode ha il vantaggio di essere supportato in modo diretto da Windows Update che propone,

[71] Progetto Mono. Link al sito: https://www.mono-project.com/.
[72] Git. Link al sito: https://git-scm.com/.

periodicamente, degli aggiornamenti dedicati, installabili direttamente dal *tool* e tramite una sezione in-app dedicata.

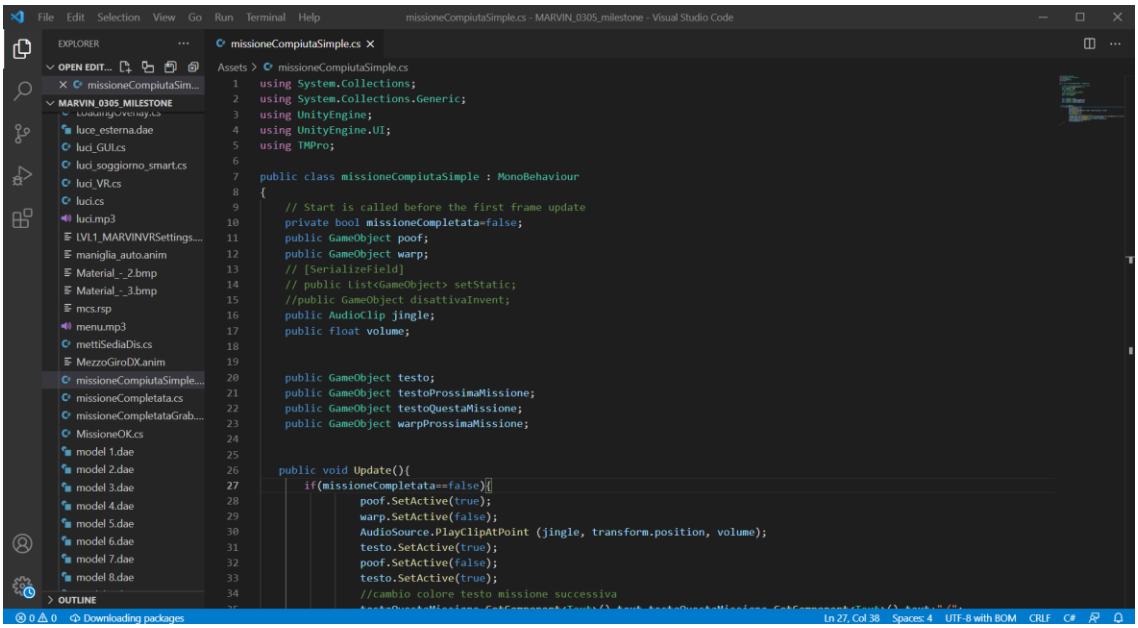

Figura 3. Visual Studio Code – Versione 1.55.2 su Windows 10

È necessario, tuttavia, affermare che non esiste una scelta univoca in termini di IDE: è sempre preferibile utilizzare quello che offre il miglior compromesso tra *comfort* visivo e funzionalità, senza escludere il lato *performance*: dovendo fruirne in affiancamento a *software* piuttosto dispendiosi di risorse quali Unity e il software Oculus, è consigliabile vertere su scelte non troppo "pesanti", per evitare di riempire la memoria RAM della macchina e di occuparne la CPU in processi impegnativi.

4.1.3.3 Il software Oculus

Per rendere attuabile la comunicazione tra il PC e il visore Oculus (nelle versioni Rift, Rift S, Quest e Quest 2) e la relativa fruizione tramite le applicazioni di Realtà Virtuale che lo supportano, è necessario installare il *software* Oculus (figura 4) sul PC, distribuito dalla stessa casa costruttrice. Come si vedrà nella prossima sezione, nel caso dei modelli Quest (utilizzato per lo sviluppo del simulatore in oggetto) e Quest 2, sarà necessario disporre di un accessorio, il cavo ad alta velocità Oculus Link, per connettere la macchina al visore e, dunque, di comunicare con il *software* dedicato, essendo questo un dispositivo *standalone*, in grado di

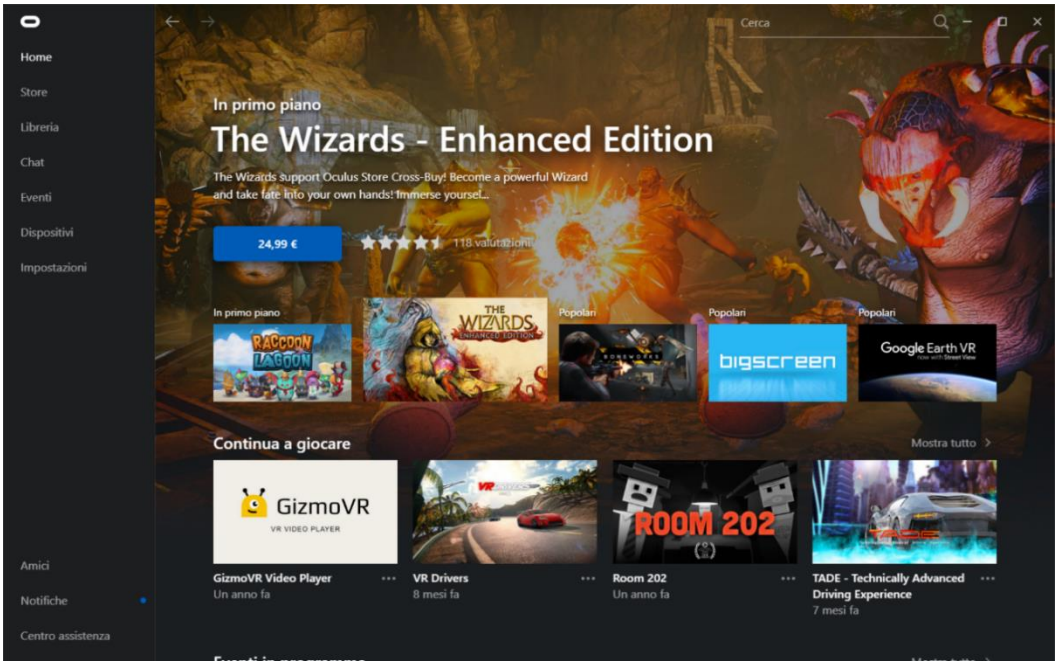

Figura 4. Oculus Software – Versione 29.0.0.54.528 su Windows 10

funzionare principalmente con applicazioni ivi installate. L'applicazione, che include i *driver* di comunicazione e dispone di uno *store* con numerosi titoli di *videogame* e simulatori da poter acquistare, include anche una procedura guidata molto curata per la connessione del visore al PC, tramite la quale è possibile configurare in modo efficiente e sicuro il visore, i *controller* ed eventuali sensori e telecomandi, se presenti. Permette, inoltre, di prendere nota di eventi inerenti esperienze dedicate alla RV e di contattare i propri amici tramite il sistema social che include, basato sulle tecnologie di Facebook. L'applicazione è fondamentale per utilizzare il visore con Unity e altre applicazioni per lo sviluppo e include anche delle funzioni per sviluppatori, oltre che un collegamento diretto alla *community*, che si avvale di appassionati, *videogamers* e *developers*.

4.1.4 Requisiti per la RV: il visore di RV

Dopo aver illustrato i requisiti software e hardware che il computer deve possedere prima di addentrarsi nello sviluppo di un'applicazione di RV, è necessario trattare la periferica che permette di accedere ai contenuti ad essa relativi: il visore di Realtà Virtuale. Come citato in precedenza, il mercato offre numerose soluzioni in grado di fornire all'utente una buona esperienza di RV: esistono visori di tutti i generi e per tutti i gusti, dai più economici nei quali è possibile posizionare lo smartphone, ai prodotti *standalone*, a progetti più interessanti come Google Cardboard[73], a interfacce più complete come, appunto, quelle di casa Oculus, come il Quest, impiegato nello svolgimento del presente progetto. La scelta di questo strumento è

[73] Google Cardboard. Link al sito: https://arvr.google.com/intl/it_it/cardboard/get-cardboard/.

stata effettuata in base alle caratteristiche *hardware* dello stesso in quanto dotato di una risoluzione di 1600x1400 (per occhio) con una frequenza di aggiornamento di 72 Hz e un processore Qualcomm Snapdragon 835, ma anche di una memoria di archiviazione interna, di 64 GB per il modello acquistato. Infatti, rispetto agli altri modelli, il visore include la possibilità di eseguire applicazioni installate direttamente sullo stesso, in quanto dispone (come il "fratello minore", l'Oculus Go) di un sistema operativo in grado di supportare giochi sviluppati *ad-hoc*; è possibile, pertanto, sviluppare anche *app standalone* dedicate (trasferibili tramite *sideload* sullo stesso), tramite l'ausilio delle librerie Android e dell'IDE Android Studio[74], anche se non se ne tratterà nei prossimi capitoli. Nonostante non sia stata prevista la creazione di un simulatore direttamente fruibile dal visore, è stato scelto un modello che assolvesse alla funzione appena descritta per eventuali sviluppi futuri del progetto; la scelta non è ricaduta sul più aggiornato Quest 2 in quanto, al momento del concepimento dell'idea, non era ancora stato lanciato sul mercato. Per i precedenti sviluppi era stato utilizzato un Oculus Rift; tuttavia, sebbene lo stesso risultasse più leggero e maneggevole e interamente dedicato alla fruizione di contenuti eseguibili sul PC, la scelta è ricaduta sul Quest poiché dispone, a differenza degli altri modelli, di una funzione di *hand tracking*, che consente all'utente di abbandonare i *controllers* e utilizzare direttamente le mani negli applicativi di RV e libera dall'ingombro dei sensori esterni, essendo dotato di telecamere ad infrarossi in grado di rilevare attivamente la posizione nello spazio. Tuttavia, la funzione di tracciamento delle mani, inclusa inizialmente nell'idea del prototipo di simulatore, è stata progressivamente scartata, poiché di difficile integrazione nel progetto e non ancora precisa, essendo tuttora in sviluppo. Per la fruizione del progetto finale, pertanto, è necessario essere in possesso di un visore di casa Oculus e non si è, dunque, vincolati univocamente alla scelta del Quest: l'unico requisito necessario è utilizzare un prodotto compatibile con le tecnologie contenute nella *build* e di *controller* altrettanto in linea con la mappatura definita in corso d'opera.

Di seguito, sarà illustrato brevemente come gestire lo spazio fisico e il sistema di controllo del visore, al fine di mettere in sicurezza un'area dove poter sviluppare senza il pericolo di

[74] Android Studio. Link al sito: https://developer.oculus.com/documentation/native/android/mobile-studio-setup-android/.

incappare in ostacoli fisici e sarà descritto, al contempo, come ottenere una configurazione ottimale del sistema di controllo del visore in merito all'ambiente circostante.

4.1.4.1 Organizzazione della postazione fisica e configurazione del sistema di controllo

Una volta effettuata la configurazione della macchina e il collegamento del visore alla stessa in modalità Link, è necessario delimitare i confini entro i quali si svolgerà la simulazione: tale settaggio è disponibile nel *wizard* di Oculus, ma richiede anche delle accortezze da seguire in base allo spazio dove si svolgerà la simulazione. In primis, bisogna scegliere una postazione libera da eventuali ostacoli fisici: non devono esserci oggetti posizionati in terra entro pochi metri e l'utente deve essere libero di muoversi. Per la progettazione e la fruizione del simulatore, non è tuttavia necessario disporre di molto spazio, poiché durante l'esperienza il giocatore può restare fermo, in quanto il movimento è quasi sempre impartito tramite il joystick (*thumbstick*) di un *controller*. Per tale ragione, sarà sufficiente stabilire un confine virtuale del *guardian* (sistema di Oculus che prevede una "griglia" entro la quale si più fruire dell'ambiente virtuale), limitato alle sole circostanze immediate dell'utente, oppure scegliere dei confini statici; bisognerà inoltre impostare un livello ottimale del pavimento: esso deve coincidere con il reale livello del suolo per evitare un'errata altezza degli oggetti di gioco durante la simulazione. È altresì richiesto uno spazio sufficiente al livello delle braccia, in quanto l'esperienza è costituita in gran parte da interazioni, dunque, anche in fase di progettazione, è necessario impostare lo spazio di lavoro a qualche decina di centimetri di distanza da PC e scrivania. In varie fasi dello sviluppo e della fruizione, sarà anche necessario doversi riposizionare poiché, come si potrà osservare nel prossimo capitolo, sono previsti livelli nei quali bisognerà sedersi e uno in cui si dovrà restare in piedi; sarà comunque possibile, in ogni momento dell'esperienza, poter riconfigurare lo spazio del *guardian* e modificare le impostazioni relative al sistema di controllo. Onde evitare la perdita del *tracking*, è necessario prestare attenzione anche alla luminosità ambientale, in quanto una stanza troppo scura può interferire negativamente sull'esperienza e, non meno importante, bisogna tenersi alla larga quanto più possibile da interferenze di onde radio e infrarossi, in quanto potrebbero interferire con i sensori dell'Oculus e con i *controllers*. È, infine, necessario utilizzare un cavo piuttosto lungo per il collegamento del visore al PC: uno troppo corto potrebbe creare intralcio e addirittura danneggiare le porte di collegamento se strattonato; il cavo originale Oculus Link assolve perfettamente alla funzione.

Una volta descritti i prerequisiti e gli strumenti di cui disporre per preparare la macchina ad accogliere la RV, sia in fase di progettazione che di fruizione, è possibile illustrare i dettagli e l'architettura del simulatore per passare, successivamente, a trattare quanto concerne la sua implementazione.

5 Il simulatore: architettura e implementazione

Premessa

Il lavoro svolto si avvale di molti script e di innumerevoli dettagli che, talvolta, risultano impossibili da descrivere nella più totale minuzia o rappresentare con precisione (anche perché appartenenti a contenuti di Realtà Virtuale); tuttavia, gli argomenti trattati verranno approfonditi quanto più possibile in maniera accurata.

Il simulatore creato alla luce delle problematiche mosse nei capitoli precedenti costituisce un prodotto destinato ad un *target* di utenti normodotati e vuole assolvere allo scopo di sensibilizzarli riguardo gli attuali disagi in relazione alle barriere architettoniche, ponendoli di fronte a situazioni che tentano di includere, quanto più realisticamente possibile, le difficoltà che dei soggetti portatori di disabilità motoria su carrozzina sono costretti a vivere nella vita quotidiana, dovendosi interfacciare con un mondo creato su misura di persone senza tale limite. Il prodotto vuole, altresì, proporre delle soluzioni ai problemi mossi: per il fruitore, sarà possibile vivere un'esperienza di RV immersiva non solo in ambienti sfavorevoli alla categoria di utenti in oggetto, ma in altrettanti dotati di strumenti in grado di facilitare alcuni tipi di azioni. Nel simulatore sono infatti previsti tre livelli che si sviluppano all'interno di un'abitazione privata su due piani, in seguito descritti, ognuno rappresentante una situazione ben precisa, entro i quali il fruitore dovrà completare alcune missioni, molto semplici e che rappresentano azioni quotidiane di base, ma che si riveleranno però essere piuttosto frustranti e complesse da portare a termine, specie nella modalità che prevede un'ambiente non accessibile e dove il giocatore sarà costretto a restare seduto (virtualmente) su una sedia a rotelle. Nell'ultimo di questi, tuttavia, saranno presenti degli strumenti e delle caratteristiche in grado di favorire l'inclusività: sarà possibile vivere nei panni di un portatore di disabilità dello spettro motorio su carrozzina, ma di accedere in modo più facile alle varie stanze della casa e di terminare con più semplicità le missioni, aiutati da alcuni ausili assistivi e dalla domotica, in totale autonomia, senza provare lo stesso senso di inadeguatezza e frustrazione che caratterizzeranno, invece, il secondo livello. Alla base del lavoro non vi sono particolari strumenti già esistenti, come già spiegato (tranne l'analisi di prodotti simili disponibili nel panorama della RV immersiva), ma sono incluse le basi gettate in progetti simili svolti in ambito accademico, con i quali condivide le fondamenta e dei quali il simulatore se ne considera un'evoluzione.

Tali sviluppi, descritti di seguito, hanno permesso al prodotto finale di giungere ad una versione più solida e sono stati fondamentali per imparare le basi necessarie all'elaborazione di un contenuto di RV più elaborato quale è il prodotto finale.

Curiosità sul nome e sul logo

Il simulatore, dalle sue origini, porta il nome in codice *MARVIN*, acronimo di ***M****odello di* ***A****bitazione per soggetti a* ***R****idotta Capacità di Mo****VI****meN****to*. Il motivo risale al primo sviluppo (che non supportava la RV), che includeva alcuni *easter eggs* e contenuti grafici tratti dal romanzo fantascientifico umoristico *Guida Galattica per gli Autostoppisti*[75] di Douglas Adams e tale nome rappresentava un riferimento diretto all'androide paranoico Marvin, protagonista di parte delle vicende del libro. Sebbene sia stata effettuata un'opera di "sbrandizzazione", man mano che il progetto cresceva, alcuni elementi "a tema" sono rimasti nel simulatore fino all'attuale versione, tra i quali il logo dell'icona della *build*, presente anche nel menu di avvio. Questo, infatti, contiene le immagini stilizzate di una persona su carrozzina e di uno smartphone con una traccia color *bordeaux* che, se osservate più da vicino, costituiscono la sagoma del numero 42, riferimento alla "risposta fondamentale sulla vita, l'universo e tutto quanto", tratta dal romanzo di Adams. Nonostante non esista una ragione precisa che giustifica il mantenimento della denominazione nel tempo, è stato scelto di conservare l'acronimo durante lo sviluppo, che nella resa grafica dell'ultima *release* riporta evidenziate le lettere centrali della parola MA**RV**IN, a marcare l'impiego della Realtà Virtuale prevista nella stessa.

5.1 Sviluppi precedenti

Fin dalle prime bozze, ci si è incentrati sull'idea di casa privata, poiché non è stato possibile reperire in rete progetti che si focalizzassero su questa tipologia di luoghi e, in particolare, che ponessero l'accento su soluzioni che si avvalessero delle più moderne tecnologie per semplificare la vita di soggetti in carrozzina. Inizialmente, il progetto consisteva in una applicazione in 3D sviluppata mediante il *software-framework* Unity, con il mero scopo di dimostrare, tramite simulazione in prima persona, come è visto il mondo da un soggetto con limiti nella deambulazione e proporre alcune strategie atte al miglioramento della sua vita. L'esperienza caratterizzante questo primo prototipo, presentato in occasione dell'esame di

[75] Adams, Douglas. 2016. *Guida Galattica per gli Autostoppisti*. Ed. tascabile. Milano, Mondadori.

Ambienti Virtuali, consisteva nella navigazione libera tra zona giorno, zona notte e il bagno di un piccolo bilocale situato a piano terra e nell'interazione con elementi dinamici, quali interruttori e telecomandi, nelle vesti di un uomo su sedia a rotelle. Il piccolo appartamento in 3D, con arredi creati su misura e ausili additivi quali rampe e sollevatori, rispondenti quanto più possibile alle caratteristiche trattate nel cap. 2, includeva anche un sistema di automazione, con il quale il personaggio poteva controllare i vari elementi della casa. Tra quelli caratterizzanti, spiccava un rudimentale sistema di controllo vocale, integrato tramite l'*SDK* di *IBM Watson*[76], attraverso il quale era possibile chiedere, ad esempio, di accendere o spegnere le luci o controllare le finestre e i condizionatori; tuttavia, questo *plugin*, ancora acerbo al tempo (2019), presentava numerose lacune sui lati *client* e *server* e non includeva ancora la lingua italiana e, per tale ragione, è stato eliminato a partire dalle release successive del simulatore. Inoltre, era possibile mostrare a video uno *smartphone*, tramite il quale si potevano controllare porte, finestre, luci e un cancello posto all'ingresso del bilocale. Il cuore dell'interazione era piuttosto basica: era possibile attivare gli elementi grafici dinamici che presentassero almeno un punto di collisione e potessero rispondere a sollecitazioni "fisiche" semplicemente usando il *mouse*, sfruttando l'evento *OnMouseDown()*[77] della classe base *MonoBehaviour* di Unity. La parte più curata era, tuttavia, costituita dal movimento del giocatore sulla sedia a rotelle, una versione modificata di un *controller* dello sviluppatore Alex Ocias[78], che permetteva di spostarsi in modo fluido e di emulare le forze fisiche agenti su una carrozzina per disabili, sensibilmente variabili in base alla massa del corpo e dell'ausilio a ruote, alla pendenza del terreno e ad altre caratteristiche ambientali. Come si può notare dagli *screenshots* che seguono, il primo prodotto presentava una grafica non troppo gradevole e molti elementi cozzavano tra loro a causa della poca cura nello scegliere lo stile grafico: alcune parti presentavano uno stile realistico, altre risultavano piuttosto povere ed erano composte, perlopiù, da solidi semplici o da oggetti con pochissimi *poligoni*. Inoltre, materiali e luci erano gestiti in modo approssimativo e senza prestare particolare

[76] *IBM Watson API*. Link al sito: https://www.ibm.com/watson
[77] Evento *OnMouseDown*, Unity Documentation: https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnMouseDown.html
[78] Unity Wheelchair Controller. A cura di Alex Ocias. Link alla pagina: https://ocias.com/blog/unity-wheelchair-controller/

attenzione alle risorse: il prodotto risultava pesante per la mancata ottimizzazione e non erano rari gli artefatti grafici (*glitch*) durante l'esecuzione.

Figura 5. Primo prototipo del simulatore 3D – Esterni

Figura 6. Primo prototipo del simulatore 3D – Zona Giorno e Smartphone

Figura 7. Primo prototipo del simulatore 3D – Zona Notte, Smartphone e comandi vocali attivi

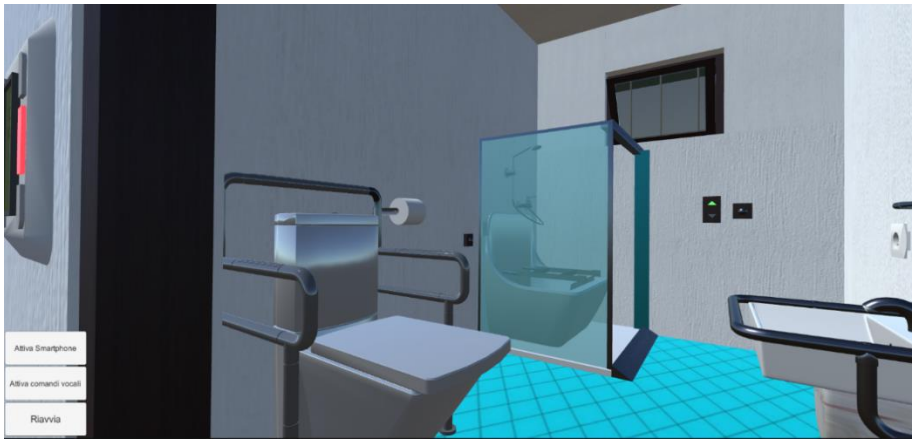

Figura 8. Primo prototipo del simulatore 3D - Bagno con ausili per disabili

Infine, la presenza di Watson lo rendeva dipendente da Internet e compatibile soltanto con le piattaforme Windows e MacOS: un limite consistente per un prodotto il cui punto di forza dovrebbe essere proprio la portabilità e la fruibilità *offline*, per poter raggiungere tutti. In definitiva questo prototipo, nonostante le lacune evidenziate, assolveva bene al suo scopo: fornire un'esperienza all'utente, seppur non immersiva, attraverso la quale egli poteva comprendere la prospettiva di un portatore di disabilità dello spettro motorio, in un ambiente creato su misura per lui, nel quale riusciva a muoversi e ad interagire senza troppa difficoltà, grazie agli ausili *in-game* proposti.

Successivamente, lo strumento è stato arricchito in itinere di un tirocinio formativo presso il PercRo e si è trasformato progressivamente, grazie all'acquisizione di nuovi concetti riguardanti la grafica 3D e la Realtà Virtuale, in un vero simulatore di Realtà Virtuale

Figura 9. Secondo prototipo del simulatore, prima versione con supporto alla RV - Esterni

immersiva. Infatti, la principale novità di questa versione era la sua fruibilità in RV, tramite i visori Oculus Rift e Oculus Quest, previa integrazione del relativo pacchetto di *plugin* in

Unity[79] (divenuto obsoleto dalle *release* 2020.x del framework) e trasposizione delle azioni tramite *mouse* e tasti direzionali in altre basate sulle forze fisiche agenti tra le mani virtuali del giocatore e gli oggetti, come si vedrà più avanti a proposito del progetto finale. Questa seconda versione prevedeva migliorie a livello grafico (non ancora sufficienti, però, a donare un *feeling* realistico agli ambienti) e offriva più spazio alle interazioni: era possibile, infatti, "impugnare" gli oggetti circostanti, "aprire" cassetti e ante, "utilizzare" l'acqua del lavello e dei sanitari del bagno e "preparare" un pasto. Inoltre, erano state apportate migliorie grafiche alla carrozzina del *player*, il cui movimento era impartibile tramite il *thumbstick* del *controller touch* destro dell'Oculus – simile a quello delle sedie a rotelle di tipo motorizzato – e risultava curato sul lato fluidità per evitare sensazioni di malessere all'utente. Le schermate presentate donano un'idea dello stato del simulatore nella fase descritta e evidenziano i cambiamenti e le migliorie apportate rispetto al lavoro precedente, in relazione alla conversione in prodotto fruibile in RV e agli incentivi sui piani grafico e interattivo:

Figura 10. Secondo prototipo del simulatore, prima versione con supporto alla RV - Zona Giorno

[79] Unity *Oculus Integration SDK*. Link alla pagina: https://developer.oculus.com/downloads/package/unity-integration/

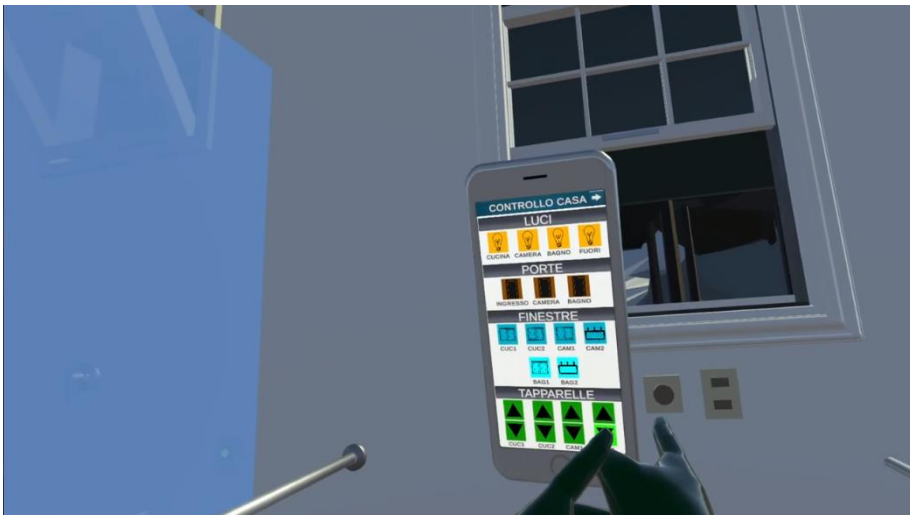

Figura 11. Secondo prototipo del simulatore, prima versione con supporto alla RV - Dettagli interazione (tapparelle motorizzate)

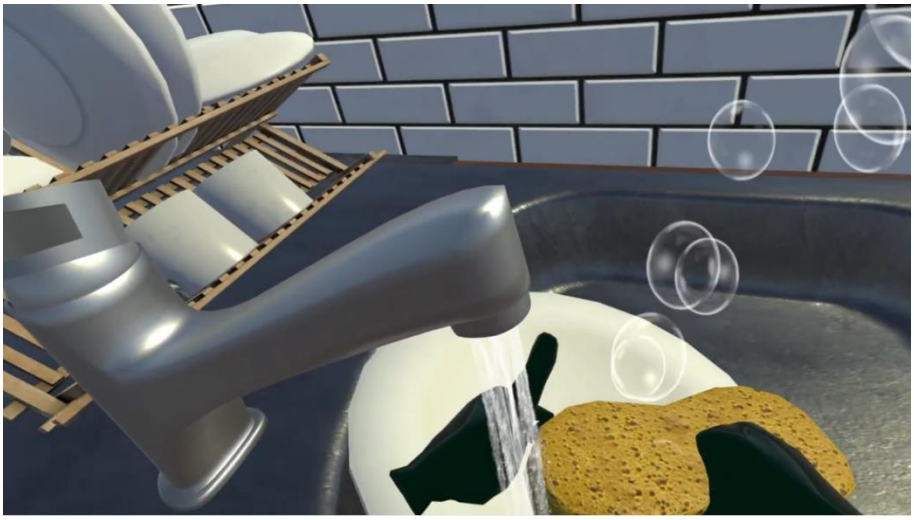

Figura 12. Secondo prototipo del simulatore, prima versione con supporto alla RV - Dettagli interazione (lavello cucina)

Tuttavia, nonostante il risultato assolvesse alla funzione di riprodurre un ambiente con specifiche caratteristiche e fosse facilmente esplorabile in RV, non risultava accattivante a causa della mancanza di un *background* ludico e, soprattutto, non poneva l'utente di fronte a sfide, importanti per incidere sulle tematiche riguardanti l'abbattimento delle barriere architettoniche. In sintesi, il simulatore – come il precedente – donava un'idea di come potesse essere adattato lo spazio per donare un certo livello di comfort e autonomia ad una persona con *deficit* a livello motorio, ma non permetteva all'utente che lo provava di avvertire le difficoltà che quest'ultima poteva incontrare nella vita reale, in un mondo creato su misura di normodotati. Era, dunque, necessario pressare sulle problematiche, affinché lo strumento

potesse diventare efficace per sensibilizzare il fruitore e risultava fondamentale donargli una chiave ludica, al fine di renderlo più accattivante e incentivare più fasce d'età al suo utilizzo.

La presenza delle lacune evidenziate ha portato ad un *brainstorming*, che ha condotto allo sviluppo di un nuovo simulatore totalmente rivisitato, strutturato secondo un meccanismo ludico a missioni, la cui architettura e implementazione verranno presentate in un secondo momento.

5.2 Gamification e presentazione dei livelli di gioco

Come accennato nel paragrafo precedente, è stato necessario donare un'impostazione ludica al lavoro, al fine di renderlo più accattivante e dotarlo di una struttura in grado di condurre il giocatore ad alcuni obiettivi prefissati consentendogli, al contempo, di usufruire di una navigazione libera all'interno degli ambienti proposti. A tal proposito, il simulatore è stato strutturato su tre livelli, due dei quali ambientati in un'abitazione privata sviluppata su due piani, dalle caratteristiche di una normale casa indipendente in stile americano con garage e uno nella stessa appositamente strutturata, arredata e dotata di sussidi dedicati a portatori di disabilità motoria su carrozzina. Per una questione di economia di risorse e per evitare ridondanze, i livelli 1 e 2 condividono la stessa scena, grazie alla selezione degli oggetti e del giocatore da caricare direttamente in tempo di esecuzione (*on runtime*), ma si differenziano per il *player*: mentre nel primo caso sarà possibile giocare nelle vesti di una persona normodotata, nel secondo si dovrà osservare il mondo dal punto di vista di un soggetto su sedia a rotelle e cambierà, di conseguenza, il modo di interagire con l'ambiente e sarà possibile constatarne anche le difficoltà che lo stesso, ricco di barriere architettoniche e privo di ausili specifici, gli causerà. Il terzo, infine, giocabile nei panni di un utente con disabilità su sedia a rotelle, si distinguerà negli arredi e negli ausili dei quali l'abitazione sarà dotata e presenterà delle differenze sul piano dell'interazione, che sarà integrata da un sistema di automazione e da dispositivi controllabili da remoto. Parte comune alle tre modalità di gioco sono le missioni, delle quali si tratterà tra poco: la trama è sviluppata su quattro *quests* per livello, includenti semplici obiettivi relativi ad azioni comuni alla vita quotidiana, quali rientrare a casa, preparare un pasto, fare la doccia e andare a letto.

Di seguito è possibile trovare un sommario di quanto sintetizzato, in grado di schematizzarne grossolanamente le analogie e le differenze, in relazione ai livelli citati:

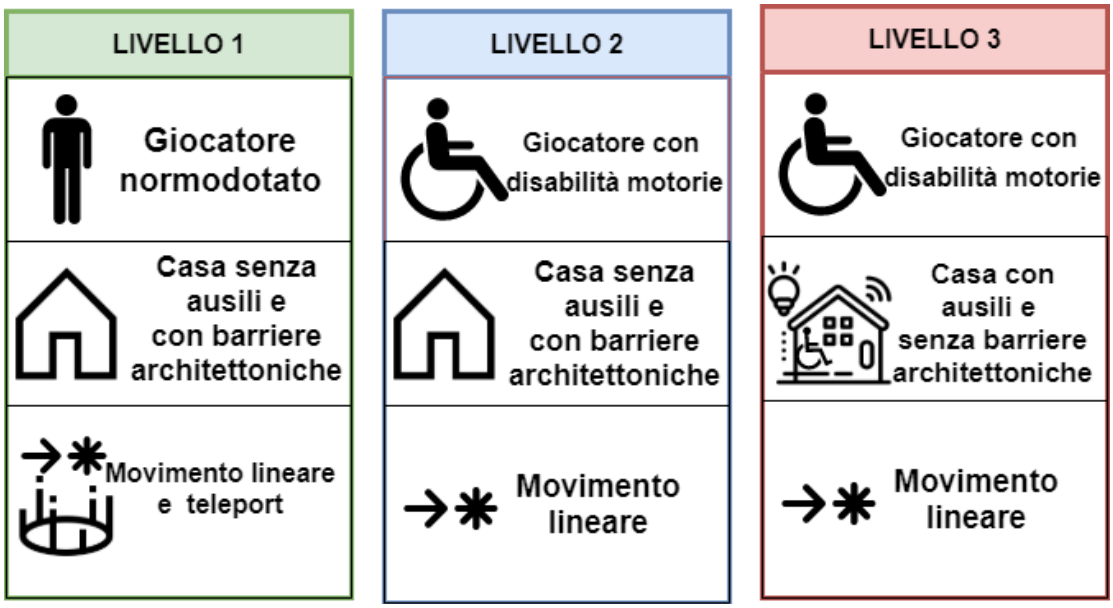

Figura 13. Rappresentazione schematica dei livelli del simulatore, con analogie e differenze

Nelle prossime righe verrà illustrata la struttura del sistema a missioni e dei livelli appena citati, divisi nelle relative scene di Unity e se ne descriveranno le caratteristiche più salienti, con un occhio di riguardo all'introduzione di determinate strategie in base alla situazione rappresentata nel simulatore. Al contempo, saranno citate alcune delle tecniche di implementazione impiegate nella realizzazione delle principali sezioni dello stesso, le quali verranno trattate in seguito nel dettaglio, con a corredo esempi e parti di codice.

5.2.1 Struttura delle missioni e obiettivi

Al fine di rendere più avvincente l'esperienza di RV immersiva, sono state inserite quattro missioni all'interno dei livelli di seguito presentati, ciascuna delle quali si compone di semplici obiettivi da portare a termine per poter, successivamente, accedere alla successiva.

Le missioni proposte (schematizzate nella figura seguente), come già accennato, consistono in semplici azioni della vita quotidiana e si dividono in obiettivi che richiedono, nella maggior parte dei casi, l'interazione con un oggetto di scena o il raggiungimento di un luogo.

Figura 14. Schema delle missioni

È stato scelto di inserire missioni di facile svolgimento per dare enfasi al problema delle barriere architettoniche, che contribuiranno a rendere più complesse anche le azioni più semplici, in particolare quando il giocatore che impersonifica un utente con disabilità dello spettro motorio deve interfacciarsi con la casa senza ausili, prevista nelle prime due modalità di gioco. Al termine di ogni obiettivo, sarà proposta a video una schermata di completamento dello stesso e si potrà, così, accedere al successivo; al termine dell'ultimo, la missione corrente sarà completata e quella dopo verrà caricata. Ognuna di queste include degli elementi di gioco inerenti alla stessa, come oggetti e portali, caricati dinamicamente in base allo stato degli obiettivi; il meccanismo di avanzamento si avvale di un pannello di interfaccia *canvas* – commutabile con il tasto A del *controller touch* destro – per il controllo delle stesse e include gli elementi di testo corrispondenti agli obiettivi delle missioni, dei quali è possibile tenerne traccia del completamento grazie al colore degli oggetti *text*; tale logica è comune ad ogni modalità di gioco proposta. Per questioni pratiche, i *canvas* inerenti le missioni e i relativi gruppi di *GameObject* da attivare al completamento degli obiettivi sono gestiti separatamente in Unity: gettando un'occhiata alla gerarchia nell'*editor*, si può notare che questi ultimi non sono annidati nella sezione *Missioni_VR*. Per la missione I "Rientra a casa", come da esempio nella figura di seguito, è prevista una parte di UI (a sinistra) con gli elementi

di testo che permettono di tenere traccia dell'avanzamento nella stessa e una con gli oggetti di gioco (a destra), raggruppati in base agli obiettivi.

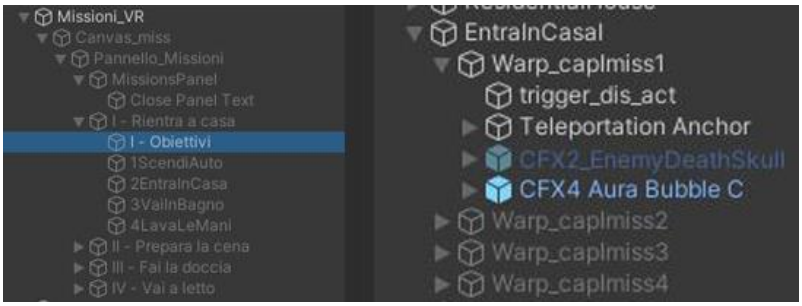

Figura 15. Gerarchia della Missione I, con elementi dell'UI e oggetti di gioco, prelevata dalla scena LVL1_LVL2

Di seguito sarà illustrato in breve in cosa consiste ciascuna missione (senza scendere nel profondo della parte implementativa[80]), per poter donare al lettore una prospettiva più chiara del tipo di esperienza che potrà essere provata fruendo del simulatore[81].

5.2.1.1 Missione I – "Rientra a casa"

In ognuno dei tre scenari di gioco, la prima missione consiste nel compiere delle semplici azioni al fine di rientrare a casa e prepararsi alla fase finale della giornata. La trama inizia *in medias res* e coglie il giocatore in auto, nel garage di casa, al rientro da un'ipotetica giornata di lavoro. Il primo obiettivo al quale deve assolvere, è proprio scendere dalla macchina: azione semplice e scontata nel primo livello, ma che si vedrà essere più complessa negli altri due. Al fine di rendere più complicato lo spostamento, l'auto è affiancata ad un'altra e lo spazio a disposizione per aprire la portiera e scendere è limitato: nel terzo livello dove è prevista l'eliminazione di alcune barriere architettoniche, la seconda è rimossa dalla scena per permettere una maggiore possibilità di movimento e consentire al *player* di montare sulla carrozzina. Una volta completato il primo obiettivo, è possibile entrare in casa: per accedere alla stessa è necessario aprire il garage (interazione non facente parte di alcuna missione, ma parte della meccanica di gioco) e posizionarsi sul pianerottolo d'ingresso, dove l'utente è invitato a sollevare uno zerbino, al fine di trovare una chiave che, al tocco (tramite l'impiego di un'animazione), si incastrerà nella serratura della porta principale, che si aprirà. Appena si verifica la collisione tra chiave e toppa, un *prompt* avvisa l'utente del termine del secondo obiettivo; nel caso del terzo livello, in questo punto del gioco viene mostrata a video

[80] L'eventuale terminologia tecnica verrà trattata e, dunque, disambiguata nel contesto dell'implementazione
[81] Per gli *screenshot* relativi alle missioni, è necessario consultare le sezioni relative ai tre livelli

un'ulteriore schermata che lo avverte della possibilità di poter fruire dello *smartphone in-game* (che verrà trattato nella sezione inerente la scena "LVL3"). Dopo lo sblocco della porta, il giocatore si trova sull'ingresso di casa ed è invitato a salire lungo una scala contrassegnata da alcune frecce di guida, le quali conducono al secondo livello dell'abitazione. Aprendo il pannello delle missioni, è possibile vedere evidenziato in giallo il terzo obiettivo, "Vai in bagno", che bisogna a questo punto portare a termine entrando nella prima porta a sinistra e posizionandosi o attivando con il raggio (previsto solo nella prima modalità di gioco) il portale di fronte al lavabo. A questo punto, è necessario avvicinarsi a quest'ultimo e "lavarsi le mani", alzando la leva del miscelatore, prendendo la saponetta e strofinando le mani alla stessa, tenendole sotto il getto d'acqua. Quando il sistema terminerà l'esecuzione di una Coroutine (vedere paragrafi sull'implementazione) con un breve timer, si conclude anche l'ultimo obiettivo ("Lava le mani") e la missione può considerarsi archiviata. In questo momento, aprendo il relativo pannello con il tasto A del *controller touch* destro, è possibile visualizzare i nuovi obiettivi della missione seguente: "Prepara la cena".

