Alma Mater Studiorum · Università di Bologna

Dipartimento di Informatica — Scienza e Ingegneria
Corso di Laurea Magistrale in Ingegneria Informatica

GAVUNIA: VIRTUAL REALITY, GAME DESIGN E HUMAN COMPUTER INTERACTION

Tesi di Laurea in "Fondamenti di Computer Graphics M"

Relatore:
Prof.ssa
Serena Morigi

Presentata da:
Bryan Bertoni

Correlatore:
Paolo Zuzolo

Sessione 26/03
Anno Accademico 2025/2026

Indice

Introduzione

Negli ultimi anni, la realtà virtuale (VR) ha conosciuto una diffusione crescente, trovando applicazione in ambito ludico, educativo e professionale. Tra queste, l'esperienza immersiva nei videogiochi richiede particolare attenzione, poiché è spesso limitata da fenomeni di Motion Sickness che riducono il comfort dell'utente e influenzano negativamente l'interazione con l'ambiente virtuale.
La scelta di concentrarsi sul mondo dei videogiochi deriva dal fatto che questo contesto, particolarmente dinamico e interattivo, rappresenta un banco di prova ideale per analizzare le criticità dell'esperienza immersiva e valutarne l'impatto sul comfort e sulla fruizione.

La presente tesi si inserisce in questo scenario con l'obiettivo di progettare e realizzare un videogioco in realtà virtuale.
Il progetto nasce dalla versione desktop del gioco *Gavunia*, uno sparatutto in prima persona (First-Person Shooter, FPS) realizzato durante l'attività progettuale dell'insegnamento di *Fondamenti di Computer Graphics M*, da cui è stato effettuato un porting verso la realtà virtuale. Questo passaggio ha permesso di rimodellare le meccaniche di gioco e le interazioni per l'ambiente immersivo, garantendo coerenza con i principi di Game Design e Human-Computer Interaction e consentendo la sperimentazione di strategie per ridurre la Motion Sickness.

A tal fine, il lavoro analizza e applica le principali best practices consolidate nel campo del VR gaming, utilizzandole come riferimento per la progettazione di meccaniche di gioco, sistemi di interazione e modalità di locomozione orientate al comfort dell'utente.
Particolare attenzione è stata dedicata al bilanciamento tecnico, valutando aspetti di performance come il baking delle luci e il frame rate, al fine di garantire un'esperienza fluida e confortevole.
In questo modo è stato realizzato un prototipo funzionale e rappresentativo, capace di testare concretamente sia le interazioni previste sia le strategie di mitigazione dei fenomeni di Motion Sickness, fornendo un caso di studio completo all'interno del contesto della tesi.

L'elaborato è organizzato come segue. Nel **Capitolo 1** viene presentato il contesto della realtà virtuale, per poi illustrare i principali giochi milestone e il modo in cui questi hanno ispirato le scelte progettuali di *Gavunia VR*.
Il **Capitolo 2** introduce le buone pratiche di Game Design, con un'analisi delle Lenses di Jesse Schell e del loro impatto sulle interazioni di gioco.
Il **Capitolo 3** approfondisce i principi di Human-Computer Interaction, illustrando le modalità di interazione in VR e le raccomandazioni per una progettazione centrata sull'utente.

Il **Capitolo 4** è dedicato ai fenomeni di Motion Sickness e alle principali strategie per mitigarne gli effetti durante l'esperienza immersiva.
A partire dal **Capitolo 5**, la tesi si concentra sul caso di studio di *Gavunia*, descrivendo le caratteristiche della versione desktop.
Il **Capitolo 6** illustra il processo di porting del gioco in VR, con particolare attenzione al redesign delle interazioni e all'integrazione delle tecniche di riduzione della Motion Sickness.
Infine, il **Capitolo 7** è dedicato all'analisi delle performance del prototipo, considerando aspetti quali frame rate e ottimizzazione del rendering, al fine di garantire un'esperienza di gioco fluida e confortevole.

Capitolo 1

Introduzione alla Realtà Virtuale

La percezione umana della realtà non costituisce una riproduzione oggettiva del mondo fisico, bensì il risultato di un complesso processo di elaborazione sensoriale e cognitiva. Gli stimoli provenienti dall'ambiente vengono trasformati in impulsi nervosi e rielaborati dal cervello, che integra informazioni visive, uditive e tattili per costruire un'esperienza coerente del mondo.

Numerosi studi dimostrano che il sistema percettivo può essere influenzato o indotto a generare illusioni. Se opportunamente stimolato, il cervello può interpretare come reale anche un insieme di segnali artificialmente prodotti.
Su questo principio si fonda la **Realtà Virtuale, ovvero la creazione di ambienti simulati capaci di evocare nell'utente una sensazione di presenza all'interno di uno spazio digitale**.

La generazione degli stimoli sensoriali rappresenta tuttavia solo una parte del problema. Un sistema di Realtà Virtuale non deve limitarsi a mostrare immagini, ma deve consentire all'utente di interagire con l'ambiente simulato. Le azioni dell'utente devono essere rilevate dal sistema, elaborate e integrate all'interno di un modello computazionale del mondo virtuale, il quale ne determina il comportamento. I risultati della simulazione influenzano a loro volta la generazione degli stimoli percettivi, creando un ciclo continuo tra percezione e azione.
La simulazione può aderire alle leggi fisiche del mondo reale oppure introdurre dinamiche alternative, permettendo la creazione di ambienti realistici o immaginari. In entrambi i casi, un requisito fondamentale è il funzionamento in tempo reale: il sistema deve aggiornare continuamente la scena virtuale in risposta alle azioni dell'utente, garantendo continuità percettiva e coerenza dell'esperienza.

Un aspetto centrale della Realtà Virtuale riguarda la possibilità di generare un senso di presenza anche quando l'utente è consapevole di trovarsi in un ambiente simulato.
Esperimenti in cui utenti vengono collocati virtualmente sul bordo di un grattacielo mostrano aumento della frequenza cardiaca, tensione muscolare e comportamenti prudenti, pur sapendo di trovarsi in un ambiente sicuro.
Il poeta e filosofo Samuel Taylor Coleridge definì tale meccanismo **"willing suspension of disbelief"**, ossia la disponibilità volontaria a sospendere l'incredulità per immergersi in un'esperienza di finzione.

1.1 Panoramica sull'Extended Reality

Nel linguaggio comune, i termini Realtà Virtuale, Realtà Aumentata e Realtà Mista vengono spesso confusi o usati in modo intercambiabile. Queste tecnologie rientrano in un più ampio spettro noto come **Extended Reality (XR)**, che include tutte le forme di interazione e sovrapposizione tra il mondo fisico e quello digitale, configurandosi come un continuum di esperienze immersive. Tuttavia, ciascuna di queste definizioni descrive un diverso rapporto tra il mondo reale e l'ambiente digitale, e comprenderne le differenze è fondamentale per inquadrare correttamente le applicazioni e le potenzialità di queste tecnologie.

La **Realtà Virtuale (VR)**, come già anticipato in precedenza, consiste nella costruzione di un ambiente completamente simulato. In VR, l'utente è immerso in un mondo digitale che sostituisce la percezione diretta del mondo reale.
Applicazioni di questo tipo si ritrovano nel gaming immersivo, come nei visori che simulano ambienti fantastici, o in training professionale, ad esempio per addestrare piloti in scenari di volo complessi senza rischi reali.

La **Realtà Aumentata (AR)**, invece, parte dal mondo reale e vi aggiunge elementi digitali, come fossero strati di informazione o oggetti virtuali. Qui il mondo fisico resta dominante: vediamo la realtà così com'è, ma alcune immagini, testi o animazioni vengono sovrapposte ad essa.
Un esempio celebre è *Pokémon Go*, in cui i Pokémon appaiono sullo schermo sopra strade e piazze reali, ma non interagiscono realmente con l'ambiente circostante. Gli oggetti digitali in AR sono dunque visibili e spesso utili, ma rimangono separati dalle leggi fisiche della realtà.

La **Realtà Mista (MR)** rappresenta un passo ulteriore lungo questo continuum. In MR, gli elementi reali e virtuali non solo coesistono nello stesso spazio percettivo, ma possono anche interagire tra loro in modo coerente. Un oggetto digitale può reagire alle caratteristiche fisiche dell'ambiente reale, come illuminazione, posizione o ostacoli, e l'utente percepisce una fusione fluida tra reale e virtuale.
Questo rende MR una tecnologia particolarmente potente per simulazioni complesse, formazione professionale avanzata o applicazioni industriali dove reale e digitale devono cooperare in modo sincronizzato.
Ad esempio, un ingegnere potrebbe visualizzare un modello 3D di un motore sopra un vero macchinario e smontarne le parti virtuali, osservando come si allineano con i componenti reali.

La Figura 1.1 illustra una serie di esempi per le tre categorie appena discusse.

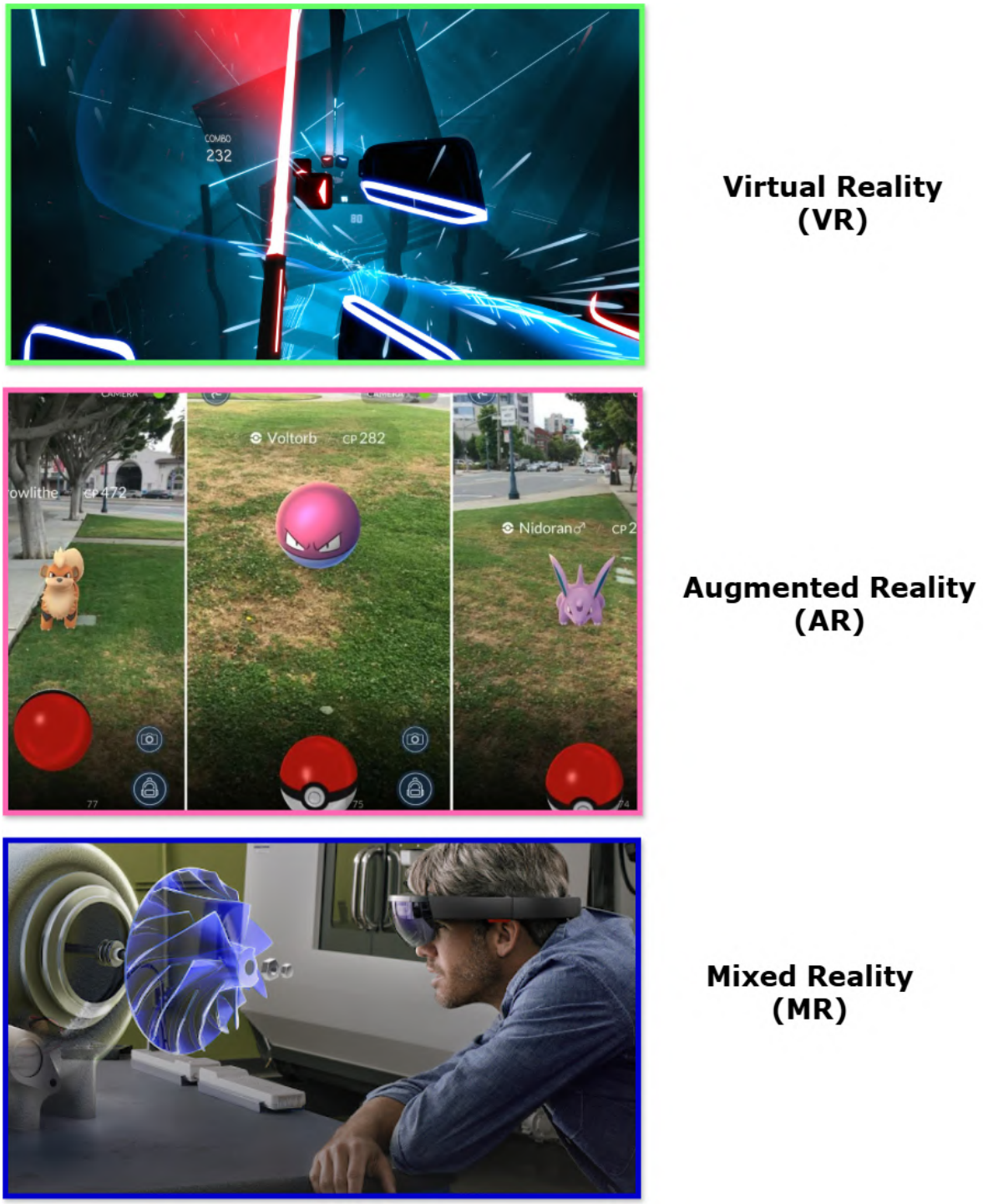

Figura 1.1: Esempi di VR, AR e MR

1.2 Tecnologia per la realtà virtuale

In questa sezione vengono presentati i principali elementi tecnologici che rendono possibile la creazione di esperienze immersive in realtà virtuale. Saranno analizzati i principi della percezione visiva su cui si basano i sistemi VR, come la visione stereoscopica, e i meccanismi che consentono di tradurre i movimenti dell'utente in azioni virtuali, attraverso tecniche di tracciamento della testa e dei dispositivi di input.

1.2.1 Stereo vision

La stereopsi, o visione stereo, è un fenomeno fondamentale per la percezione tridimensionale umana. Grazie alla presenza di due occhi separati da una certa distanza, il cervello riceve due immagini leggermente diverse della stessa scena e le elabora per ricostruire la profondità degli oggetti.

Nel funzionamento naturale, quando si fissa un punto nello spazio — ad esempio il punto A in Figura 1.2 (1) — gli occhi ruotano leggermente verso l'interno in un movimento chiamato convergenza, così da proiettare l'immagine del punto su regioni corrispondenti delle due retine. L'angolo di questa convergenza è legato alla distanza dell'oggetto osservato e contribuisce alla percezione della profondità.
Non tutti i punti della scena si trovano però alla stessa distanza del punto fissato. Punti come B e C, situati rispettivamente dietro o davanti al punto A, generano immagini che si proiettano su posizioni non corrispondenti delle due retine, producendo una differenza chiamata disparità binoculare. Questa differenza tra le immagini percepite dai due occhi viene interpretata dal cervello come segnale di profondità, permettendo la ricostruzione tridimensionale della scena.

Nella Realtà Virtuale, questa capacità percettiva viene sfruttata e manipolata per creare un'impressione di profondità su schermi bidimensionali. Come illustrato in Figura 1.2 (2), il visore mostra due immagini distinte, una per ciascun occhio, generate da punti di vista leggermente differenti. La differenza orizzontale tra le due immagini prende il nome di parallasse.
Regolando la parallasse, il sistema VR induce il cervello a fondere queste immagini in un unico punto percepito come collocato davanti o dietro lo schermo, generando così l'illusione di uno spazio tridimensionale reale. La parallasse negativa fa percepire il punto davanti allo schermo, mentre quella positiva dietro di esso.
Per una trattazione più approfondita dei fondamenti teorici della visione stereoscopica si rimanda al libro di Doerner et al. [1, pag. 42].

Un aspetto pratico importante è che, per simulare la stereopsi, la **scena virtuale deve essere renderizzata due volte da due punti di vista leggermente differenti**, corrispondenti alla posizione di ciascun occhio. Questo raddoppia di fatto il carico di lavoro grafico rispetto a una normale visualizzazione in ambienti desktop, rappresentando una delle sfide tecniche principali nella progettazione di sistemi VR performanti e realistici. Per questo motivo, nello sviluppo di applicazioni VR è necessario un lavoro di ottimizzazione maggiore rispetto alle esperienze bidimensionali, come verrà ampiamente discusso nel Capitolo 7.