5.2.1.2 Missione II – "Prepara la cena"

La presente missione inizia appena si conclude il lavaggio delle mani, ultimo obiettivo della precedente. A questo punto, è necessario scendere, tramite la rampa di scale, a piano terra della casa e svoltare a destra, dove è possibile trovare un portale luminoso all'ingresso della cucina. Dopo avervi interagito, si conclude il primo semplice obiettivo: "Vai in cucina" e, al contempo, è possibile osservare la porta del frigorifero (posizionato sulla destra) aprirsi ed evidenziare, tramite l'impiego di alcuni effetti a particella, una carota. Successivamente, l'utente deve "prendere" la stessa dal frigorifero e inserirla nell'inventario che lo segue: al momento dell'interazione con il *collider* posto sullo stesso (il cui funzionamento è gestito da *XRSocketInteractor*, in seguito trattato), il secondo obiettivo termina. È possibile, dunque, posizionare la carota sul tagliere, comparso al termine della precedente operazione e ridurla in tocchetti, tramite alcuni colpi di coltello (il cui funzionamento prevede una scissione dei poligoni che compongono la carota, illustrato più avanti). Il movimento dell'utensile e il relativo tocco con la superficie del tagliere avviano una Coroutine che apporta alcune modifiche ai tag dei "pezzi" di carota ottenuti e permette la conclusione dell'obiettivo. A questo punto il giocatore, per avanzare nella missione deve posizionare una parte dell'ortaggio nella padella *wok* che compare sul piano cottura al termine del *task* precedente

e, una volta conclusa tale azione, è avvisato del raggiungimento del presente obiettivo ed è invitato ad accendere un fornello, contrassegnato insieme alla relativa manopola di regolazione da alcuni elementi caratterizzati da un materiale di colore verde fosforescente, con emissione attiva. Dopo aver attivato il fornello, previo tocco del relativo comando, si conclude l'obiettivo e l'utente può "cuocere" l'ortaggio, posizionando la padella sulla fiamma; dopo alcuni secondi avviene la "trasformazione" del cibo e l'emissione di alcuni effetti visivi e sonori preannuncia il termine del *task*. Dopo alcuni secondi (al termine di una Coroutine che gestisce la fase di cottura) è infine possibile, con il semplice contatto della padella con un piatto posto nello scolapiatti posizionato accanto al lavello, impiattare la cena e concludere l'ultimo obiettivo, nonché la missione II.

5.2.1.3 Missione III – "Fai la doccia"

La terza missione inizia appena la precedente giunge al termine e il giocatore si trova in cucina, di fronte al piatto contenente il cibo appena preparato. Il primo obiettivo invita l'utente a trovare la doccia: per concluderlo, è necessario salire lungo la rampa di scale già percorsa in precedenza ed entrare nella porta sulla destra (precedentemente bloccata, ma con la quale è ora possibile interagire) semplicemente avvicinandosi alla stessa e commutandone l'apertura. Successivamente, bisogna entrare nella prima stanza a disposizione, che consiste in un bagno di servizio, dove è presente una doccia a *box*; interagendo con il portale posto di fronte ad essa, è possibile terminare il suddetto obiettivo. Al fine di simulare di essersi privati dei vestiti prima di entrare nel *box*, compaiono sul pavimento dei vestiti. In seguito, per avanzare nella missione, l'utente deve azionare l'acqua nella doccia e, per compiere tale azione, è necessario aprire la porta a soffietto della stessa ed entrarvi, prima di raggiungere il miscelatore. All'apertura della leva di quest'ultimo (al quale è collegato uno *script*, trattato in seguito, che definisce gli eventi da gestire in base agli angoli minimo e massimo della *Hinge Joint* ad esso connessa), è possibile osservare un flusso d'acqua scendere dal rosone posto in alto e, dopo pochi secondi, vi è l'annuncio a video del termine dell'obiettivo. Al contempo, in un angolo della doccia compare un flacone di sapone, necessario ad assolvere al *task* successivo, che recita "Insaponati e risciacqua"; toccando il *dispenser*, è possibile raccogliere una goccia del detergente che, interagendo con le collisioni posizionate sulla "testa" del giocatore (in corrispondenza della telecamera e dell'*HMD*) e con il *collider* posto al centro del fascio d'acqua, dà luogo ad una serie di commutazioni a catena dello stato di

alcuni *script* e fa partire il timer della *coroutine* che, alla sua conclusione, attiva lo *script* responsabile del termine dell'obiettivo (*missioneCompiutaSimple*, trattato in seguito[82]). Alla fine del suddetto, compare un asciugamano su un supporto a muro e, semplicemente prelevandolo, sganciandolo – via *script* dedicato – dalla *Fixed Joint* presente sull'appendino, e ponendolo sul "corpo" del giocatore (o sulla "testa", il controllo avviene verificando via *script* se il *GameObject* di destinazione con cui l'asciugamano collide ha il tag "Player" assegnato), si chiude il relativo obiettivo. Infine, il pannello delle missioni mostra in giallo un ultimo compito al quale assolvere: bisogna "indossare" i vestiti puliti; questi ultimi si trovano nella cabina armadio accanto al bagno, alla quale è necessario accedervi aprendone le ante (azione effettuabile "avvicinandovi" le mani). Una volta giunto all'interno, il giocatore può prelevare una maglietta e un pantalone e "vestirsi": i capi, scomparendo al contatto con la sua "testa" o il suo "corpo", daranno l'idea di essere indossati. L'obiettivo, a questo punto, giunge al termine insieme alla missione in oggetto e, di conseguenza, vengono abilitati gli oggetti di gioco relativi alla successiva.

5.2.1.4 Missione IV – "Vai a dormire"

Dopo aver "indossato" i vestiti, il giocatore può addentrarsi nella quarta e ultima missione del simulatore, che prevede di "andare a letto". In primis, all'utente viene richiesto di trovare il letto e, dunque, di raggiungere la camera da letto, posizionata accanto al bagno principale, nel quale prendono luogo gli ultimi obiettivi della prima missione. Il compito si conclude interagendo con il portale posto ai piedi del letto e, in seguito, viene sbloccato il successivo, che prevede di sollevare la coperta e prepararsi, dunque, a dormire. Toccando il bordo della stessa, si verifica un evento che permette al capo di tappezzeria di spostarsi su un lato; la coperta, che consiste in un piano dotato di armatura e *bones*[83] (modellato in Blender e in seguito importato come modello in Unity), assume una nuova "forma" grazie ad un'animazione che agisce su alcuni *GameObject*, corrispondenti proprio ad alcuni di questi ultimi. Una volta terminato l'obiettivo appena descritto, il giocatore può procedere con il terzo, che richiede semplicemente di spegnere la luce di una lampada posta sul comodino accanto al letto. Dopo aver assolto a questo compito, è possibile, infine, mettersi a dormire;

[82] Il nome di alcuni *script* è rimasto alla fase progettuale e può dare informazioni fuorvianti inerenti la funzione a cui assolvono. Nel caso degli script *missioneCompiuta**, questi si riferiscono al completamento degli obiettivi e non delle intere missioni.

[83] *Introduzione al rigging: armature e bones*. 3 ottobre 2013. A cura di Alfonso Annarumma. Html.it. Link alla pagina: https://www.html.it/pag/44285/introduzione-al-rigging-armature-e-bones/.

nel momento in cui il giocatore "tocca" un *collider* (settato come trigger) su un cubo reso invisibile nella scena, viene riprodotta un'animazione che lo posiziona "nel" letto e lo ruota di 90 gradi in senso orario, allo scopo di simulare l'azione di coricarsi[84]. Al termine dell'obiettivo, anche l'ultima missione giunge al termine: un messaggio a video informa l'utente della fine della simulazione e, dopo una breve transizione di dissolvenza, si è catapultati nel menu di avvio della prima scena, dal quale è possibile scegliere di ripetere il gioco nelle modalità di esperienza previste.

5.2.1.5 Esplorazione libera

Il meccanismo a missioni è, senz'altro, utile a donare al simulatore un'impostazione ludica e a permettere all'utente di visitare la maggior parte delle "zone" che prevedono un'interazione, al fine di amplificare la sua percezione dell'ambiente in merito agli obiettivi proposti, ma può risultare inefficiente nel consentirgli di esplorare liberamente l'ambiente. Vincolare il giocatore all'esplorazione delle sole mete prefissate può condurlo alla noia e, pertanto, può essere controproducente: uno dei compiti del simulatore è anche quello di sviluppare la sua creatività, ponendolo in condizioni in grado di renderlo curioso.

A tal proposito, è possibile esplorare liberamente l'ambiente virtuale proposto in ogni livello prima del termine di tutte le missioni (preferibilmente, al termine della penultima missione, dove tutte le porte della casa sono sbloccate e gli elementi relativi ai task precedenti sono ormai stati caricati del tutto); facendo ciò, il fruitore ha la possibilità di provare gli oggetti interattivi della casa che non sono parte integrante del meccanismo delle *quests*, come interruttori, finestre, tapparelle e cassetti. In particolare, nel livello 3, dove è presente un'ampia rosa di strumenti dedicati all'automazione, tra i quali spicca anche un sistema di domotica controllabile tramite uno *smartphone in-game*, è necessario donare spazio all'interazione libera dell'utente poiché, tramite la sola assoluzione degli obiettivi delle missioni, possono essere tralasciate esperienze che, invece, è interessante provare.

5.2.2 Scene e livelli

Di seguito verranno presentate le tre modalità di gioco sopra citate (alle quali ci si riferirà più semplicemente con il termine "livelli") e le relative scene di Unity entro le quali si sviluppano. Sarà anche trattato il cosiddetto "livello 0", il menu di avvio, necessario alla

[84] La resa realistica di tale obiettivo non è molto precisa in termini grafici ed è necessario apportarvi delle modifiche.

scelta del tipo di abitazione entro il si svilupperà la simulazione e, di conseguenza, i livelli relativi alla modalità di gioco selezionata. In ciascuna sezione verranno descritti i punti più salienti di ciascuna parte degli stessi e ne saranno illustrati brevemente alcuni contenuti; successivamente, quando si entrerà nel pieno dell'implementazione del simulatore, tali concetti verranno espansi e ne verranno trattate le caratteristiche nel dettaglio, con alcune parti di codice ed esempi a corredo.

5.2.2.1 Menu di avvio

La scena di apertura del simulatore (figura 12) include un menù, attraverso il quale è possibile selezionare l'ambiente da caricare, a scelta tra un'abitazione *standard* (per i livelli 1 e 2) e una con ausili assistivi integrati.

Figura 16. Schermata di avvio del simulatore

Attraverso un bottone dedicato, è possibile, inoltre, fruire delle istruzioni sui tasti funzione da poter utilizzare durante la simulazione, ovvero i pulsanti X, A e B dei *controller touch* del visore Oculus, assegnati rispettivamente al riavvio del gioco, alla visualizzazione del pannello delle missioni e all'attivazione dello smartphone in-game ove previsto, ovvero nella scena del livello 3.

Figura 17. Sezione del menù con le istruzioni sui comandi

La scena in questione è la più semplice dell'intero elaborato, poiché include esclusivamente una luce direzionale da impostazioni standard di Unity, una componente per il giocatore in prima persona *XRRig* (*prefab* del pacchetto *XR Interaction Toolkit*), nella cui gerarchia (figura 18) sono inclusi una comune *telecamera* (*Camera*), un semplice sistema di interazione via *teleport ray*, che permette al fruitore di visualizzare i contenuti del menu e un *canvas* con gli elementi in 2D dell'interfaccia utente (*UI*). In questo livello non sono previsti movimenti nello spazio; pertanto, all'elemento *XRRig* (rinominato in *VRRig*) è esclusivamente associato l'omonimo *script* che gestisce le impostazioni relative al posizionamento della telecamera nello spazio, alla cui voce *Tracking Origin Mode* è selezionata *Floor*, in quanto è necessario

impostare il livello del pavimento come punto di origine da cui misurare l'altezza del giocatore (figura 19).

Figura 18. Scena di avvio in Unity, con albero gerarchico delle componenti

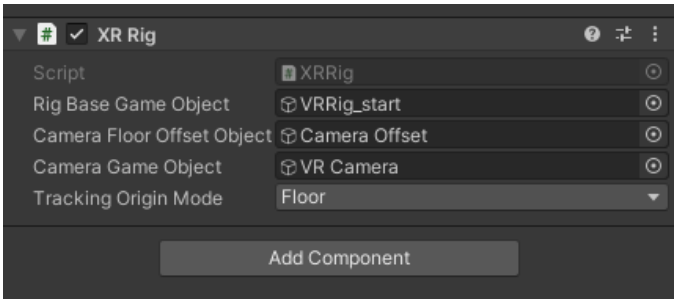

Figura 19. Script XRRig

Per permettere l'interazione con gli elementi dell'*UI*, a ciascun *controller touch* dell'Oculus (a cui corrisponde una mano nell'ambiente virtuale) è solitamente associato uno *script* che gestisce la serializzazione dei comandi impartiti; nel caso in questione il compito è svolto da *XRController*, che traccia la presenza nello spazio dei dispositivi e ne trasmette al motore grafico l'azione sui bottoni e sul *thumbstick* e ne consente la mappatura. Al *controller* virtuale destro è associata l'emissione di un raggio (*renderizzato* con lo strumento *LineRenderer* di Unity, usato per tracciare linee e traiettorie nell'ambiente 3D), che viene "sparato" per interagire, in questo caso, con le collisioni (*colliders*) degli elementi dell'*UI* del menu; tale

tecnica, detta *raycasting*, trova il suo impiego nello *script XRRayInteractor*, utile anche a definire gli eventi da gestire in base alle azioni impartite.

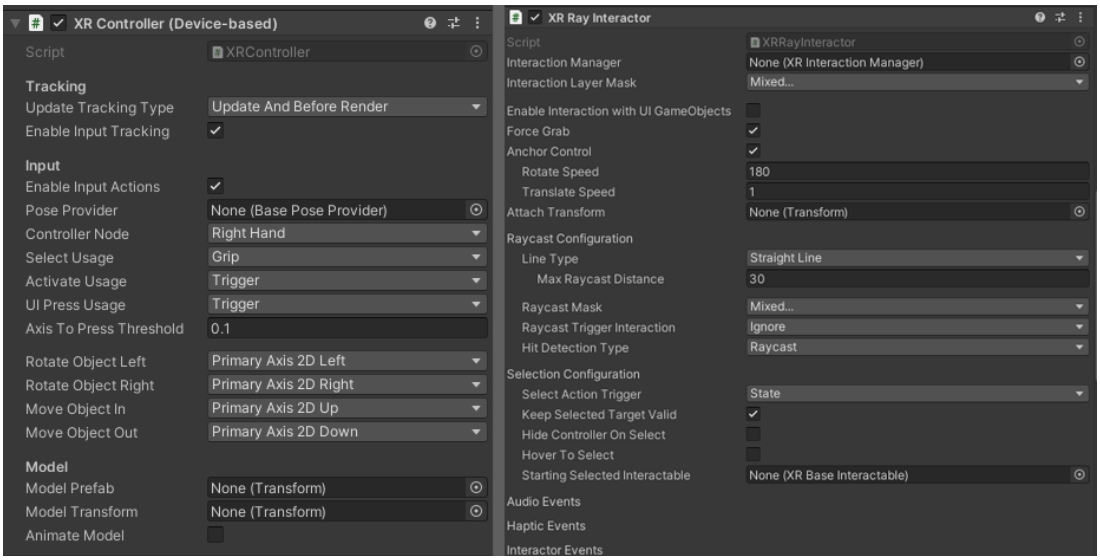

Figura 20. XR Controller e XR Ray Interactor

Alla mano destra è anche collegato lo *script XRDirectInteractor*, indispensabile per "toccare con mano" gli oggetti di gioco e interagirvi, ma si tratta di un residuo di progettazione e non sarà, dunque, trattato in nel presente caso.

Al fine di rendere più forte la presenza dell'utente nell'ambiente di RV immersivo proposto, è stato scelto di includere anche i *prefab* delle mani di Oculus (contenuti nel plugin di integrazione Oculus XR Plugin, che estende le funzionalità di XR alla piattaforma di

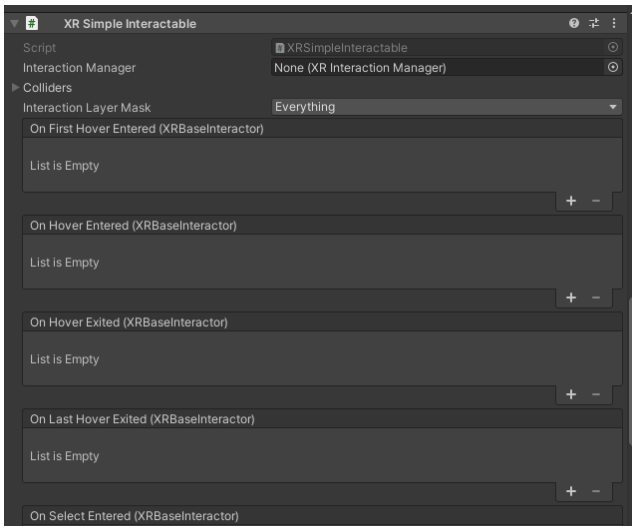

Figura 21. XR Simple Interactable e l'evento OnSelectEntered con lo script cambiaScena selezionato per l'attivazione al trigger

Facebook, del quale si tratterà più avanti). L'altro elemento chiave è il *canvas*, l'area con gli elementi della UI che include gli *sprites* (immagini 2D) che compongono l'interfaccia grafica e i pulsanti; in quanto a quelli per il caricamento delle due scene-livello, è possibile interagirvi grazie all'evento *OnSelectEntered* dell'estensione *XRSimpleInteractable* (figura 21) che, nel caso specifico dei bottoni *Abitazione Standard* e *Abitazione con Ausili*, attiva il piccolo *script cambiaScena* (riportato in appendice) collegato al *GameObject Avvia* e conduce alla scena.

Per evitare di creare uno *script* diverso a seconda della scena, esso si avvale di un campo apposito entro il quale è possibile specificare il nome della scena da caricare, operazione permessa dal metodo *LoadScene* della classe *SceneManager* di Unity, che gestisce il caricamento delle stesse.

5.2.2.2 Livelli 1-2

Selezionando la voce di menu *Abitazione Standard*, è possibile caricare la scena del simulatore che include come elemento centrale un'abitazione a due piani, entro la quale si sviluppano le missioni previste. Al caricamento, dopo un brevissimo *fadeout*, è possibile selezionare tramite due grossi pulsanti l'esperienza di gioco che si desidera provare nella casa, se impersonificando un giocatore normodotato o uno con disabilità, su sedia a rotelle;

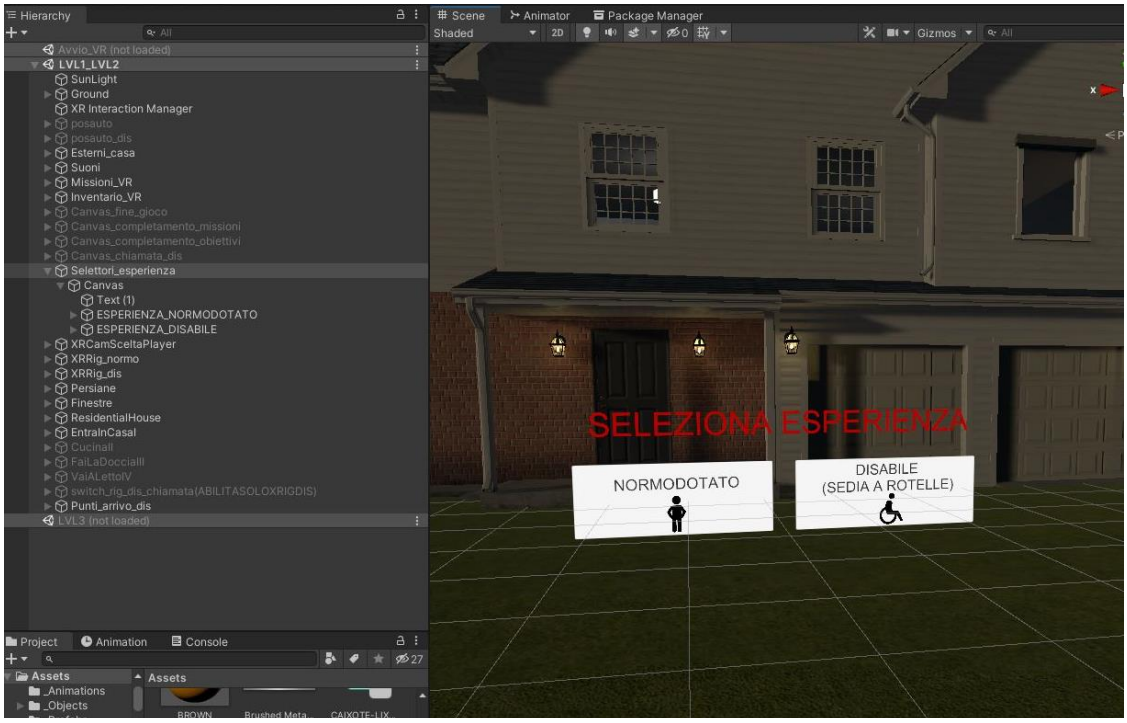

Figura 22. Albero gerarchico della scena LVL1_LVL2 nell'editor Unity e selettore dell'esperienza

in base alla selezione, verranno attivati i relativi *XRRig* (*player*) e gli elementi caratterizzanti la scelta effettuata.

I pulsanti di selezione dell'esperienza sono gestiti in modo analogo ai selettori di scena del menu di avvio, per quanto concerne l'impiego di un *canvas* e di elementi dell'*UI*, ma presentano delle sostanziali differenze riguardo l'interazione con gli stessi. Infatti, essi si avvalgono di uno *script* creato *ad hoc* per la selezione dell'esperienza (chiamato *provacambio*), che sfrutta l'evento *built-in* di Unity *OnTriggerEnter* (descritto più avanti) per interagire con le collisioni settate come *trigger* presenti sulle mani del *player*. Nella seguente figura è possibile osservare lo stesso *nell'inspector* di Unity e, nella sezione relativa all'implementazione, ne verrà analizzato il codice, riportato nella sua completezza in appendice.

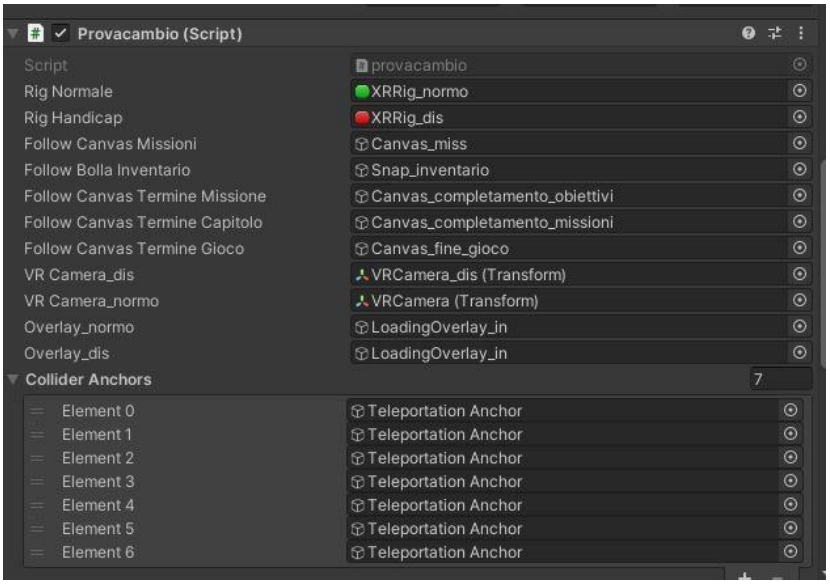

Figura 23. Script provacambio nell'inspector di Unity

In breve, lo *script*, permette di effettuare al tocco di uno dei bottoni uno *switch* di *XRRig*, di attivare *GameObject* e di assegnare valori di variabili relativi all'esperienza di gioco scelta; data la complessità di funzionamento dello stesso, se ne tratterà nel dettaglio più avanti in un'apposita sezione. All'apertura della scena, il *player* attivo, strutturato in modo analogo a quello della scena di avvio, è *XRCamSceltaPlayer*: l'unica differenza rispetto al precedente è costituita dal metodo di interazione associato ai *controllers* delle mani (*LeftHandController* e *RightHandController*), poiché stavolta il giocatore non si serve del *raycasting* ma "tocca" direttamente parte dell'interfaccia. Ad assolvere a ciò, è *XRDirectInteractor* del *toolkit* di XR

che, contrariamente a *XRRayInteractor*, include una collezione di classi che permettono di interagire direttamente con le collisioni collegate ai *GameObjects* (figura 24).

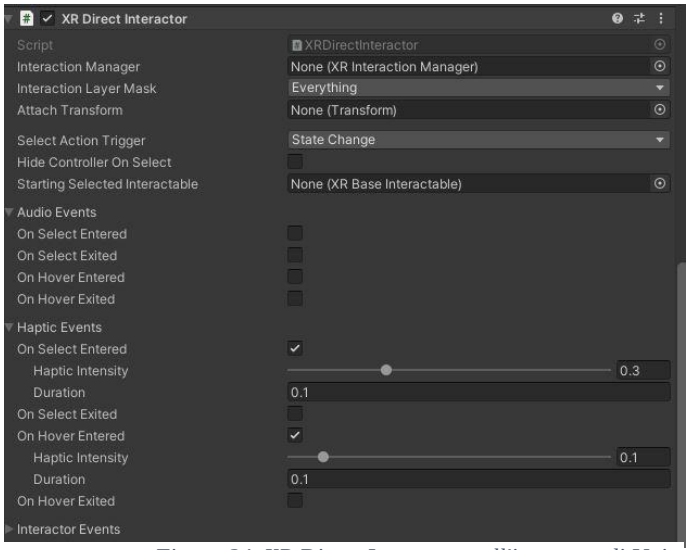

Figura 24. XR Direct Interactor nell'inspector di Unity

Una volta effettuato un *tap* con la mano nella RV sul relativo bottone, verranno, come anticipato, caricati gli elementi relativi all'esperienza scelta, tra i quali il giocatore; se sarà selezionata la voce "normodotato", *XRCamSceltaPlayer* verrà disabilitato e, istantaneamente, sarà possibile trovarsi nei panni" di *XRRig_normo*. Allo stesso modo, toccando l'altro bottone, si inizierà la simulazione nella prospettiva di una persona con disabilità motoria alle gambe e, dunque, saranno abilitati via *script* il relativo *player*, *XRRig_dis* e le componenti relative a questa seconda modalità. Dei giocatori se ne tratterà nello specifico nei paragrafi inerenti il *plugin XR Interaction Toolkit*, poiché si tratta di istanze di un *prefab* dello stesso, opportunamente modificate per assolvere ai ruoli specificati. Una volta selezionato il tipo di esperienza tramite la schermata di cui sopra, si entra nel pieno della simulazione; per entrambi i tipi di giocatore, questa inizia all'interno di un'auto, posizionata nel garage della casa, dove si è invitati a svolgere il primo obiettivo della prima missione, che prevede – appunto – di uscirvi.

Esperienza 1 – Giocatore Normodotato in casa senza ausili

Figura 25. Selezione dell'esperienza con giocatore "normodotato"

Scegliendo la prima esperienza di gioco, si prova il simulatore con gli occhi di una persona normodotata, in grado di deambulare liberamente. Il *player* che viene attivato, chiamato *XRRig_normo* nella gerarchia di Unity, presenta le stesse caratteristiche implementative (delle quali si approfondirà in seguito) degli *XRRig* citati in precedenza per quanto concernono la telecamera e le mani, ma vi si distingue per la possibilità di movimento in esso inclusa. Infatti, con questo *player* è possibile "deambulare" sfruttando due tipi di movimento: uno lineare, che impiega la componente di Unity *CharacterController* e lo *script MovementProvider* (che estende la classe *LocomotionProvider* del pacchetto *XR Interaction*

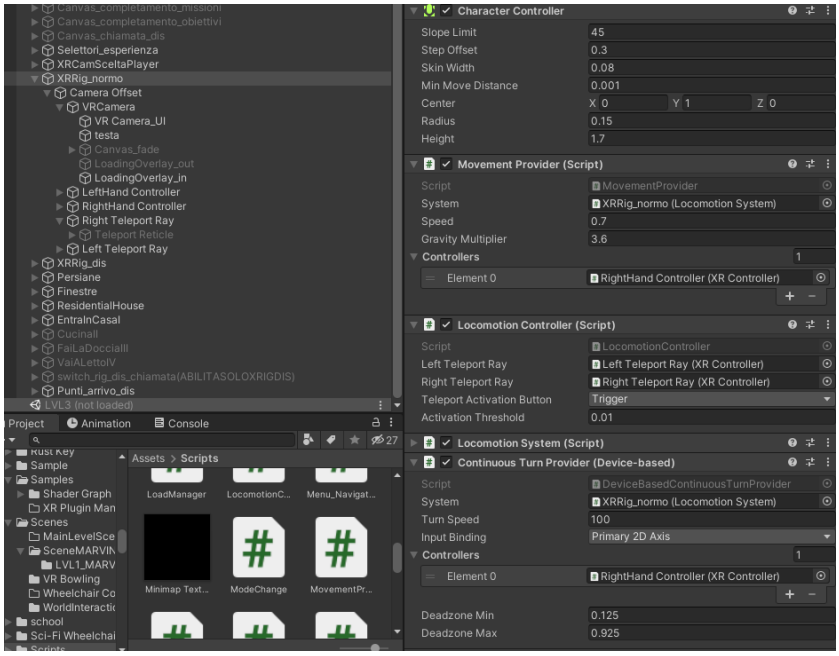

Figura 26. Gli script MovementProvider, LocomotionController e ContinuosTurnProvider associati al GameObject XRRig_normo

Toolkit) e uno via *teleport* (teletrasporto), che utilizza *LocomotionController* di XR e sfrutta

le componenti *XRRayInteractor* e i relativi raggi *renderizzati* via *LineRenderer* (connessi a i *GameObject RightTeleportRay* e *LeftTeleportRay*, annessi alle mani virtuali del giocatore) per spostarsi nelle aree e nei portali che assolvono alla funzione, ai quali sono collegati, rispettivamente, gli *script TeleportationArea* e *TeleportationAnchor* di XR. Per ruotare nello spazio senza dover girare la testa, è inoltre impiegato lo *script* di XR *DeviceBasedContinuosTurnProvider*, che consente al giocatore di ruotare verso destra e sinistra tramite il *thumbstick* di un *controller* (destro in questo caso) e permette di limitare eventuali malesseri dovuti ad un'alterazione percettiva. Nella presente modalità di gioco, come è facile poter immaginare, l'utente dovrà affrontare le missioni restando in piedi (tranne che all'inizio, dove sarà "seduto" nell'auto) e potrà muoversi liberamente nell'ambiente anche camminando – previa configurazione dei confini del *guardian* di Oculus –; tuttavia, spostarsi troppo dalla postazione di origine è sconsigliato per motivi pratici. Nella presente modalità di gioco, come sarà possibile osservare anche dagli *screenshots* proposti in seguito, sono previsti oggetti di scena che rappresentano elementi caratterizzanti l'architettura e le dotazioni interne di una classica abitazione privata. In particolare, bisogna specificare che in tale tipo di esperienza sono stati inseriti oggetti assolutamente non adattivi e, sia gli esterni che gli interni della casa, presentano barriere architettoniche anche totalmente invalicabili: citando alcuni esempi, è possibile incontrare sul percorso scale, gradini e strumenti con i quali interagire adatti esclusivamente ad un pubblico di normodotati, come finestre e porte a battente e tapparelle di tipo meccanico. Fruendo degli ambienti nelle vesti del *player* senza alcuna difficoltà nella deambulazione, non è possibile notare la maggior parte degli elementi sfavorevoli all'altra categoria di utenti in esame, poiché è facile raggiungere gli obiettivi preposti senza dover essere costretti ad essere seduti su una carrozzina. In questo livello è possibile, dunque, interagire con oggetti di tipo comune, non ottimizzati su misura di una particolare categoria di utenza: nella scena, sono infatti presenti finestre che si aprono verso l'interno, porte a battente che occupano tanto spazio in fase di apertura e chiusura, cassetti ingombranti, pensili posti in alto, interruttori ad altezza standard (con parti dedicate ai comandi molto piccole) e tapparelle a corda. È possibile "impugnare" tutti gli oggetti che prevedono questo tipo di interazione in modo molto semplice: basta che gli stessi siano dotati di un *Rigidbody*, di un *collider* e dello *script XRGrabInteractable*[85], (figura 25) un'estensione

[85] Tale script verrà trattato nell'apposita sezione su XR.

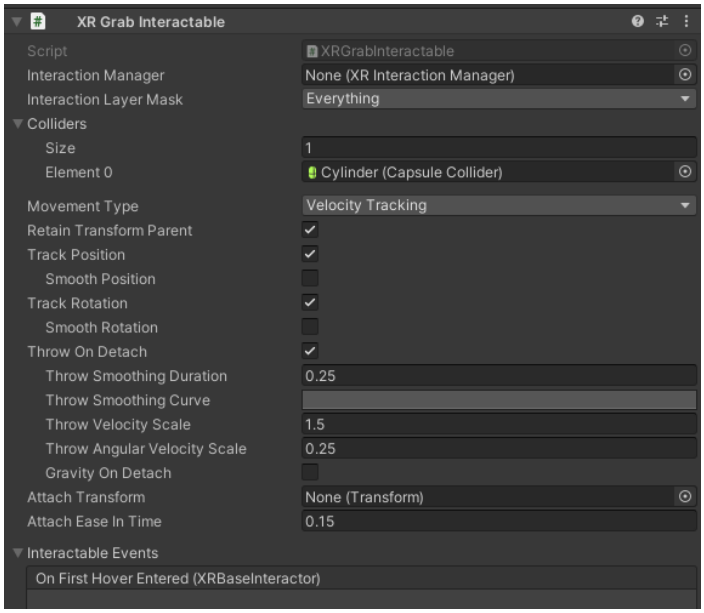

Figura 27. XRGrabInteractable nell'inspector di Unity con le impostazioni di default e un Capsule Collider caricato nel campo relativo alle collisioni da sfruttare

della classe *XRBaseInteractable* del plugin *XRInteractionToolkit* che consente agli stessi di essere "afferrati" da qualsiasi tipo di *GameObject* utilizzi *XRInteractor*. Questo tipo di interazione è applicato a tutte le maniglie e agli oggetti che richiedono di essere "sollevati", "tirati" o "spinti", applicando loro le forze fisiche impartite con il movimento dei *controller* di Oculus ed è possibile trovarlo in entrambe le scene del simulatore che prevedono elementi 3D dinamici. Saranno proposti degli *screenshots* alla fine di questa sezione, attraverso i quali si potranno osservare gli ambienti e gli oggetti di gioco inclusi nella scena in esame, il cui funzionamento tecnico dei più salienti verrà illustrato trattando gli strumenti applicati in fase di implementazione.

Esperienza 2 – Giocatore Normodotato in casa senza ausili

Figura 28. Selezione dell'esperienza con giocatore "disabile"

Gli ambienti e il tipo di interazione con gli oggetti di gioco proposti in questo tipo di esperienza sono perlopiù analoghi a quella di cui sopra, in quanto viene sfruttata la stessa scena e gli elementi aggiuntivi sono caricati dinamicamente al momento dell'esecuzione. Tuttavia, la sostanziale differenza è costituita dal *player*: in questa modalità, infatti, sarà possibile fruire degli spazi impersonificando una persona su sedia a rotelle, con difficoltà motorie agli arti inferiori. Lo stesso, chiamato nell'editor *XRRig_dis*, è costruito sulla base del giocatore normodotato per quanto concernono le impostazioni della telecamera e delle interazioni con i *controller* di Oculus (è impiegato il plugin *XRInteractionToolkit* come nel caso precedente), ma vi si differenzia principalmente per la gestione del movimento. Infatti, il caso in esame prevede nella gerarchia il modello 3D di una sedia a rotelle[86] – opportunamente personalizzata per assolvere allo scopo –, per tentare di donare all'utente un'esperienza maggiormente immersiva e la sensazione di trovarsi davvero su un ausilio per la deambulazione a ruote. A tal proposito, è stato implementato un differente tipo di movimento, che riproducesse il *feeling* di una vera carrozzina di tipo motorizzato, in quanto a sensibilità in termini di forze fisiche in rapporto al terreno ed effetti sonori e visivi tipici del funzionamento della stessa. All'inizio della simulazione, quando il giocatore si trova "in auto", sono applicate le stesse caratteristiche del precedente *player* riguardo al movimento lineare e alla rotazione ma, appena lo stesso si posiziona sul primo portale della Missione I,

[86] Sulla stessa è posizionato il *prefab* di un manichino (*dummy*), "tagliato" a livello del bacino, per permettere al giocatore di avere la parte della testa con l'*HMD* libera da ostacoli ma di donargli, al contempo, la sensazione di essere seduto, potendosi "vedere" le gambe flesse e i piedi abbassando lo sguardo.

"monta" sulla carrozzina e tutte le azioni motorie vengono, da quel momento in poi, gestite da uno *script* creato *ad hoc*, chiamato *sediaMotorizzataV2* (codice in appendice).

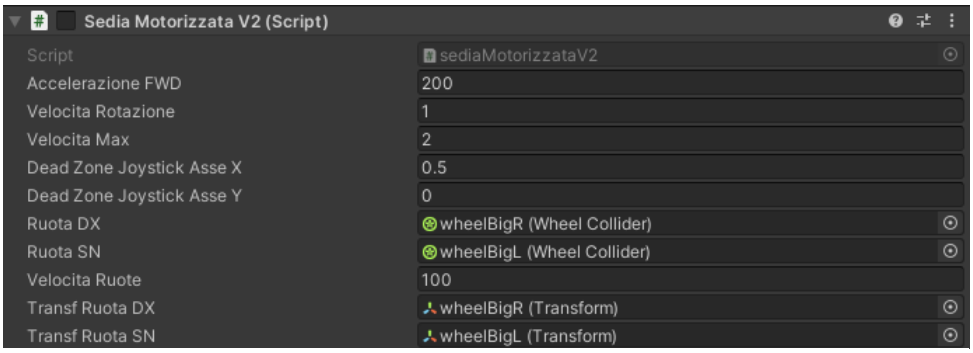

Figura 29. Script sediaMotorizzataV2 nell'inspector di Unity

Come è possibile osservare in figura, esso include una sezione dedicata alla definizione delle forze fisiche da applicare a sedia e giocatore (quali accelerazione e velocità di rotazione nello spazio) e permette alle ruote di girare a seconda della direzione di movimento, più o meno rapidamente, sfruttando le componenti *WheelCollider* associate ai *GameObject* ad esse relativi. Inoltre, prevede una definizione della *deadzone* da applicare al *joystick* con il quale vengono impartiti i movimenti, agendo sugli assi X e Y, e si occupa della riproduzione delle componenti sonore di avvio, di *idle* e di spegnimento dei "servomotori" della sedia, collegate allo stesso *XRRig_dis*.[87] Per donare all'utente una sensazione di "blocco", in grado di suscitargli persino frustrazione, su *XRRig_dis* è stato posto un *Rigidbody* con un valore di massa più alto di altri oggetti dinamici, in grado di fargli percepire, anche se astrattamente, il peso di una parte del corpo "paralizzata" e l'ingombro della sedia (i cui *colliders* applicati alle relative componenti limitano i movimenti in molte parti del gioco, cozzando con altre parti presenti nell'ambiente circostante).

Un'altra caratteristica che distingue questo tipo di esperienza dalla precedente è la presenza, nelle principali aree della casa (e in posti più complessi da raggiungere "in autonomia", come il letto o la doccia), di pulsantiere tramite le quali è possibile interagire per "chiamare" un assistente, in grado di "aiutare" il giocatore a muoversi in situazioni dove, per la presenza di barriere architettoniche, non è possibile spostarsi o compiere azioni autonomamente. Tale strategia è stata introdotta nel presente livello per marcare sul fattore dell'inadeguatezza di molti luoghi a determinate categorie di utenza non dotate di totale autonomia – resi

[87] Il relativo codice sarà descritto nella sezione relativa agli strumenti impiegato per l'implementazione.

totalmente inaccessibili a causa della mancanza di sussidi progettati su misura – ma la loro presenza si è rivelata necessaria anche per permettere effettivamente al giocatore di portare a termine le missioni e poter navigare nella casa poiché, senza una strategia simile, rimarrebbe bloccato dinanzi a parti dell'abitazione impossibili da valicare, quali scale e dislivelli anche minimi.

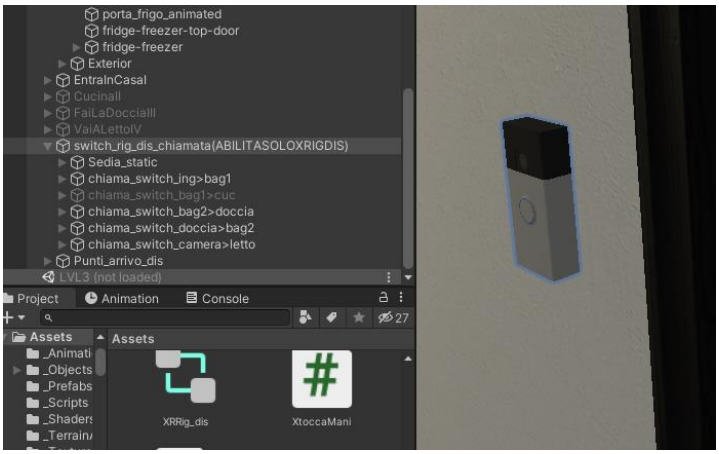

Figura 30. Interfaccia di "chiamata assistente" e relativi elementi nella gerarchia di Unity. In punti_arrivo_dis sono contenuti i GameObject con le coordinate di destinazione

Queste interfacce, che nell'aspetto ricordano dei citofoni, sono aggiunte all'ambiente virtuale al caricamento del giocatore *XRRig_dis* e includono una parte dinamica, con la quale è possibile interagire con un semplice "tocco", tramite lo *script chiamata_aiutante* (presente in appendice), che assolve alle funzioni di mostrare il *prompt* relativo al testo di chiamata dell'assistente, porre un'*overlay* sulla scena per oscurare l'ambiente e "catapultare" l'utente in un'altra posizione della casa prestabilita, alla quale è assegnato un *GameObject* con una componente *Transform*, con le coordinate x, y e z del "luogo" di destinazione.

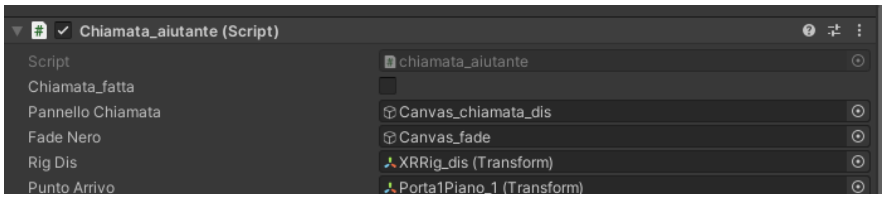

Figura 31. Script chiamata_aiutante nell'inspector di Unity

Come già accennato, tale modalità prevede le medesime tecniche di interazione con gli oggetti di scena dell'esperienza con il giocatore normodotato (basate su eventi *triggerati* da collisioni e sugli *script* per il *grabbing* del pacchetto XR); tale scelta è stata mossa dall'effettiva necessità di far interfacciare l'utente, nei "panni" di un portatore di disabilità dello spettro motorio, con un ambiente non accessibile, creato su misura di una persona senza particolari necessità.

5.2.2.3 Livello 3

È possibile fruire del livello 3 del simulatore selezionando la voce *Abitazione con ausili*, accessibile direttamente dal menu di avvio. Tale modalità di gioco è costruita, per motivi pratici, su una scena di Unity distinta, in quanto è stato necessario modificare l'ambiente virtuale rispetto a quello proposto nelle esperienze descritte precedentemente. La struttura di base della casa su due livelli è stata mantenuta (anche sul lato grafico), così come molte parti non importanti ai fini dell'interazione e delle missioni, ma sono state apportate modifiche anche radicali alle zone ove il giocatore (che interpreta un portatore di disabilità su sedia a rotelle) deve interfacciarsi con gli oggetti di scena, concentrati in zone quali cucina, bagni e camera da letto. Inoltre, sono stati aggiunti un sistema di automazione, che permette il controllo remoto – tramite *smartphone* in-*game* – di varie componenti domestiche, e riproduce il funzionamento di un vero impianto domotico) e elementi esclusivi di ausilio per la categoria in esame, tra i quali spiccano un montascale, dei condizionatori con controllo centralizzato e un aspirapolvere robot. Inoltre, porte, finestre e altre componenti mobili appartenenti all'arredo e alla struttura dell'abitazione sono stati ottimizzati per il giocatore su carrozzina, per consentirgli di muoversi senza intralci. A tal proposito, sono state scelte finestre a ghigliottina e di tipo *vasistas*, porte scorrevoli e tapparelle motorizzate e gli stessi elementi sono stati connessi ad apposite pulsantiere e allo *smartphone* per il controllo remoto, in modo da semplificarne la fruizione al giocatore. Interruttori e pulsantiere sono state, altresì, posti ad un'altezza accessibile (più in basso rispetto alla norma), in linea quanto più fedelmente (in scala) alle normative sulla progettazione di strutture dedicate a portatori di disabilità su carrozzina, come descritto precedentemente; i relativi bottoni, inoltre, sono stati designati in modo da risultare di facile interazione e visibilità: a tal proposito, è stato scelto di eliminare eventuali sistemi a leva e di lasciare spazio a grossi elementi retroilluminati, azionabili anche con un colpo poco preciso delle mani. Sia all'interno, che all'esterno, sono state inserite alcune rampe, in prossimità di dislivelli, per "dotare" il giocatore di una totale autonomia e permettergli di non bloccarsi durante la fruizione del simulatore. Le stanze adattate più radicalmente al *target* di utenza in esame sono la cucina, i due bagni e la camera da letto; dagli *screenshots* proposti al termine di queste sezioni si potranno apprezzare visivamente le differenze tra gli ambienti proposti nel simulatore, di seguito illustrate in modo sommatorio. In riferimento alla cucina, la stanza entro la quale il giocatore deve svolgere la missione più ricca di obiettivi (la II), sono state effettuate alcune modifiche ai

piani di lavoro, per permettere all'utente di inserirvi "le gambe" in modo agevole e avvicinarvisi completamente per compiere ogni tipo di azione proposta. A tal proposito, sono stati eliminati mobili "pieni" ed è stato dato spazio a compenti di arredo prive di tradizionali cassetti, scomodi da trainare da una posizione seduta, sostituiti da vassoi estraibili dotati di un'anta *vasistas* di chiusura. I pensili posti troppo in alto sono stati eliminati del tutto e, al loro posto, sono stati preferiti mobili sottopiano con più ripiani di appoggio e ante scorrevoli. Il rubinetto del lavello è stato posizionato in modo idoneo alla fruizione da parte di una persona non in grado di deambulare e il forno, l'elettrodomestico più ingombrante della cucina, è stato posto sul *top* della stessa, al fine di liberare lo spazio al di sotto del piano cottura (appositamente ridimensionato in termini di larghezza). Entrambi i bagni, il principale e quello di servizio, sono stati dotati di maniglioni ausiliari nelle prossimità del lavabo, del WC e della doccia e di campanelli di allarme, utili a chiamare un assistente in caso di necessità. Gli stessi sanitari[88] inseriti nel livello sono di tipo adatto alla categoria e conformi alle linee guida: è possibile trovare, infatti, gli stessi progettati in modo accessibile e con maniglioni e sedute apposite a corredo. La camera da letto, infine, è stata dotata di appositi sollevatori, utili nel caso di utenti con grave disabilità e la mobilia è stata ridotta all'essenziale, ovvero ad un solo comodino e ad un armadio a muro dotato di anta scorrevole.

Il *player* "con disabilità", nel presente livello, presenta caratteristiche pressoché identiche a *XRRig_dis* della precedente esperienza di gioco e lo stesso può affermarsi in quanto al sistema di interazione con gli oggetti di scena impiegato e al movimento. Sono stati, tuttavia, modificati gli *offset* e la grafica dell'inventario che segue il giocatore, al fine di impedire a quest'ultimo di interferire con le collisioni presenti sui *mesh* della sedia a rotelle, ma tale intervento non apporta alcun cambiamento al suo funzionamento.

Di seguito saranno inserite alcune schermate dei livelli appena introdotti, in modo da permettere al lettore di comprendere quanto appena citato e prepararsi in modo più agevole alla descrizione delle tecniche di implementazione impiegate.

[88] Nel terzo livello, nel primo bagno, posto accanto alla camera da letto, la vasca è stata eliminata.

5.2.3 Screenshots

Nella presente sezione è possibile consultare alcuni *screenshots* tratti dai livelli del simulatore in esecuzione, all'ultima *build* disponibile.

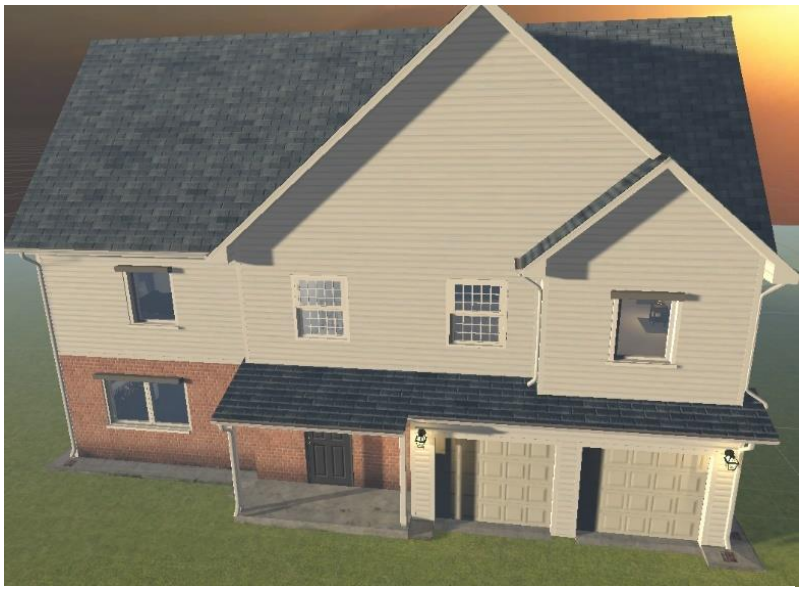

Figura 32. Livelli 1-2-3. Esterni della casa, lato frontale, nella scena di Unity.

Figura 33. Pannello prima missione e auto, livelli 1-2.

Figura 34. Ingresso, livelli 1-2

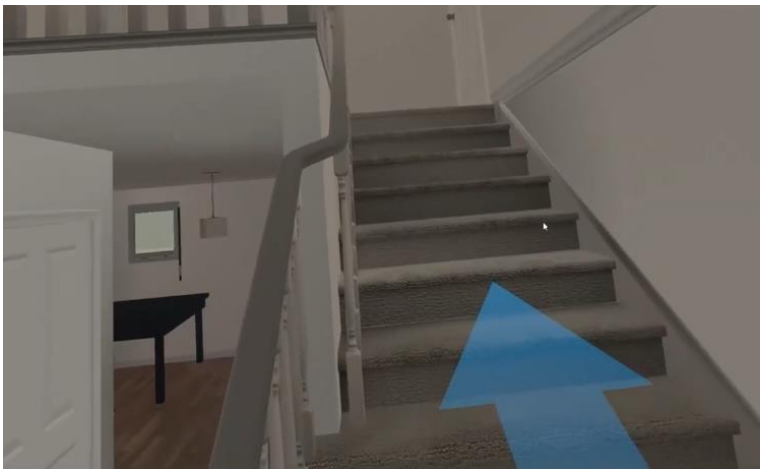

Figura 35. Zona scale, livelli 1-2

Figura 36. Lavaggio delle mani, bagno principale, livelli 1-2

Figura 37. Pensile cucina, livelli 1-2

Figura 38. Interni doccia, livelli 1-2

Figura 39. Giocatore "disabile" che scende dall'auto, livello 3

Figura 40. Zona notte, livelli 1-2

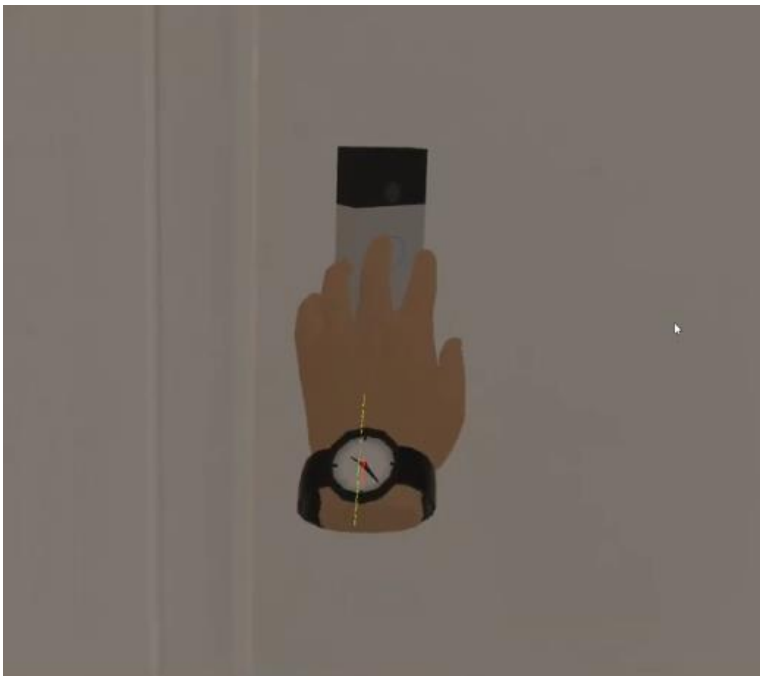

Figura 41. Dettaglio pulsante "chiamata assistente", livello 2

Figura 42. Interazione con la card per l'accesso alla casa, livello 3

Figura 43. Discesa con montascale, livello 3

Figura 44. Interazione con cassetto vasistas della cucina, livello 3

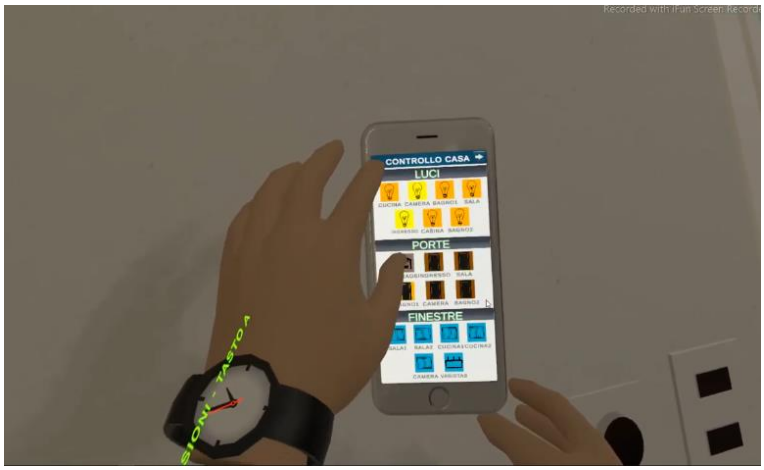

Figura 45. Smartphone e dispositivi di controllo finestre/tapparelle, livello 3

Figura 46. Interni doccia, livello 3

Interni – Vista d'insieme

In questa sottosezione sono riportati alcuni *collage* di *screenshots* relativi alle principali stanze dei livelli che compongono il simulatore; per una vista d'insieme, tali schermate sono prelevate dall'*editor* delle scene di Unity.

Livelli 1-2

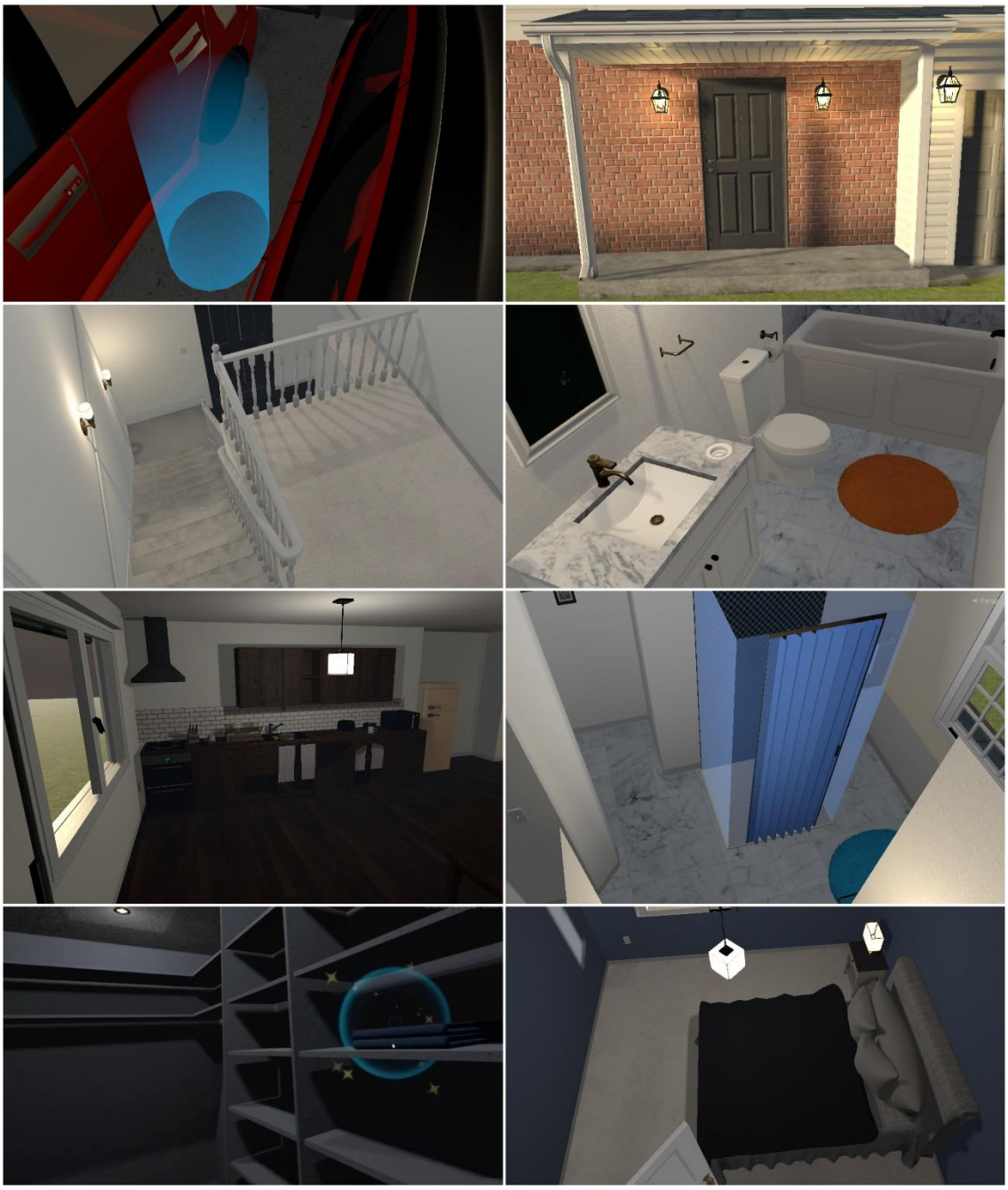

Figura 47. Riepilogo interni stanze principali livelli 1-2

Livello 3

Figura 48. Riepilogo interni stanze principali livello 3

5.3 Nel vivo dell'implementazione: tecniche e strumenti

Di seguito sono descritti gli strumenti impiegati in fase di implementazione, alcuni dei quali già citati nei paragrafi precedenti, in itinere alla presentazione dei livelli. Per la comprensione degli argomenti citati, è necessario conoscere almeno superficialmente la terminologia di base relativa al software Unity e avere una minima esperienza con la logica relativa a linguaggi di programmazione ad oggetti come il C#.

5.3.1 Linguaggio C#, classe MonoBehaviour e funzioni evento di base

È necessario introdurre questa sezione trattando il linguaggio utilizzato per la creazione degli *script* e delle tecniche di base che hanno reso possibile la realizzazione del progetto. Come già accennato nel primo capitolo, Unity è un motore grafico in grado di supportare i linguaggi C#, JavaScript e Boo, a seconda dell'architettura dello scopo della *build* da creare e la sua destinazione; nel caso del simulatore in esame, è stato utilizzato il primo citato tra questi, C#, poiché ampiamente supportato e con un'ampia documentazione a disposizione sul sito ufficiale e su canali di terze parti.

È necessario trattare di questo linguaggio in particolare riferendosi alla classe *MonoBehaviour*, dalla quale derivano secondo il relativo concetto di ereditarietà, tutti gli *script* creati, con la quale condividono le proprietà di base. Ogni *script* creato direttamente dall'*inspector* di Unity e associato a un qualsiasi *GameObject*, presenta le seguenti caratteristiche:

```
using UnityEngine;
using System.Collections;
public class ScriptDiProva : MonoBehaviour {
   //Use this for initialization
   void Start () {
}
   // Update is called once per frame
   void Update () {
}
}
```

Come è possibile notare dallo *snippet* di codice riportato, ogni *script* include di *default*, oltre alla struttura della classe (ereditata da *MonoBehaviour*), due funzioni di base, *Start* ed *Update*, entro le quali è possibile definire le istruzioni da eseguire, rispettivamente, al primo ciclo di avvio dello script, prima di qualsiasi altra funzione (insieme alla simile *Awake*, accennata di seguito) e durante l'esecuzione dello stesso, fino a quando si incontra una condizione di terminazione (che può essere gestita da un valore *booleano*, ad esempio).

Nei prossimi paragrafi verranno trattate brevemente le funzioni evento di base di Unity, impiegate durante la stesura della maggior parte degli *script* del progetto del simulatore, delle quali saranno riportate alcune parti di codice contenenti applicazioni pratiche delle stesse.

5.3.1.1 Start

La funzione *Start*, nel progetto del simulatore e in generale, è impiegata in particolare per predisporre il campo per l'esecuzione di successivi comandi: è, ad esempio, utile nell'assegnazione di valori a variabili, per la creazione di liste, dizionari e *array* di *GameObjects* e per la ricerca di oggetti di scena (e relativi componenti associate) rispondenti a determinati parametri, come nomi o *tag*. Tutto ciò che è contenuto nel corpo di `Start`, in poche parole, serve a predisporre il campo alle altre funzioni, permettendo di caricare nell'*editor* quanto necessario. Nel seguente esempio, porzione dallo *script sediaMotorizzataV2*, è possibile osservare come la funzione evento *Start* assolva al riempimento dell'*array* di *GameObject AudioSource[]*, precedentemente dichiarato e inizializzato, con le componenti di tipo *Audiosource* collegate all'oggetto *XRRig_dis* (*player* "con disabilità"), rispettivamente alle posizioni 0, 1 e 2 dello stesso:

```
void Start(){
   audioSources = GetComponents<AudioSource>();
   MotoreAvvio = audioSources[0];
   MotoreAcceso = audioSources[1];
   MotoreStop = audioSources[2];
}
```

Tale tipo di costrutto assolve, inoltre, anche all'avvio di tutto ciò sia necessario eseguire solo una volta e in modo preliminare rispetto alle altre componenti e viene in ausilio anche, ad esempio, quando bisogna avviare una *coroutine* (delle quali si tratterà più avanti) all'abilitazione in tempo di *runtime* di uno *script*, come nelle seguenti righe di codice, tratte dallo *script WaitTotSecSimple* (responsabile dell'eliminazione di alcuni oggetti di scena inseriti in un *array*, in seguito all'abilitazione dello *script missioneCompiutaSimple*, che gestisce gli eventi relativi al completamento degli obiettivi di alcune missioni):

```
void Start()
  //avvio della coroutine attendiAbilitaElimina
  StartCoroutine(attendiAbilitaElimina ());
}
```

Le applicazioni pratiche di *Start* sono pressoché infinite e variano a seconda del codice del compito al quale deve assolvere uno *script*.

5.3.1.2 Awake

Una funzione evento simile a quella appena citata è *Awake*, che differisce da *Start* per la priorità di avvio e per il suo privilegio di essere avviata inequivocabilmente ad ogni esecuzione: mentre la precedente è utile per caricare elementi di gioco in via prioritaria, questa assolve all'esecuzione di codice che necessita di "partire" senza alcuna latenza (ancora prima di *Start*) e indipendentemente dallo stato (abilitato o meno) dello *script* di appartenenza. È possibile osservarne la sua applicazione pratica nell'esempio sotto riportato, tratto dallo *script CaricaEDisattiva*, responsabile della disattivazione dei *GameObject XRRig_normo* e *XRRig_dis* se non in uso e impiegato per porre una toppa a un problema di caricamento del sistema di *teleport* sul primo *player*, dovuto ad un *bug* riscontrato nel *plugin XRInteractionToolkit*:

```
void Awake(){
if (svegliato==false){
         this.gameObject.SetActive(false);
         svegliato=true;
        }
}
```

Lo *script* in questione viene eseguito al caricamento degli oggetti sui quali è posto, indipendentemente, dunque, dalla spunta di stato (corrispondente allo stato del *booleano* enabled della classe delle componenti di Unity *Behaviour*) nell'*inspector* e ha sempre la priorità di esecuzione rispetto ad ogni altro programma collegato al *GameObject*.

5.3.1.3 Update

Il metodo *Update*, utilizzato nella maggioranza degli *script* realizzati, si occupa di aggiornare lo stato degli oggetti e componenti ad ogni *frame*, fino ad una condizione di terminazione imposta, ad esempio, tramite il cambio di stato di una variabile *booleana*. Tale funzione è ampiamente impiegata nella gestione del movimento dei *GameObject* all'interno del gioco e ha la caratteristica di dipendere dalla frequenza di aggiornamento dei fotogrammi, espressa in secondi (spesso abbreviata in *FPS*). Dunque, a differenza di *Start* o di altre funzioni evento, come quelle relative all'interazione con le collisioni successivamente descritte, *Update* esegue ciclicamente il codice contenuto al suo interno finché una condizione è soddisfatta o, nel caso non ne fosse prevista alcuna, anche durante l'intero *runtime* del gioco. Un esempio della sua applicazione è possibile trovarla nel codice dello *script levaTrigger*, che gestisce la commutazione delle leve dei rubinetti nel simulatore (trattato in seguito nell'apposita sezione):

```
void Update(){
     leverWasSwitched = false;
float offDistance = Quaternion.Angle(this.transform.localRotation,  OffHingeAngle());
float onDistance = Quaternion.Angle(this.transform.localRotation, OnHingeAngle());
     //controllo la posizione della leva e se deve essere alzata o abbassata
     bool shouldBeOn = (Mathf.Abs(onDistance) < Mathf.Abs(offDistance));
     if (shouldBeOn != leverIsOn) {
        leverIsOn = !leverIsOn;
        if (leverIsOn){
           Effetto.SetActive(true);
           suonoAcqua.Play();
        }else{
           Effetto.SetActive(false);
           suonoAcqua.Stop();
        }
           leverWasSwitched = true;
        }
}
```

Nel presente caso, il cui codice è riportato interamente in appendice, nella funzione Update vengono gestite la posizione minima e massima delle *Hinge Joint* collegate ai *GameObject* dei miscelatori dei rubinetti presenti nelle scene e vengono controllati, in base allo stato dei tre *booleani* (*shouldBeOn*, *leverIsOn* e *leverWasSwitched*), suoni ed effetti visivi che simulano l'erogazione dell'acqua. Le variabili *booleane*, assumendo i valori *true* e *false* a seconda dello stato di esecuzione, consentono di entrare e uscire dai rami *if-else* e di interrompere l'esecuzione di *Update* quando a leverWasSwitched viene assegnato true (che corrisponde al momento in cui il giocatore "chiude" il rubinetto).

5.3.1.4 FixedUpdate

FixedUpdate differisce dal funzionamento della precedente funzione per la sua caratteristica di non tener conto del *framerate*, ma di essere eseguita ogni 0.02 secondi; il suo utilizzo è consigliato quando è necessario impartire comandi relativi alle forze fisiche e che hanno bisogno, in ogni caso, di tempi di risposta minima, di un basso utilizzo di risorse e di alta precisione. Nel simulatore, l'intera gestione del movimento del giocatore su sedia a rotelle è affidata a *FixedUpdate*, presente in appendice all'interno di *SediaMotorizzataV2*. Di tale *script* si approfondirà in seguito.

5.3.2 Il framework XR

Per un migliore mantenimento nel tempo delle tecniche implementative utilizzate riguardo la RV, è stato necessario utilizzare un *plug-in* di terze parti che offrisse massima versatilità e destasse di una solida documentazione. Nei precedenti sviluppi del simulatore era stato utilizzato il pacchetto di integrazione proprietario di Oculus (*Oculus Integration SDK*) per l'accesso alla RV e alla creazione dei contenuti, ma quest'ultimo è divenuto obsoleto nel tempo e continuare ad utilizzarlo sarebbe divenuto impossibile, volendo restare al passo delle

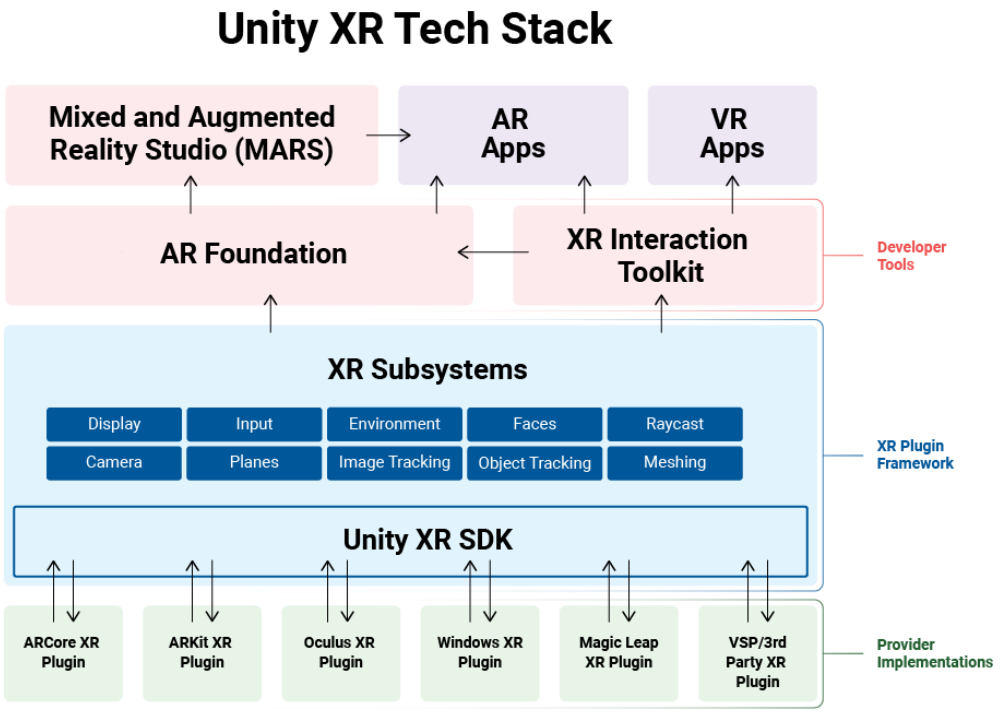

Figura 49. Architettura del plugin XR

tecnologie incluse nelle ultime versioni di Unity. La scelta è ricaduta sul *plug-in* XR[89], la cui architettura è supportata in Unity a partire dalla versione 2019.4; tale estensione comprende numerosi *script* in grado di consentire la comunicazione *hardware-software* tra il visore e il motore grafico e l'implementazione di diversi contenuti nelle scene ivi realizzate, molti dei quali riguardano il movimento e l'interazione lato giocatore. Si tratta di una collezione di strumenti multipiattaforma per la Realtà Mista, che consente agli utenti di accedere e di produrre contenuti per numerosi dispositivi per l'accesso alla RV e alla Realtà Aumentata, tra i quali rientrano anche i più recenti visori di casa Oculus, come il Quest, utilizzato durante lo sviluppo del simulatore. Il *plugin*, installabile direttamente dal *Package Manager* di Unity, insieme alla sua componente per l'integrazione nell'ambiente dell'editor (*XR Plugin Management*), include degli strumenti di base (*core*) fondamentali a comunicare con il visore e degli *script ad-hoc* utilizzabili sugli oggetti di gioco nelle scene. Tuttavia, nel caso in esame la sola integrazione dell'estensione non è sufficiente ad assolvere alla funzione di fornire gli strumenti idonei alla comunicazione con il visore Oculus utilizzato né, tantomeno, ad offrire i *tools* necessari all'interazione del *player* con le componenti di gioco. Per estendere il

[89] Per maggiori informazioni sull'architettura di XR, visitare la relativa pagina web: https://docs.unity3d.com/Manual/XRPluginArchitecture.html.

supporto ai visori di casa Oculus, è necessario, infatti, integrare in Unity il pacchetto *Oculus XR Plugin*, installabile direttamente dalla scheda in *Project Settings* di Unity di *XR Plugin Management* e, per includere gli strumenti necessari all'interazione, invece, bisogna aggiungere alla collezione il *package XR Interaction Toolkit*. Nei prossimi paragrafi verranno descritte nel dettaglio queste due estensioni, indispensabili per addentrarsi allo sviluppo del simulatore in Unity utilizzando il suddetto hardware per la RV.

5.3.2.1 Oculus XR Plugin e sistema di input

Il *plugin Oculus XR*[90] estende le funzionalità *core* del *plugin* XR, includendo il supporto ai visori di casa Oculus; in particolare, esso aggiunge un sottosistema per il rendering *stereo* al *plugin* XR, permettendone il funzionamento con alcune librerie grafiche, quali quelle riguardanti *DirectX 11* (per Windows, per il supporto ai visori Rift e Rift S), *OpenGL* e *Vulkan* (per Android, compatibili con le architetture degli *standalone* Quest e Quest 2). Inoltre, include un sistema di input – che si aggiunge a quello di *default* di Unity – che permette al motore grafico di comunicare con l'*HMD* e i *controllers* di Oculus e di tracciarne la posizione nello spazio, il movimento, il tocco e la pressione dei tasti funzione.

5.3.2.1.1 DeviceLayouts

Tra i numerosi *script* a corredo del *plugin*, è necessario citare proprio un importante programma in grado di monitorare la posizione nello spazio e l'input *dell'headset* e dei *controller* Oculus: *DeviceLayouts*, contenuto nella cartella *InputSystem* della libreria. Lo *script* include sette classi: *OculusHMD*, *OculusTouchController*, *OculusTrackingReference*, *OculusRemote*, *OculusGoController*, *OculusHMDExtended* e *GearVRTrackedController*[91].

La prima citata, *OculusHMD*, che eredita da *XRHMD*, include la ridefinizione dei controlli sulle coordinate (di tipo *Vector3* e *Quaternion*) del sistema di *input* predefinito di Unity, in modo da consentire il tracciamento del casco di RV in ogni suo movimento nello spazio, per quanto riguarda lo spostamento sui tre assi e la rotazione. Nella seguente porzione di codice è possibile osservare, ad esempio, la definizione e il relativo assegnamento (nel metodo *FinishSetup*) dei valori alle variabili di tipo *Vector3Control* e *QuaternionControl* relative alla

[90] Documentazione e *changelog* di *Oculus XR Plugin* alla versione 1.9.1. Link alla pagina web: https://docs.unity3d.com/Packages/com.unity.xr.oculus@1.9/manual/index.html.
[91] Il codice completo è consultabile al link indicato in appendice, nella sezione relativa alle librerie di XR.

posizione spaziale dell'*HMD* per quanto concerne il suo *centerEye*, "occhio centrale", corrispondente alla visione "unita" (*stereo*) delle due lenti:

```
    public class OculusHMD : XRHMD
    {
        [omissis]
        [InputControl]
        public new Vector3Control centerEyePosition { get; private set; }
        [Preserve]
        [InputControl]
        public new QuaternionControl centerEyeRotation { get; private set; }
        [Preserve]
        [InputControl]
        public Vector3Control centerEyeAngularVelocity { get; private set; }
        [Preserve]
        [InputControl]
        public Vector3Control centerEyeAcceleration { get; private set; }
        [Preserve]
        [InputControl]
        public Vector3Control centerEyeAngularAcceleration { get; private set; }

        protected override void FinishSetup()
        {
            [omissis]
            centerEyePosition = GetChildControl<Vector3Control>("centerEyePosition");
            centerEyeRotation = GetChildControl<QuaternionControl>("centerEyeRotation");
            centerEyeAngularVelocity = GetChildControl<Vector3Control>("centerEyeAngularVelo
city");
            centerEyeAcceleration = GetChildControl<Vector3Control>("centerEyeAcceleration")
;
            centerEyeAngularAcceleration = GetChildControl<Vector3Control>("centerEyeAngular
Acceleration");
        }
    }
```

La seconda, *OculusTouchController* (che eredita da *XRControllerWithRumble*), allo stesso modo della precedente assegna alle variabili di input i valori necessari alla localizzazione spaziale e alle forze fisiche da applicare ai *controller touch* destro e sinistro del visore Oculus e gestisce la mappatura dei pulsanti degli stessi. Nel seguente esempio è possibile osservare alcuni parametri della classe relativi rispettivamente alla mappatura dei *thumbsticks*, alla gestione di alcuni eventi ad essi correlati e al controllo delle forze fisiche relative alle periferiche impugnabili:

```
public class OculusTouchController : XRControllerWithRumble
    {
        [omissis]
        protected override void FinishSetup()
        {
            base.FinishSetup();

            thumbstick = GetChildControl<Vector2Control>("thumbstick");
            [omissis]
            thumbstickClicked = GetChildControl<ButtonControl>("thumbstickClicked");
            thumbstickTouched = GetChildControl<ButtonControl>("thumbstickTouched");
            [omissis]
            devicePosition = GetChildControl<Vector3Control>("devicePosition");
            deviceRotation = GetChildControl<QuaternionControl>("deviceRotation");
            deviceVelocity = GetChildControl<Vector3Control>("deviceVelocity");
```

```
            deviceAngularVelocity = GetChildControl<Vector3Control>("deviceAngularVelocity")
;
            deviceAcceleration = GetChildControl<Vector3Control>("deviceAcceleration");
            deviceAngularAcceleration = GetChildControl<Vector3Control>("deviceAngularAccele
ration");
        }
    }
```

Scorrendo nello *script*, è poi possibile leggere della classe *OculusTrackingReference*, utile a definire ulteriori parametri per il controllo della posizione dei *controller touch* e di *Oculus Remote*, per la gestione del telecomando di Oculus, previsto per alcuni modelli di visore, ma non utilizzato nel caso in esame[92]. Successivamente, sono presenti le classi minori *OculusHMDExtended* (un *helper* per l'input del *touchpad*, ereditata da *OculusHMD*), *OculusGoController* e *GearVRTrackedController*, relative al visore OculusGo e a dispositivi *standalone* affini, come il GearVR della Samsung, non trattate in questi capitoli.

5.3.2.1.2 ProvaXRInput

Per sfruttare le funzionalità offerte dal sistema di input previste nel *plugin* XR in modo più semplice, ad esempio per consentire all'utente di accedere a menu e risorse previste nel simulatore tramite i pulsanti dei *controller touch* (come il pannello delle missioni), è stato utilizzato uno *script* su misura[93], *provaXRInput*[94]. Questo, collegato ad ogni *XRRig*, raggruppa in un dizionario la mappatura dei tasti funzione e permette di associarvi, direttamente dall'*inspector* di Unity (figura 32), un comando, eseguibile alla chiamata degli eventi *built-in OnPress* e *OnRelease*[95], corrispondenti alla pressione e al rilascio dei pulsanti.

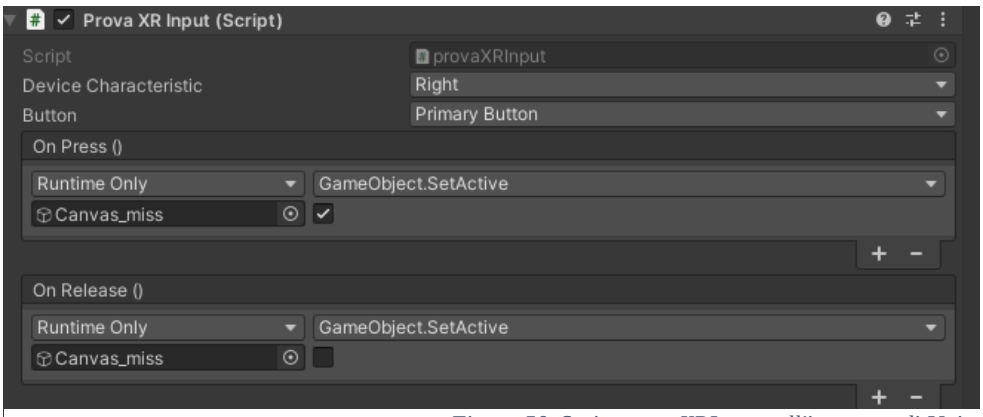

Figura 50. Script provaXRInput nell'inspector di Unity

[92] Entrambe le classi derivano dalla principale InputDevice.
[93] Il suddetto *script* è una versione modificata dell'originale a cura dell'utente Fariazz della community di Unity. Link al topic di discussione: https://forum.unity.com/threads/any-example-of-the-new-2019-1-xr-input-system.629824/.
[94] Il nome dello *script* è rimasto alla sua prima versione. È prevista una ridenominazione in eventuali future versioni.
[95] Sono stati inseriti esclusivamente gli eventi in questione, poiché erano gli unici necessari allo scopo.

Il codice (interamente riportato in appendice), prevede una logica piuttosto semplice: nello *script* vengono richiamate le caratteristiche dei *device* supportati contenute in due dizionari: deviceCharacteristic, di tipo InputDeviceCharacteristics (contenuto nell'*input system* di XR) e button, di tipo ButtonOption. Il primo serve a contenere le voci relative alle periferiche di *input*, come il *controller* destro ("Right") nella precedente immagine e il secondo, invece, include le informazioni relative al tipo di bottone da impiegare nella chiamata degli eventi, come Primary Button (dicitura corrispondente al tasto X per il *controller* sinistro ed A per il destro)[96]. Per la preparazione alla fruizione del sistema di *input* offerto e alla relativa lettura dei comandi impartiti, lo *script* si avvale della funzione evento Awake per leggere in maniera preliminare i contenuti delle liste e raccogliere, dunque, le etichette relative alla mappatura dei tasti dei dispositivi. In Update – che include un'iterazione per scorrere nelle liste –, invece, come riportato nello *snippet* a seguire, vengono invocati OnPress e OnRelease, a seconda della pressione o del rilascio dei pulsanti scelti nell'inspector; il controllo dei suddetti tasti è affidato al *booleano* IsPressed, che cambia valore in base allo stato dei bottoni.

```
void Update()
    {
        InputDevices.GetDevicesWithCharacteristics(deviceCharacteristic, inputDevices);
        for (int i = 0; i < inputDevices.Count; i++)
        {
            if (inputDevices[i].TryGetFeatureValue(inputFeature,
                out inputValue) && inputValue)
            {
                // if start pressing, trigger event
                if (!IsPressed)
                {
                    IsPressed = true;
                    OnPress.Invoke();
                    //Debug.Log("bottone premuto");
                }
            }
            // check for button release
            else if (IsPressed)
            {
                IsPressed = false;
                OnRelease.Invoke();
               // Debug.Log("bottone rilasciato");
            }
        }
    }
```

5.3.2.2 XR Interaction Toolkit

Per l'interazione del giocatore con gli oggetti di scena, è stato importato in Unity il *plugin XR Interaction Toolkit*, un *framework* che permette di utilizzare gli eventi relativi al sistema

[96] La mappatura completa di XR relativa ai dispositivi supportati è presente alla seguente pagina: https://docs.unity3d.com/Manual/xr_input.html

di *input* di Unity negli oggetti di scena (3D e UI), *renderizzati* in RV. Esso contiene un set di componenti che permettono all'utente di partecipare in maniera immersiva alle situazioni proposte nell'ambiente virtuale, in quanto gli consentono di "guardare il mondo" tramite un elemento *Camera* (appositamente configurato) e, ad esempio, "toccare", "raccogliere", "tirare" e "spingere" gli oggetti di scena. Nelle prossime righe saranno illustrate le principali componenti del pacchetto e ne verranno riportati alcuni esempi, relativi al loro impiego all'interno del simulatore.