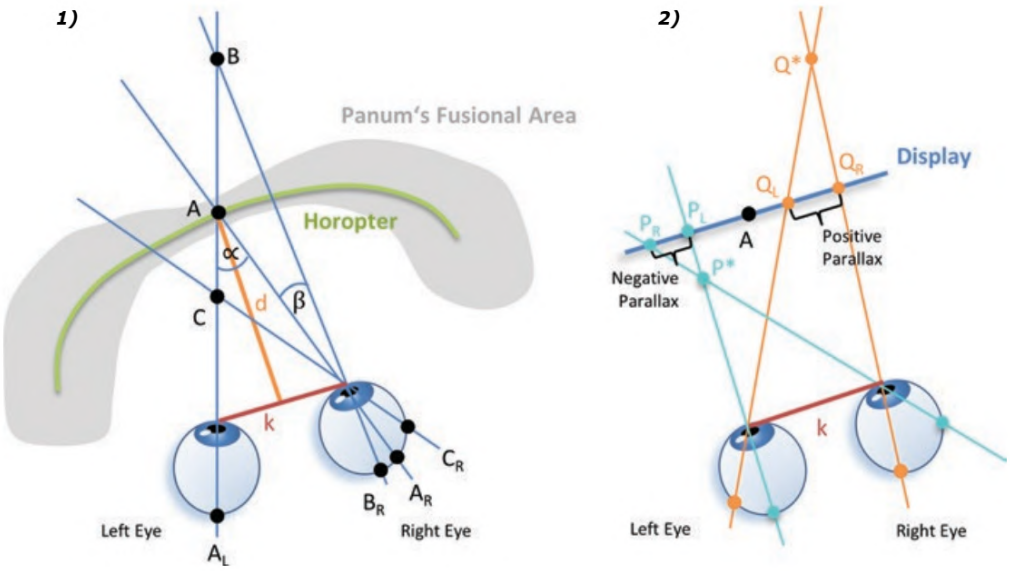

Figura 1.2: Confronto tra visione stereo nella realtà (1) e simulata in VR (2) (tratto da Doerner et al., 2022, p. 42)

1.2.2 Tracking in VR

Il tracciamento dei movimenti dell'utente è un elemento fondamentale della realtà virtuale. Il suo scopo è tradurre i movimenti reali di testa, mani e corpo in azioni corrispondenti nel mondo virtuale.
Affinché l'esperienza risulti credibile, tale traduzione deve avvenire con elevata precisione e bassa latenza: ritardi o incoerenze tra movimento reale e risposta virtuale possono compromettere la sensazione di presenza e contribuire all'insorgenza di malessere.

Esistono diverse tecnologie per il tracciamento in VR. Tra queste si trovano:

- **Tracking esterno (outside-in)**: telecamere o sensori esterni rilevano marker o LED indossati dall'utente.
- **Tracking interno (inside-out)**: telecamere montate direttamente sul visore rilevano la posizione e il movimento dell'utente rispetto all'ambiente circostante.
- **Sensori inerziali (IMU)**: accelerometri e giroscopi misurano rotazioni e variazioni di movimento ad alta frequenza.

In questa sezione l'attenzione è rivolta agli head-mounted display (HMD), in quanto rappresentano la soluzione più diffusa per l'uso personale e sono stati utilizzati per il testing del gioco *Gavunia VR* sviluppato.
Gli HMD moderni non si limitano a visualizzare la scena virtuale, ma combinano sensori inerziali e tracking inside-out per garantire sia un orientamento preciso sia una posizione spaziale corretta all'interno della scena virtuale.
Un esempio molto diffuso sul mercato consumer sono i Meta Quest, visori standalone che includono sia tracciamento della testa sia dei controller, senza necessità di telecamere esterne.

Tracciamento della testa: Gli HMD utilizzano una combinazione di sensori inerziali, come accelerometri e giroscopi, per rilevare rapidamente le variazioni di orientamento della testa. Tuttavia, i soli sensori inerziali non sono sufficienti a determinare con precisione la posizione assoluta nello spazio, poiché l'integrazione dei dati di accelerazione può introdurre errori cumulativi (drift). Per questo motivo, nei visori moderni tali sensori vengono integrati con sistemi di tracking ottico basati su telecamere, che permettono di stimare la posizione del dispositivo rispetto all'ambiente circostante.

Nei modelli dotati di eye tracking, telecamere interne rilevano il movimento degli occhi e stimano la direzione dello sguardo e il punto di fissazione. Queste informazioni possono essere utilizzate per migliorare l'interazione o per ottimizzare il rendering, ad esempio attraverso tecniche di foveated rendering [2].

Tracciamento dei motion controller: Oltre alla testa, la maggior parte dei sistemi VR commerciali traccia anche i controller manuali. Questi dispositivi integrano sensori inerziali per rilevare orientamento e variazioni di movimento e vengono localizzati nello spazio tramite il sistema di tracking del visore. Il tracciamento dei controller permette di manipolare oggetti virtuali, eseguire gesti naturali e interagire in modo coerente con l'ambiente digitale.

La Figura 1.3 mostra un esempio di tracciamento nei visori moderni.

Figura 1.3: Esempio di tracking negli HMD

1.3 Motivazioni e ambiti applicativi

La Realtà Virtuale non rappresenta soltanto una tecnologia immersiva, ma può essere interpretata come una forma avanzata di interfaccia uomo-macchina. Se la simulazione del mondo è affidata a un sistema computazionale, la VR costituisce il punto di contatto tra utente e sistema, consentendo modalità di interazione più naturali e intuitive rispetto alle tradizionali interfacce basate su tastiera e mouse. In questo senso, la ricerca in ambito VR può essere intesa come un'evoluzione delle tecniche di Human-Computer Interaction, orientata verso forme di interazione sempre più dirette e percettivamente coerenti. Maggiori dettagli saranno presentati nel Capitolo 3 dedicato.

Oltre all'ambito dell'interazione, la VR offre significativi vantaggi nella visualizzazione e nell'analisi di dati complessi. Modelli architettonici, simulazioni ingegneristiche o ambienti progettuali possono essere esplorati in modo immersivo, facilitando la comprensione spaziale e il processo decisionale.

Un ulteriore campo di applicazione riguarda la formazione e l'addestramento. Simulatori di volo, di guida o di macchinari industriali consentono di riprodurre scenari realistici riducendo costi e rischi, permettendo di esercitarsi anche in situazioni critiche difficilmente replicabili nel mondo reale. Analogamente, in ambito medico la VR può supportare sia la formazione sia il trattamento di specifiche condizioni, come le fobie, attraverso esposizioni controllate in ambienti virtuali.

Nel settore dell'intrattenimento, e in particolare nel videogioco, la VR introduce nuove possibilità espressive e interattive. L'utente non si limita a osservare uno spazio digitale, ma vi è immerso e può agire al suo interno in modo diretto. Ciò comporta sfide progettuali significative in termini di interazione, coerenza percettiva e mantenimento del senso di presenza.

Questi ambiti hanno visto nel tempo la nascita di titoli e progetti che hanno segnato tappe fondamentali nello sviluppo della realtà virtuale, offrendo esempi concreti delle sue potenzialità e delle sfide ancora da affrontare.
Nelle prossime sezioni verranno quindi analizzati alcuni giochi milestone, che hanno ispirato e guidato l'evoluzione delle esperienze VR, fornendo spunti preziosi anche per lo sviluppo di *Gavunia VR*.
La Figura 1.4 riporta una linea temporale dei principali titoli milestone che verranno analizzati.

Figura 1.4: Linea del tempo di alcuni giochi milestone presentati

1.3.1 Lone Echo (2017)

Lone Echo, sviluppato da Ready At Dawn e pubblicato nel 2017 per Oculus Rift, rappresenta uno dei titoli più significativi della prima generazione di esperienze VR ad alto budget. Si tratta di un'avventura ambientata nello spazio, in cui il giocatore interpreta Jack, un androide di servizio che affianca l'astronauta Olivia Rhodes in una missione scientifica nei pressi degli anelli di Saturno.

Uno degli aspetti più innovativi è il **sistema di locomozione in assenza di gravità, basato sul movimento delle mani**. Il movimento infatti non avviene tramite joystick o teletrasporto, ma attraverso l'interazione diretta con l'ambiente: il giocatore si sposta afferrando superfici e spingendosi nello spazio con movimenti delle mani, simulando la dinamica fisica della microgravità. Questa soluzione aumenta significativamente il senso di embodiment e presenza, poiché il movimento deriva da un'azione coerente e fisicamente motivata all'interno del mondo di gioco.

Dal punto di vista dell'interazione, *Lone Echo* propone un modello fortemente basato sulla **manipolazione diretta degli oggetti**. Pannelli, strumenti e interfacce non sono semplici elementi grafici bidimensionali, ma oggetti tridimensionali con cui l'utente interagisce tramite gesture naturali, come mostrato in Figura 1.5.
Questo approccio rafforza la credibilità dell'ambiente e anticipa molte delle linee guida che diventeranno centrali nel design VR negli anni successivi.

Anche la **componente narrativa** gioca un ruolo cruciale nel suo status di milestone. La relazione tra il giocatore e il personaggio di Olivia è costruita attraverso prossimità fisica, contatto visivo e interazioni spaziali credibili, elementi che in VR assumono un impatto emotivo molto più intenso rispetto ai media tradizionali.
Il gioco dimostra come la realtà virtuale possa potenziare il coinvolgimento narrativo non solo tramite l'immersione visiva, ma attraverso la co-presenza e la dimensione spaziale condivisa.

Figura 1.5: Esempi di interazioni in "Lone Echo"

1.3.2 Beat Saber (2018)

Beat Saber, sviluppato da Beat Games e pubblicato nel 2018, rappresenta uno dei casi più significativi di successo commerciale e diffusione della realtà virtuale nel mercato consumer. Si tratta di un "rhythm game" in cui il giocatore impugna due lame luminose virtuali e deve colpire blocchi colorati in arrivo seguendo direzione e tempo indicati dalla musica. Nonostante la semplicità concettuale, il titolo costituisce un milestone per almeno tre motivi principali: immediatezza dell'interazione, accessibilità e ottimizzazione del comfort.

Dal punto di vista dell'interazione, *Beat Saber* adotta un paradigma estremamente diretto: **i movimenti richiesti al giocatore coincidono con gesti naturali e ampi delle braccia, senza necessità di locomozione artificiale nello spazio virtuale**. L'assenza di spostamento virtuale elimina quasi completamente il rischio di motion sickness, uno dei principali ostacoli all'adozione della VR. L'esperienza si fonda su una corrispondenza chiara e coerente tra azione fisica reale e risultato virtuale, rafforzando il senso di agency e la leggibilità del sistema.

Un secondo elemento chiave è la **chiarezza del design visivo e delle regole**. I blocchi comunicano immediatamente, tramite colore e frecce direzionali, quale mano utilizzare e in quale direzione effettuare il taglio, come illustrato in Figura 1.6.
Tale codifica riduce il carico cognitivo e consente una curva di apprendimento molto rapida, rendendo il gioco accessibile anche a utenti privi di esperienza pregressa con la VR.

Dal punto di vista dell'engagement, l'integrazione tra musica, movimento corporeo e feedback audiovisivo genera uno stato di coinvolgimento elevato, spesso assimilabile al concetto di flow (si veda Sezione 2.1.6). Il corpo diventa il principale strumento di gioco, trasformando l'esperienza in una forma ibrida tra videogioco ed esercizio fisico leggero.

Il riconoscimento ottenuto, tra cui il premio "Best VR/AR Game" ai "The Game Awards 2019", testimonia il suo impatto nel settore. Tuttavia, il suo contributo più rilevante risiede nell'aver dimostrato che un'esperienza VR non necessita di complessità tecnica o narrativa per risultare efficace: un'interazione ben progettata, intuitiva e fisicamente coerente può rappresentare il principale fattore di successo.

Figura 1.6: Design del gameplay in "Beat Saber"

1.3.3 Half-Life: Alyx (2020)

Half-Life: Alyx, sviluppato e pubblicato da Valve Corporation nel 2020, rappresenta uno dei punti più alti raggiunti dalla produzione AAA in realtà virtuale. Il gioco si colloca narrativamente tra *Half-Life* e *Half-Life 2*, ponendo il giocatore nei panni di Alyx Vance durante la resistenza contro l'occupazione dei Combine.

Il suo contributo alla maturazione del medium non risiede unicamente nell'elevato livello produttivo, ma soprattutto **nell'integrazione sistematica tra gameplay tradizionale da sparatutto in prima persona e interazione tridimensionale diretta**. Il titolo dimostra come meccaniche tipiche del genere FPS possano essere ripensate in chiave embodied, sfruttando le specificità della VR anziché limitarvisi, come illustrato nell'esempio di gameplay riportato in Figura 1.7.

Uno degli aspetti più significativi riguarda il **sistema di interazione fisica con l'ambiente**. Oggetti, cassetti, leve e componenti meccaniche non sono elementi puramente decorativi, ma parti attive del sistema ludico. La ricarica delle armi, ad esempio, richiede gesti manuali sequenziali e coerenti con l'azione rappresentata (estrazione del caricatore, inserimento, armamento), trasformando una meccanica tradizionalmente automatizzata in un'azione fisicamente attiva.

Particolarmente rilevante è anche il **sistema di "afferramento a distanza" (ray grabbing)**, che consente di recuperare oggetti lontani tramite un gesto naturale di richiamo. Tale soluzione risponde a un problema tipico della VR, ovvero la necessità di evitare movimenti scomodi o eccessivamente frequenti, mantenendo al contempo coerenza diegetica e fluidità dell'esperienza. In questo modo, il gioco bilancia realismo fisico e comfort operativo, mostrando un approccio maturo alla progettazione dell'interazione.

Dal punto di vista della locomozione, il titolo offre diverse modalità (teletrasporto, movimento continuo, spostamento a scatti), adattandosi a differenti livelli di tolleranza al motion sickness. Questa flessibilità evidenzia un'attenzione progettuale orientata all'accessibilità, tema centrale nello sviluppo contemporaneo di esperienze VR.

I riconoscimenti ottenuti, tra cui il premio "Best VR/AR Game" ai "The Game Awards 2020", testimoniano il suo impatto nel settore. Tuttavia, il suo valore come milestone risiede soprattutto nell'aver dimostrato che la realtà virtuale può sostenere produzioni complesse, narrative e tecnicamente ambiziose, raggiungendo standard qualitativi comparabili ai grandi titoli non-VR.

Figura 1.7: Estratto del gameplay di "Half-Life: Alyx"

1.3.4 Contributi dei titoli analizzati e riferimenti per Gavunia VR

Nel complesso, i tre titoli analizzati evidenziano contributi differenti ma complementari allo sviluppo di giochi in realtà virtuale. *Lone Echo* ha mostrato le potenzialità di una locomozione fisicamente coerente e di un'interazione strettamente integrata con l'ambiente di gioco. *Beat Saber* ha dimostrato come un sistema di interazione semplice, immediato e confortevole possa rendere la VR accessibile a un pubblico più ampio. *Half-Life: Alyx*, infine, ha rappresentato un punto di riferimento per l'integrazione tra meccaniche da FPS e interazione fisica avanzata.

Nel caso di *Gavunia VR*, trattandosi di uno sparatutto in prima persona, il principale riferimento progettuale è stato *Half-Life: Alyx*, in particolare per quanto riguarda soluzioni come il ray grabbing, la manipolazione fisica degli oggetti e la ricarica manuale delle armi. Tali meccaniche, ormai consolidate nel design VR, verranno approfondite a livello teorico nel Capitolo 3 e riprese nel Capitolo 6, dove sarà illustrato il loro adattamento al gameplay di *Gavunia VR*.
Parallelamente, sono stati considerati anche gli apporti degli altri due titoli: da *Lone Echo* l'attenzione alla coerenza dei movimenti nello spazio virtuale, e da *Beat Saber* l'importanza della chiarezza dell'interazione e del comfort dell'utente.

Capitolo 2

Dal Game Design al VR Gaming

Il game design è la disciplina che si occupa della progettazione delle regole, delle meccaniche e dei sistemi che definiscono l'esperienza di gioco, con l'obiettivo di creare interazioni significative e coinvolgenti per il giocatore. Attraverso il game design vengono stabiliti gli obiettivi del gioco, le sfide proposte, il livello di difficoltà e le modalità con cui il giocatore riceve feedback sulle proprie azioni, contribuendo in maniera determinante alla costruzione dell'esperienza ludica complessiva.

A differenza di un approccio focalizzato esclusivamente sugli aspetti tecnici o grafici, **il game design pone al centro l'esperienza del giocatore**, analizzando come le decisioni progettuali influenzino le emozioni, la motivazione e il comportamento dell'utente all'interno del sistema di gioco. Il gioco può dunque essere interpretato come un sistema interattivo complesso, in cui ogni elemento concorre a modellare la percezione e il coinvolgimento del giocatore.

In questo quadro teorico si inserisce il contributo di Jesse Schell, autore di *"The Art of Game Design: A Book of Lenses"*[3], che propone una visione del processo di progettazione orientata all'analisi dell'esperienza del giocatore attraverso differenti punti di vista concettuali. Tale approccio risulta particolarmente rilevante nel contesto della realtà virtuale (VR), dove il livello di immersione amplifica l'impatto delle scelte progettuali sull'esperienza complessiva dell'utente.

Alla luce di queste considerazioni, il presente capitolo introduce il modello delle Lenses di Jesse Schell come strumento teorico di analisi del game design, soffermandosi in particolare su quelle prospettive ritenute maggiormente significative in relazione alla progettazione di esperienze di gioco in realtà virtuale.