5.3.2.2.1 XRRig

La componente *XRRig*, alla quale si è accennato durante la presentazione dei livelli del simulatore, è possibile trovarla associata ai *GameObject* dei *player* e ad ogni oggetto che espleta la funzione di muoversi in seguito alla lettura dell'*input* di uno o più dispositivi (i *controller touch* dell'Oculus Quest, nel caso in questione) o in ogni caso debba essere inclusa una telecamera (*Camera*) controllabile in prima persona dall'*HMD* e per la quale sia necessario stabilire un *offset*. Nel progetto del simulatore, *XRRig* è posta sull'oggetto genitore (nella gerarchia di Unity) dei giocatori in prima persona (come *XRRig_normo* e *XRRig_dis*) e consente di gestire alcuni parametri relativi alla telecamera e a impostare i riferimenti gli oggetti alla base del giocatore e della stessa *Camera*. Solitamente, alla classe *XRRig* sono associate altre componenti che consentono al giocatore di muoversi nell'ambiente virtuale e di interagire con gli oggetti ivi previsti, i quali saranno illustrati di seguito. È necessario analizzare le componenti in relazione alla struttura del giocatore (illustrata in precedenza),

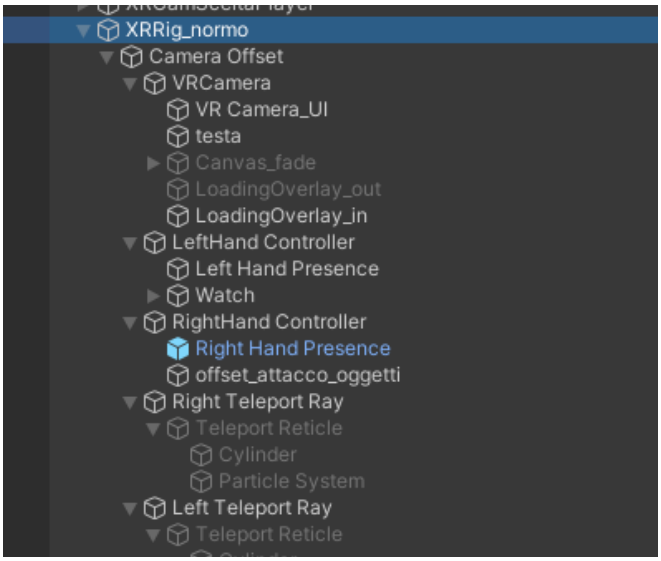

Figura 51. Gerarchia di XRig_normo

costruito in modo da essere utilizzato con gli *script* di cui sopra; il plugin *XR Interaction Toolkit* mette a disposizione nell'editor un *prefab* di *player* (chiamato come il relativo *script* associatovi, *XRRig*), editabile a piacimento, la cui gerarchia è riportata nell'immagine.

Come è possibile osservare dalla struttura illustrata, relativa ad un'istanza del *prefab* di *XRRig* opportunamente modificata per riprodurre un giocatore "normodotato", è presente come genitore un *GameObject XRRig*, al quale sono collegate le componenti per la gestione del punto di origine della telecamera e del movimento, all'interno del quale sono annidati gli oggetti relativi alla telecamera, ai *controller* e, ove previsto, ai raggi *raycast* per l'interazione e per il movimento. Indipendentemente dal livello di personalizzazione previsto per ogni *player*, il *GameObject* genitore deve avere associato il suddetto *script XRRig* per il funzionamento, il cui codice e la documentazione sono disponibili online, come da appendice[97]. Oltre a tale componente, è necessario dotare l'oggetto di un *provider* per il movimento se si vuole far muovere il giocatore nello spazio; è possibile scegliere tra due tipi di movimento, via *teleport* o lineare (di seguito trattati), a seconda dell'utilizzo, o sceglierli entrambi. Scendendo nella gerarchia dell'istanza del *prefab*, è possibile trovare un *GameObject* chiamato *VRCamera*, al quale è associata – oltre ad un *Audio Listener* e ad una *Camera*, con i *Clipping Planes* impostati rispettivamente a 0.1 (*Near*) e 1000 (*Far*) per un *feeling* di visione realistica – una componente *TrackedPoseDriver*, per il tracciamento spaziale dell'*HMD*. Quest'ultima, collegata al sistema di *input* di XR, permette la conversione dei segnali relativi a rotazione, altezza e movimento del casco di RV e li invia al motore grafico, che a sua volta li traspone nell'ambiente virtuale; *VRCamera* rappresenta, in altre parole, gli "occhi" del giocatore. Per permettere al giocatore di consultare i pannelli UI (ad esempio, per tenere traccia degli obiettivi delle missioni) senza interferire con gli oggetti previsti nell'ambiente virtuale, è stata inserita, annidata all'interno di *VRCamera*, una telecamera dedicata ai soli elementi di questo genere, *VRCamera_UI*. Per rendere ciò possibile, è stato assegnato il livello (*layer*) *UI_VR* ai soli *GameObject* da mostrare, come ad esempio i *canvas* con gli elementi dell'*UI Missioni_VR*, *Canvas_completamento_obiettivi*, *Canvas_completamento_missioni*, e nelle impostazioni della componente *Camera* allegata a *VRCamera_UI* è stato selezionato, alla voce *Culling Mask*, di *renderizzare* i soli elementi

[97] Il funzionamento di tale componente, come altre del pacchetto XR è molto complesso e non è essenziale al fine della comprensione dell'elaborato. È tuttavia disponibile una ricca documentazione a riguardo al link indicato in appendice.

relativi al livello *UI_VR*. All'interno dell'albero di *VRCamera* è presente anche un *GameObject* vuoto, testa, con annessa una sola componente *Sphere Collider*, in grado di captare le collisioni dell'area della testa del giocatore (e, dunque, all'*HMD*) con gli oggetti di scena. Inoltre, sono stati inseriti, sempre allo stesso livello gerarchico, due overlay

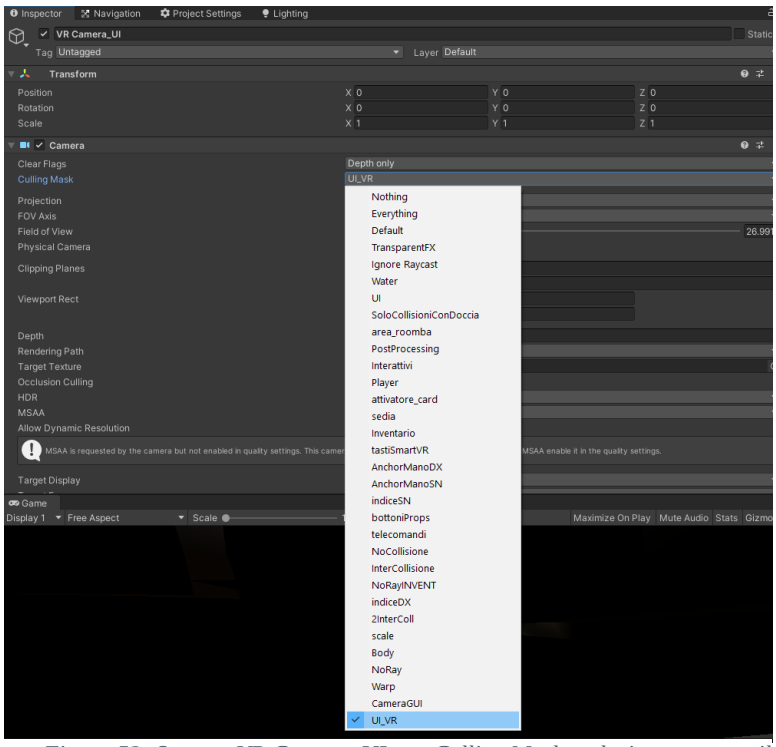

Figura 52. Oggetto VR Camera_UI con Culling Mask esclusivamente per il livello UI_VR

(*LoadingOverlayIn* e *LoadingOverlayOut*), che vengono attivati in tempo di esecuzione grazie alla commutazione di uno *script* ad essi connesso e permettono di "sfumare" tra le scene e di rendere più morbidi alcuni spostamenti del giocatore che risulterebbero, altrimenti, troppo bruschi (come l'operazione di chiamata dell'assistente per l'esperienza di gioco con il *player* "disabile"); di questi ultimi si tratterà nell'apposita sezione dedicata alla grafica del simulatore. Espandendo l'albero di *XRRig*, è possibile trovare gli oggetti relativi alla gestione dei *controller* (corrispondenti alle mani dei giocatori) e, per il solo player *XRRig_normo*, anche i raggi per il *raycasting* per il movimento via *teleport*; per rendere più vivida la presenza utente, a ciascun *controller* sono state collegati i *prefab* delle mani di Oculus e, al "polso" sinistro, è stato inserito un "orologio"[98]. Dopo aver illustrato la gerarchia che

[98] Tale oggetto è costituito da un'istanza di un *prefab* di *RealClock*, scaricabile dal *Package Manager* di Unity e opportunamente personalizzata. Il suo funzionamento non sarà trattato, in quanto non indispensabile ai fini dell'implementazione ma mera scelta grafica.

caratterizza un'istanza di XRRig, è necessario analizzare più da vicino alcune componenti del *plugin* XR relative al movimento e al lato interattivo, annesse al *parent* XRRig e ai *controllers* destro e sinistro.

5.3.2.2.2 Movimento con XR

In questo paragrafo verranno illustrate le metodologie applicate per impartire il movimento ai giocatori presenti nelle scene di Unity.

Teleportation

Per analizzare il concetto di movimento con *teleportation* (teletrasporto), è necessario considerare unicamente l'istanza *XRRig_normo*, poiché essa rappresenta l'unico *player* al quale è stata attribuita tale modalità di spostamento. Per permettere al giocatore di muoversi tramite teletrasporto, è necessario dotare il *GameObject XRRig* di due componenti: *TeleportationProvider* e *LocomotionController*.

TeleportationProvider è una componente del pacchetto *XRInteractionToolkit* che permette di associare ciascun oggetto di *XRRig* al gestore *XR Interaction Manager* (inserito automaticamente in ogni scena al primo elemento di *XRInteractionToolkit* importata) e di leggere e reimpostare i settaggi della telecamera e dell'altezza del giocatore ad ogni suo "balzo". Dopo l'importazione in Unity, è necessario associarvi via *inspector* l'*XRRig* di riferimento, al quale si vuole impartire il movimento; lo *script* contiene una funzione Update che assolve all'aggiornamento della posizione e all'orientamento del *player* ad ogni frame e sfrutta lo stato dei metodi booleani *BeginLocomotion* e *EndLocomotion* definiti nella classe principale del pacchetto per determinare l'inizio e la fine di ogni spostamento del giocatore. Di seguito è riportato uno *snippet* con la funzione *Update* contenuta in *TeleportationProvider*, con le variabili relative al *player* e allo stato del movimento in base al *target* (portale o area) da raggiungere:

```
protected virtual void Update()
  {
      if (!validRequest || !BeginLocomotion())
          return;

      var xrRig = system.xrRig;
      if (xrRig != null)
      {
          switch (currentRequest.matchOrientation)
          {
              case MatchOrientation.WorldSpaceUp:
                  xrRig.MatchRigUp(Vector3.up);
```

```
                        break;
                    case MatchOrientation.TargetUp:
                        xrRig.MatchRigUp(currentRequest.destinationRotation * Vector3.up);
                        break;
                    case MatchOrientation.TargetUpAndForward:
                        xrRig.MatchRigUpCameraForward(currentRequest.destinationRotation * V
ector3.up, currentRequest.destinationRotation * Vector3.forward);
                        break;
                    case MatchOrientation.None:
                        // Change nothing. Maintain current rig rotation.
                        break;
                    default:
                        Assert.IsTrue(false, $"Unhandled {nameof(MatchOrientation)}={current
Request.matchOrientation}.");
                        break;
                }

                var heightAdjustment = xrRig.rig.transform.up * xrRig.cameraInRigSpaceHeight
;

                var cameraDestination = currentRequest.destinationPosition + heightAdjustmen
t;

                xrRig.MoveCameraToWorldLocation(cameraDestination);
            }

            EndLocomotion();
            validRequest = false;
        }
```

L'altra componente alla quale si è accennato è *LocomotionController*, che consente di gestire l'interazione con aree e portali per il teletrasporto (in seguito trattati) e permette di associare al *provider* di XR i/il *GameObject* per effettuare il *raycasting* (*RightTeleportRay* e *LeftTeleportRay*, o uno dei due). Permette, inoltre, di selezionare il bottone per poter azionare i raggi (come il tasto trigger del *controller touch* testro di Oculus) e di stabilirne una soglia di azione. Lo *script* (riportato in appendice, poiché sfrutta *XRInteractionToolkit* ma non ne fa parte), che estende il funzionamento di XR ma deriva dalla classe *MonoBehaviour*, permette alle componenti di tipo *XRController leftTeleportRay* e *rightTeleportRay* (associabili ai relativi *GameObject* via *inspector*) di essere attivate alternativamente, a seconda della pressione del bottone scelto tramite l'*helper* che gestisce il sistema di input. Nel seguente *snippet*, è riportata come esempio parte della funzione Update che permette di commutare il raggio tramite il *controller* sinistro:

```
if(leftTeleportRay){
    leftTeleportRay.gameObject.SetActive(CheckIfActivated(leftTeleportRay));
}
   [omissis]
 }
```

e di controllare lo stato della pressione del bottone e impostarne la soglia di sensibilità all'attivazione (espressa in un valore decimale, contenuto in una variabile di tipo *float*)

```
InputHelpers.IsPressed(controller.inputDevice, teleportActivationButton, out bool isActivate
d, activationThreshold);
```

Il movimento via teletrasporto è attuabile tramite le componenti *XRRayInteractor* (delle quali si discuterà più avanti) associate ai *controllers LeftHandController* e *RightHandController*, che permettono al giocatore di proiettare nello spazio dei raggi per raggiungere aree ove è possibile teletrasportarsi o, semplicemente, interagire. Tali zone si dividono in portali e aree, a seconda dell'esigenza dettata dal gioco; nel simulatore, è possibile trovare impiegate le componenti *TeleportationArea* sui pavimenti della casa (nella scena LVL1_LVL2) e *TeleportationAnchor* sui portali relativi ad alcuni obiettivi delle missioni ove è necessario posizionarsi in un determinato punto (indicato da un cilindro luminoso) via teletrasporto. Nelle seguenti immagini è possibile osservare l'interazione via raggio in un'area abilitata al teletrasporto e ad un portale:

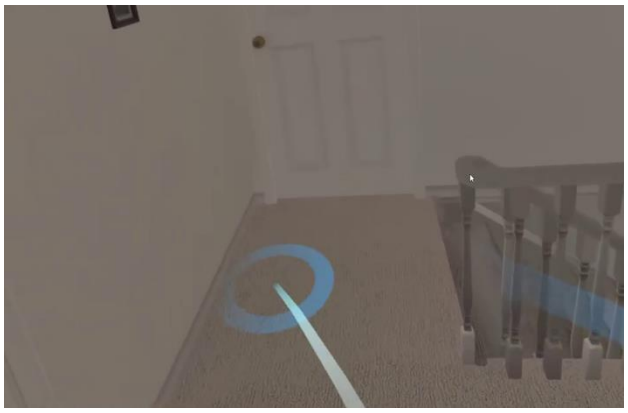

Figura 53. Raycasting di Right Teleport Ray su un'area per il teletrasporto

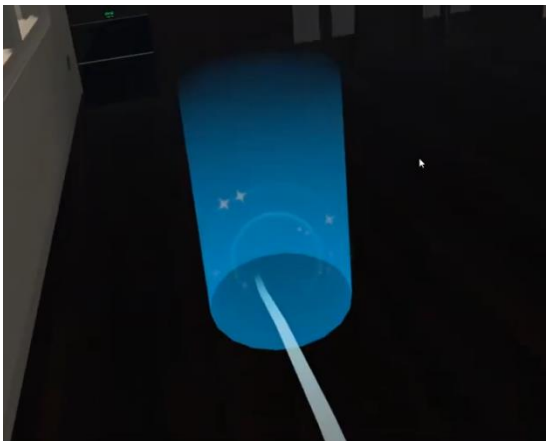

Figura 54. Raycasting di Right Teleport Ray su un portale

TeleportationArea è un piccolo *script* di XR che, se associato al *GameObject* di un qualsiasi solido o piano dotati di *collider*, permette al giocatore di interagirvi e di raggiungere il punto colpito dal *raycast* di uno dei raggi. All'interno della componente è possibile trovare un metodo che permette di definire, tramite le variabili di tipo *Transform* di XR *destinationPosition* e *destinationRotation*, la posizione e la rotazione di arrivo del giocatore, in base alle coordinate x,y e z del punto raggiunto dal raggio:

```
teleportRequest.destinationPosition = raycastHit.point;
teleportRequest.destinationRotation = transform.rotation;
```

Nel secondo caso, invece, la componente impiegata per l'interazione con i raggi è *TeleportationAnchor*, uno *script* di XR che si avvale di due metodi: *OnValidate* e *OnDrawGizmos*. Il primo, come si può osservare dallo *snippet* sotto riportato, serve a verificare l'esistenza delle coordinate del portale al quale lo *script* è associato e, in caso

contrario, a crearne una nuova istanza; il secondo, invece, serve a creare un reticolo grafico (*gizmo*) in corrispondenza della selezione.

```
protected void OnValidate()
        {
            if (m_TeleportAnchorTransform == null)
                m_TeleportAnchorTransform = transform;
        }
[omissis]
protected void OnDrawGizmos()
        {
            Gizmos.color = Color.blue;
            GizmoHelpers.DrawWireCubeOriented(m_TeleportAnchorTransform.position, m_Teleport
AnchorTransform.rotation, 1f);

            GizmoHelpers.DrawAxisArrows(m_TeleportAnchorTransform, 1f);
        }
```

Quando un'area o un portale vengono colpiti da un raggio o da un qualsiasi altro oggetto reso interattivo da uno *script* di XR, vengono eseguite le funzioni evento alle quali, tramite *inspector*, sono associate le componenti dei *GameObject* da commutare, in base alla situazione. È possibile, infatti, associare ad ogni componente *TeleportationArea* o *TeleportationAnchor* uno o più oggetti, le cui caratteristiche possono essere modificate in base allo stato degli eventi di tipo *interactable* forniti; tra questi è possibile citare *OnSelectedEntered* (alla selezione tramite tasto funzione del *controller* dell'oggetto

Figura 55. Funzione evento OnSelectEntered su un portale

corrispondente ad un portale o ad un'area) e *OnActivate* (corrispondente al momento nel quale il raggio entra in contatto con l'area o il portale).

Nell'immagine riportata, si può osservare il funzionamento di un evento collegato ad una componente *TeleportAnchor*, associata al *GameObject* del primo portale della Missione I, nella scena relativa alle prime due modalità di gioco. Nella prima sezione, è indicato di disattivare il *GameObject* corrispondente al portale luminoso alla selezione dello stesso; nella seconda, lo stesso comportamento è applicato all'oggetto contenente il *Particle System* che identifica un obiettivo da raggiungere. All'interno della terza, invece, è indicato di attivare lo *script MissioneCompiutaSimple*, che permette di passare all'obiettivo successivo (e non

alla missione, contrariamente a come possa indicare il nome), collegato allo stesso *GameObject Anchor*.

Movimento lineare con thumbstick

Sull'*XRRig_normo* sono applicate, oltre alle componenti di XR per il teletrasporto, anche quelle relative al movimento e alla rotazione lineari. Per permettere al giocatore di "spostarsi" con più precisione all'interno dell'ambiente virtuale, è stato infatti ritenuto necessario integrare il moto via teletrasporto con quello lineare, basato sull'input analogico del *thumbstick*; per assolvere alla funzione è stato scelto di utilizzare una componente che estende la classe di XR *LocomotionProvider*, *MovementProvider*, tramite la quale è possibile definire la velocità di movimento del *player* e la forza di gravità da applicarvi. Lo *script*, parte del pacchetto *XR Interaction Toolkit*, richiede la componente *CharacterController* di Unity per funzionare e si avvale di due funzioni principali: *StartMove* e *ApplyGravity*, rispettivamente dedicate allo spostamento del giocatore tramite il *controller* associabile nell'*inspector* (in questo caso, il *thumbstick* del *controller touch* Oculus destro) e alle forze da applicare alla massa dello stesso. *StartMove* utilizza tre strutture vettoriali *Vector3* per l'avvio del movimento nelle tre direzioni possibili (x, y e z): *direction* ed *headRotation*, alle quali sono associate la posizione iniziale del giocatore (definita usando il punto frontale dell'*HMD* nello spazio) e la relativa rotazione e *movement*, ottenuta dal prodotto delle variabili *direction* e *speed* precedentemente dichiarate[99].

```
private void StartMove(Vector2 position)
    {
        // Apply the touch position to the head's forward Vector
        Vector3 direction=new Vector3(position.x,0,position.y);
        Vector3 headRotation=new Vector3(0,head.transform.eulerAngles.y,0);
        // Rotate the input direction by the horizontal head rotation
        direction=Quaternion.Euler(headRotation)*direction;
        // Apply speed and move
        Vector3 movement=direction*speed;
        characterController.Move(movement*Time.deltaTime);

    }
```

Il metodo *ApplyGravity* permette, invece – tramite l'impiego di una *Vector3* –, di sfruttare il prodotto tra forza di gravità (il cui valore è impostato a 3.6, contro i 9.81 del parametro reale, per un maggior controllo della forza impartita sul giocatore in base alla sua massa) e

[99] Per dettagli ulteriori sul funzionamento, è necessario consultare il link in appendice relativo alla documentazione di XR.

l'intervallo tra i *frames* identificato dalla variabile di tipo *float time.deltaTime* per determinare il valore dell'accelerazione da impiegare.

```
   private void ApplyGravity()
    {
        Vector3 gravity = new Vector3(0,Physics.gravity.y*gravityMultiplier,0);
        gravity.y*=Time.deltaTime;
        characterController.Move(gravity*Time.deltaTime);

    }
```

Per la rotazione nello spazio, è invece impiegata la componente di *XR Interaction Toolkit ContinuosTurnProvider* di tipo *Device Based*, che sfrutta l'*input* sull'asse x (definito in un vettore *Vector2*) del *thumbstick* del *controller* selezionato da *inspector* (nel caso in questione, il destro). Nella seguente porzione di codice è riportata la funzione *ReadInput*, responsabile della lettura dell'input analogico del *thumbstick* del *controller* e la relativa conversione in valori di *output*, *renderizzati* in tempo reale:

```
   protected override Vector2 ReadInput()
        {
            [omissis]
            var input = Vector2.zero;
            var feature = k_Vec2UsageList[(int)m_InputBinding];
            for (var i = 0; i < m_Controllers.Count; ++i)
            {
                var controller = m_Controllers[i] as XRController;
                if (controller != null &&
                    controller.enableInputActions &&
                    controller.inputDevice.TryGetFeatureValue(feature, out var controllerInp
ut))
                {
                    input += GetDeadzoneAdjustedValue(controllerInput);
                }
            }

            return input;
        }
```

5.3.2.2.3 Interazione

All'interno del simulatore, ciascun *XRRig* può interagire tramite i *controllers*, con gli oggetti di scena non statici ai quali sono associati una o più componenti di tipo *collider* (come *BoxCollider*, *SpereCollider* o *MeshCollider*, di seguito trattati) e un *Rigidbody*. Ogni *GameObject* che presenta tali caratteristiche, è infatti controllabile tramite la fisica *real-time* interna al gioco. Nel simulatore sono state, tuttavia, impiegate metodologie che consentono un controllo più preciso di tali oggetti e ne permettono azioni più complesse come la presa (*grabbing*); il funzionamento di queste è a carico di alcune componenti di XR, connesse sia alle "mani" del giocatore, sia ai *GameObject* da rendere interattivi. Nelle seguenti sottosezioni verranno illustrate in breve le funzionalità sfruttate nel gioco per l'interazione

con gli oggetti di scena; tuttavia, per ulteriori dettagli sulle stesse, è necessario consultare il link in appendice relativo alla documentazione di XR.

Componenti Interactor per XRRig

Di seguito saranno descritte alcune delle componenti di XR da applicare sulle "mani" del giocatore (o, nel simulatore in esame, all'inventario), indispensabili all'interazione con i relativi oggetti di scena, trattati in un secondo momento.

XR Direct Interactor

XRDirectInteractor, come già accennato in riferimento alla scena di avvio (ma utilizzato in ognuna), è una componente da applicare ai *controller* dell'*XRRig* che rende possibile l'interazione diretta con le "mani" con tutto ciò sia reso dinamico nell'ambiente virtuale. Lo *script*, che contiene un *namespace* per la definizione di alcune caratteristiche che estendono *XRInteractionToolkit*, basa il suo meccanismo sulle funzioni evento *OnTriggerEnter* e *OnTriggerExit*, utilizzate per il riconoscimento degli oggetti dotati di *collider* con il *flag trigger* attivato e la conseguente aggiunta degli stessi ad una lista vuota entro la quale il programma colleziona i *GameObject* sui quali è possibile interagire (con *collider* e *Rigidbody*).

```
protected void OnTriggerEnter(Collider col)
     {
         var interactable = interactionManager.TryGetInteractableForCollider(col);
         if (interactable != null && !m_ValidTargets.Contains(interactable))
             m_ValidTargets.Add(interactable);
     }

     protected void OnTriggerExit(Collider col)
     {
         var interactable = interactionManager.TryGetInteractableForCollider(col);
         if (interactable != null)
             m_ValidTargets.Remove(interactable);
     }
```

Inoltre, *XRDirectInteractor* include, come altre componenti simili del pacchetto XR, dei parametri configurabili dall'*inspector* relativi al *feedback aptico* dei *controllers* di Oculus, attivabile su scelta al momento dell'attivazione dei *GameObject* interattivi, e una lista di funzioni evento entro le quali è possibile stabilire quali componenti di gioco commutare o far cambiare di stato in base ad alcune azioni dell'utente ivi definite.

XR Ray Interactor

La componente *XRRayInteractor* del pacchetto *XR Interaction Toolkit* è associata ai *TeleportRay* (destro e sinistro) previsti nei player *VRRig_start* e *XRRig_normo* e assolvono ad una duplice funzione: spostarsi in aree e portali dedicati al *teleport* e attirare a sé oggetti, anche lontani nell'ambiente virtuale. Il funzionamento dello *script*, piuttosto complesso e per il quale è necessario consultare l'estesa documentazione, si avvale della tecnica del *raycasting* dei raggi (*renderizzati* tramite la componente *LineRenderer*) per "colpire" le collisioni degli oggetti di scena ai quali sono collegate le componenti riportate nel sottoparagrafo "Elementi Interattivi". Il tipo di interazione è molto simile a *XRDirectInteractor*, ma vi differisce in quanto non si avvale di *OnTriggerEnter* e *OnTriggerExit*, ma impiega i punti di contatto di tipo *raycast* o *sphere cast* degli oggetti presenti nella RV per l'attivazione delle funzioni evento ad esso relative.

XR Socket Interactor

XRSocketInteractor è una componente che sfrutta *OnTriggerEnter* e *OnTriggerExit* in modo molto simile a *XRDirectInteractor* ed è impiegata nel funzionamento dell'inventario che segue il giocatore in ogni luogo. La caratteristica che lo differenzia dagli altri *Interactors* è quella di potervi agganciare elementi finché il giocatore desidera rimuoverli: il suo impiego nel simulatore vuole simulare una "tasca" entro la quale il *player* può "riporre" alcuni oggetti di scena, opportunamente contrassegnati dal tag *Pickup* (come la carota dell'obiettivo 2 della Missione II). Come è possibile osservare nello *snippet* riportato di seguito, il funzionamento si basa sull'aggiunta e sulla rimozione di determinati elementi da una lista, su richiesta dell'utente, in base a quanto definito nell'*inspector*, all'interno della sezione dedicata agli eventi di *XRBaseInteractable*.

```
    protected void OnTriggerEnter(Collider col)
        {
            var interactable = interactionManager.TryGetInteractableForCollider(col);
            if (interactable && !m_ValidTargets.Contains(interactable) && selectTarget != in
teractable)
                m_ValidTargets.Add(interactable);
        }

        protected void OnTriggerExit(Collider col)
        {
            var interactable = interactionManager.TryGetInteractableForCollider(col);
            if (interactable && m_ValidTargets.Contains(interactable) && selectTarget != int
eractable)
                m_ValidTargets.Remove(interactable);
```

```
}
```

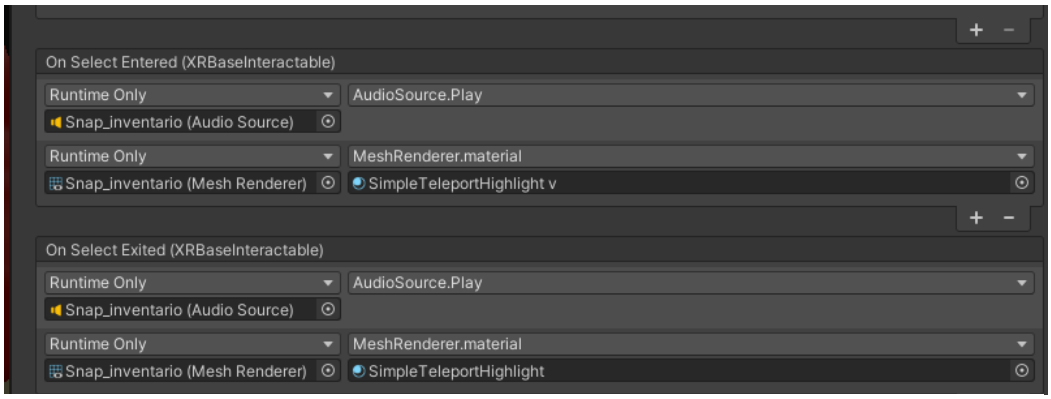

Figura 56. Eventi OnSelectEntered e OnSelectExited associati alla componente XRSocketInteractor relativa al GameObject dell'inventario

Elementi Interattivi

Dopo aver trattato gli elementi necessari all'interazione con gli oggetti di scena, bisogna descrivere quelli che permettono agli oggetti di essere riconosciuti dagli *Interactors* come *GameObjects* complementari, sui quali poter operare.

XR Simple Interactable

XRSimpleInteractable trova il suo impiego sui bottoni del menu di avvio e si avvale di una struttura molto semplice, basata sulle funzioni evento *OnTriggerEnter* e *OnTriggerExit*. Lo *script* non ha contenuto a sé stante, ma estende la classe *XRBaseInteractable*, dalla quale vi eredita la struttura e ne propone la sola funzionalità di aggiunta e rimozione dinamica dei *colliders* nella lista relativa alle collisioni presenti sull'oggetto che, dinamicamente, riempie e svuota.

XR Grab Interactable

La componente *XRGrabInteractable*, ampiamente usata sugli oggetti dinamici previsti nel simulatore, permette a questi ultimi di essere "impugnati" dal *player* per, ad esempio, essere

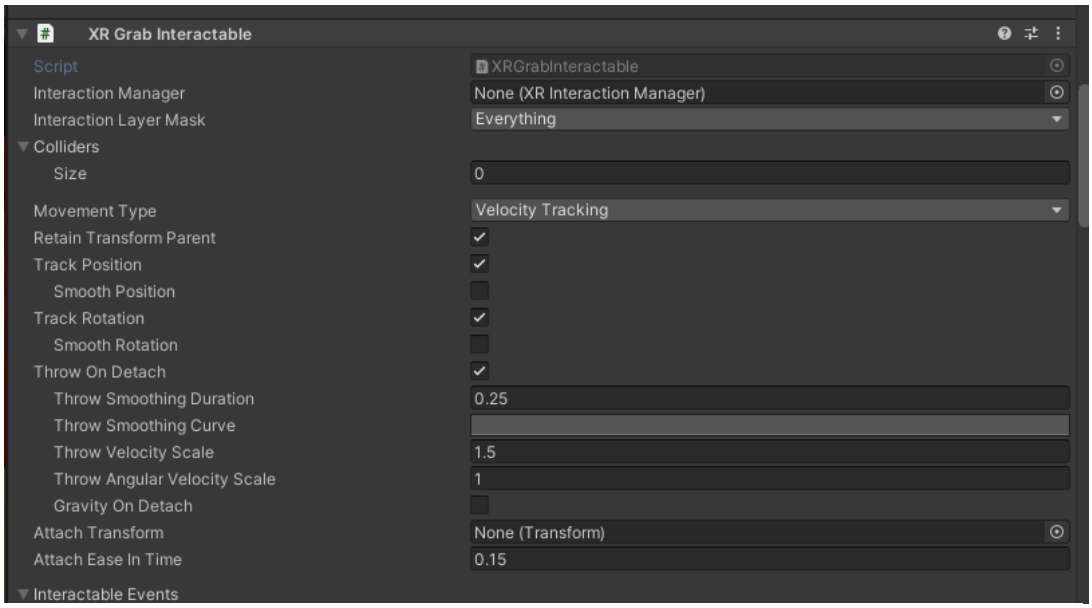

Figura 57. XRGrabInteractable

"raccolti", "tirati" o "spinti". Lo *script* contiene una definizione di *namespace* sulla base di *XRInteractionToolkit* e si avvale di diverse sezioni (come da immagine tratta *dall'inspector*), tra le quali spicca quella relativa alla definizione del tipo di movimento da impartire all'oggetto "impugnato" e prevede una serie di parametri utili a determinare la precisione di azione sullo stesso.

Nella quasi totalità dei casi, il *Movement Type* è impostato su *Velocity Tracking*, in quanto è necessario l'influsso delle forze fisiche (in primis, l'accelerazione) impartite dall'utente sugli oggetti presenti nell'ambiente virtuale. Il comportamento che tali devono assumere in relazione alle forze esterne sono definiti nel metodo *PerformVelocityTracking* dello *script* (presente al link in appendice), che permette di tracciare la velocità e la rotazione degli stessi nello spazio e di seguire, con quanta più precisione possibile, le "mani" del giocatore in relazione alla fisica applicata alle stesse. La componente, inoltre, basa il suo meccanismo su funzioni evento, come *Attach* e *Detach*, che permettono di definire il momento in cui è necessario, rispettivamente, collegare l'oggetto alla mano (corrispondente, di *default*, alla pressione del tasto *trigger* dei *controller touch*) e staccarlo (al rilascio del pulsante), con relativi parametri sulle forze da impartire ai *GameObjects* nelle due fasi.

Dopo aver illustrato, almeno sommariamente, il lato interazione basato su *XRInteractionToolkit*, è necessario addentrarsi nella descrizione di altri componenti e *script* di Unity esterni ai pacchetti di XR, ma altrettanto utili al fine della realizzazione del simulatore.

5.3.3 Controlli, eventi e coroutines: i motori del simulatore

In questa sezione verranno affrontate le principali strategie adottate in merito al contollo degli oggetti di gioco all'interno del simulatore, effettuato sfruttando gli strumenti messi a disposizione da Unity nell'inspector e in altrettanti *script*, ereditati dalla classe *MonoBehaviour*.

5.3.3.1 Tags e Layers

È necessario innanzitutto premettere che, per una migliore gestione dei *GameObject*, è stato impiegato il sistema di *tag* e livelli (*layers*) messi a disposizione dal *framework* grafico. In particolare, questi ultimi sono stati sfruttati per evitare ad alcuni *script*, in particolare quelli relativi alle interazione tra l'utente e gli oggetti, di interferire sul lato fisico tra loro o con lo stesso giocatore. Di seguito sono riportate la matrice dei livelli e parte della lista dei *tag* utilizzati nel gioco:

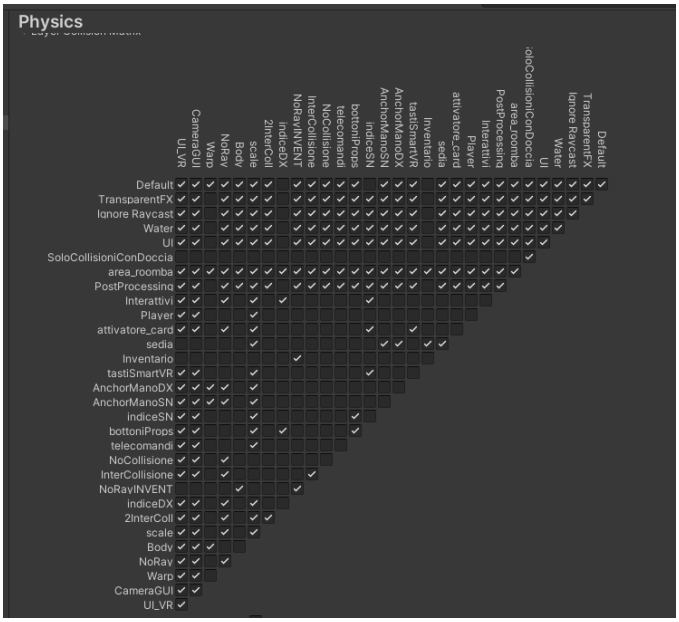

Figura 58. Matrice dei livelli in Unity

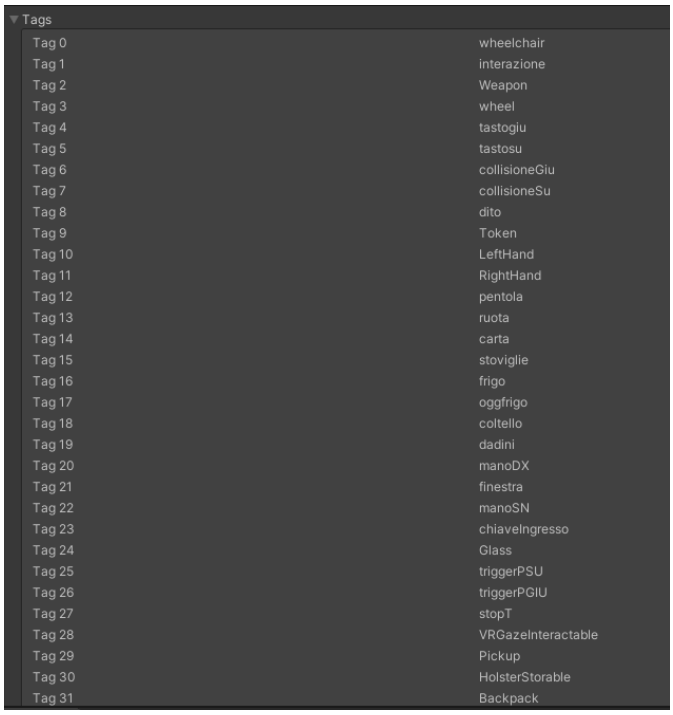

Figura 59. Parte dei 44 tag impiegati in Unity per gli oggetti del simulatore

ed è citato un esempio di *GameObject* al quale, per prevenire l'interazione tra collisioni, sono stati attribuiti un tag e un livello:

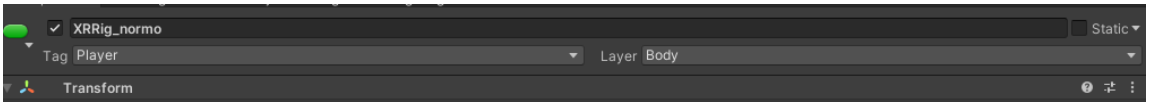

Figura 60. XRRig con tag "Player" e layer "Body" assiciato

Nel caso riportato nell'immagine, relativo ad uno dei *player* previsti nel simulatore, è stato reso necessario attribuirvi il *tag Player* per permettere ad alcuni *script* di interagirvi in maniera esclusiva, come nello *snippet* di seguito, estratto dallo *script questoToccaQuelloConf* connesso al *GameObject* sapone della Missione III, dove la componente *WaitTotSecConf* viene abilitata solo se vi interagisce un oggetto dotato di *collider taggato* come *Player* o *Player_dis*.

```
void OnCollisionEnter(Collision other){
        if(giaToccato==false){
            if(other.gameObject.tag=="Player"||other.gameObject.tag=="Player_dis"){
                GetComponent<WaitTotSecConf>().enabled=true;
                giaToccato=true;
            }
        }
    }
```

Per quanto riguarda, invece, il *layer*, vi è stato associato quello identificato dalla stringa *Body*, alla quale corrisponde (vedere matrice) un livello in grado di non interagire fisicamente con gran parte di quelli dedicati ad elementi interattivi, quali pulsanti e inventario. Tale logica è

impiegata ovunque nell'implementazione del simulatore, poiché permette di raggruppare in specifiche "categorie" i *GameObject* e di creare un discrimine tra gli stessi, in quanto a collisioni e all'azione di forze fisiche sugli stessi.

5.3.3.2 Controllo dello stato di oggetti e componenti

In molti *script* derivati dalla classe *MonoBehaviour*, sono stati impiegati dei valori booleani in grado di determinare lo stato di uno o più *GameObject* o delle relative componenti; tra questi, i più utilizzati sono stati *ActiveSelf* e *enabled*. Il primo, utilizzato per controllare se l'oggetto al quale è collegato lo *script* è attivo, trova il suo impiego in situazioni nelle quali è necessario verificare lo stato di un *GameObject*, affinché un evento possa prendere luogo. Nella seguente porzione di codice, prelevata dalla funzione *OnTriggerEnter* dello *script commutaRoomba* (responsabile dell'attivazione dell'aspirapolvere robot, presente nel livello 3), è possibile trovarlo impiegato nella verifica dello stato di un oggetto che tiene traccia dell'"accensione" e dello "spegnimento" dell'apparecchio:

```
            if(controlloStato.activeSelf==false){
                roomba.transform.Rotate(0, 180, 0, Space.Self);
                roomba.GetComponent<wanderingAi>().enabled=true;
                agent.enabled=true;
                interruttore.GetComponent<MeshRenderer>().material = materiale_int_on;
                led.GetComponent<MeshRenderer>().material = materiale_int_on;
                StartCoroutine(SuoniRoomba());
                controlloStato.SetActive(true);
                pulsante_smart.GetComponent<SpriteRenderer>().sprite = spriteOn;
                foreach (GameObject a in animazioneSpazzole){
                    a.GetComponent<Animator>().enabled=true;
              }
             [omissis]
```

Sempre all'interno dello stesso *script*, è possibile osservare la variabile di stato *enabled*: in tale caso, permette di abilitare su tutti i *GameObject* contenuti nella lista *animazioneSpazzole* l'elemento *Animator* (che controlla il movimento delle spazzole dell'aspirapolvere robot).

5.3.3.3 Eventi per le collisioni

Per la realizzazione della parte interattiva relativa, ad esempio, agli interruttori presenti nelle scene, sono state utilizzate le funzioni evento *OnTriggerEnter* e *OnCollisionEnter*, in grado di agire sugli oggetti interessati al contatto con una collisione esterna. Entrambi i metodi presentano un funzionamento simile: il primo tipo entra in gioco quando, se associato ad un *GameObject*, questo viene colpito da un altro con un *collider settato* come *trigger*; il secondo, invece, è interessato quando è coinvolta una collisione generica[100]. *OnCollisionEnter* è

[100] In entrambi i casi, almeno uno degli oggetti deve avere associato un *Rigidbody*.

utilizzata, ad esempio, nello *script onContactActivateSimple* per attivare la componente *missioneCompiutaSimple* connessa ai portali riguardanti alcuni obiettivi delle missioni. Lo *script*, collegato agli oggetti *trigger_dis_act*, permette di eliminare alcuni *GameObject* al tocco da parte di altri, contrassegnati da un *tag* passato al programma via *editor* e contenuto nella variabile stringa *GOTagAttivare*:

```
private void OnCollisionEnter(Collision other)
    {
        if(attivato==false){
            if (other.gameObject.tag==GOTagAttivare){
                portale.GetComponent<missioneCompiutaSimple>().enabled=true;
                foreach (GameObject o in oggettiDaDisattivare){
                    o.SetActive(false);
                }
                attivato=true;
                this.gameObject.SetActive(false);
            }
        }
    }
```

OnTriggerEnter è impiegata, invece, tra i tanti casi, in *PiuLuci_VR*, utilizzato nei livelli 1 e 2 per commutare lo stato degli interruttori e delle relative luci:

```
void OnTriggerEnter(Collider other){
        if (other.tag=="dito"||other.tag=="ditoDX"){
            if(controlloLuci.activeSelf==true){
                //spegni
                Light_Off();
                DisableEmission();
            }
            else{
                //accendi
                Light_On();
                EnableEmission();
            }
        }
    }
```

e permette, in base al caso, di invocare i metodi relativi all'accensione delle luci e all'attivazione del *flag Emission* nei relativi materiali.

In alcune parti del simulatore, come quella dedicata alla realizzazione del sistema di tapparelle, è stato necessario integrare le funzioni sopra descritte con altrettante che registrassero gli eventi relativi al contatto prolungato di due elementi dinamici (con *collider* e *Rigidbody* in almeno uno dei casi); ad assolvere a ciò, sono state introdotte *OnTriggerStay* e *OnCollisionStay*. Tali funzioni-evento presentano un meccanismo analogo a *OnTriggerEnter* e a *OnCollisionEnter* ma, a differenza di queste, permettono di eseguire ciclicamente e ad ogni frame i comandi. L'impiego di queste è stato necessario nello sviluppo dei due tipi di tapparelle previste nelle scene del simulatore: le prime, di tipo meccanico (contenute nella scena 1) e le seconde, commutabili tramite apposito pulsante. Nel codice

riportato è possibile osservare il funzionamento di *OnTriggerStay* nello *script Tapp_meccanicaG_VR*, applicato al *GameObject* "su" (un cubo senza *MeshRenderer*, con la sola componente *BoxCollider*), usato per ogni sistema di tapparelle a corda previste nel simulatore:

```
void OnTriggerStay(Collider other){
    if(other.tag=="interazione"){
        UpScale(parent.transform.localScale);
    }
}
```

e impiegato nello *script tapp_VR_su*, dal funzionamento analogo, utilizzato nel terzo livello e progettato per permettere al giocatore di "sollevare" i serramenti tramite il tocco sul rispettivo bottone:

```
void OnTriggerStay(Collider other){
     if(other.tag=="dito"|| other.tag=="ditoDX"){
     UpScale(parent.transform.localScale);
         tastoSmartSu.GetComponent<SpriteRenderer>().sprite = spriteOn;
     }
 }
```

In entrambi i casi, viene invocata la funzione *UpScale* (presente in appendice) responsabile, nel primo caso, dello *scaling* dei *GameObjects* delle tapparelle, finché la "corda" (che include una componente *ConfigurableJoint* e un *BoxCollider flaggato* come *trigger* e può essere "mossa", entro un limite definito, sull'asse Y) tocca il punto di contatto "su" (taggato con "interazione) e, nel secondo, fino a quanto il "dito" del giocatore non viene rimosso dal pulsante[101].

5.3.3.4 Coroutines

Nella quasi totalità degli obiettivi delle missioni previste nel progetto è stato necessario eseguire automaticamente dei comandi a catena (come la commutazione dello stato di componenti e *GameObjects*), rispettando un intervallo di tempo e una certa asincronia tra gli stessi. A tal proposito, sono state utilizzate le *coroutines*, costrutti in grado di mettere in pausa l'esecuzione e di riprenderla successivamente, che utilizzano l'interfaccia *IEnumerator* per la loro definizione e sono invocabili in qualsiasi punto del programma. È possibile trovare, ad esempio, le *coroutines* impiegate in tutti gli *script* del tipo *WaitTotSec**, responsabili dell'attivazione *on runtime* dello *script missioneCompiuta* (o di una sua variante semplificata, *missioneCompiutaSimple*):

```
void Start(){
```

[101] Gli script per il movimento nella direzione opposta hanno caratteristiche complementari rispetto a quelli illustrati e presentano un funzionamento analogo.

```
        //Start the coroutine we define below named ExampleCoroutine.
        StartCoroutine(TriggeraFineObiettivo());
    }
    IEnumerator TriggeraFineObiettivo()
    {
         yield return new WaitForSeconds(AttesaSec);
         GetComponent<missioneCompiutaSimple>().enabled=true;
         foreach(GameObject o in oggettiDaDisattivare){
             o.SetActive(false);
         }
    }
```

Nel caso in esame la *coroutine TriggeraFineObiettivo*, chiamata all'avvio dell'esecuzione di *WaitTotSecConf* (versione dello *script* con parametri configurabili dall'*inspector*, come la variabile di tipo *float AttesaSec*), prevede un'attesa di un certo numero di secondi e, solo successivamente, disabilita tutti i *GameObjects* contenuti nella lista *oggettiDaDisattivare* (relativi, ad esempio, ad oggetti di scena non più necessari al termine di un obiettivo).

5.3.4 Missioni: implementazione generale

Il meccanismo interno ai livelli del gioco è caratterizzato da quattro missioni, come già illustrato: in questa sezione ne verrà trattata, in linee generali, la struttura e saranno analizzati alcuni *script*, in particolare quelli impiegati nel passaggio tra gli obiettivi ed alcuni più complessi, dei quali è necessario fornire una descrizione.

5.3.4.1 Funzionamento di base: l'esempio della Missione II

Per la descrizione del sistema, è stato scelto di prendere in esame la Missione II – Prepara la cena, poiché costituisce l'esempio comprensivo di più obiettivi di tutto il simulatore ed include, di conseguenza, la maggior parte degli *script* comuni a tutte le missioni. Gli obiettivi previsti in quest'ultima sono sette e sono ambientati nell'ambiente virtuale relativo alla cucina della casa. Il primo, che richiede di spostarsi in tale stanza, include un portale sul quale interagire e un oggetto dotato di un *BoxCollider*, in grado di captare le collisioni del giocatore su sedia a rotelle (per il quale, in entrambe le scene, non è previsto il *raycasting*). Nei livelli che prevedono il *player* normodotato, l'oggetto che fa da *trigger* all'attivazione degli eventi correlati all'obiettivo è il portale con la relativa componente *TeleportationAnchor* che, se colpito da un raggio, lancia l'evento *OnSelectEntered*, che abilita lo *script waitTotSec* (dal funzionamento analogo a *waitTotSecConf*, trattato in precedenza). L'attivazione di tale componente, ove è previsto il giocatore sulla carrozzina, è a cura del *GameObject trigger_dis_act*, che funziona tramite lo *script onContactActivateWaitTotSec* (codice in appendice), che permette di disabilitare alcuni elementi non più necessari (quali il portale e l'effetto grafico relativo ad un obiettivo da

portare a termine) e di impostare su *enabled waitTotSecConf*, collegato al portale. In altre parole, lo *script onContactActivateWaitTotSec* compie via collisioni ciò che, nel caso del player normodotato, è possibile fare con il *raycasting*. All'attivazione di *WaitTotSec*, nel tempo di un secondo, viene attivata la componente *apriFrigo*, che avvia un ciclo di animazione sulla porta del frigorifero e ne consente l'apertura. Lo *script*, di minore importanza, si compone di una sola riga, contenuta nella funzione *Start*:

```
porta_frigo.GetComponent<Animation>().Play("porta_frigo");
```

Al contempo, come da specifiche nella *coroutine* inclusa in *WaitTotSec*, viene attivata la componente *MissioneCompiutaSimple*, responsabile – a differenza di quanto reciti il nome – della chiusura di ogni obiettivo[102], sulla quale è necessario fare luce.

Lo *script*, sopra illustrato *nell'inspector* di Unity, include una sezione relativa alla commutazione di elementi puramente grafici e sonori (il *jingle* di completamento) e all'attivazione del gruppo di elementi relativo alla successiva missione, e una più interessante, che gestisce cambio di colore della stringa di testo relativa all'obiettivo, visualizzabile nel pannello missioni. Per tenere traccia degli obiettivi, infatti, è stato scelto di colorare di giallo l'obiettivo corrente, in verde quelli già superati e in grigio (di *default*) quelli che richiedono ancora qualche *step* prima di poter essere attivati. Nel seguente *snippet* è possibile osservare il funzionamento del cambio di colore, *renderizzato* a livello *UI* nel pannello missioni di cui si tratterà più avanti:

```
public void Update(){
       if(missioneCompletata==false){
                [omissis]
                //cambio colore testo missione successiva
                 //N.B: IL TERMINE MISSIONE SI RIFERISCE AD "OBIETTIVO"!
                testoQuestaMissione.GetComponent<Text>().text=testoQuestaMissione.GetCompone
nt<Text>().text+"✓";
                testoQuestaMissione.GetComponent<Text>().color=Color.green;
                testoProssimaMissione.GetComponent<Text>().color = Color.yellow;
                [omissis]
       }
    }
```

Come è possibile osservare, nel caso che prevede il completamento dell'obiettivo in svolgimento, oltre al cambio di colore, viene aggiunto il simbolo di spunta alla rispettiva componente di testo, in modo da poterne visualizzare il compimento sullo stile di una *checklist*.

[102] Come per altri *script*, per *MissioneCompiuta** sono presenti differenti varianti, ma il funzionamento di base è offerto da *MissioneCompiutaSimple*.

Al termine del primo obiettivo della missione in oggetto, è possibile passare al secondo: in questo caso è richiesto all'utente di prelevare delle carote dal frigorifero (ridotte ad una per praticità) e di posizionarle nell'inventario. La condizione di completamento è raggiunta quando il giocatore preleva l'oggetto *carote* (con il quale è possibile interagire via *XRGrabInteractable,* come per tutti i *GameObject* "impugnabili") e lo "aggancia" all'inventario. Alla collisione della carota con l'inventario, viene abilitato *WaitTotSec* e, di conseguenza, lo *script* che sancisce il completamento dell'obiettivo. L'oggetto *carote,* come si potrà leggere in seguito, ha collegate delle componenti particolarmente complesse, che permettono di portare a termine l'obiettivo 3, nel quale al giocatore è richiesto di tagliare le verdure appena prelevate dal frigorifero. Tale parte della missione richiede di "poggiare" l'oggetto *carote*, prelevandolo dall'inventario, sul tagliere e di ridurle in pezzi. Al termine del secondo obiettivo, viene abilitato, tra gli oggetti di scena relativi a quello successivo, un coltello, con il quale è possibile tagliare la carota. L'interazione coltello-carota verrà descritta nel paragrafo inerente agli *script* più complessi, poiché non è necessario al fine di comprendere il funzionamento di base delle missioni. Al termine del terzo obiettivo, sancito dal contatto del coltello con la base del tagliere (è presente un piccolissimo *script*, non riportato, che abilita *WaitTotSec* e conduce al termine dello *step*, cambiando anche il tag associato a *carote*, da *Pickup* a *dadini*), è richiesto all'utente di inserire almeno una porzione dell'ortaggio nella padella. Al contatto con l'oggetto padella, che include numerose collisioni di tipo *BoxCollider* per l'interazione con l'istanza contenente una porzione di *carote*, l'obiettivo termina nel consueto modo già citato. Nello stile di una reazione a catena, è possibile accedere all'obiettivo 5, che richiede l'accensione del fornello. Tramite l'aggiunta in tempo di esecuzione di un *BoxCollider* su una delle manopole, identificata da un materiale di colore verde con emissione, è possibile al giocatore – a partire da tale momento –, tramite "tocco", accendere uno dei fuochi del piano cottura. Il funzionamento del fornello è gestito dallo *script accendiFornello* (riportato in appendice), che permette – oltre all'attivazione dell'effetto grafico del fuoco, all'avvio del suono delle scintille prodotte da una componente *piezoelettrica* e dell'erogazione del gas – di ruotare via *transform* la manopola di 120 gradi verso destra e di riportarla nella posizione originale, a seconda del valore booleano assegnato in tempo di *runtime* a *fornelloON* (che varia in base ai "tocchi" dell'utente, sfruttando gli *Sphere Colliders settati* come *trigger* presenti sui *controllers* delle mani o sulle dita). L'obiettivo successivo richiede il posizionamento della padella sul fornello appena "acceso":

all'esecuzione di tale azione, che comporta il contatto tra i due elementi, viene attivato lo *script spegniFuocoAutodopoC* (in appendice), che include una *Coroutine* (chiamata *Timer*) che gestisce alcuni effetti che simulano la cottura del cibo e, dopo un'attesa di tre secondi, "spegne" il fornello e abilita lo *script* per il termine dell'obiettivo, il solito *missioneCompiutaSimple*:

```
 public IEnumerator Timer(){
       yield return new WaitForSeconds(3f);
       manopola.transform.Rotate(0.0f,-120.0f, 0.0f);
       EffettoFuoco.SetActive(false);
       fornello.GetComponent<cottura>().enabled=false;
       GetComponent<missioneCompiutaSimple>().enabled=true;
   }
```

Al termine dell'obiettivo, rimane al giocatore l'"impiattamento" del pasto appena "cucinato": tramite il contatto tra un piatto (prelevabile dallo scolapiatti)[103], il cui *GameObject* è attivato solo al termine del precedente *step*, e la padella, avviene il termine dell'intera missione.

Nel caso degli ultimi obiettivi, viene abilitato (tramite una versione leggermente modificata di *WaitTotSec*) lo *script termine_capitolo*, molto simile a quelli del tipo *missioneCompletata**, che però sancisce il termine della missione. Al completamento di ogni missione prevista nel simulatore, viene abilitato anche uno *script* (*abilitaCapSucc*) che consente la commutazione degli oggetti di testo nei relativi pannelli *UI*, solitamente associato all'ultimo obiettivo di ognuna, del quale è riportata una porzione di seguito:

```
if(GetComponent<termine_capitolo>().enabled==true){
           if(abilitato==false){
               capitoloSuccessivo.SetActive(true);
               pannelloMissioneI.SetActive(false);
               pannelloMissioneII.SetActive(true);
               abilitato=true;
           }
}
```

Dopo aver illustrato il funzionamento "a catena" comune alla maggior parte degli obiettivi delle missioni previste dal simulatore, attuato presentando la Missione II, è necessario donare spazio alla descrizione del sistema che permette all'utente di tenere traccia dei *task* svolti.

Al fine di consentire all'utente di tenere a portata di mano i progressi di gioco, sono stati inseriti all'interno del simulatore dei pannelli di livello *UI* (interfaccia utente), commutabili tramite il tasto A del *controller* Oculus destro tramite lo *script provaXRInput*, descritto in precedenza, associato alla radice di ogni istanza di *XRRig*. Come è possibile osservare nell'immagine riportata durante la presentazione dei livelli, il *canvas* dei progressi delle

[103] Il contatto tra la padella e il piatto è gestito dallo *script* minore ciboprontо_piatto, presente in appendice.

missioni include quattro gruppi, ciascuno contenente gli elementi di testo (*Text*) relativi ad ogni obiettivo[104]. Come per la maggior parte dei canvas del simulatore, anche questo è *renderizzato* esclusivamente al livello *VR_UI*, per evitare sovrapposizioni con gli elementi 3D e la conseguente sparizione dello stesso in alcune parti dell'ambiente virtuale.

5.3.4.1.1 FollowToTheSide

Per permettere al giocatore di non perdere di vista alcun pannello (e consentirgli di avere "a portata di mano" la sfera dell'inventario), è stato costruito un piccolo *script*, che permette di allineare al *runtime* la posizione dei suddetti oggetti rispetto all'*HMD*. Lo *script*, *FollowToTheSide*, associabile a qualsiasi *GameObject*, si avvale di una funzione *FixedUpdate* per il ricalcolo progressivo della posizione e della rotazione degli elementi, effettuato tramite un'operazione che interviene sui relativi *Vector3*[105]:

```
void FixedUpdate()
    {
        transform.position = target.position + Vector3.up * offset.y
            + Vector3.ProjectOnPlane(target.right,Vector3.up).normalized * offset.x
            + Vector3.ProjectOnPlane(target.forward, Vector3.up).normalized * offset.z;
        transform.eulerAngles = new Vector3(0, target.eulerAngles.y, 0);
    }
```

Per poter utilizzare lo *script*, è necessario associarlo all'elemento che si vuole visualizzare davanti alla telecamera e inserire alla voce *Target* il *GameObject* con la componente *Camera* del *player* desiderato.

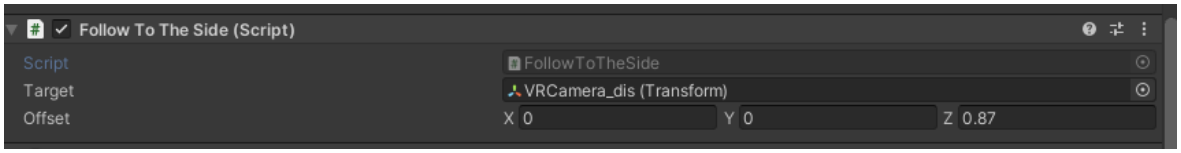

Figura 61. Script FollowToTheSide nell'inspector di Unity,

Come è possibile leggere a proposito dello *script ProvaCambio* proposto nella prossima sezione, nella scena relativa ai livelli 1-2, questa componente subisce delle modifiche in tempo di esecuzione, in base al giocatore scelto alla selezione dell'esperienza.

[104] Il controllo delle stringhe di testo e la commutazione degli elementi del canvas da mostrare in base ai progressi è affidato agli *script MissioneCompiuta** e *abilitaCapSucc*.
[105] Lo *script* non è presente in appendice, in quanto il corpo è composto esclusivamente dalla *FixedUpdate* riportata.

5.3.5 Componenti e scenari complessi

Di seguito verranno trattati alcune componenti relative al simulatore che necessitano di una presentazione, poiché includono un tipo di funzionamento non intuitivo e costituiscono parte importante del meccanismo previsto all'interno dello stesso.

5.3.5.1 ProvaCambio

Lo *script ProvaCambio*, già citato in precedenza, è la componente creata per permettere le operazioni di selezione dinamica degli elementi appartenenti ai livelli 1 e 2. La sua realizzazione è stata necessaria poiché le due esperienze di gioco condividono la stessa scena (e, dunque, gli stessi ambienti) e, in base alla selezione tramite bottone proposta al caricamento della scena, bisogna commutare gruppi di elementi o compiere associazioni di variabili, in base al *player* scelto. Il relativo codice, riportato per intero in appendice, include nel suo corpo due parti: una contenente tutte le associazioni da compiere nel caso di selezione dell'esperienza nel ruolo del giocatore normodotato (*XRRig_normo*) e un'altra quelle inerenti al *player XRRig_dis*. In particolare, è necessario citare la porzione di codice riguardante la selezione della telecamera da utilizzare come *target* per le componenti *FollowToTheSide* connesse ai *GameObject* del pannello missioni, dell'inventario e delle schermate con i messaggi da mostrare a video durante i progressi di gioco. Nel seguente esempio di codice, relativo agli elementi da attivare se si seleziona l'esperienza con il giocatore *XRRig_dis*, è possibile osservare alcuni degli assegnamenti di variabile sul quale si basa lo *script*:

```
followCanvasMissioni.GetComponent<FollowToTheSide>().target=VRCamera_dis;
followBollaInventario.GetComponent<FollowToTheSide>().target=VRCamera_dis;
followCanvasTermineCapitolo.GetComponent<FollowToTheSide>().target=VRCamera_dis;
followCanvasTermineMissione.GetComponent<FollowToTheSide>().target=VRCamera_dis;
followCanvasTermineGioco.GetComponent<FollowToTheSide>().target=VRCamera_dis;
```

Inserendo i corretti *GameObject* dall'*inspector*, sarà dunque possibile assegnare dinamicamente la telecamera relativa al *player* con disabilità a tutti quegli elementi che devono seguire questo tipo di giocatore durante il suo itinerario. Inoltre, la componente ha anche il ruolo di commutare i *colliders* presenti sui portali (iterando in un *array* di *GameObjects*), in base al tipo di giocatore; tale funzione è stata implementata poiché *XRRig_dis* ha bisogno di interagire con i *trigger_dis_act* ad esso dedicati, altrimenti inaccessibili se sovrapposti ad altre collisioni. In chiusura dello *script*, sono definite due *coroutines*, che abilitano – sempre a seconda del tipo di esperienza – gli oggetti delle *overlay* di sfumatura, trattate in seguito.

5.3.5.2 ifRigDisAbilitaInterruttoreChiamata

Un altro *script* di selezione dinamica, ma dal comportamento più semplice, è quello in oggetto, abilitato al caricamento della scena dei livelli 1 e 2 e progettato per l'attivazione dei pulsanti di "chiamata assistente" nel solo caso della scelta dell'esperienza nel ruolo di un giocatore con disabilità motorie. Il codice include una sola funzione da eseguire all'abilitazione dello *script* (*Start*), il cui corpo prevede la ricerca e la successiva popolazione di una lista con tutti gli elementi taggati con *Player_dis*:

```
    IstanzeRigDis = GameObject.FindGameObjectsWithTag("Player_dis");
```

Successivamente, tramite un ciclo *foreach*, viene verificato se gli oggetti trovati e aggiunti nell'*array* (corrispondenti a qualsiasi istanza di *XRRig_dis* ed elementi relativi al giocatore con disabilità) sono attivi e, in caso positivo, vengono aggiunte all'ambiente virtuale le suddette pulsantiere[106].

5.3.5.3 SediaMotorizzataV2

Uno *script* inserito nel progetto del simulatore di cui è necessario scrivere è *SediaMotorizzataV2*, la componente responsabile del movimento del *player* su sedia a rotelle. A differenza degli altri simili impiegati, questo non sfrutta il plugin *XR Interaction Toolkit*, ma basa il suo funzionamento esclusivamente sulle forze fisiche. Buona parte dello *script* è dedicata alla definizione delle operazioni di calcolo inerenti le forze che devono agire sulla massa del giocatore e alla gestione dell'input via *thumbstick*. Nello *snippet* a seguire, ad esempio, è possibile osservare l'assegnamento della stringa corrispondente all'asse verticale del *thumbstick* destro di Oculus alla variabile *forwardForce* (riguardante la forza frontale che permette al giocatore di spingersi in avanti e indietro in base all'accelerazione impartita):

```
forwardForce = Input.GetAxis("Oculus_CrossPlatform_SecondaryThumbstickVertical") * this.Acce
lerazioneFWD;
```

in modo analogo viene effettuato, invece, l'assegnamento all'asse orizzontale del *thumbstick*, responsabile della rotazione (espressa in valore *float* nella variabile *VelocitaRotazione*):

```
float adjustedAxis = Input.GetAxis("Oculus_CrossPlatform_SecondaryThumbstickHorizontal");
```

Una seconda parte dello *script* è, invece, dedicata alla gestione della parte audio della carrozzina motorizzata: come è possibile leggere dal codice riportato in appendice, sono presenti dei rami controlli condizionali che gestiscono la riproduzione dei suoni di avvio,

[106] Lo *script*, al termine del primo ciclo di esecuzione, si autodistrugge.

esecuzione e stop, in base alla pressione del *thumbstick* in una delle direzioni. È inoltre gestita la rotazione delle ruote della sedia, sempre sfruttando le assi di input e le componenti di tipo *Transform* e *WheelCollider* presenti sulle stesse:

```
RuotaDX.motorTorque = Input.GetAxis("Oculus_CrossPlatform_SecondaryThumbstickVertical")*Velo
citaRuote;
RuotaSN.motorTorque = Input.GetAxis("Oculus_CrossPlatform_SecondaryThumbstickVertical")*Velo
citaRuote;
```

Per determinare un movimento più o meno veloce delle stesse, è presente una variabile di tipo *float*, impostata a 100 di *default*, ma che ne influenza il comportamento solo se la massa delle ruote non è troppo alta.

5.3.5.4 Interruttori push-in

Nel livello 3 sono state impiegate delle strategie per tentare di donare un *feeling* realistico ad alcune tipologie di pulsanti, rendendoli in grado di simulare i movimenti di pressione e rilascio degli stessi. Come è possibile osservare nello *script* associato ai bottoni per l'apertura e la chiusura delle porte scorrevoli *PortaCamVR* (codice in appendice), è incluso un sistema in grado di gestire la pressione dei bottoni a livello grafico, sfruttando la posizione di origine degli stessi su uno dei tre assi (a seconda del posizionamento degli stessi).

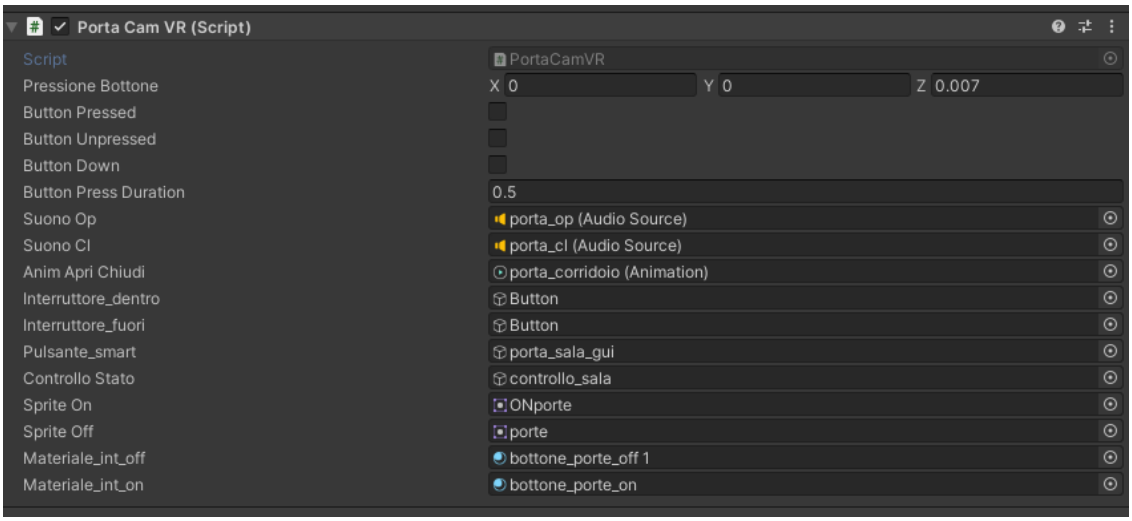

Figura 62. PortaCamVR nell'inspector di Unity

Lo *script*, oltre alla funzione *OnTriggerEnter*, comune a tutti gli interruttori presenti nel livello 3, gestisce nei rami *if-else* presenti in Update la casistica per determinare la posizione corrente del bottone e il relativo stato, grazie all'assegnamento dinamico dei valori *true* e *false* alle variabili booleane previste. Se un pulsante è premuto dal giocatore, esso viene più o meno "spinto" in base al valore stabilito su una delle coordinate di tipo *Transform* (riguardante l'asse Z, in questo caso particolare); tale comportamento è gestito da:

```
this.transform.position += this.transform.TransformDirection(-PressioneBottone);
```
eseguito al termine di ogni verifica dell'evento *OnTriggerEnter*.

5.3.5.5 Aspirapolvere robot e Navmesh

Uno degli elementi interattivi più particolari, ma che non presenta alcuna utilità in merito all'avanzamento nelle missioni, è quello che, nel livello 3, tenta di riprodurre un aspirapolvere robot. Il funzionamento dello stesso si suddivide in due parti: una riguardante il telecomando di controllo e l'altra lo *script wanderingAI*, responsabile del movimento su un tracciato *Navmesh* presente sulla superficie del pavimento della stanza. Innanzitutto, è stato necessario aggiungere un oggetto *NavMeshSurface* alla scena, ove sono stati determinati i confini entro i quali consentire al robot di muoversi in modo casuale. Successivamente, si è passati all'importazione di un modello in formato *Collada* (con estensione .dae) di un aspirapolvere robotizzato e alla definizione dei parametri relativi alla componente *NavMeshAgent* ivi aggiunta, quali la velocità di movimento e gli intervalli entro i quali effettuare delle pause. Solo in un secondo momento, è stato associato lo *script wanderingAI* (disabilitato di *default*) al *prefab* dell'aspirapolvere, tratto da un vecchio progetto[107] che prevedeva delle entità ambulanti senza alcun controllo. Il programma, che richiede una componente *NavMeshAgent*, basa il suo meccanismo su una funzione *Update*, che aggiorna ad ogni *frame* la posizione dell'oggetto (sfruttando un vettore *Vector3*) e su *RandomNavSphere*, che permette di stabilire un *range* entro il quale l'entità *NavMesh* può operare[108]. Una volta preparato il *GameObject* relativo all'apparecchio, è stato realizzato lo *script* per il relativo telecomando, collegato ad un pulsante posto sullo stesso (con associati un *Rigidbody*, *flaggato* come *kinematic*, e un *BoxCollider)*, indispensabile per commutare lo stato dell'aspirapolvere. Lo *script* associato al bottone, chiamato *commutaRoomba*, concentra il suo funzionamento nella funzione *OnTriggerEnter*, entro la quale sono riportati i comandi responsabili dell'attivazione *on runtime* delle componenti *wanderingAI* e *NavMeshAgent*, alla riproduzione dei suoni, all'animazione delle spazzole e al riposizionamento del dispositivo sulla base di ricarica. In particolare, per rendere ciò attuabile è stata adottata la medesima strategia attraverso la quale sono stati gestiti i punti di arrivo del

[107] La fonte dello *script* è andata, purtroppo, perduta. Pertanto, nessun diritto in merito viene detenuto.
[108] Nel codice in appendice è possibile trovare le funzioni nella loro completezza.

giocatore con disabilità nel livello 2: al cambio di stato corrispondente allo spegnimento, è stato inserito il comando

```
roomba.transform.position = puntoArrivo.position;
```

per il riposizionamento dell'aspirapolvere sulla base di ricarica. Lo *script*, tuttavia, necessita di ulteriori limature, per permettere di ammorbidire l'operazione appena descritta, troppo brusca sul lato grafico.

5.3.5.6 Mesh slicing: l'interazione coltello/carota

Come trattato in itinere alla Missione II, nell'obiettivo ove è richiesto all'utente di ridurre in pezzi la carota prelevata dal frigorifero è presente uno *step* in cui, tramite l'impiego del coltello, viene effettuato un taglio sulle *mesh* dell'ortaggio, che viene ridotto in istanze dello stesso.

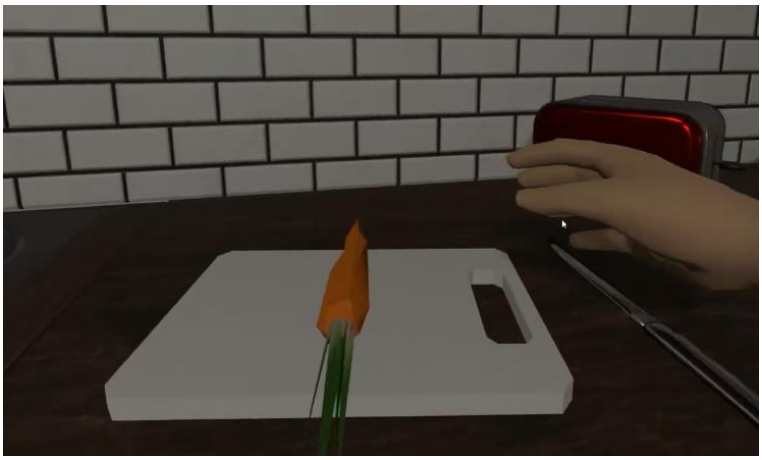

Figura 63. Tagliere con carota, missione II

Tralasciando i dettagli affini alla missione, come il cambio *tag* in tempo di esecuzione al quale si era accennato, è possibile descrivere il funzionamento di tale tecnica, considerando i *GameObjects carote* e *coltello*. Il primo è un solido *lowpoly* (composto da pochi *poligoni*, che consistono in *triangoli*), scelto proprio per avere un maggior controllo sulle sottocomponenti grafiche che lo caratterizzano, al quale è associato uno *script*, *Sliceable* (presente in appendice), derivato dal pacchetto *LightSaber* presente nel *Package Manager* di Unity. Tale componente, che deriva da *MonoBehaviour*, permette di associare all'oggetto 3D alcune proprietà che gli permettono di interagire con un'altra componente associata al coltello, della quale si tratterà più avanti. Queste ultime, gestite tramite valori booleani, permettono di determinare il comportamento dell'oggetto carote al colpo del coltello: *isSolid*, definisce la sua idoneità ad interagire con un altro corpo "solido", *reverseWindTriangles*,

caratterizza la direzione di scissione dei *poligoni* in rapporto alle forze fisiche agenti, *useGravity*, se selezionata, indica di utilizzare la forza di gravità (come avviene nel funzionamento dei *Rigidbody*) e le due *shareVertices* e *smoothVertices* sono parametri riguardanti la disposizione dei vertici dei *poligoni*. La componente che però gestisce il taglio è *Lightsaber*, uno *script* dalla sintassi complessa, associato al *GameObject coltello* che, per funzionare correttamente, necessita della creazione di una specifica gerarchia entro la quale inserire un *GameObject* con il solo *MeshRenderer* e uno chiamato *Parent*, entro il quale annidare l'oggetto *blade* con la sola componente *Transform* posizionata in corrispondenza della lama e *hilt*, come la precedente, ma da porre sull'impugnatura. All'interno di *blade*, che include un *MeshCollider* con il *flag* abilitato sulle voci *Convex* e *isTrigger*, si trovano altri due oggetti con associata la sola *Transform*: *Tip* e *Base*, posti sulla punta e sulla base della lama.[109]

Lo *script Lightsaber* (anch'esso presente in appendice) permette la divisione in sottounità delle *mesh* dei *GameObjects* ai quali è associata la componente *Sliceable*; ogni parte generata corrisponde ad una sezione della superficie e rappresenta un'istanza dell'oggetto. All'interno del programma sono presenti, tra le variabili, due *array*: uno di valori interi (*_triangles*) e l'altro di vettori *Vector3* (*_vertices*). La funzione *Start* serve ad istanziare i riferimenti relativi ai *triangoli* e ai *vertici* che compongono l'oggetto 3D e a caricare le informazioni sui materiali dell'effetto scia (*trailMaterial*, non implementato) e della lama del coltello (*bladeMaterial*). Nei frammenti sotto riportati, inoltre, viene assegnata alle variabili *_previousTipPosition* e *_previousBasePosition* la posizione degli elementi *Transform* relativi, rispettivamente, alla punta e alla base della lama.

```
_previousTipPosition = _tip.transform.position;
_previousBasePosition = _base.transform.position;
```

La seconda funzione inclusa nello *script* è *LateUpdate* (che ha una priorità di caricamento inferiore rispetto alla simile Update) e gestisce la creazione di *vertici*, *triangoli* e superfici a partire dal *mesh* del coltello e aggiorna, ad ogni *frame*, le posizioni di punta e base della relativa lama. Sono presenti, inoltre, le funzioni *OnTriggerEnter* e *OnTriggerExit*, che rispondono al contatto tra i *MeshCollider* del coltello e della carota (questo settato come

[109] La "scia di taglio" *TrailMesh* non è stata implementata nel funzionamento.

trigger); alla verifica del primo tipo di evento, viene assegnata alle relative variabili la posizione entro la quale è avvenuta la collisione con la carota:

```
_triggerEnterTipPosition = _tip.transform.position;
_triggerEnterBasePosition = _base.transform.position;
```

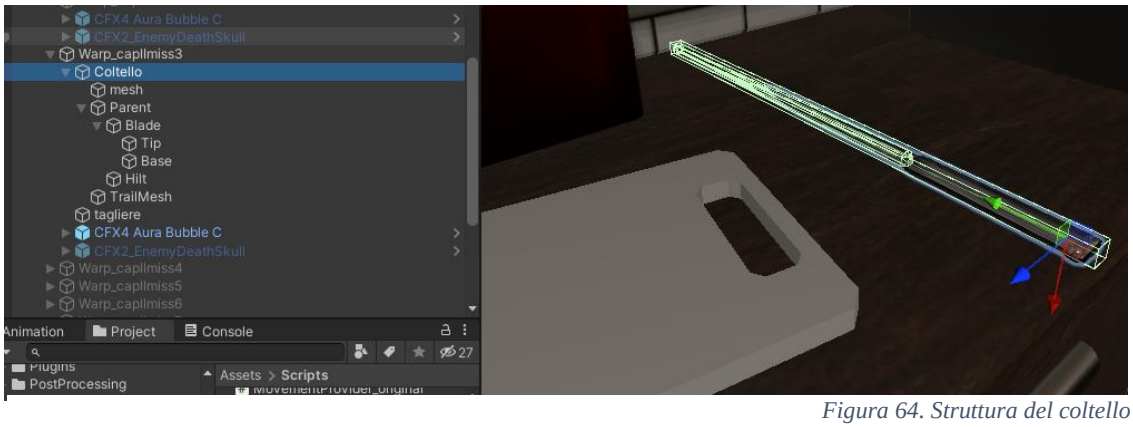

Figura 64. Struttura del coltello

All'interno di *OnTriggerExit*, invece, viene creato un nuovo triangolo tra le posizioni di base e punta della lama e viene eseguito un ricalcolo sulla *normale* (il vettore perpendicolare alla superficie del poligono), in modo da allinearlo all'oggetto da tagliare:

```
Vector3 side1 = _triggerExitTipPosition - _triggerEnterTipPosition;
Vector3 side2 = _triggerExitTipPosition - _triggerEnterBasePosition;
Vector3 normal = Vector3.Cross(side1, side2).normalized;
Vector3 transformedNormal = ((Vector3)(other.gameObject.transform.localToWorldMatrix.transpo
se * normal)).normalized;
```

e, in seguito, è associata al vettore *transformedStartingPoint* la posizione di origine (convertita da *world space* in *local space*) dell'oggetto (*carota*) al momento del "taglio":

```
Vector3 transformedStartingPoint = other.gameObject.transform.InverseTransformPoint(_trigger
EnterTipPosition);
```

In coda alla funzione, viene riempito l'*array slices* con tutte le nuove istanze, corrispondenti alle parti dell'oggetto divise in triangoli e l'oggetto originale viene distrutto:

```
GameObject[] slices = Slicer.Slice(plane, other.gameObject);
Destroy(other.gameObject);
```

Infine, sempre all'uscita dal *trigger*, viene creato un nuovo *Rigidbody*, ottenendo la componente associata all'elemento in posizione 1 dell'*array* appena creato e vi è applicata una forza istantanea, che nel caso in esame scaraventa più o meno lontano (a seconda della vigorosità del colpo impartito) i pezzi della carota:

```
Rigidbody rigidbody = slices[1].GetComponent<Rigidbody>();
[omissis]
rigidbody.AddForce(newNormal, ForceMode.Impulse);
```

Per tale ragione, al fine di non disperdere tutti i "pezzi" e poter restare bloccati nella missione, è necessario cadenzare la forza, scuotendo appena il *controller touch*.

5.3.5.7 Lo smartphone in-game

A conclusione di tale sezione, è necessario descrivere, almeno sommariamente, la struttura dello *smartphone in-game* previsto nel livello 3; sebbene tale non costituisca un tipo di componente specifico ma sia un insieme di elementi *UI* disposti su un *canvas*, desta attenzione, in quanto presenta una struttura gerarchica interessante.

L'oggetto *parent* dello *smartphone*, annidato all'interno del *RightHandController* di *XRRig_dis*, è commutabile dall'utente tramite il tasto B (solo nel terzo livello) ed è costituito da una base fissa con la componente *MeshRenderer* e da due schermate, di tipo *UI*, commutabili tramite il "tocco" sulle freccette poste in alto. Ciascuna pagina contiene, raggruppati in categorie, i *GameObjects* con associati un *Rigidbody* e un BoxCollider, rappresentanti i bottoni da premere per azionare i relativi oggetti di scena. Ad ognuno è assegnata una componente *Sprite Renderer*, attraverso la quale viene *renderizzata* l'icona corrispondente, di tipo *Sprite2D*. L'interazione con questi ultimi è possibile con il solo dito della mano sinistra; alla quasi totalità degli oggetti è associato il medesimo *script* utilizzato nei bottoni presenti nella casa, che assolvono alla stessa funzione e constano degli stessi

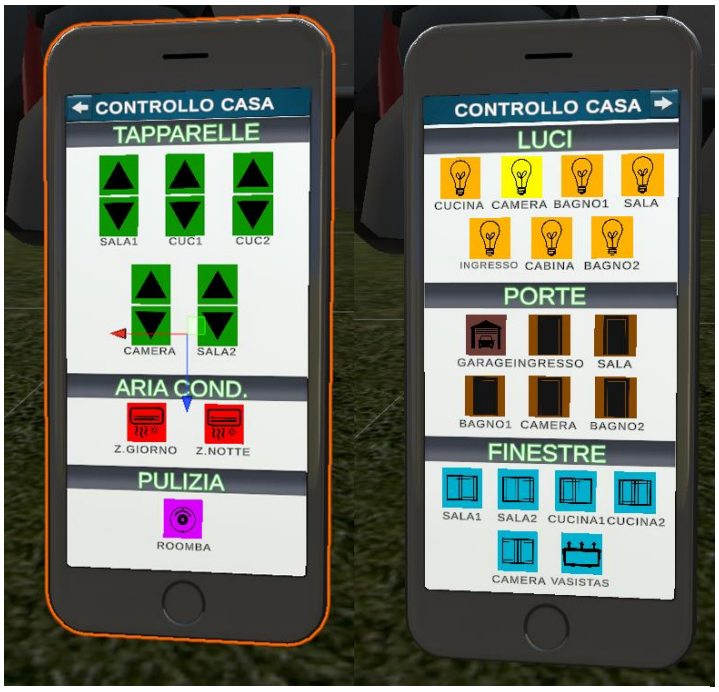

Figura 65. Pagine 2 e 1 dello smartphone in-game

Figura 66. Gerarchia dello smartphone (con entrambe le pagine attivate)

elementi di controllo o di versioni leggermente modificate per essere adattate alla *UI*[110]. Per una gestione ottimale, non sono stati inclusi nello stesso i tasti funzione relativi a oggetti che, realisticamente, sarebbe scomodo controllare da uno *smartphone* anche nel contesto di un impianto domotico reale, come ad esempio il movimento del montascale.