La selezione delle Lenses analizzate tiene conto del contesto del presente studio, ovvero un progetto universitario finalizzato alla realizzazione di un gioco semplice ma articolato in VR, dove risultano rilevanti le Lenses legate all'esperienza del giocatore, al controllo, all'interazione e all'interfaccia. Non vengono approfondite, invece, quelle Lenses che riguardano aspetti di storytelling complesso, documentazione commerciale o strategie di marketing, in quanto esulano dagli obiettivi e dalle caratteristiche di questo progetto.

2.1 Le Lenses di Jesse Schell

Nel suo modello teorico, Jesse Schell introduce il concetto di *Lens* come strumento di supporto all'analisi e alla progettazione del gioco. La metafora della lente richiama l'idea di un filtro interpretativo attraverso il quale il designer può osservare il sistema di gioco da una specifica prospettiva, focalizzandosi su determinati aspetti dell'esperienza del giocatore.

Le Lenses non si configurano come un insieme di regole rigidamente prescritte, bensì come un framework concettuale basato su domande guida. Tali domande stimolano una riflessione critica sulle scelte progettuali adottate, favorendo un processo iterativo in cui il gioco viene costantemente valutato e rielaborato in funzione degli obiettivi prefissati. Questo approccio consente al designer di individuare eventuali criticità e di allineare in modo più coerente le meccaniche di gioco con l'esperienza che si intende offrire al giocatore.

L'insieme delle Lenses individuate da Schell copre un ampio spettro di dimensioni del game design, includendo aspetti legati alle meccaniche, alla narrazione e al coinvolgimento emotivo e cognitivo del giocatore. Tale pluralità di prospettive risulta particolarmente significativa nel contesto della realtà virtuale, dove l'elevato grado di immersione rende l'esperienza del giocatore fortemente sensibile alle decisioni progettuali.
Sebbene Schell proponga un centinaio di Lenses differenti, nel presente elaborato l'analisi sarà circoscritta a quelle ritenute maggiormente pertinenti al contesto della VR.

2.1.1 Lens of Essential Experience

La Lens of Essential Experience [3, pag. 21] rappresenta uno strumento concettuale che invita il game designer a operare un cambio di prospettiva nel processo progettuale, spostando l'attenzione dal gioco in sè all'esperienza che si intende far vivere al giocatore. In questa ottica, il design di un videogioco non dovrebbe partire dalle sue funzionalità o caratteristiche tecniche, bensì dalla definizione dell'esperienza desiderata e dagli elementi essenziali necessari a renderla efficace. Schell sottolinea come la progettazione orientata all'esperienza consenta di distinguere ciò che è fondamentale da ciò che è accessorio, **evitando di inserire elementi che, pur aumentando il livello di realismo o complessità, non contribuiscono in modo significativo all'esperienza ludica.**

Un esempio emblematico di questo approccio è fornito dal gioco *Wii Sports - Baseball*. In fase iniziale, i designer avevano l'obiettivo di riprodurre una simulazione il più possibile fedele del baseball reale; tuttavia, nel corso dello sviluppo, emerse l'impossibilità e la non necessità di rappresentarne ogni aspetto in modo accurato. L'attenzione venne quindi focalizzata sull'elemento ritenuto realmente distintivo dell'esperienza di gioco, ovvero il gesto fisico del colpire la palla attraverso il movimento del controller. Di conseguenza, numerosi elementi tipici del baseball tradizionale, come la struttura completa delle partite o alcune regole avanzate, furono eliminati in quanto non essenziali all'esperienza che il gioco intendeva offrire. Questo esempio evidenzia come il realismo non costituisca un valore intrinseco, ma assuma rilevanza solo nella misura in cui contribuisce all'esperienza desiderata.
La Figura 2.1 riporta un estratto della presentazione del Dr. Christopher Hopkins, che sintetizza in forma schematica le scelte di game design appena discusse.

Figura 2.1: Processo di game design di Wii Sports - Baseball

Nel contesto della realtà virtuale, la Lens of Essential Experience assume un ruolo particolarmente rilevante, poichè la VR amplifica l'impatto delle scelte progettuali sull'esperienza del giocatore, in quanto coinvolge direttamente il corpo dell'utente. Aspetti come presenza, immersione ed embodiment diventano centrali, rendendo fondamentale l'identificazione degli elementi davvero essenziali dell'esperienza ludica. Applicare questa Lens al design VR consente di concentrare l'attenzione su interazioni fisiche e percettive significative, evitando l'introduzione di dettagli o meccaniche che, pur apparendo realistici, non contribuiscono all'esperienza desiderata.

2.1.2 Lens of Fun

La Lense of Fun [3, pag. 27] affronta uno degli aspetti centrali ma più complessi del game design: il concetto di divertimento. Pur essendo una qualità desiderabile nella maggior parte dei giochi, il fun risulta difficile da definire in modo univoco e sistematico. Schell analizza diverse teorie del gioco e del comportamento ludico, evidenziandone i limiti: alcune, come quella dell'energia in eccesso di Schiller, riducono il gioco a uno sfogo; altre, come le definizioni basate sul piacere o sulla libertà all'interno di regole, risultano troppo ampie e rischiano di includere attività non propriamente ludiche.
Da queste riflessioni emerge un punto cruciale: **il divertimento non dipende tanto dalla natura dell'attività, quanto dall'atteggiamento del giocatore nei confronti di essa**. Schell sottolinea la distinzione tra gioco e lavoro non in termini di azione, ma di motivazione: un'attività può essere percepita come gioco quando viene affrontata liberamente e per scelta personale, mentre assume le caratteristiche del lavoro quando è vissuta come obbligatoria.

L'esempio di Rico Medellin, operaio in una fabbrica, ne è emblematico: un compito ripetitivo di assemblaggio diventa gioco quando l'operaio si pone l'obiettivo di battere il proprio record, trasformando un dovere in una sfida personale. In questo modo, la libertà di definire gli obiettivi e la possibilità di monitorare i risultati generano un'esperienza ludica intensa, anche in contesti normalmente percepiti come lavoro.

Sulla base di queste considerazioni, Schell propone una definizione operativa di gioco: *"manipolazione che indulge la curiosità"*. Questa formulazione mette in evidenza due elementi fondamentali: l'azione volontaria e l'interazione, attraverso le quali il giocatore esplora, sperimenta e apprende, generando coinvolgimento e piacere.

Applicare la Lens of Fun nel game design significa quindi valutare criticamente **quali elementi del gioco stimolino la curiosità del giocatore, incoraggino l'interazione volontaria e favoriscano un senso di libertà, evitando sensazioni di obbligo o ripetitività**. In pratica, questa lente invita a domandarsi: quali meccaniche spingono il giocatore a esplorare? Quali sfide risultano stimolanti senza essere frustranti? In che modo le regole del gioco permettono di creare esperienze gratificanti e scoperte personali?

Nel contesto della realtà virtuale, la Lens of Fun assume una rilevanza ancora maggiore. Qui, il divertimento è strettamente legato alla qualità delle interazioni corporee e sensoriali: la manipolazione diretta degli oggetti, il feedback immediato e la possibilità di esplorare liberamente l'ambiente virtuale amplificano la percezione di libertà e curiosità. Al contrario, controlli poco intuitivi, ritardi nei feedback o vincoli eccessivi riducono rapidamente il piacere di gioco, evidenziando come il design dell'interazione influenzi direttamente la percezione ludica.

2.1.3 Lens of Endogenous Value

La Lens of Endogenous Value invita il designer a riflettere su ciò che i giocatori considerano veramente significativo all'interno di un gioco, su come rendere quegli elementi più preziosi e su come il valore percepito si leghi alle motivazioni interne dei giocatori [3, pag. 32]. Secondo Schell, non tutti gli oggetti o i punteggi presenti in un gioco possiedono valore intrinseco: un elemento diventa rilevante solo quando supporta gli obiettivi reali del giocatore o risponde alle sue motivazioni principali.

Il confronto tra i giochi *Bubsy* e *Sonic the Hedgehog 2* evidenzia il concetto: nel primo caso i gomitoli aumentano solo il punteggio, senza incidere sul completamento dei livelli, perdendo rapidamente rilevanza; nel secondo, gli anelli proteggono dai danni e garantiscono vite extra, risultando direttamente funzionali agli obiettivi di sopravvivenza e progresso. La differenza mostra come un elemento acquisisca valore solo quando supporta concretamente le motivazioni principali del giocatore.

Applicare la Lens of Endogenous Value nel game design significa **valutare attentamente quali elementi, oggetti o ricompense del gioco siano percepiti come significativi dai giocatori, come supportino le loro motivazioni interne e come favoriscano il senso di progresso e gratificazione**. In questo modo si garantisce che ogni interazione abbia uno scopo chiaro, evitando azioni percepite come arbitrarie o prive di impatto.

Applicata alla realtà virtuale, la Lens of Endogenous Value assume un ruolo particolarmente rilevante. In VR, il giocatore interagisce fisicamente con oggetti e ambienti, e la

percezione di valore di queste interazioni influenza direttamente il divertimento e il senso di immersione. Se un gesto, come afferrare un oggetto o attivare un meccanismo, non ha ricadute concrete sugli obiettivi del giocatore, l'azione rischia di risultare gratuita o frustrante. Al contrario, quando le azioni fisiche e le ricompense sono direttamente collegate alle motivazioni interne, il feedback corporeo immediato e la manipolazione dell'ambiente rafforzano il legame tra azione e risultato, generando un'esperienza significativa e gratificante. Esempi contemporanei includono giochi come *Beat Saber*, dove colpire blocchi al ritmo della musica ha conseguenze dirette sulla performance, o *Half-Life: Alyx*, in cui la manipolazione fisica di oggetti e strumenti è essenziale per risolvere puzzle e progredire nella storia.

2.1.4 Lens of the Elemental Tetrad

Schell, per introdurre questa lens, propone un modello concettuale volto a scomporre il videogioco nei suoi elementi fondamentali, denominato *Elemental Tetrad.*[3, pag. 41]
Questo modello nasce dall'esigenza di superare approcci riduzionisti al game design, che tendono a privilegiare un singolo aspetto del gioco — spesso le meccaniche — a scapito di una visione sistemica dell'esperienza ludica. Secondo Schell, ogni gioco è il risultato dell'interazione equilibrata di quattro elementi essenziali: meccaniche, storia, estetica e tecnologia. Nessuno di essi è intrinsecamente più importante degli altri; al contrario, è proprio la loro interdipendenza a determinare la qualità dell'esperienza di gioco:

- Le **meccaniche** comprendono regole, procedure e sistemi che definiscono le azioni possibili e le loro conseguenze, introducendo interattività e agency, intesa come la percezione, da parte del giocatore, di poter compiere azioni intenzionali e di influenzare in modo significativo lo stato e l'evoluzione del sistema di gioco.

- La **storia** è la sequenza di eventi che si sviluppa durante il gioco, sia in forma predefinita sia emergente, e trova significato attraverso il dialogo con le meccaniche.

- L'**estetica** riguarda la dimensione sensoriale ed emotiva dell'esperienza, contribuendo in modo diretto all'immersione.

- La **tecnologia**, intesa in senso ampio (hardware, software o supporti materiali), costituisce il mezzo espressivo che abilita e vincola gli altri elementi.

Schell dispone i quattro elementi in una struttura a rombo per evidenziare un ulteriore concetto chiave: **il gradiente di visibilità**. La tecnologia tende a essere l'elemento meno percepito direttamente dal giocatore, mentre l'estetica è il più immediato; meccaniche e storia occupano una posizione intermedia. Questa disposizione non implica una gerarchia di valore, ma serve a ricordare che anche gli elementi meno visibili hanno un impatto profondo sull'esperienza complessiva.
La Figura 2.2 illustra graficamente questa disposizione, rappresentando i quattro elementi secondo la struttura a rombo e rendendo visibile il gradiente appena descritto.

Da questo modello deriva la Lens of the Elemental Tetrad, uno strumento analitico che invita il designer a esaminare il gioco considerando separatamente ciascun elemento e successivamente la loro interazione come sistema unitario. Le domande proposte da Schell mirano a verificare la presenza di tutti e quattro gli elementi, a individuare

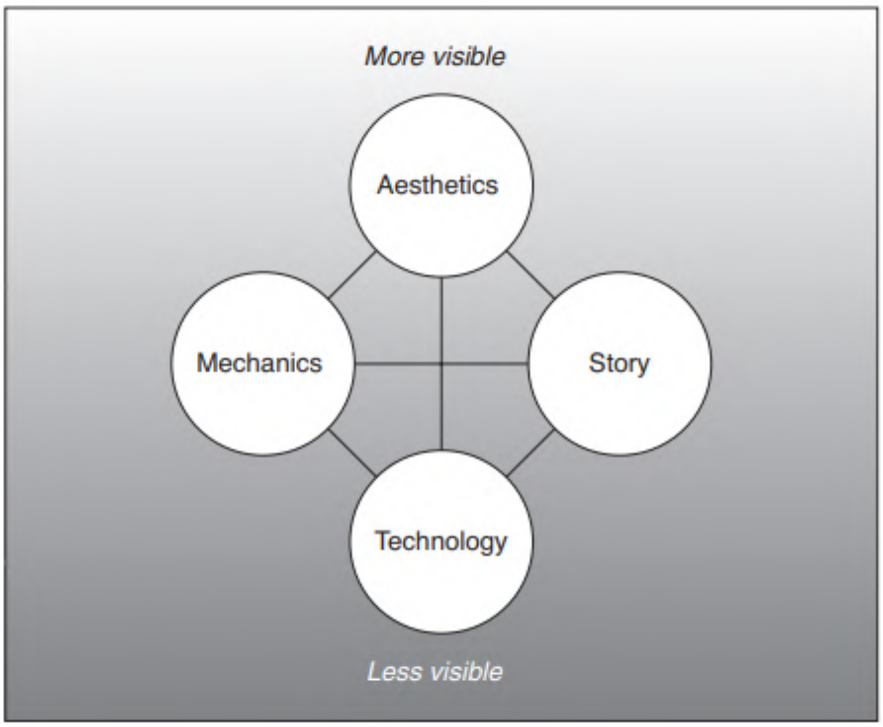

Figura 2.2: Elemental Tetrad (tratto da Schell, 2008, p. 42)

eventuali squilibri e a valutare il grado di armonia tra di essi. Applicare questa lente **consente di comprendere come le diverse componenti del gioco collaborino nel trasmettere un tema comune e nel costruire un'esperienza coerente per il giocatore**.

Un esempio significativo è *Space Invaders* (Taito, 1978). Il gioco mostra come tecnologia, meccaniche, storia ed estetica operino in modo integrato. Le possibilità tecniche dell'epoca consentirono la gestione di un'armata aliena in avanzamento continuo, generando meccaniche basate su pressione crescente e aumento progressivo della difficoltà. La narrazione minimale — la difesa della Terra da un'invasione — attribuisce senso all'azione, mentre l'estetica visiva e il ritmo sonoro incalzante amplificano la tensione.
Nel complesso, Space Invaders dimostra come l'armonia tra i quattro elementi del Elemental Tetrad sia determinante nel definire l'esperienza di gioco, confermando l'utilità della Lens come strumento di analisi del game design.

Applicata alla realtà virtuale, la Lens of the Elemental Tetrad assume un'importanza ancora più marcata, poiché l'esperienza del giocatore dipende in modo diretto dall'equilibrio tra elementi spesso gestiti separatamente. In VR, meccaniche, estetica, storia e tecnologia non possono essere considerate componenti indipendenti: qualsiasi incoerenza tra di esse rischia di compromettere l'immersione e di provocare disorientamento o discomfort.
Dal punto di vista tecnologico, la VR introduce vincoli e possibilità specifiche — come il tracciamento dei movimenti, la latenza, il campo visivo e i dispositivi di input — che influenzano profondamente le meccaniche di gioco. Meccaniche progettate senza tenere conto dei limiti fisici del corpo o delle capacità dell'hardware possono risultare faticose, innaturali o persino causare motion sickness. Al contrario, quando la tecnologia è scelta

e sfruttata in funzione delle meccaniche, l'interazione diventa intuitiva e credibile, rafforzando la sensazione di presenza.
L'estetica in VR non è soltanto un elemento visivo, ma contribuisce in modo decisivo alla percezione spaziale e corporea del giocatore. Scala, profondità, illuminazione e audio spazializzato devono essere coerenti sia con le meccaniche sia con la tecnologia utilizzata, poiché discrepanze percettive possono interrompere l'illusione di trovarsi in un ambiente tridimensionale credibile.
Anche la storia assume in VR una configurazione peculiare: la forte sensazione di presenza rende problematiche narrazioni eccessivamente lineari o intrusive. La Lens evidenzia come la narrazione debba emergere attraverso l'interazione e l'ambiente, piuttosto che essere imposta dall'esterno, affinché non entri in conflitto con le meccaniche e con il coinvolgimento fisico del giocatore.