5.3.6 Componenti dinamiche composite

Ad arricchire le scene, sono stati inseriti nei due ambienti virtuali previsti alcuni particolari oggetti che sfruttano le forze fisiche impartite via *controller* per riprodurre in modo quanto più realistico alcuni tipi di movimento. La maggior parte di questi si avvale di almeno una parte fissa e di una o più mobili e sfrutta le componenti *HingeJoint* e *ConfigurableJoint* di Unity per la gestione delle zone di snodo.

5.3.6.1 Cassetti

Ogni cassetto presente nella cucina dei livelli 1 e 2 è azionabile tramite la componente *XRGrabInteractable*, con assegnata la maniglia di essi come punto di presa; la gestione dello scorrimento entro la parte fissa del mobile sottopiano è affidata alla componente *ConfigurableJoint*, che permette di stabilire l'asse sul quale poter eseguire il movimento (in questo caso, z) e di impostare, tra i vari parametri a disposizione, il limite entro il quale poter effettuare le azioni di spinta e di traino. Tale componente è possibile trovarla associata, con

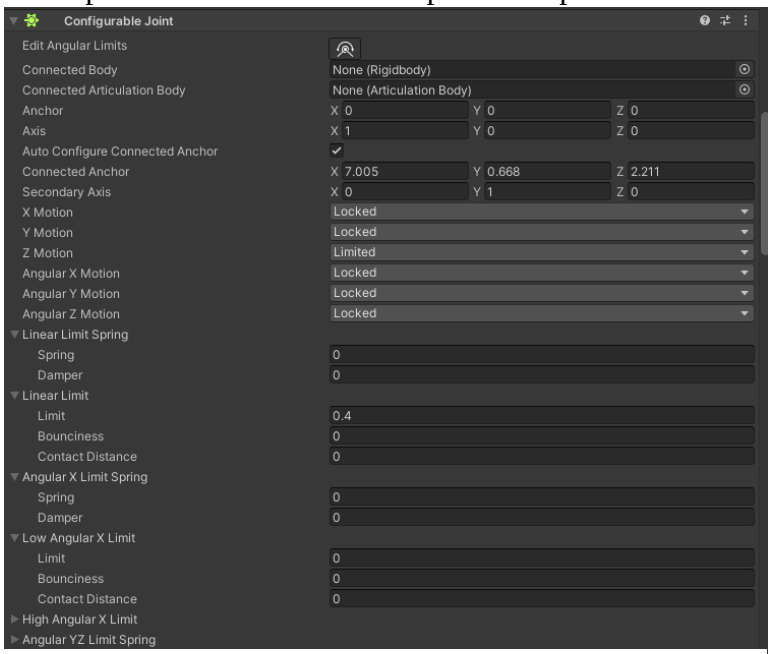

Figura 67. Configurable Joint su un oggetto Drawer

[110] Tutti gli *script* relativi alla gestione degli elementi interattivi tramite gli eventi legati alle collisioni sono disponibili al link in appendice

settaggi leggermente differenti, anche alle ante scorrevoli previste nel livello 3 e nelle corde-slider delle tapparelle nei livelli 1 e 2. In rapporto al corretto funzionamento delle giunture, è necessario gestire in modo idoneo le collisioni: il contatto tra due superfici o con il *player* può causare infatti un comportamento anomalo e originare conflitti di forze fisiche. I cassetti sono tra gli oggetti che, durante la progettazione, hanno manifestato la maggior parte dei problemi di questo tipo.

5.3.6.2 Ante

Nelle ante delle finestre della scena *LVL1_LVL2*, così come in altri punti del simulatore ove è richiesta un'articolazione a cerniera (es: pensili), è impiegata la componente *HingeJoint*, che fornisce uno snodo di un certo numero di gradi angolari, in una certa direzione ed entro un certo limite, se configurato. Nel caso delle finestre, viene interessato l'asse y ed è necessario limitare l'apertura delle ante a 90 gradi verso l'interno: per tale ragione, come da figura, sono impostati i valori -1 sulla y e -90 e 0 rispettivamente ai limiti minimo e massimo.

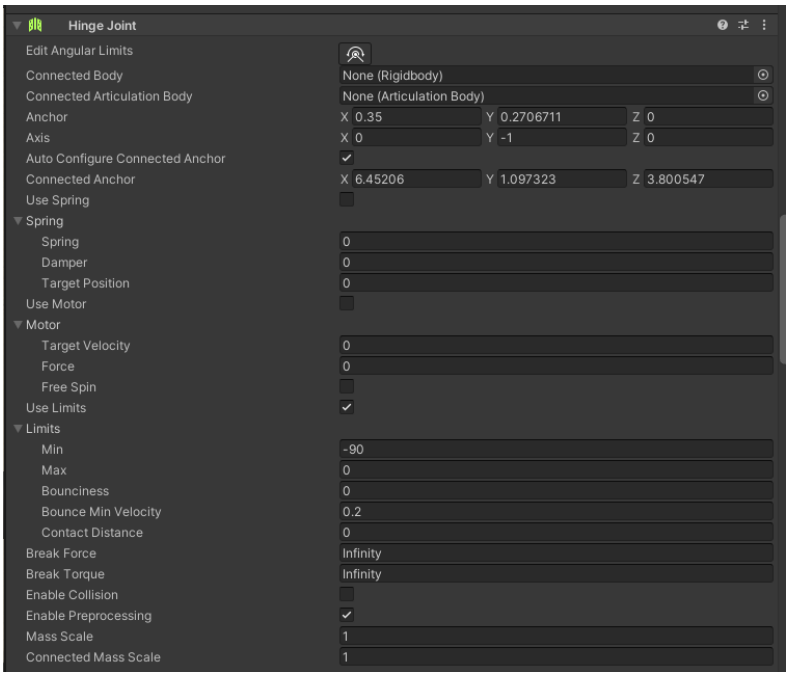

Figura 68. Hinge Joint applicata all'anta di una finestra nei livelli 1 e 2

Come nel caso precedente, è necessario gestire bene le collisioni, per evitare alle stesse di "rimbalzare" violentemente sotto l'influsso di forze fisiche esterne.

5.3.6.3 Leve

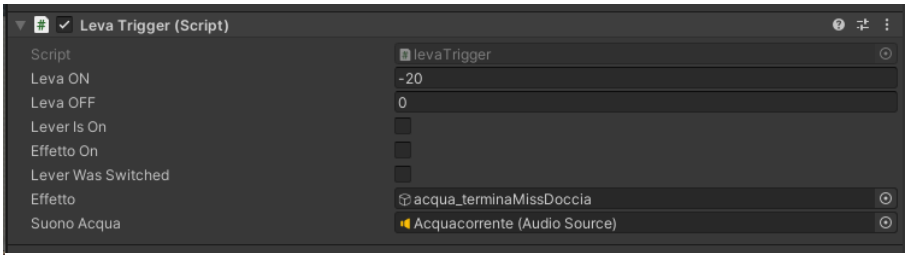

Figura 69. Script levaTrigger

La componente appena presentata è impiegata anche nella parte dinamica relativa alle leve che nel simulatore sono correlate ai miscelatori della rubinetteria presente all'interno dell'abitazione. In tutti i casi, il movimento avviene intorno all'asse delle ascisse ed è limitato su un angolo che, da una posizione a -20, arriva a 0. A tutti i *GameObject* che gestiscono l'apertura e la chiusura dell'acqua dei sanitari e del lavello, è associato anche uno *script* in grado di leggere il valore corrente dell'angolo e di attivare o meno l'effetto relativo all'acqua corrente. Tale componente è *levaTrigger* (presente in appendice), un programma che tramite l'impiego di valori *booleani* e la definizione di due metodi di tipo *Quaternion*, *OnHingeAngle* e *OffHingeAngle*, è in grado di commutare lo stato del rubinetto sulla base dei valori minimo e massimo immessi dall'*inspector*:

```
private Quaternion OnHingeAngle() {
      return Quaternion.Euler(this.leverHingeJoint.axis * levaON + startingEuler);
  }
  private Quaternion OffHingeAngle() {
      return Quaternion.Euler(this.leverHingeJoint.axis * levaOFF + startingEuler);
  }
```

5.3.7 Stile grafico

Come è possibile notare dagli *screenshots* proposti in precedenza, nello sviluppo del simulatore è stato adottato uno stile realistico, che include ambienti fedeli alla realtà e permette all'utente di vivere un'esperienza ancora più immersiva. Per gli interni e gli esterni della casa su due livelli è stato impiegato il pacchetto *HQ Residential House*[111] (disponibile a pagamento nell'*Asset Store* di Unity), dal quale è stato prelevato il relativo *prefab* completo di molti oggetti di arredo, del sistema di *LightProbes* e alcuni *script*, parte dei quali è andata rimossa. Le zone relative alla cucina, alla camera da letto e ai bagni sono state parzialmente o totalmente rinnovate, per far spazio a arredi in linea alle modalità di gioco; in particolare, per il terzo livello, sono stati rimossi tutti gli elementi di intralcio al *player*, a favore di sanitari

[111] *HQ Residential House*. Link alla pagina web: https://assetstore.unity.com/packages/3d/environments/urban/hq-residential-house-48976.

e oggetti a misura di una persona con disabilità motorie. La quasi totalità dei *prefabs* importati nella cartella *Assets* è prelevata gratuitamente da siti web che offrono il download di materiale 3D libero in formato *Collada* (.dae) o *3DS Max* (con estensione .3ds). L'adozione di uno stile realistico presenta, tuttavia, numerosi svantaggi in merito alla performance: ove sono presenti troppe luci o oggetti dinamici, è possibile osservare dei *lag* durante l'esecuzione se il simulatore viene provato con l'ausilio di macchine non troppo potenti. Per ridurre il problema, dopo alcuni test tramite lo strumento Profiler messo a disposizione da Unity che confermava, purtroppo, una scarsa resa in alcune fasi della simulazione, sono stati *staticizzati* tutti gli oggetti non coinvolti nella parte dinamica.

5.3.7.1 Materiali, textures e shaders

Al fine di ottenere uno stile grafico quanto più omogeneo in ogni ambiente previsto nelle scene, sono stati impiegati quasi esclusivamente i materiali e le textures a corredo del pacchetto *HQ_ResidentialHouse*, ai quali sono stati aggiunti alcuni materiali semplici (come quelli emissivi sui bottoni del livello 3) o che utilizzassero particolari *textures*, come alcuni che riproducono differenti tipi di legno. Il materiale della *Skybox* di *default* è stato rimpiazzato da uno che rappresenta un cielo nuvoloso al tramonto, per restare nel tema delle missioni, che si svolgono ad ora di cena. Inizialmente, si era scelto di puntare sulla pipeline *Lightweight Render Pipeline* (*LWRP*)[112], per poter utilizzare gli *shaders* relativi a quest'ultima che offrivano una resa migliore di alcuni materiali, ma l'idea è stata presto abbandonata e sono stati ripristinati quelli *standard*, in quanto causavano alcuni conflitti a livello grafico.

5.3.7.2 Luci

Al fine di migliorare la performance in termini di generazione delle *Lightmaps* durante lo sviluppo, è stato scelto come *Light Manager* il deprecato *Enlighten*; tuttavia, in eventuali nuove *release* del simulatore tale sistema va sostituito da un gestore più all'avanguardia, poiché questo non è in grado di supportare né le nuove pipeline grafiche. Come metodo di illuminazione, si è optato per quello *Subtractive*, per ottenere un buon supporto alla modalità di rendering delle luci *Mixed* (in tempo reale e di tipo *baked*, con le relative *lightmaps* relative agli elementi statici). Nelle scene è stata inserita una sola *Directional Light* per la simulazione

[112] *Shaders* di *LWRP*. Link alla pagina web: https://docs.unity3d.com/Packages/com.unity.render-pipelines.lightweight@5.10/manual/shaders-in-lwrp.html.

della luce solare, con *rendering* in tempo reale; per quelle artificiali, invece (poste all'interno dell'abitazione, come nelle appliques a muro) sono state impiegate delle *PointLights* e, nel caso dei faretti, alcune *SpotLights*. Per una maggiore resa realistica, tutte le lampade con annessa una *PointLight* presentano le plafoniere con applicato un materiale *emissivo*, commutabile insieme alle luci tramite lo *script* applicato ai relativi interruttori.

5.3.7.2.1 Reflection probes

In entrambe le scene sono stati impiegati alcuni *Reflection Probes* per gestire al meglio la luminosità nelle stanze e donare l'idea al fruitore di trovarsi in un ambiente con un'illuminazione quanto più realistica, anche in termini di gestione delle luci riflesse. Ai fini di non appesantire ulteriormente la grafica, è stato scelto di non *renderizzarli* in tempo reale, ma di conservarne le caratteristiche nelle rispettive *Lightmaps*.

5.3.7.3 Animazioni

All'interno del simulatore sono state inserite molte animazioni, alcune delle quali per puro gusto grafico, altre, invece, per assolvere a funzioni pratiche, come al controllo di porte, finestre e montascale. In particolare, nel livello 3, si è fatto largo uso di questa tecnica su diversi oggetti, ai quali sono correlati anche degli *script* per la relativa gestione. Salvo alcune gestite tramite l'interfaccia di controllo *Mecanim* (come nel caso della porta di ingresso nei livelli 1-2), la maggior parte delle animazioni presenti nelle scene è di tipo *legacy* ed è commutabile direttamente via *script*.[113] Di seguito sono riportate alcune di esse che presentano comportamenti più complessi e importanti ai fini del funzionamento della meccanica di gioco.

5.3.7.3.1 Porte e finestre

Le porte e le finestre del livello 3 sono rese dinamiche unicamente tramite l'impiego delle animazioni, avviabili tramite gli *script* dedicati. Nel caso delle prime, il meccanismo implica il solo movimento sull'asse z e, dunque, si tratta di un tipo di animazione molto semplice; come è possibile osservare, ad esempio, dallo *script PortaCamVR*, la commutazione del meccanismo è impartita tramite lo *script* presente sul relativo bottone, sfruttando lo stato di una variabile booleana:

[113] Tale tecnica di implementazione andrebbe abbandonata, in vista di aggiornamenti.

```
            //apertura
            if (controlloStato.active==false){
                AnimApriChiudi.Play("portaCamOp");
                [omissis]
            //chiusura
            }
            else{
                AnimApriChiudi.Play("portaCamCl");
                [omissis]
            }
```

Tale logica è utilizzata in tutte le porte presenti nell'abitazione ed è molto simile a quella applicata per la gestione delle finestre a ghigliottina e di tipo *vasistas*, il cui codice è riportato in appendice agli *script ControlloFinestraGhigliottina_VR* e *Vasistas_1_VR*. Per le finestre del livello 3 (a ghigliottina) sono state utilizzate le animazioni incluse nel pacchetto *WindowsPack*[114], che agiscono quasi esclusivamente sull'asse y per simulare lo scorrimento della parte mobile verso l'alto. Per le *vasistas*, invece, è stata realizzata un'animazione su misura, che agisce sul punto *pivot* corrispondente alle cerniere e ne permette l'apertura (e la conseguente chiusura, nella *clip* complementare) di 40 gradi rispetto all'angolo originale.

5.3.7.3.2 Il montascale

Il movimento del montascale, presente nel livello 3, è interamente gestito da un'animazione, le cui *clip* sono avviate tramite lo *script muoviEscalatore* applicato ai relativi pulsanti. Le fasi di salita e discesa (che alterano il solo parametro *Position* sugli assi x e y) dello stesso sono sviluppate all'interno di due *clip*, ciascuna la versione inversa dell'altra e dalla durata di ben 20 secondi. È stato scelto di utilizzare un valore così alto poiché, in caso ne fosse stato impostato uno inferiore, il corpo del *player* sarebbe stato sbalzato via durante i due tipi di movimento a causa dell'azione delle forze fisiche agenti tra la piattaforma e la sedia a rotelle.

All'interno delle scene sono, tuttavia, presenti numerose animazioni anche complesse, come quella riguardante la porta del garage, ma non è necessario trattarle, poiché non assolvono ad alcuna utilità pratica.

5.3.7.4 Effetto dissolvenza

Come già accennato in precedenza, al fine di non rendere troppo bruschi i passaggi tra una scena all'altra o, comunque, dove fosse previsto lo "spostamento" immediato da una posizione all'altra della casa, è stato annidato nella *VRCamera* di ogni *XRRig* un *GameObject* chiamato *LoadingOverlay*, in grado di donare un effetto dissolvenza. Tale oggetto, che

[114] *Windows Pack*, Asset Store. Link alla pagina: https://assetstore.unity.com/packages/3d/props/interior/windows-pack-72257.

include solo una componente *Mesh Renderer* con collegato un materiale di colore nero, di tipo *Unlit*, ha associato uno *script* chiamato *SceneFader* (codice in appendice) che, se abilitato dinamicamente in tempo di esecuzione, permette al giocatore di visualizzare una sfumatura verso il nero (o verso la trasparenza, a seconda dei parametri inseriti) e di percepire meno le situazioni di "stacco", caratterizzanti alcune parti della simulazione. La componente sfrutta il canale *alpha* del colore del materiale di *LoadingOverlay* per consentire allo stesso di passare dal nero pieno al trasparente e viceversa, in base alla direzione di dissolvenza (in o out) definita all'interno della funzione *OnEnable*, che viene eseguita appena lo *script* viene abilitato:

```
   void OnEnable()
    {
        StartCoroutine(Fade(FadeDirection.In));
    }
```

Lo *script* basa il suo funzionamento sulla *coroutine Fade* che, all'invocazione, richiama il metodo *SetColorImage*, responsabile del caricamento delle caratteristiche del materiale e della regolazione del canale *alpha* dello stesso in base alla direzione di sfumatura e al tempo stabilito nella variabile *fadeSpeed*:

```
 private void SetColorImage(ref float alpha, FadeDirection fadeDirection)
    {
        materiale_overlay=overlay.GetComponent<MeshRenderer>().material;
        materiale_overlay.color= new Color (materiale_overlay.color.r,materiale_overlay.colo
r.g,materiale_overlay.color.b, alpha);
        alpha += Time.deltaTime * (1.0f / fadeSpeed) * ((fadeDirection == FadeDirection.Out)
? -1 : 1) ;
    }
```

La *coroutine* (vedere appendice), al raggiungimento delle due condizioni di terminazione previste gestite da altrettanti cicli *while*, entro i quali sono utilizzate le variabili di tipo *float alpha* e *fadeEndValue* come *range* (alle quali sono assegnati i valori di 1 e 0 in base al caso), permette di commutare lo stato dell'oggetto *LoadingOverlay*. Per ottenere un effetto di dissolvenza da nero opaco a trasparente, è sufficiente chiamare la *coroutine Fade* con associata una delle due costanti (*In*, corrispondente al valore massimo del canale alpha e dunque al colore totalmente opaco e *Out*, che rappresenta il minimo dello stesso e indica la trasparenza completa) contenute nel *set* dell'*enumerazione FadeDirection* definito in cima alla classe:

```
StartCoroutine(Fade(FadeDirection.In));
```

Per la componente, non essendo presenti campi *serializzabili* e regolabili dall'*inspector*, è necessario intervenire sul codice per cambiare direzione di sfumatura; tuttavia, nel simulatore

è sfruttata ovunque la sola *FadeDirection.in*, in quanto era necessario esclusivamente riprodurre delle dissolvenze al nero e non viceversa.

In queste sezioni sono state trattate le tecniche e le scelte di implementazione che hanno permesso di rendere concreta l'idea del simulatore, precedentemente studiata dopo un'analisi sul campo, e hanno condotto alla creazione di una *build* stabile per la piattaforma Windows, prodotto successivamente sottoposto ad alcuni *test* di usabilità, per valutarne l'effettivo impatto su un campione controllato di utenti. A tal proposito, nel prossimo capitolo è trattato il processo di *user study* condotto, del quale sono analizzati i risultati e ne sono tracciate alcune delle caratteristiche più interessanti, rilevate anche tramite l'incrocio di alcuni dei dati rilevati.

6 User study

Al termine dello sviluppo del simulatore, è stata creata una *build* compatibile con il sistema Microsoft Windows ed è stata somministrata ad un campione di utenti, per poterne comprendere punti di forza e limiti, ma anche al fine di raccogliere *feedback* relativi all'esperienza utente in rapporto al prodotto.

6.1 Contesto e questionario

Per la totalità delle prove effettuate, è stata impiegata la strumentazione utilizzata anche per lo sviluppo, composta dal *notebook* MSI Leopard 95E e dal visore Oculus Quest, ad esso connesso via cavo Link. Ogni utente è stato supervisionato durante l'intero svolgimento dei *test*, ma non è stato aiutato, per evitare di alterare i risultati. Per la quantificazione e la conseguente analisi, è stato preparato un questionario[115], sottomesso poi alle persone coinvolte dopo la fruizione.

Il questionario proposto si compone di tre sezioni principali: una dedicata alla raccolta dei dati relativi ad ogni profilo, una sull'analisi dell'esperienza utente in merito ai tre livelli giocati e, infine, una inerente le impressioni generali ottenute al termine di ogni prova.

Per la definizione del profilo utente, sono stati individuati quattro parametri: età, professione, livello di esperienza con la Realtà Virtuale e conoscenza del problema relativo alle barriere architettoniche. Nella seconda sezione, relativa ai tre livelli, sono state strutturate altrettanti gruppi di domande, relativi all'esperienza di gioco: per la valutazione, sono state impiegate una *scala di Likert* con valori da 1 a 5 (con 1 equivalente a "Per niente" e 5 a "Molto") e una casella di testo, al fine di permettere al fruitore di inserire dati in merito alla complessità di gioco, al tempo impiegato e ad eventuali problemi riscontrati che lo hanno bloccato nel proseguire nelle missioni proposte. Pertanto, in caso di mancato completamento di un livello, l'utente è invitato a selezionare il valore massimo (5) nella sezione della complessità e a indicare, in ogni caso, il tempo impiegato per il tentativo e la motivazione per la quale è stato costretto a fermarsi. Al termine della sezione sulla valutazione dell'esperienza di gioco, nel questionario è stata inserita una parte dedicata ad alcune impressioni generali, in particolare concernenti la sfera emotiva dell'utente: impiegando sempre una scala da 1 a 5 punti, ciascun

[115] *Questionario usabilità simulatore di RV immersiva*. Link alla pagina: https://forms.gle/HFheDDXrAUGqJg2t6.

partecipante al *test* è stato invitato a rispondere a quattro domande riguardanti il livello di "immersività" trovata nel ruolo dei tre giocatori previsti, lo stato d'animo, il desiderio di interrompere la simulazione ed eventuali sensazioni fisiche sgradevoli provate. A queste, sono stati aggiunti altri quattro quesiti, tra i quali il primo relativo alla valutazione dell'utilità del simulatore nello scopo di sensibilizzare sul problema delle barriere architettoniche, il secondo, sull'efficacia riguardo l'abbattimento di queste ultime in merito all'integrazione dei portatori di disabilità motoria, il terzo, sulle tecnologie assistive e al conseguente impiego per il miglioramento del tenore di vita di tali soggetti e l'ultimo, sul giudizio a proposito della totale abolizione della figura di *caregiver*, a favore dell'automazione domestica. Infine, è stato inserito un campo di testo, entro il quale ciascun partecipante ha potuto suggerire alcune migliorie da apportare al simulatore.

6.2 Report dei risultati

In questa sezione saranno presentate le tabelle relative al *dataset* di *output*, ottenute dalla pagina della piattaforma Google Forms relativa al questionario al termine della somministrazione dello stesso a tutti i partecipanti. Dopo un'opportuna operazione di pulizia, verranno illustrati i dati, raggruppati in base alla sezione di interesse e, in un momento successivo, i più salienti saranno oggetto di alcune importanti analisi.

Lo studio è stato condotto su sette utenti, appartenenti a fasce di età, background culturali e professionali differenti e con diversi livelli di esperienza in merito alla conoscenza del mondo della RV e dei problemi inerenti le barriere architettoniche. Per garantire l'anonimato, ciascun partecipante è stato contrassegnato da un identificativo (ID).

Di seguito è riportato un riepilogo dei dati "sporchi", ottenuti a partire dal questionario, i quali saranno, successivamente, ripuliti e standardizzati per permettere di effettuare un'analisi ordinata.

Profilo utente

Inserisci la tua età	Inserisci la tua professione	Quanta esperienza hai con la Realtà Virtuale?	Quanto conosci il problema delle barriere architettoniche?
30	docente	1	4
67	Professoressa di lingue	1	5
62	Vicepreside	1	5
25	studente di economia	2	3
21	studentessa	1	3
21	studente	1	2
34	biologo	2	4

Tabella 1. Dati profilo utente

Esperienza di gioco

Livello 1

Quanto è stato complesso completare il livello 1?	Quanto tempo hai impiegato, approssimativamente, per completarlo? (in minuti)	Se non sei riuscito/a a completare il livello, descrivi il problema e indica dove ti sei bloccato/a (indicando la missione o l'obiettivo):
3	20	
5	25	
4	15	
2	10	
3	17	
4	15	
3	18	

Tabella 2. Dati esperienza di gioco - Livello 1

Livello 2

Quanto è stato complesso completare il livello 2?	Quanto tempo hai impiegato, approssimativamente, per completarlo? (in minuti)	Se non sei riuscito/a a completare il livello, descrivi il problema e indica dove ti sei bloccato/a (indicando la missione o l'obiettivo):
4	28	
5	30	
3	26	
3	15	
4	24	
5	24	
3	21	

Tabella 3. Dati esperienza di gioco - Livello 2

Livello 3

Quanto è stato complesso completare il livello 3?	Quanto tempo hai impiegato, approssimativamente, per completarlo? (in minuti)	Se non sei riuscito/a a completare il livello, descrivi il problema e indica dove ti sei bloccato/a (indicando la missione o l'obiettivo):
3	15	
5	22	
2	18	
2	10	
3	20	
3	16	
2	15	

Tabella 4. Dati esperienza di gioco - Livello 3

Impressioni generali e suggerimenti

Per motivi di spazio, le richieste poste in merito all'ultima sezione del questionario sono riportate nella seguente legenda:

A. Quanto sei riuscito/a ad "immergerti" nei vari giocatori e nelle diverse situazioni presentate nel simulatore?

B. Quanto eri rilassato durante lo svolgimento?

C. Quanto hai preso in considerazione di abbandonare la prova?

D. Se hai provato sensazioni fisiche sgradevoli durante lo svolgimento, dovute all'utilizzo del visore (motion sickness), parlane qui:

E. Quanto credi che un simulatore di realtà virtuale immersiva possa aiutare le persone a familiarizzare con il problema delle barriere architettoniche e possa fungere da strumento di sensibilizzazione?

F. Quanto credi che l'abbattimento delle barriere architettoniche possa favorire l'integrazione dei portatori di disabilità motoria?

G. Quanto credi che la domotica e le tecnologie assistive possano aiutare le persone con difficoltà motorie ad essere più autonome?

H. Quanto credi che domotica e tecnologie assistive possano sostituire completamente l'essere umano (caregiver, assistente "fisico")?

I. Cosa, secondo te, potrebbe essere utile integrare nel simulatore? Se hai suggerimenti, inseriscili qui:

A	B	C	D	E	F	G	H	I
4	3	2	lieve mal di testa. è la prima volta che provo il visore	5	4	4	2	sollevatore x auto
3	1	5	mi girava la stanza dopo aver tolto il casco e mi tremavano le mani	4	5	3	1	il prodotto è buono ma non è per tutti se non sei abituato puoi sentirti male e non riuscire
4	1	2	Nulla mi sono divertito	4	4	4	2	Mettere google home perchè aiuta tanto anche me che sono anziano
4	5	1	leggero bruciore agli occhi	4	5	4	2	
3	1	4	no	4	5	3	1	
4	3	1	no	4	5	3	2	
3	3	2	no	4	5	3	2	

Tabella 5. Dati sulle impressioni generali

Come sarà possibile osservare nella sezione sull'analisi, non tutti i dati raccolti, in particolare in quest'ultima sezione, verranno utilizzati, poiché meramente dedicati alla conoscenza di impressioni relative all'esperienza utente e non fini all'osservazione di fenomeni statistici.

6.3 Analisi dei risultati

Prima di addentrarsi nel lavoro di analisi, è stato necessario ripulire le tabelle dai dati non quantificabili, relativi al feedback dei singoli utenti, i quali saranno poi trattati in coda al capitolo. È stata, inoltre, effettuata un'operazione di *standardizzazione* sulle intestazioni dei parametri esaminati, per questioni legate alla leggibilità. Al termine delle procedure appena citate, è stata ottenuta la seguente tabella:

ID	Età	Esp. RV	Conosc. BA	Compl. LVL1	Tempo LVL1	Compl. LVL2	Tempo LVL2	Compl. LVL3	Tempo LVL3	Relax
1	30	1	4	3	20	4	28	3	15	3
2	67	1	5	5	25	5	30	5	22	1
3	62	1	5	4	15	3	26	2	18	1
4	25	2	3	2	10	3	15	2	10	5
5	21	1	3	3	17	4	24	3	20	1
6	21	1	2	4	15	5	24	3	16	3
7	34	2	4	3	18	3	21	2	15	3

Tabella 6. Dati ripuliti e standardizzati

Come è possibile osservare, è stata rimossa anche la colonna relativa alla professione: non essendosi rivelata utile nell'evidenziare aspetti interessanti, si è preferito ometterla, in modo da ottenere una visione ancora più chiara sui dati da impiegare nell'analisi. Al contrario, è

stata inserita l'unica variabile che, tra quelle inerenti le impressioni generali dell'utente al termine della fruizione del simulatore, risultava di una certa utilità per osservare alcuni fenomeni di correlazione: il livello di rilassamento (*relax*) durante lo svolgimento della prova. Contrariamente a quanto progettato inizialmente, non è stato possibile effettuare studi sul tasso di successo e di fallimento, in quanto tutti i partecipanti sono riusciti, seppur con qualche difficoltà, a portare a termine ogni obiettivo previsto. Alla fine della prova, ogni utente ha anche scelto se proseguire nel gioco nella modalità libera (dopo lo sblocco di ogni missione), continuando a provare le interazioni previste nell'ambiente virtuale, e qualcuno ha accettato tale proposta. Tuttavia, il tempo impiegato nella fruizione "extra" non è stato sommato ai minuti cronometrati durante lo svolgimento del *test*.

Al fine di trovare aspetti interessanti a partire dai dati ripuliti e donare una quantificazione ai risultati ottenuti, è stato scelto di selezionare alcune delle variabili e di incrociarne i dati, in modo da osservare alcuni fenomeni di dipendenza (se esistenti) tra loro.

6.3.1 Correlazioni

In questa sezione verranno analizzate alcune delle correlazioni[116] più interessanti tra le variabili presentate, utili ad evidenziare alcuni aspetti di rilievo a partire dai dati raccolti. Per definizione, il coefficiente utilizzato per determinare se esiste o meno una correlazione è una misura che serve a quantificare la forza della relazione lineare tra due variabili. Esso rappresenta un valore (*r*) senza unità di misura, collocabile nell'intervallo delimitato da -1 o 1; più *r* si avvicina allo zero, più la correlazione è debole; al contrario, più rasenta l'1, più essa è forte e quindi, di conseguenza, è possibile osservare una crescita parallela delle due variabili studiate. Infine, se *r* assume un valore inferiore allo 0, significa che vi è una correlazione negativa che è denotata dalle variabili che vanno in direzione opposta e, ad esempio, all'aumento della prima corrisponde una diminuzione della seconda.

6.3.1.1 Età – Livello di esperienza con la RV

Il primo aspetto che ha destato curiosità e si è, dunque, ritenuto utile esaminare è stata la possibile esistenza di un nesso in grado di correlare l'età dei partecipanti al loro livello (autovalutato) di esperienza con la RV. Senza troppe astrazioni in merito, l'attesa sarebbe quella di un forte nesso tra le due variabili, poiché solitamente la Realtà Virtuale tocca quasi

[116] Ogni calcolo statistico è stato effettuato tramite il software Microsoft Excel.

esclusivamente i giovani; tuttavia, i dati hanno offerto un quadro leggermente differente dalle aspettative.

Correlazione: -0.2702848

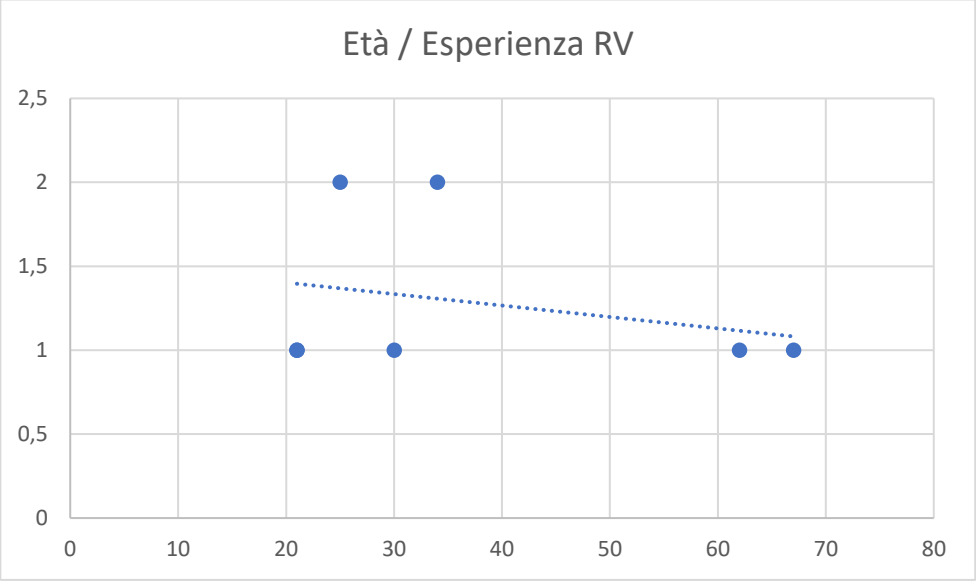

Grafico 1. Correlazione Età/Esperienza RV

Come è possibile osservare dal grafico (con sulle ascisse l'età e, sulle ordinate, il livello di esperienza, espresso in valori da 1 a 5), le due variabili sono legate tra loro da una proporzione inversa: il valore della correlazione è infatti leggermente sotto lo 0 e denota, dunque, una lieve dipendenza inversa tra le stesse. Tracciando la linea di tendenza, è possibile notare che, all'aumentare dell'età, diminuisce drasticamente il livello di esperienza nel campo della Realtà Virtuale. Tuttavia, si osserva un'eccezione alla tendenza, poiché sono presenti due valori non in linea con l'andamento generale: questi, relativi ai due utenti di 21 anni, servono da discrimine per mettere in evidenza un importante aspetto. Quest'ultimo riguarda il mondo della Realtà Virtuale e della sua settorialità: non è facile trovare utenti giovani e con una buona conoscenza in tale campo e l'esperimento condotto ne rappresenta una prova. Allo stesso modo, non essendo stati coinvolti esperti del settore, non sono stati evidenziati fenomeni caratterizzati da una forte correlazione tra le due variabili.

6.3.1.2 Età – Conoscenza del problema delle Barriere Architettoniche

Allo stesso modo della precedente, l'analisi riguardante la possibile correlazione tra età e conoscenza inerente le barriere architettoniche e i problemi ad esse connessi ha messo in luce

aspetti interessanti. Di seguito è presente il valore della correlazione tra le due variabili e il relativo grafico, con annessa la linea di tendenza:

Correlazione: 0.8939426

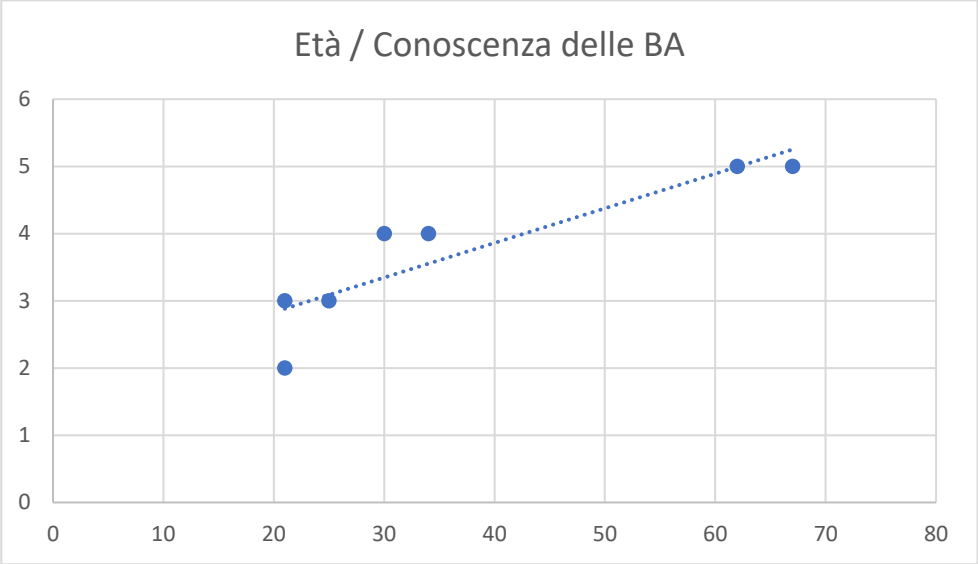

Grafico 2. Correlazione Età / Conoscenza delle BA

In questo caso, il valore della correlazione è vicino a 1: come anche evidenziato dalla linea di tendenza, le due variabili sono dipendenti tra loro e presentano un andamento ascendente; tale fenomeno è influenzato da due fattori, che verranno di seguito esplicitati. Il primo deriva dal campione di utenti: essendo stati coinvolte ben tre persone operanti nel campo della didattica, tra i quali due docenti interessati in progetti riguardanti proprio la tematica esaminata, i risultati hanno subito l'influsso di valori spiccatamente alti relativi a tali soggetti, appartenenti alla fascia d'età più avanzata. Il secondo, deriva dalla tendenza media dei ragazzi nell'essere poco informati sul problema delle barriere architettoniche: come è stato, infatti, più volte analizzato nei primi capitoli, le campagne di sensibilizzazione inerenti tale campo possono risultare talvolta noiose o troppo settoriali per essere raggiunte dai più giovani.

6.3.1.3 Tempo – Complessità di gioco

Uno degli aspetti più interessanti analizzati è il rapporto tra il tempo impiegato per il superamento di ogni livello previsto nel simulatore e la complessità riscontrata in ogni fase di gioco. Per poter esaminare insieme le due variabili, è stato però necessario dapprima analizzarle singolarmente; a tal proposito, sono state calcolate la media e la deviazione

standard dei valori relativi ad ognuna di esse, in relazione ad ogni livello. Quest'ultimo indice è stato inserito in modo da poter osservare anche la dispersione statistica (evidenziata nei grafici) dei dati in merito ai valori medi calcolati; tuttavia, tale parametro non è utile ai fini dello studio sulla forza che lega le due variabili.

Tempo

Tempo	**Livello 1**	**Livello 2**	**Livello 3**
Media	17,14285714	24	16,57142857
Deviazione standard	4,670066789	4,932882862	3,90969491

Tabella 7. Dati relativi al tempo di gioco

Complessità di gioco

Complessità	**Livello 1**	**Livello 2**	**Livello 3**
Media	3,428571429	3,857142857	2,857142857
Deviazione standard	0,975900073	0,899735411	1,069044968

Tabella 8. Dati relativi alla complessità di gioco

Come è possibile osservare dai valori riportati, relativi alle tre parti del simulatore, i livelli 1 e 3 hanno richiesto, in media, meno tempo del 2 per essere portati a termine[117]. Tale fenomeno si è verificato poiché il secondo livello include alcune situazioni difficili da portare a termine, poiché si riveste il ruolo di un giocatore con disabilità motorie che, costretto su carrozzine, deve affrontare un mondo ove sono previste numerose insidie dovute alle barriere architettoniche. Al contrario, il primo e il terzo sono stati più veloci da completare, per due differenti ragioni, tra le quali spicca la facilità di compimento delle azioni richieste nel livello 1 (nel quale si impersonifica una persona libera di muoversi e di deambulare senza particolari problemi) e, nel 3, la presenza dell'automazione interna al gioco e di una casa creata su scala del *player*. Ad influire sulla rapidità in merito al terzo, è anche un aspetto riguardante l'apprendimento utente: è infatti noto che un'azione, se ripetuta, richiede sempre meno tempo se completata con successo più di una volta. Tale ragionamento è in linea con lo sviluppo del simulatore, nel quale sono previsti più livelli che condividono lo stesso ambiente, accessibili soltanto se il precedente è stato completato: il fruitore arriva a provare l'ultimo (il terzo)

[117] Fortunatamente, nessun partecipante ha deciso di abbandonare la prova o si è bloccato durante lo svolgimento delle missioni, dunque i dati non presentano punti di rottura o valori sommari alterati dovuti a tali evenienze.

quando è già preparato sulle aspettative e, di conseguenza, tende a finirlo più velocemente. Come è possibile notare dal valore della correlazione e dai grafici sotto riportati, i dati inerenti il tempo e la complessità di gioco sono perfettamente assimilabili e presentano un'alta correlazione tra di loro:

Correlazione – Livello 1: 0,679149867

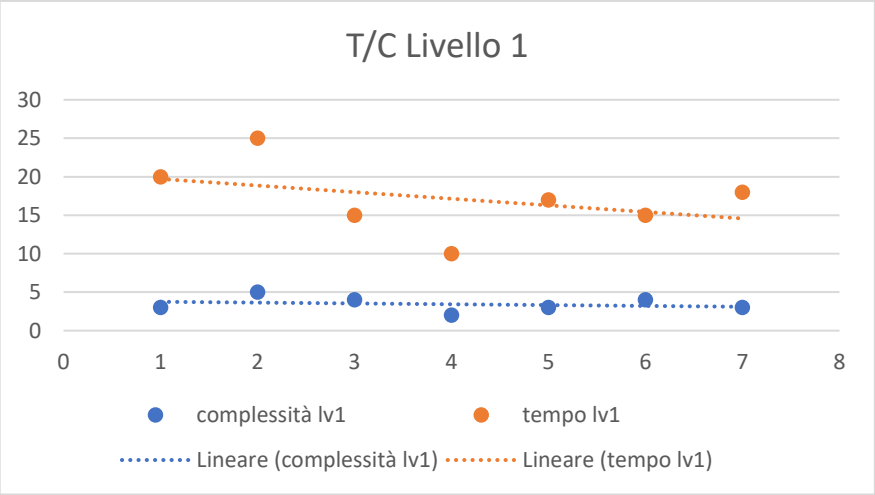

Grafico 3. Correlazione Tempo/Complessità - Livello 1

Correlazione – Livello 2: 0,600832085

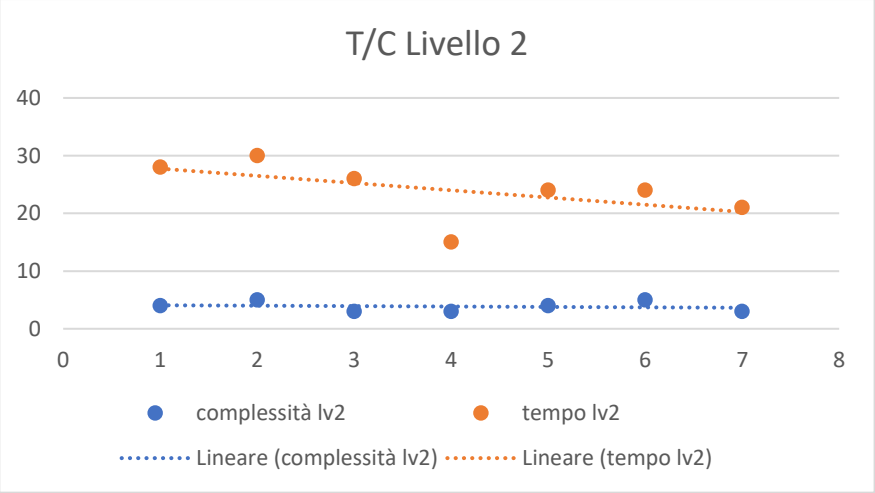

Grafico 4. Correlazione Tempo/Complessità - Livello 2

Correlazione – Livello 3: 0,700675575

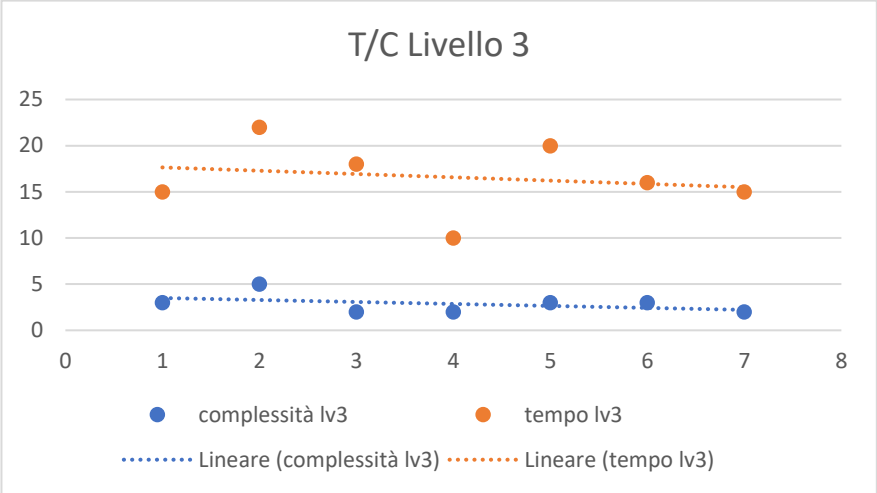

Grafico 5. Correlazione Tempo/Complessità - Livello 3

Alla luce dei valori e dei grafici riportati, è possibile dunque giustificare quanto appena scritto: tempo e complessità di gioco, espressi in valori medi, sono due variabili dipendenti tra loro e confermano le aspettative inerenti il caso, già ipotizzate durante la fase di sviluppo del simulatore.

6.3.1.4 Età – Livello di relax

L'ultimo tipo di studio condotto sulle correlazioni è quello riguardante l'età dei partecipanti e il livello di rilassamento (*relax*) manifestato durante lo svolgimento della prova. Tale variabile è l'unica ad essere stata prelevata, a questo scopo, dai risultati in merito alle impressioni generali, poiché rappresenta quella in grado di fornire deduzioni più significative inerenti l'analisi.

Di seguito sono riportati il valore della correlazione dei dati sull'età dei partecipanti e il relativo livello di *relax* manifestato durante la fruizione del simulatore e il grafico ad esso inerente:

Correlazione: -0,578842647

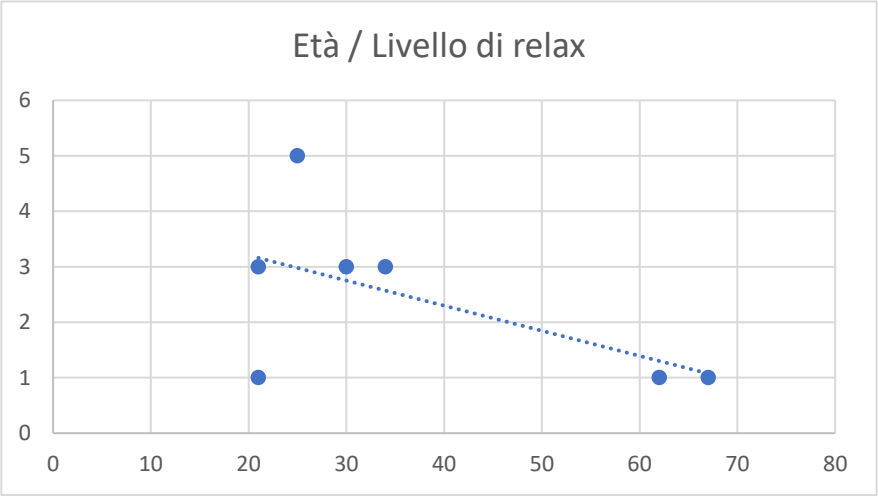

Grafico 6. Correlazione Età/Livello di relax

Il valore della correlazione tra le due variabili è negativo: il fenomeno si verifica poiché l'età, distribuita – come di consueto – sulle ordinate, presenta una tendenza del tutto opposta rispetto al livello di *relax*. Tale numero rispecchia perfettamente le attese: come è facile intuire, una persona di età più avanzata tende ad innervosirsi più facilmente di una giovane, se posta ad interfacciarsi con uno strumento di natura informatica che richiede una certa dimestichezza d'utilizzo. È infatti noto che i non nativi digitali, o comunque quei soggetti che hanno visto la loro giovinezza svilupparsi prima del *boom* informatico che ha caratterizzato gli anni Novanta e i primi Duemila, siano più inclini ad essere ostili e a demonizzare le più moderne tecnologie. Durante il *test*, gli utenti più grandi hanno avuto bisogno di fermarsi più volte e hanno necessitato di qualche *input* (pur non essendo stati aiutati, per non invalidare i risultati) per procedere nelle missioni; in concomitanza con tali momenti, essi hanno manifestato (in particolare, il soggetto con ID 2) sentimenti di frustrazione e disagio, che fortunatamente però non hanno condotto all'interruzione della prova.

6.3.2 Feedback utenti e impressioni generali

Grazie all'ultima sezione del questionario, nonché all'osservazione diretta degli utenti durante lo svolgimento delle prove, è possibile delineare, in linee generali, le impressioni ottenute in merito alla loro esperienza.

Livello di immersività

La prima domanda relativa alla sezione del questionario riguardante le impressioni generali degli utenti chiede loro di esprimere, con un punteggio da 1 a 5, quanto siano riusciti ad immedesimarsi nel giocatore protagonista di ciascuna esperienza prevista dal simulatore. In linea di massima, tutti i partecipanti hanno espresso un punteggio di 3 o 4, ma nessuno ha ammesso di essersi sentito pienamente coinvolto – emotivamente e sul piano sensoriale – durante lo svolgimento della prova. Tuttavia, considerando che non si tratta di uno strumento professionale, i dati sono sufficienti a denotare questo come possibile punto di forza.

Livello di rilassamento

Il livello di relax, già citato in merito all'analisi delle correlazioni, è pienamente in linea con le aspettative, come accennato e c'è da sottolineare quanto alcuni utenti, soprattutto i più anziani, manifestassero disagio durante la simulazione, provati anche da alcune spiacevoli malesseri tipici di chi si interfaccia per la prima volta al mondo della RV.

Tentazione di abbandonare il gioco

Seppur qualche utente abbia avuto bisogno di alcuni *input* per poter proseguire, nessun partecipante ha deciso di abbandonare la prova; tuttavia, due di loro (ID 2 e ID 6) hanno descritto, inserendo i valori 5 e 4 nella scala proposta, di essere stati molto tentati. In particolare, il partecipante di 67 anni ha ripetutamente manifestato una sensazione di disagio, ma ha resistito fino al termine dell'ultimo livello.

Sensazioni fisiche sgradevoli

Come da aspettative su coloro che si interfacciano per la prima volta con la RV, o non sono abituati a fruire di strumenti non ancora ottimizzati per evitare totalmente l'influsso negativo dei contenuti sullo spettro sensoriale, tre partecipanti all'indagine su sette hanno provato sensazioni fisiche sgradevoli durante e in seguito al *test*. I sintomi più lamentati sono stati nausea, vertigini, tremori e stanchezza oculare; tuttavia, la totalità degli utenti colpiti sono riusciti a stare immediatamente meglio in seguito ad alcuni suggerimenti proposti, quali fare brevi pause durante l'esecuzione respirando profondamente e, nel caso degli utenti più anziani, di toccare a intervalli regolari alcuni oggetti circostanti, per non perdere il contatto con la realtà. L'utente numero 3, durante il *test*, ha mostrato un comportamento totalmente

inatteso: nonostante le difficoltà riscontrate nell'avanzamento nei livelli proposti, si è mostrato particolarmente divertito dalla situazione e non ha provato sensazioni spiacevoli.

Al termine di ogni *test*, a ciascun utente è stato suggerito di riposare gli occhi, assumere zuccheri e restare per un po' di tempo all'aria aperta, per eliminare ogni possibile traccia di malessere dovuto a fenomeni quali la *motion sickness*.

Integrazione dei portatori di disabilità motoria vs abbattimento delle barriere architettoniche

La terza richiesta di tale sezione riguarda l'espressione di un'opinione in merito all'efficacia dell'abbattimento delle barriere architettoniche, a favore dell'integrazione sociale dei portatori di disabilità motoria. Tutti i partecipanti allo studio hanno risposto presentando un punteggio ottimo (con valori di 4 e 5 della *scala di Likert*) in merito alla questione: tutti sono stati d'accordo sul miglioramento che tale strategia può apportare a tali soggetti con difficoltà riguardanti lo spettro motorio.

Domotica vs autonomia

Il quesito riguardante l'impatto positivo di strumenti di domotica sull'autonomia di persone con disabilità motorie ha riportato, in linea con quello precedente, buone impressioni: tutti gli utenti sono d'accordo sull'efficacia di tali tecnologie sul campo in esame. Tuttavia, in riferimento alla domanda successiva, che richiede se queste ultime possano sostituire totalmente la figura umana del *caregiver*, i risultati sono stati tutti espressi da un punteggio basso: nessuno è stato d'accordo sull'eliminazione di un assistente fisico a favore di uno totalmente digitale o di un insieme di strumenti atti a rimpiazzarlo.

Suggerimenti per il simulatore

Tre utenti hanno proposto di integrare delle migliorie, in una possibile futura versione del simulatore; in particolare, il primo ha suggerito di inserire un sollevatore a corredo dell'auto presente nel terzo livello, il secondo ha espresso alcuni disagi relativi alla poca ottimizzazione dello strumento in merito all'eliminazione dei disturbi fisici dovuti al movimento, il terzo ha richiesto di integrare un sistema di controllo vocale nel simulatore (idea presente nelle versioni precedenti, ma abbandonata per lasciare spazio all'interazione manuale nella RV).

In linee generali, lo *user study* effettuato ha condotto ad una raccolta dati più o meno organica, sulla quale è stato possibile effettuare qualche interessante studio; tuttavia, se fossero stati coinvolti più partecipanti, sarebbe stato attuabile (nonché più sensato) preparare più domande affini all'esperienza di gioco e, di conseguenza, analizzare ulteriori parametri che, nel caso di un numero ridotto di utenti – si veda il campo "Professione" – non sono risultati sfruttabili per mancanza di dati disomogenei[118].

[118] Il campione di utenti raccolto è risultato assai omogeneo sul piano "Professione": tra sette partecipanti, erano presenti tre operatori scolastici, tre studenti e un biologo.

Conclusioni e sviluppi futuri

Lo sviluppo del simulatore di Realtà Virtuale immersiva presentato si pone come proposta per incentivare l'impiego di tecniche alternative nel campo della sensibilizzazione, al fine di coadiuvarne l'efficacia. Come già accennato in più punti dell'elaborato, la RV rappresenta un mondo piuttosto settoriale; tuttavia, il costante abbassamento dei prezzi di produzione, che caratterizza il naturale decorso del progresso tecnologico – entro il quale i prodotti divengono obsoleti ad un ritmo sempre più veloce – , sta permettendo a molte più persone di acquistare prodotti dedicati, quali visori ed accessori ad essi correlati. Grazie proprio a questo fenomeno, è facile aspettarsi tra pochi anni la RV impiegata sempre in più settori; un tale adattamento potrebbe condurre, in tempi più o meno rapidi, all'impiego su larga scala di tali tecniche e strumenti, che entrerebbero a far parte della quotidianità umana. Sebbene tale visione conduca facilmente al pensiero di una realtà distopica, allo stesso modo di scenari protagonisti di contenuti in stile *cyberpunk*, la diffusione della RV non è da vedersi come un'azione distruttiva atta ad alterare la percezione umana, ma come una naturale estensione dei sensi, che permette all'uomo di vedere e "toccare" in modo realistico ciò che gli è difficile (se non impossibile) incontrare nella realtà. Il simulatore è progettato proprio partendo da quest'idea: dotare l'essere umano di una sorta di *plugin*, in grado di consentirgli di immergersi nel ruolo di più personaggi-tipo, al fine di permettergli di provare differenti punti di vista e di coinvolgere l'aspetto sensoriale il più possibile di fronte a determinati scenari. Nel caso in questione, lo strumento si pone l'obiettivo di sensibilizzare sulle tematiche inerenti il mondo delle barriere architettoniche, in relazione ad alcuni tipi di disabilità motoria che impediscono la deambulazione ai soggetti che ne sono portatori; in seguito agli studi condotti in merito al caso e all'effettuazione del *test*, è possibile giungere alla conclusione che tale stratagemma risulta efficace nella diffusione del messaggio. Tuttavia, il prodotto necessita di alcuni accorgimenti di progettazione: per un possibile impiego su larga scala, andrebbe ottimizzato sotto diversi aspetti, tra i quali spiccano la stabilizzazione del movimento *in-game* (in alcuni casi, causa di *motion sickness*) e il miglioramento di alcune interazioni con gli oggetti dell'ambiente virtuale, che presentano talvolta qualche artefatto. Al fine di valutare meglio l'esperienza in merito alla fruizione dello stesso, sarebbe necessario coinvolgere più utenti nei *test*, in particolare, in previsione di eventuali impieghi pratici sul campo (ad esempio, in itinere di un'iniziativa a favore dell'abbattimento delle

barriere architettoniche o ad una immissione sul mercato), poiché è improponibile somministrare in maniera non controllata un prodotto che può disturbare fisicamente e suscitare sentimenti di frustrazione. Perfezionando il livello 3, integrando misure in scala ancora più fedeli alla realtà, il simulatore potrebbe, inoltre, fungere da strumento di supporto per gli operatori nel campo dell'*interior design*; esistono, tuttavia, prodotti specifici in merito e adattarne le caratteristiche per l'assoluzione a tale scopo finirebbe per trasformarlo in uno dei tanti strumenti simili, già ampiamente radicati sul mercato. Uno dei possibili sviluppi attuabili, al fine di rendere il prodotto maggiormente fruibile a più fasce di videogiocatori che acquistano i propri dispositivi per l'accesso alla RV, sarebbe di produrre più versioni del simulatore, in base alla piattaforma di destinazione; con l'avvento dei moderni dispositivi *standalone*, come l'Oculus Quest e Quest II, si va sempre più verso l'elaborazione di contenuti in grado di essere eseguiti senza la mediazione di un supporto esterno (PC, *console*, ecc). Tuttavia, tale idea richiederebbe un solido lavoro di squadra per essere attuata, poiché non si tratta di operazioni immediate da effettuare, ma implicherebbe la reimplementazione di buona parte dei contenuti e, al fine di avere successo, necessiterebbe di un ampio campione di utenti per le fasi di *testing* post-produzione, operazioni dispendiose in termini di fondi e risorse. Infine, potrebbe essere necessario applicare una licenza alle *build* del simulatore e al relativo codice sorgente, al fine di regolamentarne la distribuzione; l'intenzione è di impiegare quella sostenuta dalla Free Software Foundation, la GNU GPL[119], ma sono da rivedere attentamente le clausole relative all'attribuzione, poiché il simulatore include contenuti non liberi all'origine e, a loro volta, sottoposti a licenza proprietaria. Per consentirne l'accesso a tutti coloro volessero provarlo, resta comunque viva l'intenzione di rilasciare gratuitamente le versioni dello stesso, almeno in questa fase di lancio: è da ricordare che il prodotto è progettato per sensibilizzare su una tematica seria e piuttosto dibattuta e, la gratuità, costituisce sicuramente un incentivo allo scopo al quale, lo stesso, mira.

[119] GNU GPL. Link al sito web: https://www.gnu.org/licenses/licenses.it.html.

Appendice

Nelle seguenti pagine è riportato il codice incluso al simulatore presentato. Sono escluse dalla presente appendice le librerie di base di Unity non utilizzate o che non richiedono di essere riportate ai fini della comprensione della fase di implementazione dello strumento.

Il codice che, per motivi di spazio, non è stato incluso, è possibile trovarlo al *repository* del progetto su GitHub di seguito riportato, insieme al file dell'ultima build per Windows:

https://github.com/CZamberti/MARVIN_LATEST_0521

abilitaCapSucc.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class abilitaCapSucc : MonoBehaviour
{
    public GameObject capitoloSuccessivo;
    public GameObject pannelloMissioneI;
    public GameObject pannelloMissioneII;
    public bool abilitato=false;
    void Update(){
        if(GetComponent<termine_capitolo>().enabled==true){
            if(abilitato==false){
                capitoloSuccessivo.SetActive(true);
                pannelloMissioneI.SetActive(false);
                pannelloMissioneII.SetActive(true);
                abilitato=true;
            }
        }
    }
}
```

accendiFornello.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class accendiFornello : MonoBehaviour
{
    public AudioSource avvio_fornello;
    public GameObject fornelloAcceso;
    public bool fornelloON=false;
    void Start(){

    }
     void OnTriggerEnter(Collider other){

        if(other.tag=="manoSN" || other.tag=="manoDX"){
            if(fornelloON==false){
                    avvio_fornello.Play();
                    fornelloAcceso.SetActive(true);
                    transform.Rotate(0.0f,120.0f, 0.0f);
                    fornelloON=true;
            }else{
                avvio_fornello.Stop();
```

```
                fornelloAcceso.SetActive(false);
                transform.Rotate(0.0f,-120.0f, 0.0f);
                fornelloON=false;

            }
        }
    }

}
```

cambiaScena.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
public class cambiaScena : MonoBehaviour
{
   [SerializeField]
   private string livello;
   public GameObject overlay;
   public bool fatto=false;
   void Start(){
   }
   public void Update(){
          if(fatto==false){
          StartCoroutine(VaiAScenaDopoOverlay());
          fatto=true;
          }
   }
   //coroutine per attivazione dello screen overlay e cambio scena
   private IEnumerator VaiAScenaDopoOverlay(){
          overlay.SetActive(true);
          yield return new WaitForSeconds(2f);
          SceneManager.LoadScene(livello);
   }
}
```

caricaEdisattiva.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class caricaEdisattiva : MonoBehaviour{

    public bool svegliato=false;
    // Update is called once per frame
    void Awake(){
        if (svegliato==false){
             this.gameObject.SetActive(false);
             svegliato=true;
        }
    }
}
```

chiamata_aiutante.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

```
public class chiamata_aiutante : MonoBehaviour
{

    public bool chiamata_fatta=false;
    public GameObject pannelloChiamata;
    public GameObject fadeNero;
    public Transform rigDis;
    public Transform puntoArrivo;
    // Start is called before the first frame update
    void Start()
    {

    }
    //avvia la coroutine per la chiamata dell'assistente al tocco
    void OnTriggerEnter(Collider other)
    {
        if(other.tag=="manoSN"||other.tag=="manoDX"){
            StartCoroutine(avvioChiamata());
            chiamata_fatta=true;
        }
    }
    private IEnumerator avvioChiamata()
    {
        pannelloChiamata.SetActive(true);
        fadeNero.SetActive(true);
        yield return new WaitForSeconds(3f);
        rigDis.position=puntoArrivo.position;
        yield return new WaitForSeconds(2f);
        pannelloChiamata.SetActive(false);
        fadeNero.SetActive(false);
    }
}
```

ciboprontо_piatto.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class spegniFuocoAutodopoC : MonoBehaviour
{
    private bool missioneCompletata=false;
    public GameObject manopola;
    public GameObject EffettoFuoco;
    public GameObject fornello;
    //public GameObject[] punti_contatto;

    void Start(){
         if(missioneCompletata==false){
            StartCoroutine("Timer");
            missioneCompletata=true;

        }else{
            StopCoroutine("Timer");
        }
    }
    public IEnumerator Timer(){
        yield return new WaitForSeconds(3f);
        manopola.transform.Rotate(0.0f,-120.0f, 0.0f);
        EffettoFuoco.SetActive(false);
        fornello.GetComponent<cottura>().enabled=false;
        GetComponent<missioneCompiutaSimple>().enabled=true;

    }
}
```

commutaRoomba.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.AI;
public class commutaRoomba : MonoBehaviour
{
    private NavMeshAgent agent;
    public GameObject roomba;
    public GameObject led;
    public GameObject pulsante_smart;
    public GameObject interruttore;
    public GameObject controlloStato;
    public Sprite spriteOn;
    public Sprite spriteOff;
    public Material materiale_int_off;
    public Material materiale_int_on;
    public Transform puntoArrivo;
    public GameObject[] animazioneSpazzole;
    public AudioSource suonoAvvio;
    public AudioSource suonoIdle;

    // Start is called before the first frame update
    void Start(){
        agent = roomba.GetComponent<NavMeshAgent>();
    }
    void OnTriggerEnter(Collider other){
        if (other.tag=="dito"||other.tag=="ditoDX"){
            if(controlloStato.activeSelf==false){
                roomba.transform.Rotate(0, 180, 0, Space.Self);
                roomba.GetComponent<wanderingAi>().enabled=true;
                agent.enabled=true;
                interruttore.GetComponent<MeshRenderer>().material = materiale_int_on;
                led.GetComponent<MeshRenderer>().material = materiale_int_on;
                StartCoroutine(SuoniRoomba());
                controlloStato.SetActive(true);
                pulsante_smart.GetComponent<SpriteRenderer>().sprite = spriteOn;
                foreach (GameObject a in animazioneSpazzole){
                    a.GetComponent<Animator>().enabled=true;
                }
            }else{
                interruttore.GetComponent<MeshRenderer>().material = materiale_int_off;
                led.GetComponent<MeshRenderer>().material = materiale_int_off;
                suonoAvvio.Play();
                suonoIdle.Stop();
                roomba.GetComponent<wanderingAi>().enabled=false;
                agent.enabled=false;
                roomba.transform.position=puntoArrivo.position;
                controlloStato.SetActive(false);
                pulsante_smart.GetComponent<SpriteRenderer>().sprite = spriteOff;
                foreach (GameObject a in animazioneSpazzole){
                    a.GetComponent<Animator>().enabled=false;
                }


            }
        }
    }

      public IEnumerator SuoniRoomba(){
        suonoAvvio.Play();
        yield return new WaitForSeconds(2f);
        suonoIdle.Play();

    }
}
```

ControlloFinestraGhigliottina_VR.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class ControlloFinestraGhigliottina_VR : MonoBehaviour{
    public bool buttonPressed = false;
    [SerializeField] private Vector3 PressioneBottone;
    public bool buttonUnpressed = false;
    public bool buttonDown = false;
    public float buttonPressDuration;
    private float buttonPressedTime;
    private bool buttonImmediatelyPressed;  // used to hold onto the pressed state for one f
rame.
    private bool buttonImmediatelyUnpressed;  // used to hold onto the pressed state for one
 frame.
    public Animation apriChiudi;
    public AudioSource suonoOp;
    public AudioSource suonoCl;
    public GameObject controlloFinestra;
    public GameObject pulsante_smart;
    public Sprite spriteOn;
    public Sprite spriteOff;
  void Update() {
        if (this.buttonDown) {
            // Only hold button pressed for one frame
            if (this.buttonPressed && !this.buttonImmediatelyPressed) {
                this.buttonPressed = false;
            }
            this.buttonImmediatelyPressed = false;

            if (Time.time - this.buttonPressedTime > buttonPressDuration) {
                this.transform.position += this.transform.TransformDirection(PressioneBotton
e);
                this.buttonDown = false;
                this.buttonImmediatelyUnpressed = true;
            }
        } else {
            if (this.buttonImmediatelyUnpressed) {
                this.buttonUnpressed = true;
                this.buttonImmediatelyUnpressed = false;
            } else {
                this.buttonUnpressed = false;
            }
        }
    }
   private void OnTriggerEnter(Collider other) {
        this.buttonPressedTime = Time.time;
            if (this.buttonDown) {
                return;
            }
        if(other.tag=="dito"|| other.tag=="ditoDX"){
            if (controlloFinestra.active==false){
                apriChiudi.Play("Open");
                suonoOp.Play();
                pulsante_smart.GetComponent<SpriteRenderer>().sprite = spriteOn;
                controlloFinestra.SetActive(true);

            }
            else{
                apriChiudi.Play("Close");
                suonoCl.Play();
                pulsante_smart.GetComponent<SpriteRenderer>().sprite = spriteOff;
                controlloFinestra.SetActive(false);
            }
        }
        this.transform.position += this.transform.TransformDirection(-PressioneBottone);
        this.buttonImmediatelyPressed = true;
        this.buttonPressed = true;
        this.buttonDown = true;
    }
}
```

ifRigDisAbilitaInterruttoreChiamata.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ifRigDisAbilitaInterruttoreChiamata : MonoBehaviour
{

    public GameObject[] IstanzeRigDis;
    public GameObject bottoni_assistenza;
    void Start(){
    IstanzeRigDis = GameObject.FindGameObjectsWithTag("Player_dis");
        foreach (GameObject g in IstanzeRigDis){
            if (g.activeSelf==true){
                bottoni_assistenza.SetActive(true);
            }
            Destroy(this.GetComponent<ifRigDisAbilitaInterruttoreChiamata>());
        }
    }
}
```

levaTrigger.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class levaTrigger : MonoBehaviour
{
    public float levaON;
    public float levaOFF;
    public bool leverIsOn = false;
    public bool effettoOn = false;
    public bool leverWasSwitched = false;
    private HingeJoint leverHingeJoint;
    public GameObject Effetto;
    public AudioSource suonoAcqua;
    private Vector3 startingEuler;
    void Start(){
        leverHingeJoint = GetComponent<HingeJoint>();

    }
    // Update is called once per frame
    void Update()
    {
        //Debug.Log("LEVA ASSE"+leverHingeJoint.axis);
        leverWasSwitched = false;
        float offDistance = Quaternion.Angle(this.transform.localRotation, OffHingeAngle());
        float onDistance = Quaternion.Angle(this.transform.localRotation, OnHingeAngle());
    //controllo la posizione della leva e se deve essere alzata o abbassata
        bool shouldBeOn = (Mathf.Abs(onDistance) < Mathf.Abs(offDistance));
        if (shouldBeOn != leverIsOn) {
            leverIsOn = !leverIsOn;
            if (leverIsOn){
             Effetto.SetActive(true);
             suonoAcqua.Play();

            }else{
                Effetto.SetActive(false);
                suonoAcqua.Stop();
            }
            leverWasSwitched = true;
        }
    }
private Quaternion OnHingeAngle() {
        return Quaternion.Euler(this.leverHingeJoint.axis * levaON + startingEuler);
```

```
    }

    private Quaternion OffHingeAngle() {
        return Quaternion.Euler(this.leverHingeJoint.axis * levaOFF + startingEuler);
    }
}
```

Lightsaber.cs

```
using Assets.Scripts;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Lighsaber : MonoBehaviour
{
    //The number of vertices to create per frame
    private const int NUM_VERTICES = 12;
    [SerializeField]
    [Tooltip("The blade object")]
    private GameObject _blade = null;
    [SerializeField]
    [Tooltip("The empty game object located at the tip of the blade")]
    private GameObject _tip = null;
    [SerializeField]
    [Tooltip("The empty game object located at the base of the blade")]
    private GameObject _base = null;
    [SerializeField]
    [Tooltip("The mesh object with the mesh filter and mesh renderer")]
    private GameObject _meshParent = null;
    [SerializeField]
    [Tooltip("The number of frame that the trail should be rendered for")]
    private int _trailFrameLength = 3;
    [SerializeField]
    [Tooltip("The amount of force applied to each side of a slice")]
    private float _forceAppliedToCut = 3f;
    private Mesh  mesh;
    private Vector3[] _vertices;
    private int[] _triangles;
    private int _frameCount;
    private Vector3 _previousTipPosition;
    private Vector3 _previousBasePosition;
    private Vector3 _triggerEnterTipPosition;
    private Vector3 _triggerEnterBasePosition;
    private Vector3 _triggerExitTipPosition;
    void Start()
    {
        //Init mesh and triangles
        _meshParent.transform.position = Vector3.zero;
        _mesh = new Mesh();
        _meshParent.GetComponent<MeshFilter>().mesh = _mesh;
        Material trailMaterial = Instantiate(_meshParent.GetComponent<MeshRenderer>().shared
Material);
        _meshParent.GetComponent<MeshRenderer>().sharedMaterial = trailMaterial;
        Material bladeMaterial = Instantiate(_blade.GetComponent<MeshRenderer>().sharedMater
ial);
        _blade.GetComponent<MeshRenderer>().sharedMaterial = bladeMaterial;
        _vertices = new Vector3[_trailFrameLength * NUM_VERTICES];
        _triangles = new int[_vertices.Length];
        //Set starting position for tip and base
        _previousTipPosition = _tip.transform.position;
        _previousBasePosition = _base.transform.position;
    }
    void LateUpdate()
    {
        //Reset the frame count one we reach the frame length
        if(_frameCount == (_trailFrameLength * NUM_VERTICES))
        {
            _frameCount = 0;
        }
```

```
        //Draw first triangle vertices for back and front
        _vertices[_frameCount] = _base.transform.position;
        _vertices[_frameCount + 1] = _tip.transform.position;
        _vertices[_frameCount + 2] = _previousTipPosition;
        _vertices[_frameCount + 3] = _base.transform.position;
        _vertices[_frameCount + 4] = _previousTipPosition;
        _vertices[_frameCount + 5] = _tip.transform.position;
        //Draw fill in triangle vertices
        _vertices[_frameCount + 6] = _previousTipPosition;
        _vertices[_frameCount + 7] = _base.transform.position;
        _vertices[_frameCount + 8] = _previousBasePosition;
        _vertices[_frameCount + 9] = _previousTipPosition;
        _vertices[_frameCount + 10] = _previousBasePosition;
        _vertices[_frameCount + 11] = _base.transform.position;
        //Set triangles
        _triangles[_frameCount] = _frameCount;
        _triangles[_frameCount + 1] = _frameCount + 1;
        _triangles[_frameCount + 2] = _frameCount + 2;
        _triangles[_frameCount + 3] = _frameCount + 3;
        _triangles[_frameCount + 4] = _frameCount + 4;
        _triangles[_frameCount + 5] = _frameCount + 5;
        _triangles[_frameCount + 6] = _frameCount + 6;
        _triangles[_frameCount + 7] = _frameCount + 7;
        _triangles[_frameCount + 8] = _frameCount + 8;
        _triangles[_frameCount + 9] = _frameCount + 9;
        _triangles[_frameCount + 10] = _frameCount + 10;
        _triangles[_frameCount + 11] = _frameCount + 11;
        _mesh.vertices = _vertices;
        _mesh.triangles = _triangles;
        //Track the previous base and tip positions for the next frame
        _previousTipPosition = _tip.transform.position;
        _previousBasePosition = _base.transform.position;
        _frameCount += NUM_VERTICES;
    }
    private void OnTriggerEnter(Collider other)
    {
        if (other.tag=="dadini"){
            _triggerEnterTipPosition = _tip.transform.position;
            _triggerEnterBasePosition = _base.transform.position;
        }
    }
    private void OnTriggerExit(Collider other)
    {
        if (other.tag=="dadini"){
            _triggerExitTipPosition = _tip.transform.position;
            //Create a triangle between the tip and base so that we can get the normal
            Vector3 side1 = _triggerExitTipPosition - _triggerEnterTipPosition;
            Vector3 side2 = _triggerExitTipPosition - _triggerEnterBasePosition;
            //Get the point perpendicular to the triangle above which is the normal
            //https://docs.unity3d.com/Manual/ComputingNormalPerpendicularVector.html
            Vector3 normal = Vector3.Cross(side1, side2).normalized;
            //Transform the normal so that it is aligned with the object we are slicing's tr
ansform.
            Vector3 transformedNormal = ((Vector3)(other.gameObject.transform.localToWorldMa
trix.transpose * normal)).normalized;
            //Get the enter position relative to the object we're cutting's local transform
            Vector3 transformedStartingPoint = other.gameObject.transform.InverseTransformPo
int( triggerEnterTipPosition);
            Plane plane = new Plane();
            plane.SetNormalAndPosition(
                    transformedNormal,
                    transformedStartingPoint);
            var direction = Vector3.Dot(Vector3.up, transformedNormal);
            //Flip the plane so that we always know which side the positive mesh is on
            if (direction < 0)
            {
                plane = plane.flipped;
            }
            GameObject[] slices = Slicer.Slice(plane, other.gameObject);
            Destroy(other.gameObject);
            Rigidbody rigidbody = slices[1].GetComponent<Rigidbody>();
            Vector3 newNormal = transformedNormal + Vector3.up * _forceAppliedToCut;
```

```
            rigidbody.AddForce(newNormal, ForceMode.Impulse);
        }
    }
}
```

LocomotionController.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.XR.Interaction.Toolkit;

public class LocomotionController : MonoBehaviour
{
    public XRController leftTeleportRay;
    public XRController rightTeleportRay;
    public InputHelpers.Button teleportActivationButton;
    public float activationThreshold = 0.1f;

    // Update is called once per frame
    void Update()
    {
        if(leftTeleportRay)
        {
            leftTeleportRay.gameObject.SetActive(CheckIfActivated(leftTeleportRay));
        }

        if (rightTeleportRay)
        {
            rightTeleportRay.gameObject.SetActive(CheckIfActivated(rightTeleportRay));
        }
    }

    public bool CheckIfActivated(XRController controller)
    {
        InputHelpers.IsPressed(controller.inputDevice, teleportActivationButton, out bool is
Activated, activationThreshold);
        return isActivated;
    }
}
```

missioneCompiutaSimple.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;

public class missioneCompiutaSimple : MonoBehaviour
{
    // Start is called before the first frame update
    private bool missioneCompletata=false;
    public GameObject poof;
    public GameObject warp;
    // [SerializeField]
    // public List<GameObject> setStatic;
    //public GameObject disattivaInvent;
    public AudioClip jingle;
    public float volume;

    public GameObject testo;
    public GameObject testoProssimaMissione;
    public GameObject testoQuestaMissione;
    public GameObject warpProssimaMissione;

   public void Update(){
       if(missioneCompletata==false){
```

```
                poof.SetActive(true);
                warp.SetActive(false);
                AudioSource.PlayClipAtPoint (jingle, transform.position, volume);
                testo.SetActive(true);
                poof.SetActive(false);
                testo.SetActive(true);
                //cambio colore testo missione successiva
                testoQuestaMissione.GetComponent<Text>().text=testoQuestaMissione.GetCompone
nt<Text>().text+"✓";
                testoQuestaMissione.GetComponent<Text>().color=Color.green;
                testoProssimaMissione.GetComponent<Text>().color = Color.yellow;
                warpProssimaMissione.SetActive(true);
                missioneCompletata=true;
        }
    }
}
```

muoviEscalatore.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class muoviEscalatore : MonoBehaviour
{
    public GameObject montascale;
    public GameObject controllo_piattaforma;
    public GameObject bordo;
    void Start(){
    }
    // Start is called before the first frame update
    void OnTriggerEnter(Collider other){
        if(other.tag=="manoSN"||other.tag=="manoDX"){
            if(controllo_piattaforma.activeSelf==false){
                montascale.GetComponent<Animation>().Play("montascal_giu 1");
                // montascale.GetComponent<AudioSource>().Play();
                controllo_piattaforma.SetActive(true);
                StartCoroutine("coroutineColl");
            }
            else if(controllo_piattaforma.activeSelf==true){
                montascale.GetComponent<Animation>().Play("montascal_su");
                // montascale.GetComponent<AudioSource>().Play();
                controllo_piattaforma.SetActive(false);
                StartCoroutine("coroutineColl");
            }
        }
    }
     public IEnumerator coroutineColl(){
        bordo.SetActive(true);
        yield return new WaitForSeconds(20f);
        bordo.SetActive(false);
    }
}
```

onContactActivateSimple.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class onContactActivateSimple : MonoBehaviour
{
    [SerializeField]
    public string GOTagAttivare;
    public GameObject[] oggettiDaDisattivare;
    public GameObject portale;
    public bool attivato=false;
    void Start(){
        //vuoto
```

```
    }
    private void OnCollisionEnter(Collision other)
    {
        if(attivato==false){
            if (other.gameObject.tag==GOTagAttivare){
                portale.GetComponent<missioneCompiutaSimple>().enabled=true;
                foreach (GameObject o in oggettiDaDisattivare){
                    o.SetActive(false);
                }
                attivato=true;
                this.gameObject.SetActive(false);
            }
        }
    }
}
```

onContactActivateWaitTotSec.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class onContactActivateWaitTotSec : MonoBehaviour
{
    [SerializeField]
    public string GOTagAttivare;
    public GameObject[] oggettiDaDisattivare;
    public GameObject portale;
    public bool attivato=false;
    void Start(){

    }
    private void OnCollisionEnter(Collision other)
    {
        if(attivato==false){
            if (other.gameObject.tag==GOTagAttivare){
                portale.GetComponent<waitTotSec>().enabled=true;
                foreach (GameObject o in oggettiDaDisattivare){
                    o.SetActive(false);
                }
                attivato=true;
                this.gameObject.SetActive(false);
            }
        }
    }
}
```

PiuLuci_VR.cs

```
using System.Collections;using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PiuLuci_VR : MonoBehaviour {
    public GameObject pulsante_smart;
    public Sprite spriteOn;
    public Sprite spriteOff;
    public Material materiale_int_off;
    public Material materiale_int_on;
    public AudioSource clickInterruttore;
    public GameObject[] Lights;
    public GameObject[] Interruttori;
    public ReflectionProbe[] ReflectionProbes;
    public Renderer[] EmissiveObjects;
    private Color[] EmissColors;
    public GameObject controlloLuci;

    private bool inZone = false;
```

```
    private float _timer = 0.5f;
    void Start () {

        foreach (ReflectionProbe _probe in ReflectionProbes) {
            if (_probe) {
                _probe.mode = UnityEngine.Rendering.ReflectionProbeMode.Realtime;
                _probe.refreshMode = UnityEngine.Rendering.ReflectionProbeRefreshMode.ViaScr
ipting;
                Invoke ("BakeProbes", _timer);
            }
        }

        EmissColors = new Color[EmissiveObjects.Length];

        for(int i = 0; i < EmissiveObjects.Length; i++){
            EmissColors[i] = EmissiveObjects[i].material.GetColor("_EmissionColor");
        }

        if (controlloLuci.activeSelf==false) {
            DisableEmission ();
        }
        else {
            StartCoroutine (EnableEmissionLate());
            //Debug.Log("Emission enabled");
        }
    }

    IEnumerator EnableEmissionLate () {
        yield return null;
        for(int i = 0; i < EmissiveObjects.Length; i++){
            EmissiveObjects[i].material.SetColor ("_EmissionColor", EmissColors[i]);
            DynamicGI.SetEmissive(EmissiveObjects[i], EmissColors[i] * 0.36f);
        }
    }

    void DisableEmission(){

        for (int i = 0; i < EmissiveObjects.Length; i++) {
                EmissiveObjects[i].GetComponent<Renderer>().material.DisableKeyword("_EMISSI
ON");
        }

    }
    void EnableEmission(){
        for(int i = 0; i < EmissiveObjects.Length; i++){
                EmissiveObjects[i].GetComponent<Renderer>().material.EnableKeyword("_EMISSIO
N");
        }
    }

    void BakeProbes(){
        foreach (ReflectionProbe _probe in ReflectionProbes) {
            if (_probe) {
                _probe.RenderProbe ();
            }
        }
    }
    void Light Off(){
        foreach (GameObject interruttore in Interruttori){
            interruttore.GetComponent<MeshRenderer>().material = materiale_int_off;
        }
        foreach (GameObject _light in Lights) {
            if(_light){
                _light.SetActive (false);
                controlloLuci.SetActive(false);
                clickInterruttore.Play();
                pulsante_smart.GetComponent<SpriteRenderer>().sprite = spriteOff;
            }
        }
    }
    void Light_On(){
        foreach (GameObject interruttore in Interruttori){
```

```
            interruttore.GetComponent<MeshRenderer>().material = materiale_int_on;
        }
        foreach (GameObject _light in Lights) {
            if(_light){
                _light.SetActive (true);
                controlloLuci.SetActive(true);
                clickInterruttore.Play();
                pulsante_smart.GetComponent<SpriteRenderer>().sprite = spriteOn;
            }
        }
    }

    void OnTriggerEnter(Collider other){
        if (other.tag=="dito"||other.tag=="ditoDX"){
            if(controlloLuci.activeSelf==true){
                //spegni
                Light_Off();
                DisableEmission();
            }
            else{
                //accendi
                Light_On();
                EnableEmission();
            }
        }
    }
}
```