In questo contesto, la Lens of the Elemental Tetrad si rivela uno strumento essenziale per valutare la coerenza complessiva di un'esperienza VR, permettendo di analizzare come le scelte tecnologiche influenzino meccaniche, estetica e narrazione, e come il loro allineamento contribuisca a un'esperienza immersiva, confortevole e significativa. Titoli come *Half-Life: Alyx* o *Lone Echo* mostrano come un attento bilanciamento dei quattro elementi sia fondamentale per sfruttare appieno il potenziale espressivo della realtà virtuale.

2.1.5 Lens of Holographic Design

La Lens of Holographic Design si pone in continuità con la Lens of the Elemental Tetrad, ampliandone la prospettiva. Se la Tetrade consente di analizzare la coerenza tra meccaniche, storia, estetica e tecnologia, l'approccio olografico mette tali elementi in relazione diretta con l'esperienza vissuta dal giocatore [3, pag. 45]. L'attenzione si sposta quindi dalla sola struttura del sistema agli effetti percettivi ed emotivi che essa genera.

Schell utilizza il termine holographic per indicare la necessità di una visione integrata, in cui ogni scelta progettuale è valutata in base al suo impatto sull'esperienza complessiva. Utilizzare questa lente significa **osservare il gioco nella sua interezza, mantenendo contemporaneamente consapevolezza sia degli elementi strutturali sia degli effetti che essi producono sul giocatore**, interrogandosi non solo su come il gioco è costruito, ma su quali effetti produce e perché.

Nel contesto della realtà virtuale, tale prospettiva risulta particolarmente significativa, poiché immersione e presenza rendono immediatamente percepibili eventuali incoerenze tra meccaniche, tecnologia ed estetica. La lente olografica consente quindi di valutare in che modo ciascun elemento contribuisca o ostacoli sensazioni di comfort, coinvolgimento e presenza.

2.1.6 Lens of Flow

La Lens of Flow [3, pag. 118] invita il designer a progettare "l'esperienza mentale" del giocatore, concentrandosi sulla capacità del gioco di catturare e mantenere l'attenzione. Il cervello umano seleziona attivamente gli stimoli rilevanti, ignorando quelli secondari; un gioco efficace deve quindi organizzare obiettivi, sfide e feedback in modo da sostenere la concentrazione.

Schell richiama il concetto di **flow** elaborato dallo psicologo Mihaly Csikszentmihalyi, ovvero uno stato di coinvolgimento totale caratterizzato da concentrazione intensa, perdita della percezione del tempo e gratificazione intrinseca. Il flow emerge quando sono presenti obiettivi chiari, feedback immediato e un equilibrio tra difficoltà e abilità del giocatore.

Il concetto di flow si articola ulteriormente nel modello del **flow channel**, ossia il margine ideale compreso tra noia e ansia/frustrazione. Se la difficoltà dell'attività è troppo bassa rispetto alle abilità del giocatore, si verifica la noia; se è eccessiva, subentra ansia o frustrazione. Il compito del designer è quindi quello di guidare il giocatore lungo questo canale, aumentando gradualmente le sfide in linea con il miglioramento delle sue abilità. La Figura 2.3 evidenzia questo canale, definito dall'insieme dei punti nel grafico che bilanciano le abilità del giocatore (asse X) con le sfide del gioco (asse Y).

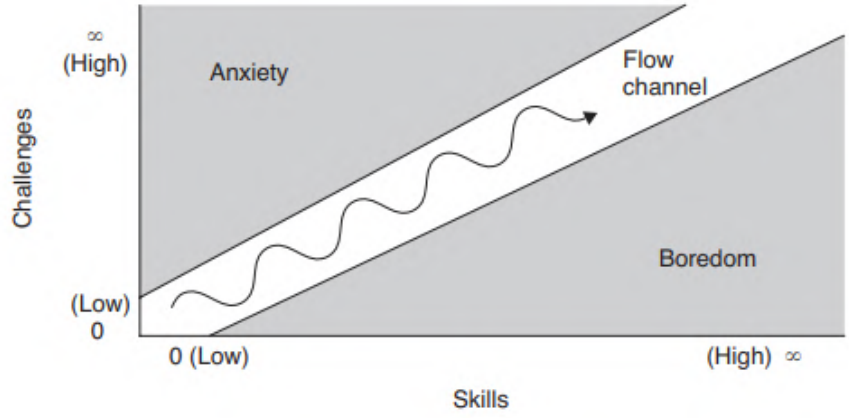

Figura 2.3: Flow Channel (tratto da Schell, 2008, p. 121)

Nei videogiochi, questa dinamica viene spesso gestita tramite livelli a difficoltà crescente o attraverso cicli di tensione e rilascio, nei quali periodi di intensa sfida sono seguiti da momenti di gratificazione e relativa facilità. Questo alternarsi di difficoltà e ricompensa genera piacere e varietà, stimolando la motivazione intrinseca del giocatore e favorendo la crescita delle sue abilità. Il flow, quindi, non è solo uno stato di divertimento immediato, ma anche un meccanismo attraverso cui il giocatore apprende, migliora e si sente gratificato, mantenendo un coinvolgimento duraturo con il gioco.

La Lens of Flow invita il designer a osservare attentamente cosa cattura l'attenzione del giocatore e a verificare che obiettivi, sfide e feedback siano adeguati. **Comprendere e gestire il flow consente di progettare esperienze più gratificanti, mantenendo il giocatore costantemente impegnato nel canale ottimale tra sfida e abilità**.

Nel contesto della realtà virtuale, la Lens of Flow assume un ruolo particolarmente cruciale, poiché la VR è caratterizzata da elementi come il movimento corporeo, l'interazione spaziale e il senso di embodiment che introducono nuove dimensioni nella gestione del flow. Le sfide devono essere calibrate non solo in base alle abilità cognitive del giocatore, ma anche in relazione alla sua capacità motoria e alla tolleranza alla stimolazione sensoriale. Un'azione troppo complessa o non intuitiva può facilmente portare a frustrazione, mentre una sfida troppo semplice rischia di interrompere l'immersione e generare noia.

2.1.7 Lens di controllo e interfaccia

Nel game design, l'interfaccia costituisce il punto di mediazione tra intenzione del giocatore e risposta del sistema. La qualità di questa mediazione incide direttamente sulla continuità dell'esperienza: un'interfaccia coerente e prevedibile rafforza il senso di controllo, mentre una risposta imprecisa o ambigua interrompe l'immersione.
Per descrivere questo aspetto, Schell definisce quattro Lens differenti che consentono al game designer di analizzare l'interazione non come un insieme di componenti isolate, ma come un sistema integrato orientato all'esperienza: [3, pag. 222]

- La **Lens of Control** invita il designer a interrogarsi sul grado di controllo percepito dal giocatore. Non si tratta semplicemente di garantire che i comandi rispondano correttamente agli input, ma di fare in modo che **le azioni producano effetti comprensibili, proporzionati e coerenti con le aspettative**. Un'interfaccia efficace consente al giocatore di attribuire correttamente successi e fallimenti alle proprie decisioni, rafforzando il senso di agency.
  Ad esempio, in un gioco d'azione, un sistema di movimento impreciso o poco reattivo può generare frustrazione non perché il gioco sia difficile, ma perché il giocatore non percepisce più una relazione diretta tra input e risultato.

- Questa relazione viene ulteriormente articolata distinguendo tra interfaccia fisica e interfaccia virtuale. La **Lens of Physical Interface** riguarda i dispositivi attraverso cui il giocatore interagisce con il gioco, come controller, tastiera, mouse o sistemi di motion tracking. Applicare questa Lens significa **valutare quanto il mapping tra gesto fisico e azione di gioco sia naturale e adeguato al tipo di esperienza proposta**. Un esempio evidente è rappresentato dai controller basati sul movimento, come il Wiimote o i controller VR, che permettono di associare direttamente un gesto corporeo a un'azione ludica, riducendo la distanza cognitiva tra giocatore e sistema.

- La **Lens of Virtual Interface**, invece, si concentra sugli elementi intermedi che non appartengono direttamente al mondo di gioco, ma ne facilitano la comprensione e la gestione, come HUD, menu, indicatori di stato o feedback visivi e sonori. Un'interfaccia virtuale ben progettata **fornisce le informazioni necessarie nel momento opportuno, senza sovraccaricare il giocatore o distrarlo dall'azione**. Ad esempio, una barra dell'energia sempre visibile può essere utile in un gioco frenetico, mentre in un'esperienza più esplorativa può risultare invasiva e ridurre il senso di immersione.

- Queste componenti convergono nella **Lens of Transparency**, che rappresenta l'obiettivo ideale del design dell'interfaccia. Un'interfaccia trasparente è un'interfaccia che "scompare", diventando così familiare da non richiedere più attenzione conscia. In questa condizione, il giocatore non pensa più ai comandi o ai menu, ma agisce direttamente nel mondo di gioco, descrivendo l'esperienza in prima persona piuttosto che in termini di input e output. La trasparenza non implica semplicità assoluta, ma apprendibilità progressiva: un buon esempio è un sistema di controllo che inizialmente richiede attenzione, ma che, una volta interiorizzato, permette azioni complesse senza sforzo cognitivo.

Nel contesto della realtà virtuale, le Lens di interfaccia assumono caratteristiche particolari.
La Lens of Physical Interface diventa centrale: il giocatore interagisce con il mondo di gioco attraverso controller di movimento, tracciamento delle mani o gesture naturali, e ogni azione deve risultare immediatamente riconoscibile nel mondo virtuale. Ad esempio, afferrare un oggetto con le mani virtuali deve corrispondere esattamente alla percezione tattile o visiva attesa, altrimenti l'illusione del mondo VR si rompe.
La Lens of Virtual Interface richiede attenzione speciale nella progettazione di HUD, indicatori e menu, che devono essere posizionati nello spazio tridimensionale in modo da non interrompere il senso di presenza. Un classico esempio è un indicatore di salute o energia fluttuante vicino al corpo dell'avatar invece che in un angolo dello schermo: è accessibile e visibile senza distogliere lo sguardo dal mondo di gioco.
Infine, la Lens of Transparency assume un ruolo ancora più critico in VR: l'interfaccia deve "scomparire" completamente alla percezione del giocatore, consentendo un'immersione totale. Ritardi negli input, feedback visivi incoerenti o elementi dell'interfaccia mal posizionati interrompono immediatamente il senso di presenza, evidenziando quanto la trasparenza in VR non sia solo desiderabile, ma fondamentale per la qualità dell'esperienza.
La Figura 2.4 riassume le Lenses di interfaccia appena descritte, mostrando come interagiscono tra di loro.

Sebbene tali Lens siano formulate all'interno di una prospettiva di game design, esse mostrano evidenti punti di contatto con i principi della Human-Computer Interaction, in particolare per quanto riguarda il controllo percepito, la trasparenza dell'interazione e il ruolo del feedback. Nel presente capitolo, questi aspetti sono affrontati come strumenti progettuali finalizzati all'esperienza di gioco, mentre nel capitolo successivo verranno ripresi e approfonditi all'interno di un quadro teorico HCI più ampio, che consentirà di analizzare in modo sistematico i processi di interazione uomo-sistema, anche alla luce delle specificità introdotte dalla realtà virtuale.

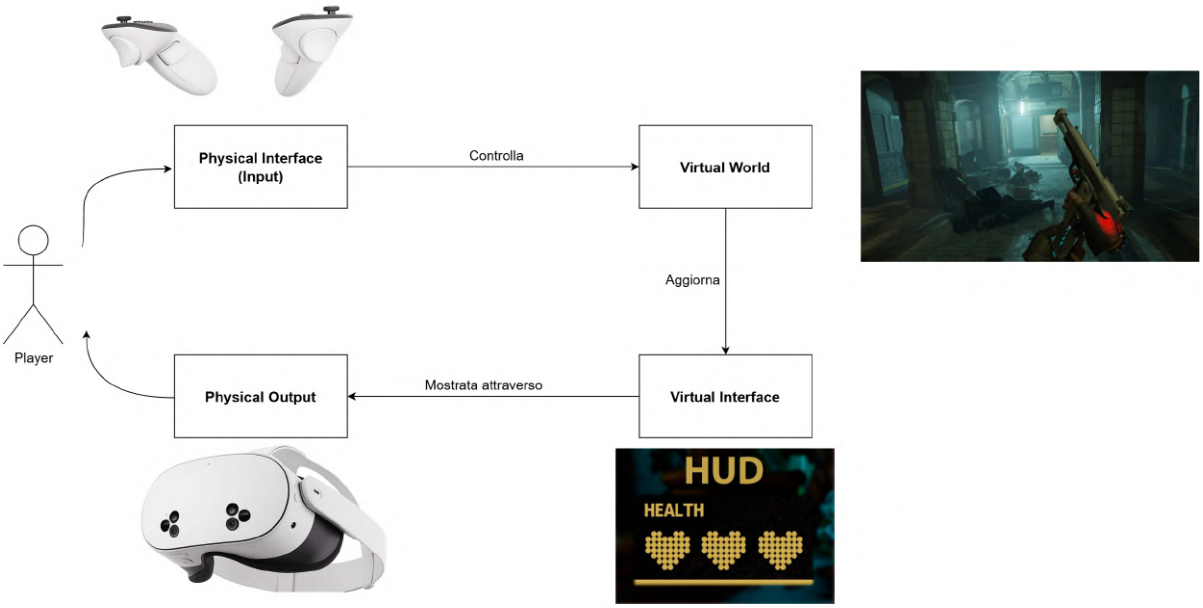

Figura 2.4: Schema riassuntivo delle Lenses di interfaccia

2.1.8 Relazioni tra le Lenses analizzate

Le Lenses di game design introdotte da Jesse Schell, descritte nelle sezioni precedenti, non costituiscono strumenti indipendenti né una sequenza prescrittiva di fasi progettuali, ma differenti prospettive attraverso cui osservare e valutare un'unica esperienza di gioco. Ciascuna lens enfatizza un aspetto specifico del design — dall'esperienza essenziale del giocatore, al divertimento, al valore generato dalle meccaniche, fino allo stato di flow e all'interazione mediata dall'interfaccia — ma il loro reale potenziale emerge quando vengono considerate come parti di un sistema interconnesso.

La seguente sezione ha dunque l'obiettivo di ricapitolare le lenses affrontate, evidenziandone le relazioni e le interdipendenze, al fine di offrire una visione d'insieme coerente del framework teorico proposto da Schell.
A tal proposito, la Figura 2.5 sintetizza tali relazioni:

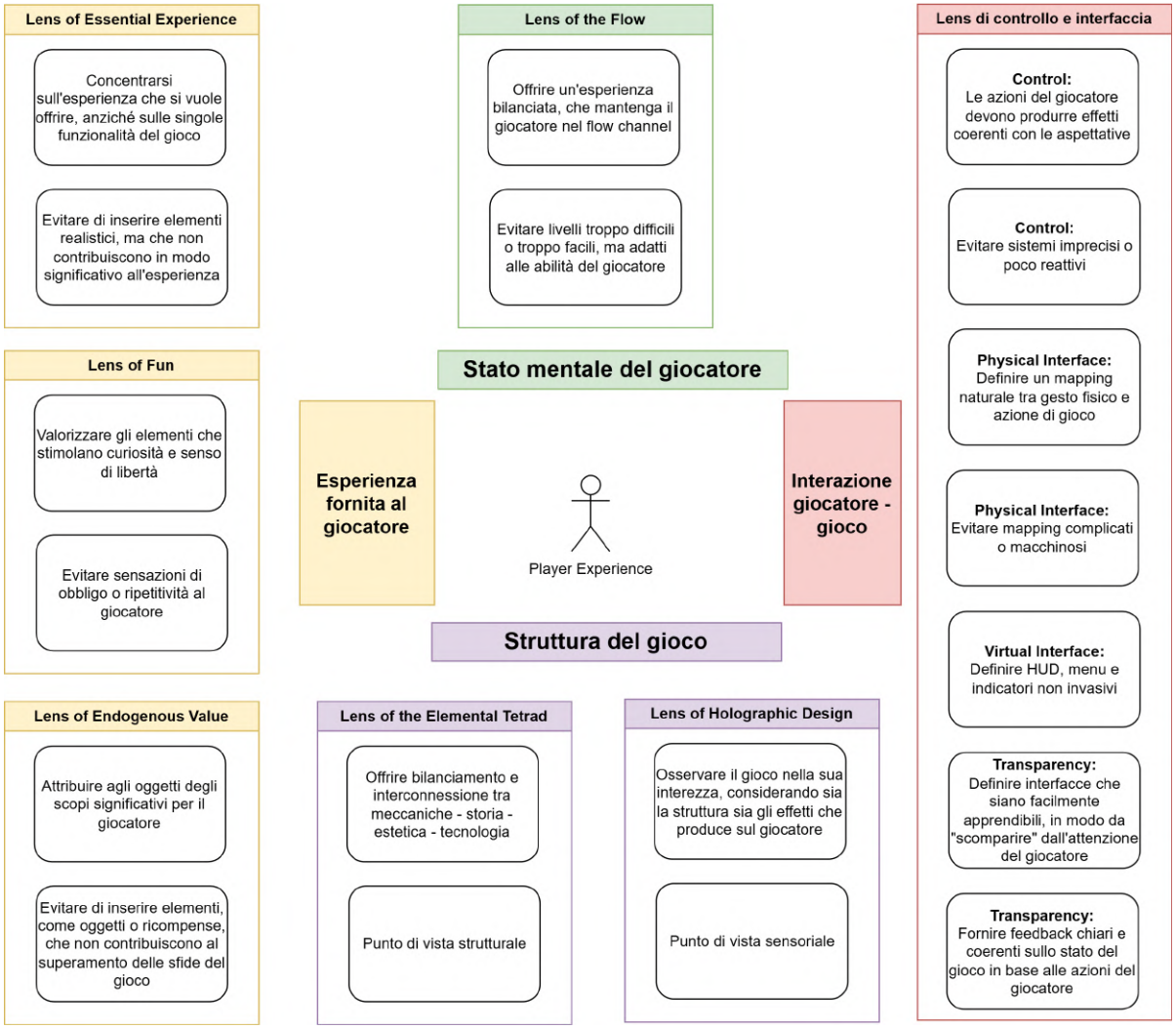

Figura 2.5: Riassunto e relazioni tra le Lenses trattate

Capitolo 3

Human Computer Interaction

La Human-Computer Interaction (HCI) rappresenta l'insieme delle discipline e dei principi che governano l'interazione tra esseri umani e sistemi informatici, con l'obiettivo di progettare interfacce che risultino efficaci, efficienti e intuitive.
Nell'ambito della realtà virtuale (VR), la HCI assume un ruolo centrale poiché questi sistemi mirano a creare modalità di interazione più naturali rispetto alle interfacce tradizionali. Una possibile definizione di HCI VR, infatti, può essere formulata in termini di obiettivo: *la realizzazione di interfacce uomo-computer in grado di consentire un'interazione intuitiva con un ambiente simulato tridimensionale, superando i limiti delle interfacce bidimensionali tradizionali.*(Doerner et al., 2022)[1, pag. 14]

Per comprendere l'evoluzione dell'interazione uomo-computer, è utile richiamare il paradigma dominante nelle interfacce desktop, rappresentato dalle GUI (Graphical User Interfaces) e dallo stile di interazione WIMP (Windows, Icons, Menus, Pointing).
Questo paradigma, nato per attività di elaborazione documentale, risulta tuttavia inefficiente quando applicato alla manipolazione di contenuti tridimensionali.

Un esempio esplicativo è il compito di riposizionare un oggetto nello spazio 3D: in un ambiente VR tale operazione può essere svolta in modo diretto e naturale afferrando e spostando l'oggetto con la mano, mentre in un'interfaccia 2D tradizionale è necessario scomporre il compito in più sottotask (es. spostamento sul piano xy, poi spostamento lungo l'asse z). Oltre al maggiore sforzo motorio, questa scomposizione richiede un ulteriore impegno cognitivo, poiché l'utente deve apprendere e gestire lo stato di controllo del sistema (ad esempio, come indicare che il movimento del mouse debba essere interpretato come traslazione lungo l'asse z). Di conseguenza, il paradigma WIMP richiede anche un maggiore sforzo di apprendimento per svolgere compiti tridimensionali, risultando meno adatto alle esigenze della VR.

In questo contesto, **VR può essere interpretata come un'interfaccia post-WIMP, ovvero un sistema che si basa su tecniche di interazione capaci di sfruttare conoscenze e abilità già acquisite dall'utente nel contesto delle interazioni quotidiane con oggetti fisici**. Questo approccio consente di ridurre il carico cognitivo e lo sforzo di apprendimento, poiché l'utente può utilizzare schemi motori e aspettative comportamentali già presenti nella propria esperienza pregressa.

Inoltre, il concetto di metafora riveste un ruolo significativo nella progettazione, in quanto permette di spiegare e comprendere l'interfaccia attraverso analogie con esperienze di vita quotidiana. Nel paradigma WIMP, esempi noti di metafore sono il desktop,

le cartelle e le operazioni di taglia-e-incolla. Anche il termine "Virtual Reality" stesso può essere interpretato come una metafora, poiché suggerisce un'analogia con la realtà fisica, implicando che gli oggetti presenti nell'ambiente simulato si comportino in modo realistico e che siano supportate forme di interazione naturali. La metafora della VR implica inoltre che l'utente sia immerso nell'ambiente simulato, sperimentandolo "dall'interno" anziché osservandolo "dall'esterno" attraverso una finestra, come avviene nei sistemi desktop tradizionali.
La Figura 3.1 visualizza questo cambiamento di paradigma nelle modalità di interazione, confrontando i sistemi desktop tradizionali con le interfacce di realtà virtuale.

Figura 3.1: Metafora HCI (tratto da Doerner et al., 2022, p. 16)

La realizzazione di interazioni naturali e intuitive rappresenta dunque un obiettivo primario nella progettazione di sistemi VR, sebbene non sempre tale obiettivo sia completamente raggiunto nelle applicazioni contemporanee, come osservato da Doerner et al. (2022) citando Robert Stone. [1, pag. 15]
Nonostante ciò, grazie all'uso di dispositivi di input e output 3D, le interazioni nelle attuali applicazioni VR risultano comunque generalmente più naturali rispetto a quelle basate su interfacce 2D convenzionali.

A partire dalle considerazioni introduttive presentate finora, la sezione successiva approfondirà i principi fondamentali dell'interazione nei sistemi di realtà virtuale, così come descritti nel libro *"Virtual and Augmented Reality (VR/AR)"* di Doerner et al. [1].

3.1 Principi di interazione in VR

Un ambiente virtuale diventa significativo per l'utente solo nel momento in cui risulta interattivo. Rendere un sistema VR interattivo significa consentire all'utente di agire sull'ambiente e ricevere risposte in tempo reale, instaurando un continuo scambio di informazioni tra l'essere umano e il sistema che governa la componente virtuale dell'esperienza. Questo processo di comunicazione tra uomo e computer rientra nell'ambito della Human-Computer Interaction (HCI), disciplina che si occupa della progettazione, valutazione e implementazione di sistemi interattivi, nonché dei fenomeni cognitivi e percettivi coinvolti nel loro utilizzo.

Un aspetto centrale della HCI è la progettazione user-oriented delle interfacce, che si fonda non solo su conoscenze informatiche, ma anche su contributi provenienti da ambiti quali la psicologia, le scienze cognitive, l'ergonomia, la sociologia e il design.
Negli ultimi anni, la ricerca in HCI ha inoltre ampliato il proprio campo di interesse includendo il concetto di user experience, che considera l'insieme delle sensazioni, delle

emozioni e delle percezioni maturate dall'utente durante l'interazione, comprendendo elementi quali l'estetica dell'interfaccia e il piacere d'uso.

3.1.1 Tipologie di interazione

Un tema centrale nella progettazione dell'interazione in ambienti virtuali riguarda la **scelta tra interazione naturale e interazione magica**.[1, pag. 203]
L'interazione naturale mira a riprodurre il più fedelmente possibile le modalità di azione del mondo reale, consentendo all'utente di sfruttare competenze motorie e percettive già acquisite. Esempi tipici includono il movimento fisico reale per la navigazione o la manipolazione di oggetti virtuali entro la portata delle braccia.
L'interazione magica, al contrario, introduce meccaniche che superano i limiti della realtà fisica, come il teletrasporto istantaneo o la manipolazione a distanza di oggetti, ampliando le possibilità funzionali del sistema. La scelta tra questi due approcci dipende dal contesto d'uso, dall'esperienza desiderata e dal grado di naturalità ritenuto opportuno.

Anche quando si opta per un elevato livello di naturalità, l'interazione in VR non può mai essere completamente diretta come nel mondo reale, poiché è sempre mediata da strati tecnologici quali dispositivi di input, sistemi di tracking e processi di elaborazione che introducono una distanza tra azione fisica e risposta del sistema.

In questo contesto, il principio della **direct manipulation** assume un ruolo fondamentale: un'interfaccia supporta la manipolazione diretta quando l'utente può modificare una rappresentazione grafica degli oggetti ricevendo un feedback immediato e continuo sulle proprie azioni. In VR, la direct manipulation contribuisce in modo significativo a mascherare la mediazione tecnologica, riducendo la distanza cognitiva tra azione e risultato e rafforzando la percezione di controllo sull'ambiente virtuale.

3.1.2 System Control

Il system control nei sistemi di realtà virtuale comprende tutte quelle interazioni che consentono all'utente di modificare lo stato del sistema. [1, pag. 203]

A differenza delle interazioni orientate al contenuto dell'ambiente virtuale (3.1.1), il system control opera a un livello meta, ponendosi in una posizione concettualmente più distante dall'esperienza immersiva. Nelle interfacce bidimensionali tradizionali, tali funzioni vengono generalmente realizzate attraverso elementi consolidati come menu, pulsanti, toolbar, comandi testuali o interazioni di tipo drag and drop. Tuttavia, il trasferimento diretto di queste soluzioni in ambienti tridimensionali risulta problematico, poiché manca un supporto bidimensionale naturale sul quale collocare tali elementi e definirne le modalità di utilizzo.

Un problema concettuale rilevante del system control in VR riguarda il conflitto con la *willful suspension of disbelief*, ovvero la disponibilità dell'utente ad accettare il mondo virtuale come coerente e credibile. Molti comandi di sistema non trovano un equivalente diretto nel mondo reale, rendendo difficile una loro implementazione naturale o una rappresentazione realistica uno-a-uno. Di conseguenza, numerosi sistemi adottano soluzioni ad hoc, come menu tridimensionali fluttuanti o pulsanti che compaiono improvvisamente nello spazio virtuale, la cui usabilità risulta tuttora poco esplorata dal punto di vista scientifico.

In letteratura, è possibile individuare cinque principali categorie di tecniche di system control:

- I **menu** costituiscono la tecnica più diffusa e possono essere classificati in base alla loro collocazione nello spazio, alla modalità di rappresentazione e alla tecnica di selezione utilizzata.
  Un menu può essere fisso nello spazio virtuale, associato a un oggetto specifico (menu contestuale), ancorato all'utente — ad esempio alla mano — oppure collegato a oggetti reali. Dal punto di vista della rappresentazione, i menu possono essere organizzati in una, due o tre dimensioni, adottando strutture lineari, bidimensionali o volumetriche. A seconda di tali scelte progettuali, variano anche le modalità di selezione delle voci, con un impatto diretto su accessibilità e carico cognitivo.

- I **3D widgets** sono strettamente correlati alle tecniche viste per i menu e sono oggetti tridimensionali inseriti nell'ambiente virtuale con lo scopo esplicito di controllare il sistema. A differenza degli elementi del mondo virtuale, essi non rappresentano contenuti della scena, ma fungono da strumenti di interazione. La loro geometria tridimensionale rende visibile la funzionalità interattiva e fornisce affordance, ovvero indicazioni percettive sulla modalità d'uso (ad esempio rotazione, spostamento o pressione) che guidano l'utente nell'utilizzo.

- I **tangibles (o props)** sono oggetti fisici reali che l'utente può impugnare per attivare modalità di interazione o strumenti nel sistema VR. A differenza dei controller generici, i tangibles sono progettati per rappresentare o richiamare funzionalmente strumenti specifici del mondo reale, sfruttando la corrispondenza tra forma fisica e funzione virtuale.
  Nel contesto del system control, il loro ruolo non è tanto quello di rappresentare contenuti del mondo virtuale, quanto quello di modificare lo stato o la modalità di funzionamento dell'applicazione. Un esempio può essere un'impugnatura che, quando afferrata, attiva la modalità "strumento", consentendo di manipolare oggetti virtuali con un comportamento specifico (ad esempio un "pennello" o un "trapano"). La presenza di un oggetto fisico conferisce un feedback tattile immediato e facilita l'apprendimento delle funzionalità, ma limita il numero di strumenti disponibili e la flessibilità del sistema, rendendo questa soluzione più adatta a contesti specifici come simulazioni di training o applicazioni professionali.

- I **comandi vocali** permettono un'interazione hands-free e possono essere efficacemente combinati con altre modalità di input. Un vantaggio significativo è l'assenza di elementi grafici aggiuntivi che potrebbero occludere la scena virtuale. D'altra parte, l'utente deve apprendere e ricordare i comandi disponibili, poiché essi non sono generalmente rappresentati in modo esplicito nell'interfaccia.

- Infine, i **gesti** rappresentano una tecnica di system control potente e naturale, soprattutto se combinata con il riconoscimento vocale o con altri input. Analogamente ai comandi vocali, i gesti non richiedono elementi grafici aggiuntivi, ma rendono meno immediatamente visibili le funzionalità disponibili, che devono essere apprese dall'utente.

3.1.3 Selezione

La selezione rappresenta una delle operazioni fondamentali nell'interazione con mondi virtuali. Si tratta di un compito che consiste nel determinare un punto, un'area o un volume nello spazio circostante, oppure un oggetto virtuale per eseguire operazioni successive come spostamento, modifica o attivazione di funzionalità.

Tuttavia, la selezione in un contesto tridimensionale presenta sfide significativamente maggiori rispetto alle interfacce 2D tradizionali. Il testo di Doerner et al. evidenzia come la complessità della selezione in VR sia dovuta a diversi fattori. In primo luogo, l'input in 3D introduce un numero superiore di gradi di libertà, rendendo più difficile il controllo preciso rispetto all'interazione su superfici 2D. In secondo luogo, l'occlusione costituisce un problema rilevante: gli oggetti possono essere nascosti da altri elementi della scena, impedendo una visione diretta del target e quindi compromettendo la capacità di selezionarlo con precisione. Infine, la fase di sviluppo delle interfacce VR può soffrire di una minore maturità metodologica: le tecniche non sempre sono testate e standardizzate, per cui possono rimanere problemi di usabilità non identificati.

Per attenuare tali difficoltà, una strategia frequentemente adottata consiste nel limitare la selezione a una superficie di interazione, sia essa parallela al piano dell'immagine o incorporata nello spazio virtuale come un pannello 2D. Questo approccio sfrutta la familiarità dell'utente con le modalità operative classiche del computer, semplificando il compito di selezione e riducendo la complessità cognitiva richiesta. Al contempo, **la realtà virtuale offre l'opportunità di esplorare modalità di selezione innovative, che si discostano dalle convenzioni tradizionali e si avvicinano maggiormente alle esperienze del mondo reale**.

Un esempio significativo di applicazione combinata di diverse strategie di selezione è rappresentato da *Half-Life: Alyx*, che adotta un approccio ibrido adattando le modalità di interazione al contesto.
Gli oggetti vicini vengono manipolati tramite afferramento diretto, coerente con l'esperienza fisica e orientato alla massimizzazione della presenza. Per oggetti distanti, il sistema introduce le "Gravity Gloves", una tecnica world-based non realistica che consente di attirare elementi lontani attraverso un gesto intenzionale, estendendo virtualmente la portata dell'utente senza ricorrere a interfacce bidimensionali. Le funzioni di menu e inventario sono invece implementate tramite pannelli 2D nello spazio virtuale, privilegiando precisione e controllo.
La Figura 3.2 illustra due scene di gioco in cui si osserva l'integrazione delle modalità di selezione 2D e 3D, bilanciando realismo, usabilità ed efficienza.

Figura 3.2: Esempi di selezione nel gioco VR "Half-Life: Alyx"

3.1.3.1 Pointing

Una componente centrale di molte tecniche di selezione in VR è l'uso di un dispositivo di puntamento (pointing device), che può essere il dito dell'utente oppure un controller dedicato, come un hand controller o un 3D mouse. La selezione avviene quando l'utente mira il target e conferma l'azione: il sistema deve quindi determinare quale entità tridimensionale è stata selezionata. Questo processo non è banale, perché spesso richiede di tradurre un input bidimensionale (la posizione sullo schermo o nel campo visivo) in una posizione tridimensionale e di identificare quale oggetto si trova in quel punto.