PortaCamVR.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PortaCamVR : MonoBehaviour {

    [SerializeField] private Vector3 PressioneBottone;

    public bool buttonPressed = false;
    public bool buttonUnpressed = false;
    public bool buttonDown = false;

    public float buttonPressDuration;
    private float buttonPressedTime;
    private bool buttonImmediatelyPressed;  // used to hold onto the pressed state for one f
rame.
    private bool buttonImmediatelyUnpressed;  // used to hold onto the pressed state for one
 frame.

    public AudioSource suonoOp;
    public AudioSource suonoCl;

    public Animation AnimApriChiudi;

    public GameObject interruttore_dentro;
    public GameObject interruttore_fuori;
    public GameObject pulsante_smart;
    public GameObject controlloStato;

    public Sprite spriteOn;
    public Sprite spriteOff;

    public Material materiale_int_off;
    public Material materiale_int_on;

    void Update() {
        if (this.buttonDown) {
            // Only hold button pressed for one frame
            if (this.buttonPressed && !this.buttonImmediatelyPressed) {
                this.buttonPressed = false;
```

```
            }
            this.buttonImmediatelyPressed = false;

            if (Time.time - this.buttonPressedTime > buttonPressDuration) {
                this.transform.position += this.transform.TransformDirection(PressioneBotton
e);
                this.buttonDown = false;
                this.buttonImmediatelyUnpressed = true;
            }
        } else {
            if (this.buttonImmediatelyUnpressed) {
                this.buttonUnpressed = true;
                this.buttonImmediatelyUnpressed = false;
            } else {
                this.buttonUnpressed = false;
            }
        }
    }

    private void OnTriggerEnter(Collider other) {
        this.buttonPressedTime = Time.time;
            if (this.buttonDown) {
                return;
            }

        //apertura
        //controllo tutte le istanze degli interruttori (dentro-fuori)
        if(other.tag=="dito"||other.tag=="ditoDX"){
            if (controlloStato.active==false){
                AnimApriChiudi.Play("portaCamOp");
                suonoOp.Play();
                interruttore_dentro.GetComponent<MeshRenderer>().material = materiale_int_on
;
                interruttore_fuori.GetComponent<MeshRenderer>().material = materiale_int_on;
                pulsante_smart.GetComponent<SpriteRenderer>().sprite = spriteOn;
                controlloStato.SetActive(true);

            //chiusura
            }
            else{
                AnimApriChiudi.Play("portaCamCl");
                suonoCl.Play();
                interruttore_dentro.GetComponent<MeshRenderer>().material = materiale_int_of
f;
                interruttore_fuori.GetComponent<MeshRenderer>().material = materiale_int_off
;
                pulsante_smart.GetComponent<SpriteRenderer>().sprite = spriteOff;
                controlloStato.SetActive(false);
            }
        }
            this.transform.position += this.transform.TransformDirection(-PressioneBottone);
            this.buttonImmediatelyPressed = true;
            this.buttonPressed = true;
            this.buttonDown = true;

    }
}
```

provacambio.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class provacambio : MonoBehaviour
{
    public GameObject rigNormale;
    public GameObject rigHandicap;
    public GameObject followCanvasMissioni;
```

```
    public GameObject followBollaInventario;
    public GameObject followCanvasTermineMissione;
    public GameObject followCanvasTermineCapitolo;
    public GameObject followCanvasTermineGioco;
    public Transform VRCamera_dis;
    public Transform VRCamera_normo;
    public GameObject overlay_normo;
    public GameObject overlay_dis;
    public GameObject[] colliderAnchors;
    // Start is called before the first frame update
    void Start()
    {
    }
    // Update is called once per frame
    void OnTriggerEnter(Collider other)
    {
        if(this.gameObject.tag=="selettore_disabile"){
            //seleziono rig disabile
            rigNormale.SetActive(false);
            rigHandicap.SetActive(true);
            followCanvasMissioni.GetComponent<FollowToTheSide>().target=VRCamera_
dis;
            followBollaInventario.GetComponent<FollowToTheSide>().target=VRCamera
_dis;
            followCanvasTermineCapitolo.GetComponent<FollowToTheSide>().target=VR
Camera_dis;
            followCanvasTermineMissione.GetComponent<FollowToTheSide>().target=VR
Camera_dis;
            followCanvasTermineGioco.GetComponent<FollowToTheSide>().target=VRCam
era_dis;
            StartCoroutine(overlayDis());
            //disabilita tutti i mesh collider sulle teleport anchors
            foreach (GameObject c in colliderAnchors){
                c.GetComponent<MeshCollider>().enabled=false;
            }
        }
      if(this.gameObject.tag=="selettore_normo"){
            //seleziono rig disabili
            rigNormale.SetActive(true);
            rigHandicap.SetActive(false);
            followCanvasMissioni.GetComponent<FollowToTheSide>().target=VRCamera_
normo;
            followBollaInventario.GetComponent<FollowToTheSide>().target=VRCamera
_normo;
            followCanvasTermineCapitolo.GetComponent<FollowToTheSide>().target=VR
Camera_normo;
            followCanvasTermineMissione.GetComponent<FollowToTheSide>().target=VR
Camera_normo;
            followCanvasTermineGioco.GetComponent<FollowToTheSide>().target=VRCam
era_normo;
            StartCoroutine(overlayNormo());
            //abilita tutti i mesh collider sulle teleport anchors
            foreach (GameObject c in colliderAnchors){
                c.GetComponent<MeshCollider>().enabled=true;
            }
        }
    }
    private IEnumerator overlayDis(){
        yield return new WaitForSeconds(1.5f);
        overlay_dis.SetActive(true);
    }
    private IEnumerator overlayNormo(){
        yield return new WaitForSeconds(1.5f);
        overlay_normo.SetActive(true);
```

```
    }
}
```

provaXRinput.cs

```
using System;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Events;
using UnityEngine.XR;
using Unity.XR.Oculus;
public class provaXRInput : MonoBehaviour
{
    static readonly Dictionary<string, InputFeatureUsage<bool>> availableButtons = new Dicti
onary<string, InputFeatureUsage<bool>>
    {
        {"triggerButton", CommonUsages.triggerButton },
        {"thumbrest", OculusUsages.thumbrest },
        {"primary2DAxisClick", CommonUsages.primary2DAxisClick },
        {"primary2DAxisTouch", CommonUsages.primary2DAxisTouch },
        {"menuButton", CommonUsages.menuButton },
        {"gripButton", CommonUsages.gripButton },
        {"secondaryButton", CommonUsages.secondaryButton },
        {"secondaryTouch", CommonUsages.secondaryTouch },
        {"primaryButton", CommonUsages.primaryButton },
        {"primaryTouch", CommonUsages.primaryTouch },
    };
    public enum ButtonOption
    {
        triggerButton,
        thumbrest,
        primary2DAxisClick,
        primary2DAxisTouch,
        menuButton,
        gripButton,
        secondaryButton,
        secondaryTouch,
        primaryButton,
        primaryTouch
    };
    [Tooltip("Tipo di controller (destro/sinistro)")]
    public InputDeviceCharacteristics deviceCharacteristic;
    [Tooltip("Selezione del bottone")]
    public ButtonOption button;
    [Tooltip("Evento da eseguire quando il bottone è premuto")]
    public UnityEvent OnPress;
    [Tooltip("Evento da eseguire quando il bottone è rilasciato")]
    public UnityEvent OnRelease;
    // to check whether it's being pressed
    public bool IsPressed { get; private set; }
    // to obtain input devices
    List<InputDevice> inputDevices;
    bool inputValue;
    InputFeatureUsage<bool> inputFeature;

    void Awake()
    {
        // get label selected by the user
        string featureLabel = Enum.GetName(typeof(ButtonOption), button);
        // find dictionary entry
        availableButtons.TryGetValue(featureLabel, out inputFeature);
        // init list
        inputDevices = new List<InputDevice>();
    }
    void Update()
    {
        InputDevices.GetDevicesWithCharacteristics(deviceCharacteristic, inputDevices);

        for (int i = 0; i < inputDevices.Count; i++)
        {
```

```
            if (inputDevices[i].TryGetFeatureValue(inputFeature,
                out inputValue) && inputValue)
            {
                // if start pressing, trigger event
                if (!IsPressed)
                {
                    IsPressed = true;
                    OnPress.Invoke();
                    //Debug.Log("bottone premuto");
                }
            }
            // check for button release
            else if (IsPressed)
            {
                IsPressed = false;
                OnRelease.Invoke();
               // Debug.Log("bottone rilasciato");
            }
        }
    }
}
```

questoToccaQuello.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class questoToccaQuelloConf : MonoBehaviour
{
    void Start(){

    }

    public bool giaToccato=false;
    // Start is called before the first frame update
    void OnCollisionEnter(Collision other){
        if(giaToccato==false){
            if(other.gameObject.tag=="Player"||other.gameObject.tag=="Player_dis"){
                GetComponent<WaitTotSecConf>().enabled=true;
                giaToccato=true;
            }
        }

    }
}
```

SceneFader.cs

```
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using System.Collections;
public class SceneFader : MonoBehaviour {
    #region FIELDS
    //public Image fadeOutUIImage;
    public float fadeSpeed = 0.8f;
    public GameObject overlay;
    private Material materiale_overlay;
    public enum FadeDirection
    {
        In, //Alpha = 1
        Out // Alpha = 0
    }
    #endregion
    #region MONOBHEAVIOR
    void OnEnable()
    {
```

```
        StartCoroutine(Fade(FadeDirection.In));
    }
    #endregion

    #region FADE
    private IEnumerator Fade(FadeDirection fadeDirection)
    {
        float alpha = (fadeDirection == FadeDirection.Out)? 1 : 0;
        float fadeEndValue = (fadeDirection == FadeDirection.Out)? 0 : 1;
        if (fadeDirection == FadeDirection.Out) {
            while (alpha >= fadeEndValue)
            {
                SetColorImage (ref alpha, fadeDirection);
                yield return null;
            }
            overlay.SetActive(false);
        } else {
             overlay.SetActive(true);
            while (alpha <= fadeEndValue)
            {
                SetColorImage (ref alpha, fadeDirection);
                yield return null;
            }
        }
    }
    #endregion

    // #region HELPERS
    // public IEnumerator FadeAndLoadScene(FadeDirection fadeDirection, string sceneToLoad)
    // {
    //     yield return Fade(fadeDirection);
    //     SceneManager.LoadScene(sceneToLoad);
    // }

    private void SetColorImage(ref float alpha, FadeDirection fadeDirection)
    {
        materiale_overlay=overlay.GetComponent<MeshRenderer>().material;
        materiale_overlay.color= new Color (materiale_overlay.color.r,materiale_overlay.colo
r.g,materiale_overlay.color.b, alpha);
        //fadeOutUIImage.color = new Color (fadeOutUIImage.color.r,fadeOutUIImage.color.g, f
adeOutUIImage.color.b, alpha);
        alpha += Time.deltaTime * (1.0f / fadeSpeed) * ((fadeDirection == FadeDirection.Out)
? -1 : 1) ;
    }
    // #endregion
}
```

sediaMotorizzataV2.cs

```
using UnityEngine;
using System.Collections;

public class sediaMotorizzataV2 : MonoBehaviour {
    //forze fisiche
    public float AccelerazioneFWD = 120.0f;
    public float VelocitaRotazione = 0.25f;
    public float VelocitaMax = 1.1f;
    public float DeadZoneJoystickAsseX = 0.5f;
    public float DeadZoneJoystickAsseY = 0f;
    AudioSource[] audioSources;
    AudioSource MotoreAvvio;
    AudioSource MotoreAcceso;
    AudioSource MotoreStop;
    //rotazione ruote
    public WheelCollider RuotaDX;
    public WheelCollider RuotaSN;
    public float VelocitaRuote = 100;
    public Transform TransfRuotaDX;
    public Transform TransfRuotaSN;
```

```
    // Use this for initialization
    void Start()
    {
          audioSources = GetComponents<AudioSource>();
          MotoreAvvio = audioSources[0];
          MotoreAcceso = audioSources[1];
          MotoreStop = audioSources[2];
    }


     void Update()
     {
        TransfRuotaSN.Rotate ( RuotaSN.rpm / 60 * 360 * Time.deltaTime, 0, 0);
        TransfRuotaDX.Rotate ( RuotaDX.rpm / 60 * 360 * Time.deltaTime,0,0);
     }

    //per le forze, meglio FixedUpdate. in caso di problemi, sostituire con Update
    void FixedUpdate()
    {
        float forwardForce = 0.0f;//actual amount we will move this frame
        forwardForce = Input.GetAxis("Oculus_CrossPlatform_SecondaryThumbstickVertical") *
this.AccelerazioneFWD;

        //The faster we go, the less force that is applied by the engine
        if (this.GetComponent<Rigidbody>().velocity.magnitude > VelocitaMax)
        {
            forwardForce = 0;
        }
        else
        {
            forwardForce *= (VelocitaMax -
this.GetComponent<Rigidbody>().velocity.magnitude) / VelocitaMax;
        }

        float turnAmount = 0.0f;//actual amount we will turn this frame
        float adjustedAxis =
Input.GetAxis("Oculus_CrossPlatform_SecondaryThumbstickHorizontal");
        if (Mathf.Abs(adjustedAxis) < this.DeadZoneJoystickAsseX)
        {
            adjustedAxis = 0;
        }
        else if (adjustedAxis > 0)
        {
            adjustedAxis = (adjustedAxis - this.DeadZoneJoystickAsseX) * (1.0f /
this.DeadZoneJoystickAsseX);

        }
        else
        {
            adjustedAxis = (adjustedAxis + this.DeadZoneJoystickAsseX) * (1.0f /
this.DeadZoneJoystickAsseX);
        }

        turnAmount = adjustedAxis * this.VelocitaRotazione;

          if (Mathf.Abs (Input.GetAxis
("Oculus_CrossPlatform_SecondaryThumbstickHorizontal")) > this.DeadZoneJoystickAsseX) {
                if (!MotoreAvvio.isPlaying && !MotoreAcceso.isPlaying &&
!MotoreStop.isPlaying) {
                        MotoreAvvio.Play ();
                        MotoreAcceso.Play ();
                }
          } else if (Mathf.Abs (Input.GetAxis
("Oculus_CrossPlatform_SecondaryThumbstickVertical")) > 0) {
                if (!MotoreAvvio.isPlaying && !MotoreAcceso.isPlaying &&
!MotoreStop.isPlaying) {
                        MotoreAvvio.Play ();
                        MotoreAcceso.Play ();
                }
          } else {
                if (MotoreAvvio.isPlaying) {
                        MotoreAvvio.Stop ();
```

```
                        MotoreStop.Play ();
                } else if (MotoreAcceso.isPlaying) {
                        MotoreAcceso.Stop ();
                        MotoreStop.Play ();
                }
          }
        this.transform.Rotate(0, turnAmount, 0);
        this.GetComponent<Rigidbody>().AddRelativeForce(forwardForce * Vector3.forward);

        //ruote
        //thumbstick dx
        RuotaDX.motorTorque =
Input.GetAxis("Oculus_CrossPlatform_SecondaryThumbstickVertical")*VelocitaRuote;
        RuotaSN.motorTorque =
Input.GetAxis("Oculus_CrossPlatform_SecondaryThumbstickVertical")*VelocitaRuote;

    }
}
```

Sliceable.cs

```
using System.Collections;
using System.Collections.Generic;
using System.Security.AccessControl;
using UnityEngine;
public class Sliceable : MonoBehaviour
{
    [SerializeField]
    private bool _isSolid = true;
    [SerializeField]
    private bool _reverseWindTriangles = false;
    [SerializeField]
    private bool _useGravity = false;
    [SerializeField]
    private bool _shareVertices = false;
    [SerializeField]
    private bool _smoothVertices = false;
    public bool IsSolid
    {
        get
        {
            return _isSolid;
        }
        set
        {
            _isSolid = value;
        }
    }
    public bool ReverseWireTriangles
    {
        get
        {
            return _reverseWindTriangles;
        }
        set
        {
            _reverseWindTriangles = value;
        }
    }
    public bool UseGravity
    {
        get
        {
            return _useGravity;
        }
        set
        {
            _useGravity = value;
        }
```

```
    }
    public bool ShareVertices
    {
        get
        {
            return _shareVertices;
        }
        set
        {
            _shareVertices = value;
        }
    }
    public bool SmoothVertices
    {
        get
        {
            return _smoothVertices;
        }
        set
        {
            _smoothVertices = value;
        }
    }
}
```

spegniFuocoAutodopoC.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class spegniFuocoAutodopoC : MonoBehaviour
{
    private bool missioneCompletata=false;
    public GameObject manopola;
    public GameObject EffettoFuoco;
    public GameObject fornello;
    //public GameObject[] punti_contatto;

    void Start(){
         if(missioneCompletata==false){
            StartCoroutine("Timer");
            missioneCompletata=true;

        }else{
            StopCoroutine("Timer");
        }
    }
    public IEnumerator Timer(){
        yield return new WaitForSeconds(3f);
        manopola.transform.Rotate(0.0f,-120.0f, 0.0f);
        EffettoFuoco.SetActive(false);
        fornello.GetComponent<cottura>().enabled=false;
        GetComponent<missioneCompiutaSimple>().enabled=true;

    }

}
```

tapp_meccanica_G.cs

```
using UnityEngine;

 public class tapp_meccanicaG_VR : MonoBehaviour {
     [SerializeField] private Vector3 muoviY;
    public GameObject parent;
    public GameObject tastoGiu;
```

```
void Start(){
}

   void OnTriggerStay(Collider other){
       if(other.tag=="interazione"){
            UpScale(parent.transform.localScale);
       }
   }

    void UpScale(Vector3 scale) {
        //limiti transform y giu
        if ((parent.transform.localScale.y<=12)){
            scale += muoviY;
            parent.transform.localScale = scale;
        }
    }

 }
```

tapp_VR_su.cs

```
using UnityEngine;

 public class VR_su : MonoBehaviour {
    [SerializeField] private Vector3 muoviY;
    public AudioClip SoundToPlay;
    public float volume;
    public bool alreadyPlayed;
    AudioSource audio;
    public GameObject parent;
    public GameObject tastoSu;
    public GameObject tastoSmartSu;
    public Sprite spriteOn;
    public Sprite spriteOff;

    void Start(){
        audio=GetComponent<AudioSource>();
    }

   void OnTriggerStay(Collider other){
        if(other.tag=="dito"|| other.tag=="ditoDX"){
        UpScale(parent.transform.localScale);
            tastoSmartSu.GetComponent<SpriteRenderer>().sprite = spriteOn;
        }
    }

    void OnTriggerEnter(Collider other){
        if(other.tag=="dito"|| other.tag=="ditoDX"){
            if(!alreadyPlayed){
                audio.PlayOneShot(SoundToPlay, volume);
                alreadyPlayed=true;
            }
        }

    }

   void OnTriggerExit(Collider other){
        if(other.tag=="dito"|| other.tag=="ditoDX"){
            tastoSmartSu.GetComponent<SpriteRenderer>().sprite = spriteOff;
            alreadyPlayed=false;
            audio.Stop();
        }
    }


    void UpScale(Vector3 scale) {
        //limiti transform y su
        if ((parent.transform.localScale.y>0 && parent.transform.localScale.y<=13)){
            scale += muoviY;
            parent.transform.localScale = scale;
```

```
        }else{
            audio.Stop();
        }
    }
 }
```

Vasistas_1_VR.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Vasistas_1_VR : MonoBehaviour{
    public bool buttonPressed = false;
    [SerializeField] private Vector3 PressioneBottone;
    public bool buttonUnpressed = false;
    public bool buttonDown = false;

    public float buttonPressDuration;
    private float buttonPressedTime;
    private bool buttonImmediatelyPressed;  // used to hold onto the pressed state for one f
rame.
    private bool buttonImmediatelyUnpressed;  // used to hold onto the pressed state for one
 frame.
    public Animation apriChiudi;
    public AudioSource suonoOp;
    public AudioSource suonoCl;

    public GameObject controlloFinestra;
    public GameObject pulsante_smart;
    public Sprite spriteOn;
    public Sprite spriteOff;
    void Update() {
        if (this.buttonDown) {
            // Only hold button pressed for one frame
            if (this.buttonPressed && !this.buttonImmediatelyPressed) {
                this.buttonPressed = false;
            }
            this.buttonImmediatelyPressed = false;

            if (Time.time - this.buttonPressedTime > buttonPressDuration) {
                this.transform.position += this.transform.TransformDirection(PressioneBotton
e);
                this.buttonDown = false;
                this.buttonImmediatelyUnpressed = true;
            }
        } else {
            if (this.buttonImmediatelyUnpressed) {
                this.buttonUnpressed = true;
                this.buttonImmediatelyUnpressed = false;
            } else {
                this.buttonUnpressed = false;
            }
        }
    }
   private void OnTriggerEnter(Collider other) {
        this.buttonPressedTime = Time.time;
            if (this.buttonDown) {
                return;
            }
        if(other.tag=="dito"||other.tag=="ditoDX"){
            if (controlloFinestra.active==false){
                apriChiudi.Play("vasistas_leva_op");
                suonoOp.Play();
                pulsante_smart.GetComponent<SpriteRenderer>().sprite = spriteOn;
                controlloFinestra.SetActive(true);
            }
            else{
                apriChiudi.Play("vasistas_leva_cl");
                suonoCl.Play();
                pulsante_smart.GetComponent<SpriteRenderer>().sprite = spriteOff;
```

```
                controlloFinestra.SetActive(false);
            }
        }
        this.transform.position += this.transform.TransformDirection(-PressioneBottone);
        this.buttonImmediatelyPressed = true;
        this.buttonPressed = true;
        this.buttonDown = true;
    }
}
```

WaitTotSecConf.cs

```
using UnityEngine;
using System.Collections;

public class WaitTotSecConf : MonoBehaviour
{  [SerializeField]
    public GameObject[] oggettiDaDisattivare;
    [SerializeField]
    public float AttesaSec;
    void Start()
    {
        //Start the coroutine we define below named ExampleCoroutine.
        StartCoroutine(TriggeraFineObiettivo());
    }

    IEnumerator TriggeraFineObiettivo()
    {
         yield return new WaitForSeconds(AttesaSec);
         GetComponent<missioneCompiutaSimple>().enabled=true;
         foreach(GameObject o in oggettiDaDisattivare){
             o.SetActive(false);
         }
    }
}
```

wanderingAI.cs

```
using UnityEngine;
using System.Collections;
using UnityEngine.AI;
public class wanderingAi : MonoBehaviour {
    public float wanderRadius;
    public float wanderTimer;
    private Transform target;
    private NavMeshAgent agent;
    private float timer;
    // Use this for initialization
    void OnEnable () {
        agent = GetComponent<NavMeshAgent> ();
        timer = wanderTimer;
    }
    // Update is called once per frame
    void Update () {
        timer += Time.deltaTime;

        if (timer >= wanderTimer) {
            Vector3 newPos = RandomNavSphere(transform.position, wanderRadius, -1);
            agent.SetDestination(newPos);
            timer = 0;
        }
    }
    public static Vector3 RandomNavSphere(Vector3 origin, float dist, int layermask) {
        Vector3 randDirection = Random.insideUnitSphere * dist;

        randDirection += origin;
        NavMeshHit navHit;
```

```
        NavMesh.SamplePosition (randDirection, out navHit, dist, layermask);
        return navHit.position;
    }
}
```

Plugins di XR

Il codice delle componenti dei plugin di XR è molto esteso ed è possibile, pertanto, scaricarlo dal *PackageManager* di Unity e consultarne la relativa documentazione alle seguenti pagine web:

XR Plugin Framework

https://docs.unity3d.com/Manual/XRPluginArchitecture.html

Oculus XR Plugin

https://docs.unity3d.com/Manual/com.unity.xr.oculus.html

XR Interaction Toolkit

https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@0.9/manual/index.html

Bibliografia

- Adams, Douglas. 2016. *Guida Galattica per gli Autostoppisti*. Ed. tascabile. Milano, Mondadori.
- Bortolotti, Elena. 2018. *Le disabilità motorie e sensoriali.* Presentazione. Università degli Studi di Trieste. Link alla presentazione in PDF: https://moodle2.units.it/pluginfile.php/306908/mod_resource/content/1/lez%203a%20Disabilit%C3%A0%20mot.pdf.
- Cellini, Paolo. 2018. *La rivoluzione digitale. Economia di internet dallo Sputnik al machine learning*. Luiss University Press.
- Commissione per l'analisi delle problematiche relative alla disabilità nello specifico settore dei beni e delle attività culturali. 16 maggio 2008. *Linee guida per il superamento delle barriere architettoniche nei luoghi di interesse culturale*. Supplemento ordinario alla Gazzetta Ufficiale. Allegato A. Link alla versione in PDF: https://www.gazzettaufficiale.it/eli/gu/2008/05/16/114/so/127/sg/pdf.
- Ente Italiano di Normazione. 29 novembre 2016. *Abbattimento barriere architettoniche. Linee guida per la riprogettazione del costruito in ottica Universal Design*. Link alla versione in PDF: http://www.cng.it/CMSContent/Consiglio-Nazionale/Repository/files_news/uni_pdr_24_2016.pdf.
- Jenkins, Harry. 2014. *Cultura Convergente*. Apogeo.
- Tavosanis, Mirko. 2018. *Lingue e intelligenza artificiale.* Il riconoscimento del parlato. La trascrizione delle conversazioni. Carocci Editore.

Sitografia

- *Abili a proteggere. Tecnologie assistive* – AbiliAProteggere. Consultato il 29 giugno 2021. https://www.abiliaproteggere.net/informazioni-utili/tecnologie-assistive/.
- *About the Oculus XR Plugin* - Oculus XR Plugin - 1.9.1. Consultato il 29 giugno 2021. https://docs.unity3d.com/Packages/com.unity.xr.oculus@1.9/manual/index.html.
- *Accessibilità di impianti e accessori* - Disabili.com. Consultato il 29 giugno 2021. https://www.disabili.com/mobilita-auto/speciali-mobilita-a-auto/barriere-architettoniche-e-disabilita/barrarch09-accessori-e-impianti.
- *«Accessibilità»* - Enciclopedia Treccani. Consultato il 29 giugno 2021. https://www.treccani.it/enciclopedia/accessibilita.
- *«Adattivo»* - Vocabolario Treccani. Consultato il 29 giugno 2021. https://www.treccani.it/vocabolario/adattivo.
- *Android Development Software Setup* - Oculus Developers. Consultato il 29 giugno 2021. https://developer.oculus.com/documentation/native/android/mobile-studio-setup-android.
- *Any example of the new 2019.1 XR input system?* - Unity Forum. Consultato il 29 giugno 2021. https://forum.unity.com/threads/any-example-of-the-new-2019-1-xr-input-system.629824/.
- *Automazione domestica* - Disabili.com. Consultato il 29 giugno 2021. https://www.disabili.com/prodotti/speciali-prodotti-a-ausili/domotica/17040-domotica-automazione.
- *Barriere architettoniche e disabilità* - Disabili.com. Consultato il 29 giugno 2021. https://www.disabili.com/mobilita-auto/speciali-mobilita-a-auto/barriere-architettoniche-e-disabilita.
- *Barriere architettoniche. Progettare una casa accessibile* - Disabili.com. Consultato il 29 giugno 2021. https://www.disabili.com/mobilita-auto/speciali-mobilita-a-auto/barriere-architettoniche-e-disabilita/barrarch09-spazi-interni.
- *Architettura per tutti. Che cos'è l'Universal Design* - Architutti.15 dicembre 2016. Consultato il 29 giugno 2021. https://www.architutti.it/che-cose-luniversal-design/.

- *Con il progetto EXPERIENCE la realtà virtuale entra nei social network* - Unipi. Consultato il 29 giugno 2021. https://www.unipi.it/index.php/news/item/19862-con-il-progetto-experience-la-realta-virtuale-entra-nei-social-network.
- *Cos'è la CAA - Comunicazione Aumentativa Alternativa.* - FareLeggereTutti. Consultato il 29 giugno 2021. https://www.fareleggeretutti.it/cosa-e-la-caa-comunicazione-aumentativa-alternativa.
- *Cos'è la Sindrome di Escobar, una malattia quasi sconosciuta*. – AbbassamentoVeicoli. Consultato il 29 giugno 2021. https://www.abbassamentoveicoli.it/la-sindrome-di-escobar-una-malattia-quasi-sconosciuta/.
- *CZamberti/MARVIN_LATEST_0521* – Github. Consultato il 29 giugno 2021. https://github.com/CZamberti/MARVIN_LATEST_0521.
- *Disabili e Barriere Architettoniche* – Fondazione Serono. Consultato il 29 giugno 2021. https://www.fondazioneserono.org/disabilita/disabilita-diritti-e-normativa/disabili-e-barriere-architettoniche/disabilita-e-diritti-disabili-e-barriere-architettoniche/.
- *Disturbi della deambulazione negli anziani* – Manuali MSD Edizione Professionisti. Consultato il 29 giugno 2021. https://www.msdmanuals.com/it-it/professionale/geriatria/disturbi-della-deambulazione-negli-anziani/disturbi-della-deambulazione-negli-anziani.
- *Guide introduttive, esercitazioni e informazioni di riferimento.* – Documentazione di C#. Consultato il 29 giugno 2021. https://docs.microsoft.com/it-it/dotnet/csharp/.
- *Domotica, una casa intelligente per persone con disabilità* – Cronaca, La Nazione. Consultato il 29 giugno 2021. https://www.lanazione.it/empoli/cronaca/domotica-casa-intelligente-persone-disabilita-1.4999994/.
- *Getting Started* – FIABA Onlus. Consultato il 29 giugno 2021. https://www.fiaba.org/.
- *Git-Scm* – Git. Consultato il 29 giugno 2021. https://git-scm.com/.
- *Google and Mattel Relaunch View-Master as Virtual Reality Headset* – eTeknix. Consultato il 29 giugno 2021. https://www.eteknix.com/google-mattel-relaunch-view-master-virtual-reality-headset/.
- «*Hackaton*» – Vocabolario Treccani. Consultato il 29 giugno 2021. https://www.treccani.it/vocabolario/hackathon.
- *Home* – Mono. Consultato il 29 giugno 2021. https://www.mono-project.com/.
- *Home* – INAIL. Consultato il 29 giugno 2021. https://www.inail.it/cs/internet/home.html

- *Home* - La Macchina del Tempo. Consultato il 29 giugno 2021. https://www.lamacchinadeltempo.eu/#esperienze.
- *How Virtual Reality Takes Scientists Inside New Molecules* – Pfizer. Consultato il 29 giugno 2021. https://www.pfizer.com/news/featured_stories/featured_stories_detail/how_virtual_reality_takes_scientists_inside_new_molecules.
- *HP® OMEN Gaming PCs and Laptops* – HP. Consultato il 29 giugno 2021. https://www.hp.com/us-en/shop/slp/omen-gaming.
- *HQ Residential House* – Unity Asset Store. Consultato il 29 giugno 2021. https://assetstore.unity.com/packages/3d/environments/urban/hq-residential-house-48976.
- *I numeri della disabilità in Italia* – Le Nius. Consultato il 29 giugno 2021. https://www.lenius.it/disabilita-in-italia/.
- Interfacce di comando – Disabili.com. Consultato il 29 giugno 2021. https://www.disabili.com/prodotti/speciali-prodotti-a-ausili/domotica/17042-domotica-interfacce-di-comando.
- «*Interfaccia*» – Vocabolario Treccani. Consultato il 29 giugno 2021. https://www.treccani.it/vocabolario/interfaccia.
- *Introduzione al rigging: armature e bones.* - Grafica HTML.it. Consultato il 29 giugno 2021. https://www.html.it/pag/44285/introduzione-al-rigging-armature-e-bones/.
- *Kinect, Sviluppo di app di Windows.* – Microsoft. Consultato il 29 giugno 2021. https://developer.microsoft.com/it-it/windows/kinect/.
- *Leadership nell'elaborazione basata su intelligenza artificiale* – NVIDIA. Consultato il 29 giugno 2021. https://www.nvidia.com/it-it/.
- *Leap Motion introduce Orion, tracciamento delle mani in VR di nuova generazione* – Hardware Upgrade». Consultato il 29 giugno 2021. https://www.hwupgrade.it/news/wearables/leap-motion-introduce-orion-tracciamento-delle-mani-in-vr-di-nuova-generazione_61026.html.
- *Learn game development w/* – Unity Learn». Consultato il 29 giugno 2021. https://learn.unity.com/.
- *Legge 13 1989: Abbattimento delle Barriere Architettoniche* – ContactSRL. Consultato il 29 giugno 2021. https://www.contactsrl.it/normative/legge-13-89/.

- *Licenze, Progetto GNU* – GNU Free Software Foundation. Consultato il 29 giugno 2021. https://www.gnu.org/licenses/licenses.it.html.
- *L'interfaccia vocale e la figura dello UX Designer.* – Geeks Academy. Consultato il 29 giugno 2021. https://www.geeksacademy.it/articolo-28/l_interfaccia-vocale-e-la-figura-dello-ux-designer/.
- *Lo scivolo per disabili senza corrimano è a norma?* – La Legge per Tutti. Consultato il 29 giugno 2021. https://www.laleggepertutti.it/143687_lo-scivolo-per-disabili-senza-corrimano-e-a-norma.
- Miur – Ministero dell'istruzione, dell'università e della ricerca. Consultato il 29 giugno 2021. https://www.miur.gov.it/.
- *MonoDevelop Tool* – MonoDevelop. Consultato il 29 giugno 2021. https://www.monodevelop.com/.
- *Movidabilia, spazi senza barriere* – Movidabilia. Consultato il 29 giugno 2021. https://www.movidabilia.it.
- MSI Italy. Consultato il 29 giugno 2021. https://it.msi.com/.
- *Una casa domotica per la vita autonoma.* – La Nazione. Consultato il 29 giugno 2021. https://www.lanazione.it/empoli/cronaca/una-casa-domotica-per-la-vita-autonoma-1.6204234.
- *Oculus Integration SDK* – Oculus. Consultato il 29 giugno 2021. https://developer.oculus.com/downloads/package/unity-integration/.
- *Oculus Rift + Kinect = follia psichedelica.* – Vigamus Magazine. Consultato il 29 giugno 2021. https://www.vigamusmagazine.com/138243/oculus-rift-kinect-follia-psichedelica/.
- Perceptual Robotics, PERCRO Lab – Scuola Superiore Sant'Anna». Consultato il 29 giugno 2021. https://www.santannapisa.it/it/istituto/tecip/perceptual-robotics-percro-lab.
- *Primi passi* – Oculus». Consultato il 29 giugno 2021. https://support.oculus.com/.
- *Progetta la tua cameretta con la realtà virtuale* – Mondo Camerette. Consultato il 29 giugno 2021. https://www.mondocamerette.it/progetta-la-tua-cameretta-con-la-realta-virtuale/.
- *Questionario di valutazione dell'usabilità e dell'esperienza utente.* – Google Forms. Consultato il 29 giugno 2021.

https://docs.google.com/forms/d/e/1FAIpQLSdoIcExjuGNt6qOeRy_jw5pB8M0p9UBU1DZldnbaOd49qPHXA/viewform.

- *Quindici Molfetta, un lettore scrive a "Quindici": il marciapiede per i disabili di Corso Dante fa inciampare i pedoni.* – Quindici-Molfetta. Consultato il 29 giugno 2021. http://www.quindici-molfetta.it/molfetta-un-lettore-scrive-a-quindici-il-marciapiede-per-i-disabili-di-corso-dante-fa-inciampare-i-pedoni_20140.aspx.
- *Rampa troppo ripida, disabile in carrozzella ferito*– Il Resto del Carlino. Consultato il 29 giugno 2021. https://www.ilrestodelcarlino.it/pesaro/cronaca/disabile-ferito-san-lazzaro-1.3073457.
- *Rea. Anza Lone and. Giosue Carducci* – Opera Omnia. Consultato il 29 giugno 2021. http://carducci.letteraturaoperaomnia.org/.
- *Realtà aumentata, cos'è e come utilizzarla per l'assistenza da remoto.* – Digital4Biz. Consultato il 29 giugno 2021. https://www.digital4.biz/executive/realta-aumentata-cose-come-funziona-e-ambiti-applicativi-in-italia/.
- *Realtà Virtuale tra sensibilizzazione ed empatia: un progetto di Uqido e WWF* – Benessere Tecnologico». Consultato il 29 giugno 2021. https://benesseretecnologico.it/realta-virtuale-uqido-wwf/.
- *Rendering 3D fotorealistico per l'architettura e il design* – NativeDesign. Consultato il 29 giugno 2021. https://www.nativedesign.it/servizi/rendering-3d.
- *Resa accessibile ai disabili la stazione di Pontedera (PI).* – VorreiPrendereIlTreno. Consultato il 29 giugno 2021. https://vorreiprendereiltreno.it/progetti/resa-accessibile-stazione-pontedera-pi/.
- *Riabilitazione con Telepresenza Immersiva Virtuale* – Auxologico». Consultato il 29 giugno 2021. https://www.auxologico.it/prestazione/riabilitazione-telepresenza-immersiva-virtuale.
- *Scegli il tuo Cardboard* – Google VR. Consultato il 29 giugno 2021. https://arvr.google.com/intl/it_it/cardboard/get-cardboard/.
- *Shaders in LWRP. Package Manager UI website* – Unity. Consultato il 29 giugno 2021. https://docs.unity3d.com/Packages/com.unity.render-pipelines.lightweight@5.10/manual/shaders-in-lwrp.html.
- *Silvia scrive: domotica, un aiuto contro le barriere domestiche.* – FIABA Onlus. Consultato il 29 giugno 2021. https://www.fiaba.org/domotica-e-barriere-domestiche/.

- *SIM card IoT globale e connettività M2M*. – Things Mobile. Consultato il 29 giugno 2021. https://www.thingsmobile.com/it/business.
- *Social model of disability. Disability charity.* – Scope UK. Consultato il 29 giugno 2021. https://www.scope.org.uk/about-us/social-model-of-disability/.
- *Sport e disabilità: il moncalvese Samuele Bosco intervista Arturo Mariani* – ATNews.it. Consultato il 29 giugno 2021. https://www.atnews.it/2021/04/sport-e-disabilita-il-moncalvese-samuele-bosco-intervista-arturo-mariani-142441/.
- SuperAbile INAIL. Consultato il 29 giugno 2021. https://www.superabile.it/cs/superabile/home.
- *Unity Manual: Oculus XR Plugin*. – Unity. Consultato il 29 giugno 2021. https://docs.unity3d.com/Manual/com.unity.xr.oculus.html.
- *Unity Manual: Unity XR Input*. – Unity. Consultato il 29 giugno 2021. https://docs.unity3d.com/Manual/xr_input.html.
- *Unity Manual: XR Plug-in Framework.* – Unity. Consultato il 29 giugno 2021. https://docs.unity3d.com/Manual/XRPluginArchitecture.html.
- *Unity Scripting API: MonoBehaviour*. – Unity. Consultato il 29 giugno 2021. https://docs.unity3d.com/ScriptReference/MonoBehaviour.html.
- *Unity Scripting API: MonoBehaviour.OnMouseDown().* – Unity. Consultato il 29 giugno 2021. http://docs.unity3d.com/ScriptReference/MonoBehaviour.OnMouseDown.html.
- Unity Real-Time Development Platform. Consultato il 29 giugno 2021. https://unity.com/.
- *Unity Wheelchair Controller* – Alex Ocias Blog. Consultato il 29 giugno 2021. https://ocias.com/blog/unity-wheelchair-controller/.
- *«Usabilità»* – Enciclopedia Treccani. Consultato il 29 giugno 2021. https://www.treccani.it/enciclopedia/usabilita.
- Visual Studio Code. Code Editing. Redefined. Consultato il 29 giugno 2021. https://code.visualstudio.com/.
- *What Is Virtual Reality?* – The Interaction Design Foundation. Consultato il 29 giugno 2021. https://www.interaction-design.org/literature/topics/virtual-reality.
- *Wheelchair Basketball VR - Animation & Game*. – Medien Campus. Consultato il 29 giugno 2021. https://ag.mediencampus.h-da.de/projekt/wheelchair-basketball-vr/.

- *Wheelchair Simulator VR* – Steam. Consultato il 29 giugno 2021. https://store.steampowered.com/app/841120/Wheelchair_Simulator_VR/.
- *Windows Pack* – Unity Asset Store. Consultato il 29 giugno 2021. https://assetstore.unity.com/packages/3d/props/interior/windows-pack-72257.
- *XR Interaction Toolkit.* – Unity. Consultato il 29 giugno 2021. https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@0.9/manual/index.html
- *Zotero. Consultato il 29 giugno 2021. https://www.zotero.org/*

Bibliografia e Sitografia curate con Zotero[120]

[120] Zotero. Link al sito: https://www.zotero.org/