La tecnica di base per realizzare questa operazione è chiamata *picking*. Esistono due approcci principali: uno basato sul colore e uno basato sul raggio.
Nel primo caso, il sistema genera un'immagine "nascosta" della scena in cui ogni oggetto è disegnato con un colore unico; individuando il pixel verso cui l'utente punta, è possibile risalire all'oggetto corrispondente (color picking).
Il secondo approccio, più preciso e oggi più diffuso, utilizza il ray-casting: un raggio viene proiettato dalla posizione dell'osservatore (o dall'estensione del dito) verso la scena e si calcolano le intersezioni con la geometria 3D. L'oggetto selezionato è quello il cui punto di intersezione è più vicino all'osservatore. Con tecniche di ottimizzazione, come le gerarchie di volumi di delimitazione, questa operazione può essere eseguita in tempo reale, rendendo il ray-casting una tecnica fondamentale anche per altri aspetti dell'interazione, come il rilevamento delle collisioni.

Per aiutare l'utente durante la selezione, è essenziale fornire un feedback visivo. Questo può avvenire evidenziando l'oggetto selezionato oppure mostrando un cursore o un punto di mira (target). In VR, il cursore 3D rappresenta l'equivalente del puntatore del mouse nelle interfacce 2D, permettendo di puntare oggetti anche a distanza.
La Figura 3.3 rappresenta un esempio di selezione di un oggetto, mettendo in evidenza il feedback visivo che supporta l'interazione dell'utente.

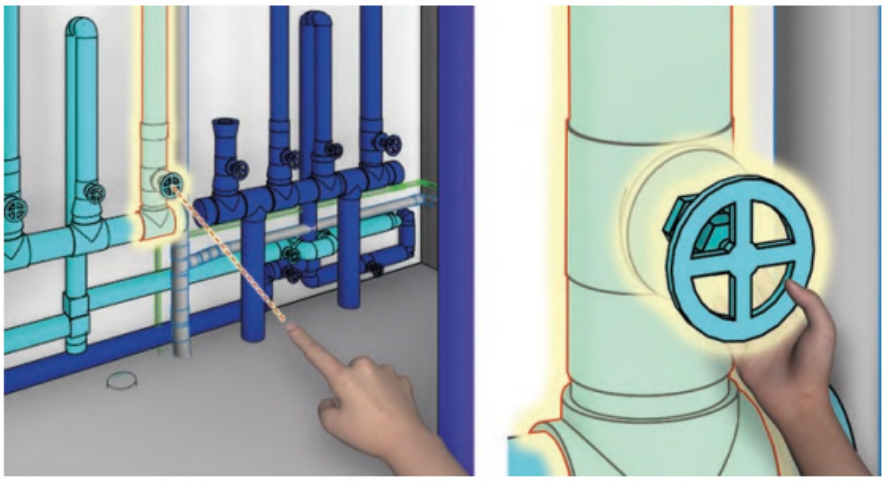

Figura 3.3: Selezione in VR e feedback visivo (tratto da Doerner et al., 2022, p. 207)

3.1.3.2 Esempi di tecniche di selezione

In VR esistono diverse tecniche per selezionare e manipolare oggetti nello spazio tridimensionale, ciascuna con caratteristiche e scenari d'uso differenti.
Una delle più comuni è il ***ray-casting***, già vista in precedenza, in cui un raggio virtuale parte dal cursore 3D o dalla mano dell'utente e individua gli oggetti lungo la sua traiettoria. Gli oggetti attraversati dal raggio diventano candidati per la selezione e generalmente viene scelto quello più vicino. Questa tecnica è molto efficace per interazioni ravvicinate, ma la precisione diminuisce con la distanza, rendendola meno adatta a oggetti lontani. Una variante, chiamata ***flashlight technique***, sostituisce il raggio singolo con un cono simile al fascio di una torcia, considerando anche la distanza dal centro del cono per selezionare l'oggetto più rilevante.
A partire da questi principi, sono state sviluppate ulteriori tecniche:

- **go-go technique**: estende virtualmente la portata del braccio oltre il limite fisico mediante uno scaling non lineare, consentendo di raggiungere oggetti distanti senza spostamento corporeo.
- **HOMER (Hand-centered Object Manipulation Extending Raycasting)**: combina ray-casting e manipolazione diretta, trasferendo la mano virtuale sull'oggetto selezionato per migliorarne precisione e controllo, soprattutto a distanza.
- **World-in-Miniature (WIM)**: rappresenta l'ambiente in miniatura, permettendo la selezione da una prospettiva esterna.

La limitata trattazione di queste tecniche nel presente elaborato è dovuta al fatto che, pur essendo valide in altri contesti, risultano meno adatte a un gioco FPS come nel mio caso di studio.
Tuttavia, per i lettori interessati ad esplorare maggiormente queste soluzioni, si rimanda al libro di Doerner et al. [1, pag. 211].

3.1.4 Navigazione

La navigazione rappresenta uno dei compiti fondamentali e allo stesso tempo più complessi negli ambienti VR. Analogamente alla navigazione nel mondo reale, essa implica la capacità di determinare la propria posizione nello spazio, pianificare un percorso e compiere le azioni necessarie per raggiungere una destinazione.
In ambito di Human-Computer Interaction, la navigazione è un'attività centrale, poiché consente all'utente di esplorare, comprendere e interagire con ambienti complessi. In particolare, la qualità della navigazione influisce direttamente sul senso di presenza e sull'efficacia complessiva dell'esperienza immersiva.

La navigazione viene comunemente suddivisa in due componenti concettualmente distinti ma strettamente interdipendenti: il ***wayfinding*** e il ***traveling***. Tale distinzione permette di separare gli aspetti cognitivi della navigazione da quelli motori, facilitando l'analisi e la progettazione delle tecniche di interazione.

Il wayfinding rappresenta la componente cognitiva della navigazione e ha come obiettivo principale la costruzione di una *cognitive map*, ovvero una rappresentazione mentale semplificata dello spazio virtuale. Questo processo è in larga parte inconscio e soggettivo: utenti diversi possono sviluppare mappe cognitive differenti dello stesso ambiente, anche a parità di esperienza. Per questo motivo, il ruolo del design non è quello di imporre una specifica rappresentazione spaziale, ma piuttosto di supportare l'utente nella progressiva acquisizione della conoscenza spaziale.
Questa conoscenza coinvolta nel wayfinding può essere suddivisa in tre tipologie:

- La *landmark knowledge* riguarda la conoscenza di punti di riferimento salienti e facilmente riconoscibili all'interno dell'ambiente. I landmark sono generalmente oggetti unici o particolarmente evidenti, che fungono da ancore spaziali e risultano più facili da ricordare rispetto ad altri elementi.

- La *route knowledge*, detta anche conoscenza procedurale, descrive la conoscenza della sequenza di azioni o passaggi necessari per seguire un determinato percorso. Essa è fortemente orientata all'azione e non richiede una comprensione globale della struttura dell'ambiente.

- La *overview knowledge* riguarda invece la conoscenza topologica dell'ambiente e fornisce una visione d'insieme delle relazioni spaziali tra luoghi, percorsi e punti di riferimento. Si tratta della forma di conoscenza più astratta e completa, la cui acquisizione richiede generalmente più tempo e l'integrazione delle altre due tipologie.

Il traveling costituisce la componente motoria della navigazione e comprende le azioni di base necessarie a modificare la posizione e l'orientamento dell'utente o della camera virtuale. La possibilità di muoversi nello spazio virtuale è un prerequisito essenziale per molte altre tecniche di interazione tridimensionale, come la manipolazione degli oggetti o l'ispezione dettagliata di elementi della scena.

3.1.4.1 Tecniche per controllare il Traveling

A differenza dei sistemi di Realtà Aumentata, nei quali il movimento fisico dell'utente nello spazio reale costituisce implicitamente il meccanismo principale di navigazione, i sistemi di Realtà Virtuale richiedono una progettazione esplicita delle tecniche di controllo del traveling. In VR, infatti, l'utente può rimanere fisicamente fermo mentre la camera virtuale si muove all'interno dell'ambiente, rendendo necessarie tecniche specifiche per la gestione del movimento.

Una delle modalità più diffuse di locomozione in VR, chiamata ***virtual reality locomotion***, è basata sulla definizione di un vettore di direzione lungo il quale viene spostata la camera virtuale. Questo approccio è comunemente implementato tramite dispositivi di input tridimensionali, come flystick, wand o i controller degli HMD commerciali, che consentono di rilevare in modo accurato la posizione e l'orientamento della mano dell'utente.

Un caso particolarmente rappresentativo di questa categoria è la *smooth locomotion*, tipica dei giochi in prima persona, in cui l'utente si muove in modo continuo e fluido nello spazio virtuale tramite lo stick del controller, mantenendo il controllo della direzione e della velocità di movimento. Sebbene tali tecniche risultino relativamente semplici da implementare e intuitive da comprendere, esse presentano lo svantaggio di vincolare una delle mani al controllo del movimento, riducendo la disponibilità per altre forme di interazione.

Un'altra tecnica ampiamente utilizzata è il ***eye-directed control***, nel quale la direzione di movimento è allineata alla direzione di sguardo dell'utente. Nei sistemi VR immersivi, tale direzione viene tipicamente determinata tramite il tracciamento della testa o, nei casi più avanzati, dello sguardo. È importante sottolineare che, nella pratica, il movimento non viene attivato automaticamente dal semplice orientamento dello sguardo al fine di evitare spostamenti involontari dovuti ai naturali movimenti esplorativi della testa o degli occhi, ma l'avanzamento viene generalmente abilitato tramite un comando separato, come la pressione di un pulsante o l'attivazione di una modalità di locomozione. Un esempio di questa tecnica è rappresentato da un ambiente VR in cui l'utente guarda nella direzione desiderata e avanza solo mantenendo premuto un pulsante sul controller. Questo approccio risulta naturale e di facile apprendimento, ma presenta un limite strutturale, poiché vincola il movimento alla direzione di osservazione, riducendo la possibilità di muoversi in una direzione mentre si esplora visivamente l'ambiente in un'altra.

Infine, un caso particolare di controllo del traveling è rappresentato dalla ***teleportation***, una tecnica che consente all'utente di spostarsi istantaneamente tra due posizioni senza attraversare lo spazio intermedio. Un esempio tipico è costituito da applicazioni VR in cui l'utente seleziona una posizione sul pavimento tramite un raggio o un arco proiettato dal controller e viene immediatamente trasportato in quel punto.
La teleportation è diventata una tecnica ampiamente adottata nei sistemi VR consumer, poiché riduce in modo significativo il rischio di motion sickness ed è supportata direttamente dai principali motori di sviluppo. Tuttavia, l'assenza di un movimento continuo può compromettere la percezione delle distanze e ridurre la continuità spaziale, con possibili effetti sul senso di presenza.

Nel complesso, le tecniche di controllo del traveling sono generalmente consolidate e facilmente implementabili nei modern sistemi VR, ma introducono una criticità fondamentale legata alla discrepanza tra movimento visivo e assenza di movimento fisico reale. Questo conflitto tra il sistema visivo e altre modalità percettive, come il senso dell'equilibrio, rappresenta una delle principali cause di discomfort negli ambienti immersivi. Per attenuare tale conflitto, sono state proposte strategie progettuali che coinvolgono maggiormente il corpo dell'utente, come l'uso di gesti naturali (ad esempio il movimento alternato delle braccia per simulare la camminata) o l'introduzione di feedback visivi stabilizzanti durante il movimento. Tali approcci evidenziano come la progettazione delle tecniche di traveling in VR debba tenere conto non solo degli aspetti di controllo, ma anche delle caratteristiche percettive, cognitive e fisiologiche dell'utente (ulteriori informazioni sulla relazione tra movimento virtuale e motion sickness verranno affrontanti nel Capitolo 4 dedicato).
Di seguito, la Figura 3.4 offre una categorizzazione delle tecniche di controllo precedentemente analizzate, mentre la Figura 3.5 ne mostra un'applicazione pratica implementata nel gioco *Half-Life.Alyx*.

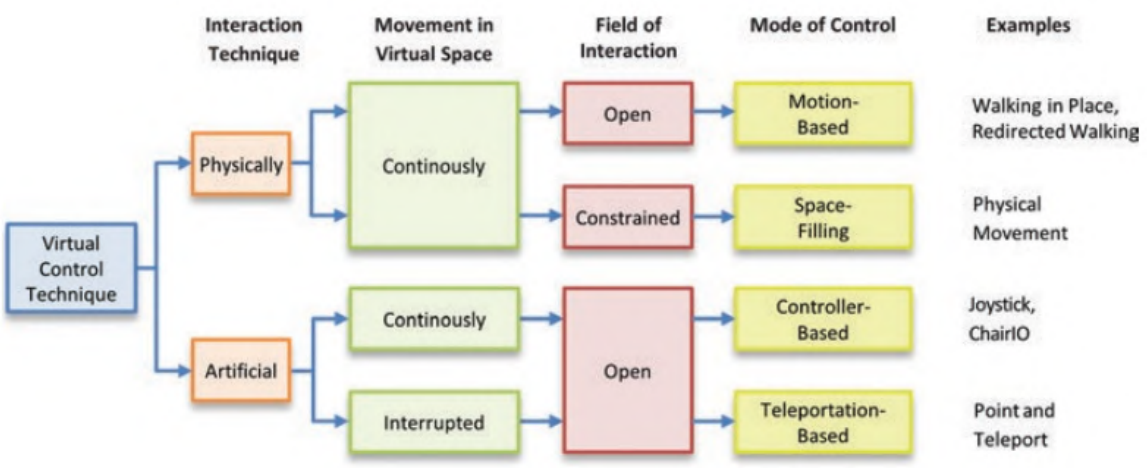

Figura 3.4: Categorizzazione delle tecniche di controllo di traveling in VR. (tratto da Doerner et al., 2022, p. 217)

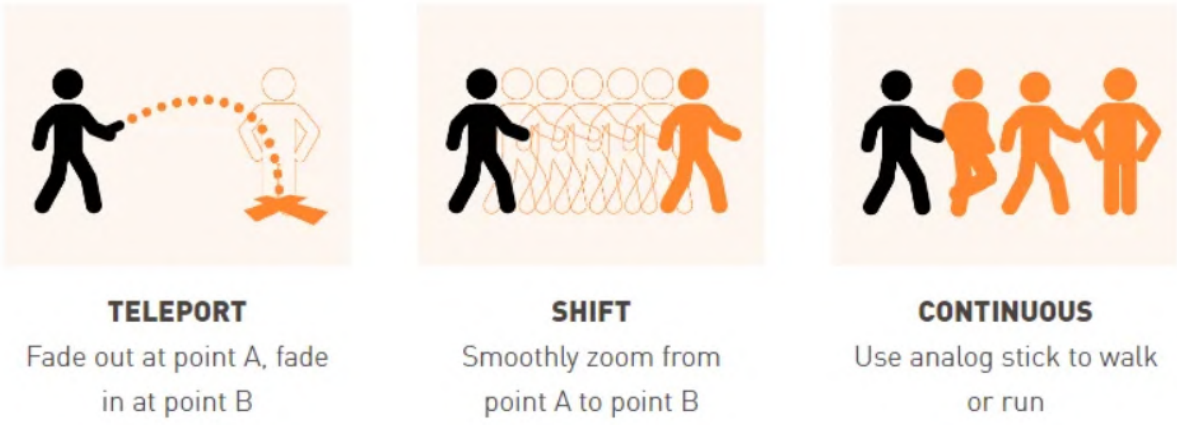

Figura 3.5: Esempi di movimento proposti nel gioco VR "Half-Life: Alyx"

3.2 Interaction Design

Dopo aver analizzato i principi fondamentali dell'interazione in VR, risulta chiaro che la loro semplice conoscenza non garantisce di per sé un'esperienza fluida e intuitiva.
L'Interaction Design si occupa di trasformare questi principi in soluzioni concrete, stabilendo come combinare le diverse modalità di interazione e fornire feedback chiari e coerenti, in modo da ridurre errori, frustrazione e disorientamento dell'utente.
Tuttavia, questo processo non è banale, soprattutto in VR. Rispetto a interfacce desktop o web, le interfacce immersive presentano alcune differenze chiave che rendono la progettazione più complessa:

- **Mancanza di standardizzazione**: mentre nel desktop e nel web esistono convenzioni consolidate (widget, controlli, layout), in VR tali standard sono ancora in fase di definizione. Di conseguenza, molte tecniche devono essere progettate e implementate ex novo, aumentando la dipendenza dal contesto hardware e dalle scelte progettuali.

- **Interazione hardware-software strettamente integrata**: la scelta di controller, sensori e modalità di input influenza direttamente le possibilità di interazione. La progettazione dell'hardware deve quindi essere considerata parte integrante del processo di design, al pari dello sviluppo software.

- **Limitata disponibilità di toolkit e strumenti di prototipazione**: mentre nel web esistono ambienti consolidati per prototipare rapidamente interfacce, in VR strumenti analoghi sono meno diffusi. Ciò rende la prototipazione e la valutazione più costose e richiede un processo iterativo più attento.

Queste specificità rendono evidente che l'Interaction Design in VR non può limitarsi a una scelta teorica delle tecniche, ma è necessario **adottare un processo di progettazione iterativo e centrato sull'utente**, che consenta di esplorare alternative, prototipare rapidamente e validare le soluzioni con test reali.
Per questo motivo, nella sezione successiva verrà presentato un approccio sistematico basato sull'Human-Centered Design (HCD), che rappresenta un metodo consolidato per garantire usabilità e coerenza nelle interazioni immersive.

3.2.1 HCD - Human Centered Design

L'Interaction Design in VR non può limitarsi alla selezione di tecniche "buone sulla carta", in quanto la complessità sensoriale e l'assenza di standard consolidati rendono necessaria una progettazione iterativa e orientata all'utente. In questo contesto, il Human-Centered Design (HCD) rappresenta un approccio sistematico e consolidato per sviluppare sia singole tecniche di interazione sia sistemi completi con elevata usabilità.

Un elemento centrale dell'HCD è l'adozione di procedure iterative, considerate la best practice nello sviluppo di interfacce. I processi iterativi suddividono lo sviluppo in fasi ripetute, in cui il risultato di una iterazione viene valutato tramite test con utenti reali e i feedback raccolti guidano le modifiche nelle iterazioni successive. Tale approccio è particolarmente importante in VR, dove molte decisioni progettuali dipendono da percezioni e comportamenti difficili da prevedere senza osservazione diretta.

Nella letteratura esistono diversi modelli iterativi, alcuni dei quali riconosciuti come standard ISO (ad esempio DIN EN ISO 9241-210, spesso citato insieme al suo predecessore ISO-13407, o ISO/PAS 18152).
Nella pratica, i progetti VR tendono a personalizzare il processo in base alle specificità del contesto applicativo e alle risorse disponibili.

3.2.1.1 Fasi nella ISO 9241- 210

Secondo la norma ISO 9241-210, il processo HCD è strutturato in quattro attività principali, ripetute ciclicamente fino al raggiungimento di un risultato soddisfacente:

1. **Analisi del contesto d'uso**: In questa fase vengono analizzati e documentati gli utenti target, i compiti da supportare e l'ambiente applicativo. Nel caso di VR, è particolarmente importante includere anche l'analisi dell'ambiente tecnico, ad esempio i sensori disponibili e le modalità di input. L'analisi del contesto non è un documento statico: durante lo sviluppo può essere continuamente aggiornato, soprattutto quando emergono nuove tecnologie o nuovi requisiti.

2. **Specifica dei requisiti**: Sulla base del contesto d'uso vengono definiti requisiti funzionali e non funzionali, tenendo conto non solo delle esigenze dell'utente e del committente, ma anche di vincoli tecnici e normativi (ad esempio requisiti di sicurezza o ergonomia). Nel campo VR, dove spesso si esplorano soluzioni innovative, risultano utili tecniche esplorative come lo *scenario-based design*: brevi storie descrivono possibili interazioni, consentendo a utenti e esperti di fornire feedback già in una fase iniziale, anche quando il sistema non è ancora implementato.

3. **Concept, design e interazione**: Questa attività riguarda la creazione concreta delle soluzioni. In VR è particolarmente efficace una strategia di *rapid prototyping iterativo*: nelle prime iterazioni si utilizzano sketch, storyboard e mock-up non implementati, che vengono poi valutati e migliorati nelle iterazioni successive.

4. **Valutazione**: La valutazione è un elemento imprescindibile dell'HCD. Le soluzioni implementate vengono testate con utenti reali e i risultati dei test guidano le iterazioni successive.

La Figura 3.6 illustra il processo appena descritto, evidenziandone la natura iterativa finalizzata al raggiungimento del risultato desiderato.

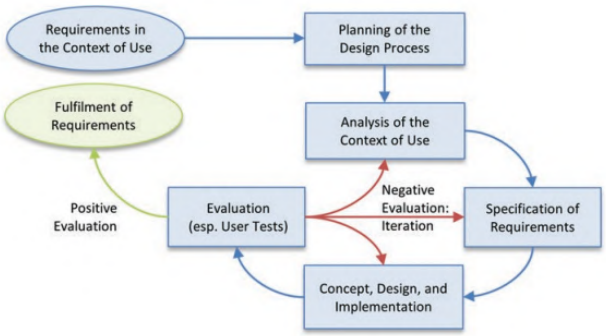

Figura 3.6: Processo iterativo in accordo con ISO 9241-210. (tratto da Doerner et al., 2022, p. 226)

3.2.2 Vincoli, modalità e feedback

La progettazione dell'interazione nei sistemi di realtà virtuale richiede un'attenta gestione dei gradi di libertà messi a disposizione dell'utente. In un ambiente tridimensionale completo, un'interazione può teoricamente coinvolgere sei gradi di libertà: tre relativi alla posizione nello spazio e tre relativi all'orientamento. Tuttavia, non tutte le attività richiedono l'utilizzo simultaneo di tutte queste dimensioni di controllo. In molte operazioni di selezione o puntamento, ad esempio, la rotazione dell'oggetto o del cursore attorno all'asse del raggio non influisce sul risultato dell'interazione e può quindi essere considerata ridondante. In tali casi, vincolare una o più componenti del movimento consente di semplificare il compito dell'utente, come avviene quando un cursore tridimensionale mantiene automaticamente un orientamento stabile mentre viene traslato nello spazio.

Per questo motivo, **nel design delle interazioni VR è pratica comune ridurre o limitare deliberatamente i gradi di libertà, introducendo vincoli che guidano il comportamento dell'utente**. Ad esempio, durante la manipolazione di un oggetto virtuale, il sistema può consentire solo traslazioni lungo un piano o bloccare temporaneamente la rotazione, evitando cambiamenti accidentali dell'orientamento dovuti a movimenti imprecisi della mano o del controller. Tali vincoli non rappresentano una limitazione delle potenzialità del sistema, bensì uno strumento progettuale volto a migliorare la precisione e la prevedibilità dell'interazione, riducendo il carico cognitivo e la probabilità di errori involontari.
Un esempio di ciò, verrà descritto in modo dettagliato nella sezione 3.2.4.

Un problema centrale nella progettazione delle tecniche di selezione in ambienti immersivi è il cosiddetto *Midas touch problem* [vedi 1, pagina 209], che si manifesta quando il sistema interpreta ogni azione dell'utente come un comando intenzionale. In un ambiente VR, ad esempio, lo sguardo dell'utente può essere costantemente rivolto verso oggetti interattivi o le mani possono attraversare numerosi elementi dello scenario; se ciascuno di questi eventi producesse un'azione, l'utente attiverebbe frequentemente comandi non desiderati. Per ridurre questo rischio, vengono spesso introdotte modalità di interazione che distinguono tra diverse fasi operative, come navigazione, selezione e manipolazione. L'uso delle modalità, tuttavia, introduce una complessità aggiuntiva. Se lo stato corrente del sistema non è chiaramente percepibile, l'utente può trovarsi in una modalità diversa da quella attesa, generando i cosiddetti *mode errors*. Ad esempio, un utente potrebbe tentare di afferrare un oggetto mentre il sistema si trova ancora in una modalità di navigazione, senza comprendere immediatamente il motivo dell'azione mancata. Per limitare questo tipo di errori, diversi approcci di interaction design suggeriscono di ridurre l'uso di modalità persistenti e di adottare **quasimodes temporanee**, attivate in modo esplicito dall'utente e mantenute solo per la durata dell'azione, come nel caso della pressione continua di un grilletto per abilitare la selezione o dell'esecuzione di un gesto specifico per avviare una manipolazione.

Infine, la progettazione dell'interazione in VR può essere guidata da principi orientati alla chiarezza comunicativa tra utente e sistema. È essenziale che il sistema renda evidente quando l'utente sta effettivamente interagendo, a quale oggetto o riferimento si applica un comando, e che fornisca un riscontro immediato dell'azione eseguita, oltre a meccanismi chiari per l'annullamento o la correzione degli errori.

3.2.3 Raccomandazioni per la navigazione

Oltre ai principi generali di navigazione discussi nella sezione 3.1.4, è utile considerare alcune raccomandazioni consolidate per la progettazione dell'interazione spaziale in VR. Tali indicazioni, basate principalmente su Bowman et al. (2004)[4], forniscono un insieme di linee guida operative che aiutano a tradurre i concetti di wayfinding e traveling in scelte concrete di design.

In primo luogo, i landmark virtuali dovrebbero essere chiaramente distinguibili e posizionati in punti strategici e ben visibili dell'ambiente. La presenza di punti di riferimento riconoscibili facilita l'acquisizione di landmark knowledge e contribuisce alla costruzione di una mappa cognitiva coerente. Allo stesso modo, le mappe rappresentano un supporto efficace per l'orientamento solo se sono leggibili, mostrano la posizione corrente dell'utente e sono orientate correttamente rispetto allo spazio. È inoltre fondamentale scegliere una dimensione adeguata, in modo che la mappa non copra eccessivamente l'ambiente circostante.

Per quanto riguarda il controllo del traveling, la tecnica di locomozione dovrebbe essere scelta in funzione delle capacità fisiologiche dell'utente e dei dispositivi di input disponibili. Inoltre, la scelta deve essere coerente con l'obiettivo dell'applicazione e con il compito dell'utente: in scenari di esplorazione o ricerca libera, ad esempio, tecniche di movimento continuo e controllate dall'utente (come la smooth locomotion) risultano particolarmente adatte, mentre per compiti orientati a raggiungere una destinazione specifica ("Go to X") sono più efficaci strategie basate su piani di percorso, mappe o teletrasporto. Nel caso in cui la navigazione non rappresenti l'obiettivo principale dell'esperienza, è consigliabile adottare tecniche di movimento il più possibile semplici e poco invasive, in modo da non distrarre l'utente dal task principale.

Un altro aspetto importante dell'interaction design riguarda la possibilità di offrire diverse modalità di navigazione per utenti con abilità e preferenze differenti. Quando i profili degli utenti sono molto diversi, può essere utile mettere a disposizione sia tecniche semplici sia tecniche più complesse, permettendo all'utente di scegliere quella più adatta. In questo senso, anche le tecniche "magiche" tipiche della VR (ad esempio teletrasporto o spostamenti istantanei) possono risultare altrettanto valide quanto tecniche realistiche, purché siano compatibili con le altre modalità di interazione previste (es. manipolazione, selezione).

Infine, se le tecniche di navigazione sono particolarmente complesse, è consigliabile prevedere una fase di training o un onboarding, in modo da permettere all'utente di acquisire una conoscenza di overview dell'ambiente e di ridurre errori legati all'inesperienza. Questa fase è particolarmente utile quando l'interazione richiede l'uso di strumenti o gesti non immediatamente intuitivi, oppure quando l'obiettivo dell'applicazione richiede un alto grado di precisione e controllo.

3.2.4 Caso di studio: Gun Interaction in FPS Games

Durante la fase di ricerca delle pratiche consolidate di Interaction Design nel contesto dei videogiochi, con focus sul genere FPS (First Person Shooter), è stato individuato un approccio di particolare interesse proposto da Przemysław Krompiec e Kyoungju Park[5]. Tale approccio risulta pienamente coerente con il principio di limitazione dei gradi di libertà dell'interazione utente, discusso nella sezione 3.2.2.

Nello specifico, il lavoro degli autori evidenzia la necessità di individuare un equilibrio tra realismo dell'interazione e comodità d'uso per il giocatore. Essi mostrano come meccaniche di ricarica semplificate, quali la pressione di un singolo pulsante o il movimento di "shaking" del controller, risultino immediatamente accessibili dal punto di vista dell'usabilità, ma siano carenti sotto il profilo del realismo percettivo. Al contrario, una ricarica basata su interazioni fisiche realistiche — come l'inserimento manuale del caricatore e l'azionamento del carrello dell'arma — offre un'elevata fedeltà simulativa, ma introduce una complessità significativa, richiedendo movimenti altamente precisi e potenzialmente frustranti per l'utente.
Per rispondere a questa problematica, Krompiec e Park propongono un meccanismo di ricarica ibrido che si fonda sul principio della limitazione dei gradi di libertà del movimento, con l'obiettivo di guidare l'utente durante l'interazione e ridurre il carico cognitivo e motorio, mantenendo al contempo un adeguato livello di realismo.

3.2.4.1 Strategia di presa degli oggetti (grabbing)

Prima di introdurre nel dettaglio le fondamenta del meccanismo di ricarica proposto, risulta utile descrivere l'approccio adottato dagli autori per l'interazione di presa (grabbing) degli oggetti presenti nel mondo virtuale.
La strategia distingue esplicitamente tra *gun objects* e *non-gun objects*. Nel caso degli oggetti che non rappresentano un'arma — come un semplice cubo — la posizione e la rotazione dell'oggetto rispetto al motion controller rimangono invariate al momento dell'interazione. Per gli oggetti che rappresentano un'arma, invece, la trasformazione dell'oggetto viene modificata in modo da allinearne posizione e orientamento ai vettori forward e up del motion controller.
La Figura 3.7 illustra questa differenza: in "a)" l'arma viene posizionata e ruotata per orientarsi già in direzione di sparo, mentre in "b)" il cubo rimane in posizione libera.

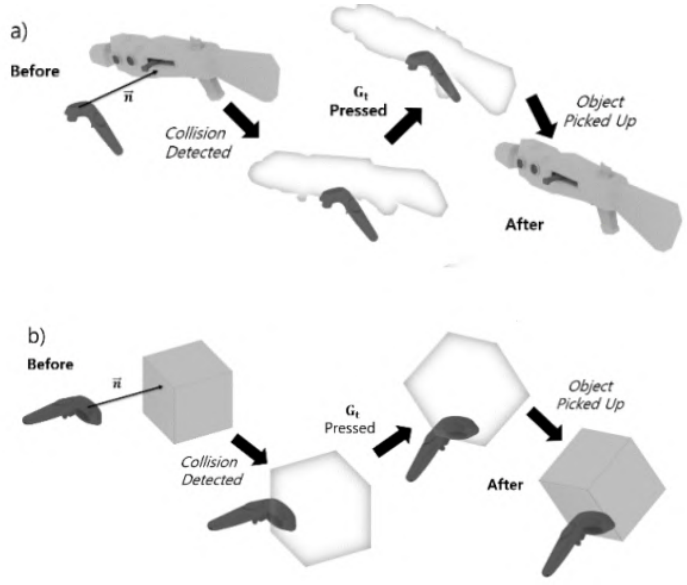

Figura 3.7: Illustrazione del grabbing. (tratto da Krompiec e Park, 2019)

Questo approccio consente di definire una posa predefinita e coerente per l'arma, garantendo al giocatore una presa visivamente consistente e riducendo la variabilità del-

l'interazione. Tale scelta progettuale non solo migliora il feedback visivo e la leggibilità dell'azione, ma semplifica anche la gestione delle meccaniche di ricarica, fungendo da ulteriore applicazione del principio di limitazione dei gradi di libertà.

3.2.4.2 Meccanismo di ricarica dell'arma

Come anticipato in precedenza, il meccanismo di ricarica è progettato con l'obiettivo di mantenere un adeguato livello di realismo, evitando al contempo un'eccessiva complessità dell'azione per il giocatore. In particolare, il sistema mira a essere robusto rispetto all'errore umano nei movimenti del motion controller, fenomeno frequente nelle interazioni in realtà virtuale.

Il principio alla base della ricarica consiste nel vincolare l'azione lungo un asse di ricarica predefinito, sfruttando il concetto di proiezione del movimento del controller — potenzialmente impreciso — lungo una traiettoria precisa e controllata. In questo modo, il movimento effettivamente compiuto dall'utente viene reinterpretato dal sistema in funzione della direzione attesa per l'azione di ricarica. Tale vincolo viene implementato mediante la formula standard di proiezione vettoriale:

$$\text{proj}_{\vec{n}}\vec{m} = \frac{\vec{n} \cdot \vec{m}}{\|\vec{m}\|} \tag{3.1}$$

dove $\vec{n}$ rappresenta la direzione di movimento rilevata dal motion controller del giocatore e $\vec{m}$ la direzione ideale prevista per l'azione di ricarica.
L'applicazione di questa formula consente, da un lato, di verificare che l'utente stia muovendo il controller nel verso corretto e, dall'altro, di vincolare il movimento lungo la traiettoria corretta, riducendo l'impatto delle imprecisioni e migliorando l'esperienza complessiva di interazione.
Di seguito, la Figura 3.8 illustra il ruolo della proiezione del movimento nel mitigare l'errore umano, vincolando l'interazione lungo la traiettoria di ricarica prevista.

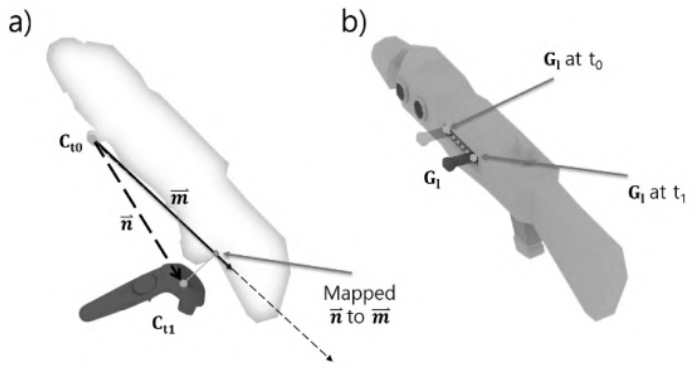

Figura 3.8: Illustrazione della ricarica dell'arma. (tratto da Krompiec e Park, 2019)

3.2.4.3 Applicazione del principio in giochi commerciali

L'approccio sviluppato da Krompiec e Park non è rimasto confinato alla letteratura accademica, ma ha influenzato anche titoli VR di successo. In giochi come *Half-Life: Alyx* e *Pavlov VR*, le meccaniche di ricarica riflettono lo stesso principio di limitazione dei gradi di libertà: il movimento del caricatore è vincolato lungo traiettorie predefinite, permettendo al giocatore di compiere l'azione in modo realistico senza richiedere movimenti estremamente precisi.
La Figura 3.9 mostra il processo di ricarica dell'arma nel gioco *Half-Life: Alyx*, dimostrando come i concetti teorici di HCI possano essere tradotti in soluzioni pratiche, migliorando simultaneamente il realismo percettivo e l'usabilità nell'interazione VR.

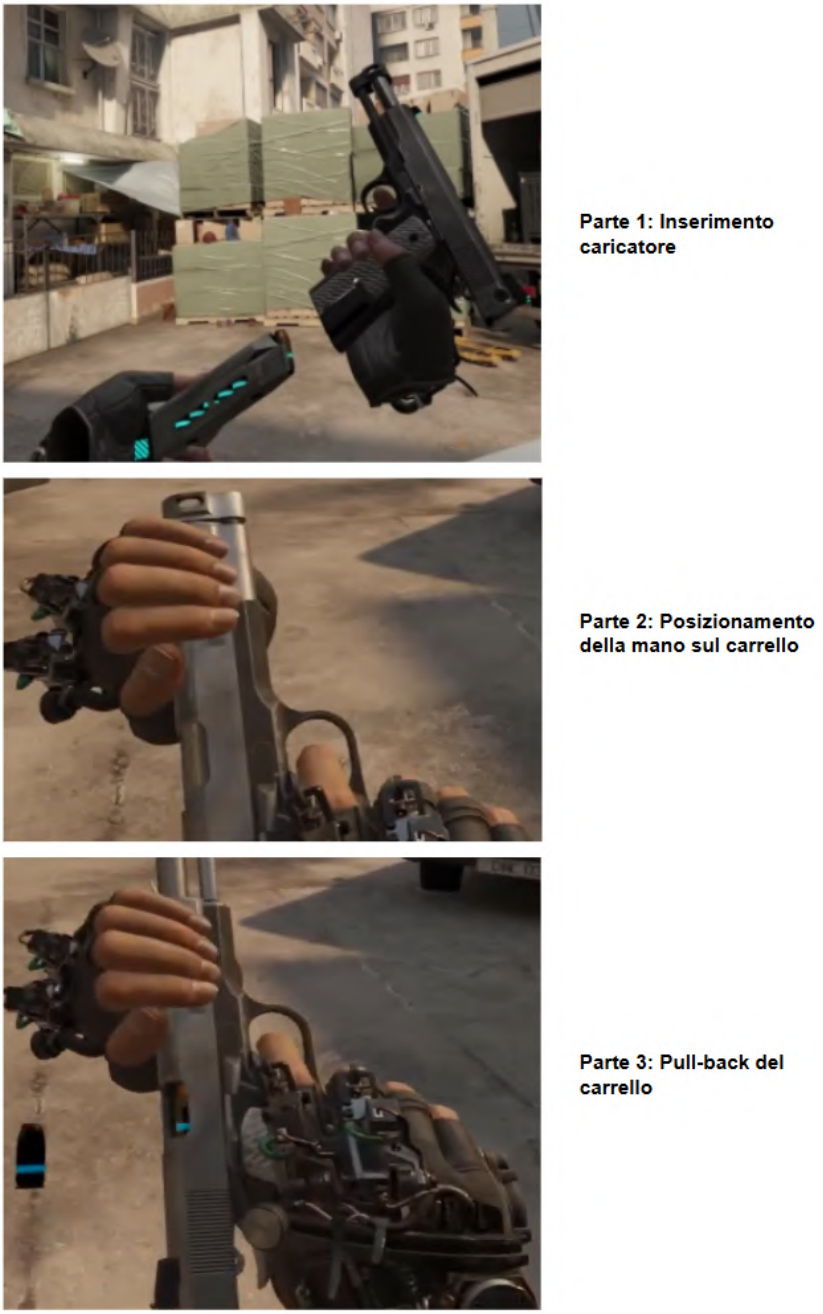

Figura 3.9: Ricarica dell'arma nel gioco VR "Half-Life: Alyx"

3.3 Relazione tra Game Design e HCI

Game Design e Human-Computer Interaction condividono un obiettivo fondamentale: creare esperienze significative e coinvolgenti per l'utente. Sebbene affrontino questo obiettivo da prospettive diverse, le due discipline risultano strettamente interconnesse, soprattutto nel contesto della realtà virtuale.
Il Game Design definisce le regole, le meccaniche e le sfide che strutturano l'esperienza di gioco, determinando obiettivi, livelli di difficoltà e modalità di feedback, mentre la HCI si occupa di rendere queste interazioni accessibili, intuitive e compatibili con le capacità motorie e cognitive del giocatore. In ambienti immersivi, la corretta integrazione di questi due approcci è cruciale: meccaniche complesse o altamente realistiche rischiano di risultare frustranti se non supportate da interfacce progettate secondo principi di HCI, che sfruttano schemi motori già acquisiti e riducono il carico cognitivo.

Questa relazione è bidirezionale. Da un lato, le scelte di Game Design impongono vincoli all'interazione, definendo quali azioni devono essere possibili, quali percorsi di movimento devono essere seguiti e quale livello di precisione è richiesto. Dall'altro, i limiti umani e le caratteristiche dei dispositivi, evidenziati dall'HCI, guidano il design delle meccaniche, suggerendo semplificazioni o strategie di assistenza che migliorino l'esperienza senza compromettere il realismo o l'immersione.

Questo dialogo tra teoria e implementazione si riflette anche nel Game Logic Layer (Sezione 5.4), uno dei quattro layer classici nello sviluppo di videogiochi. Qui le scelte di Game Design definiscono regole e meccaniche, mentre i principi di HCI ne guidano la realizzazione pratica, assicurando che le interazioni siano coerenti, intuitive e soddisfacenti per l'utente.

Un esempio concreto di questa integrazione tra Game Design e HCI si è evidenziato durante il porting da *Gavunia desktop* a *Gavunia VR*. In questa fase, le interazioni con gli oggetti sono state riprogettate per sfruttare appieno le potenzialità immersive del medium. In particolare, la ricarica della pistola è stata trasformata da un'azione semplice — la pressione di un tasto — a un meccanismo manuale che richiede l'inserimento fisico del caricatore. Tale modifica ha comportato anche una revisione della gestione dell'inventario delle munizioni, dimostrando come le scelte di interazione influenzino direttamente le meccaniche di gioco. Per maggiori dettagli, si rinvia alla Sezione 6.3.4.6.

Capitolo 4

Motion Sickness

L'uso di applicazioni VR può provocare effetti fisiologici e cognitivi indesiderati nei giocatori, fenomeno noto come cybersickness o VR sickness. Tra i sintomi più comuni si annoverano nausea, vomito, mal di testa, vertigini, affaticamento e disorientamento. Questi effetti non si manifestano esclusivamente durante il movimento nell'ambiente virtuale, ma talvolta anche in assenza di spostamenti, semplicemente osservando immagini o scene simulate.

Le cause della cybersickness non sono completamente note e sono probabilmente multifattoriali. Le principali teorie a supporto della spiegazione del fenomeno sono le seguenti:

- **Sensory conflict theory** (Treisman, 1977)[6]: Secondo questa teoria, i sintomi derivano da incoerenze tra informazioni sensoriali percepite dal cervello. Ad esempio, un utente seduto in VR può osservare visivamente il movimento dell'ambiente, mentre il sistema vestibolare (responsabile dell'equilibrio) segnala che il corpo è fermo. Questo conflitto tra segnali visivi e vestibolari genera sintomi quali nausea, vertigini e disorientamento.

- **Postural instability theory** (Riccio and Stoffregen, 1991)[7]: Secondo questa teoria, la cybersickness emerge quando una persona non riesce a mantenere il controllo della postura in situazioni insolite o non familiari. In VR, l'ambiente tridimensionale e la mancanza di riferimenti stabili possono compromettere il controllo posturale, provocando instabilità, disagio e sintomi associati come nausea e affaticamento.
  Questa teoria spiega perché utenti che camminano o ruotano in VR senza un adeguato feedback visivo o senza punti di riferimento stabili tendano a sviluppare sintomi più rapidamente. Inoltre, suggerisce che la capacità di prevedere e anticipare i movimenti riduce la probabilità di cybersickness: ad esempio, gli utenti che hanno pieno controllo della propria navigazione (come guidare una navetta virtuale con feedback visivo coerente) soffrono meno dei sintomi rispetto a chi subisce movimenti imposti dalla simulazione.

Oltre alle teorie, è possibile identificare una serie di fattori che favoriscono la comparsa della cybersickness, raggruppabili in tre categorie principali:

- **Fattori individuali**: età, sesso, predisposizione alla motion sickness, esperienze pregresse con VR o AR. La suscettibilità varia significativamente tra individui, e la stessa persona può reagire in modo differente a scenari simili in momenti diversi.

- **Fattori legati al sistema VR**: qualità del display (refresh rate, contrasto, stereo display), latenza tra movimento della testa e aggiornamento dell'immagine, ampiezza del campo visivo e coinvolgimento della visione periferica. In generale, un campo visivo più ampio aumenta la probabilità di insorgenza dei sintomi.
- **Fattori legati all'applicazione**: durata e intensità dell'esperienza, mancanza di riferimenti stabili nell'ambiente virtuale, tecniche di locomozione scelte (ad esempio smooth locomotion vs. teleportation) e grado di controllo e anticipazione dei movimenti da parte dell'utente. L'illusione di movimento (vection) può amplificare la sensazione di sickness se non è accompagnata da feedback coerente.

Dal punto di vista del design, mentre i fattori individuali sono difficilmente controllabili, esistono numerose strategie per mitigare gli effetti negativi agendo sui fattori legati al sistema e all'applicazione.
Nelle sezioni successive saranno analizzate tecniche sperimentate per ridurre la cybersickness, garantendo al contempo un'esperienza immersiva e confortevole per l'utente.

4.1 Tecniche per mitigare la motion sickness

Il problema della cybersickness, e in particolare della motion sickness, rappresenta una delle principali criticità nello sviluppo di esperienze in realtà virtuale, soprattutto nel contesto videoludico, dove il comfort dell'utente riveste un ruolo fondamentale.
Per questo motivo, numerosi studi si sono concentrati sull'individuazione di strategie volte a mitigare tali effetti, intervenendo prevalentemente su fattori legati all'applicazione e alle scelte di design dell'esperienza.
Tra le diverse soluzioni proposte in letteratura, in questa sezione verranno analizzate tre tecniche ritenute particolarmente rilevanti: *Teleportation*, *Geometry Deformation* e *Tunnel Vision*.

4.1.1 Teleportation

La Teleportation, già menzionata nella sezione 3.1.4.1, è una tecnica di locomozione ampiamente utilizzata nelle applicazioni VR, poiché numerosi studi hanno dimostrato che contribuisce a ridurre significativamente le sensazioni di motion sickness.
Il funzionamento si basa sul trasferimento istantaneo dell'utente da un punto all'altro dell'ambiente virtuale, evitando il movimento fluido continuo. In questo modo, il mismatch tra le informazioni visive e quelle provenienti dall'apparato vestibolare viene minimizzato, riducendo la percezione di disagio durante la navigazione nello spazio virtuale. Tuttavia, la Teleportation può comportare una riduzione del senso di presenza e immersione rispetto alla locomozione fluida, poiché l'utente percepisce interruzioni nel movimento e una certa discontinuità temporale nell'esperienza.

Concentrandoci sul contesto dei videogiochi FPS, uno studio di Monteiro et al. [8] ha evidenziato come le performance dei giocatori (misurate come rapporto tra avversari colpiti e danni subiti) siano inferiori quando si utilizza la Teleportation rispetto alla locomozione continua (smooth locomotion). Ciò suggerisce che, in ambienti altamente dinamici e competitivi, il teletrasporto può introdurre sfide aggiuntive se non progettato e gestito attentamente.
Un esempio di gestione efficace è rappresentato dal gioco *Half-Life: Alyx*, in cui il teletrasporto è stato implementato in modo tattico e immersivo: il giocatore può vedere una rappresentazione dei propri piedi nella destinazione, orientarsi correttamente, evitare ostacoli e sfruttare l'ambiente per coprirsi o spostarsi rapidamente.
Il sistema gestisce automaticamente la validità del percorso e della posizione finale, dando priorità al pavimento e al contatto con l'ambiente, riducendo così gli errori di puntamento in situazioni di combattimento frenetico. Inoltre, viene fornito un feedback audio coerente con i movimenti del corpo, che contribuisce a rinforzare la sensazione di presenza senza distrarre il giocatore. Grazie a queste soluzioni, i giocatori possono utilizzare il teletrasporto in modo naturale e sicuro, riducendo la motion sickness senza compromettere immersione e giocabilità, anche negli spostamenti rapidi tipici degli FPS.
La Figura 4.1 riporta un estratto del video di presentazione di Valve, evidenziando il funzionamento della modalità di Teleportation.

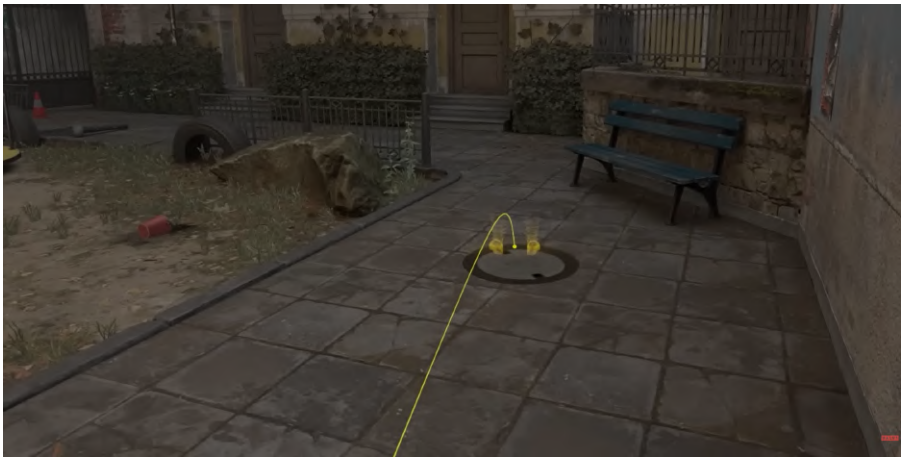

Figura 4.1: Teleportation nel gioco VR "Half-Life:Alyx" (tratto da video youtube "Half-Life: Alyx - Locomotion Deep Dive": https://youtu.be/TX58AbJq-xo?si=F4JYUCskUIwavcJ5)

4.1.2 Geometry Deformation

La Geometry Deformation è una tecnica proposta da Ruding [9] per ridurre la motion sickness in ambienti VR, intervenendo sulla percezione del movimento autoindotto visivamente. La motivazione alla base della tecnica nasce dal fatto che molti episodi di cybersickness derivano da un conflitto sensoriale: l'utente percepisce il movimento attraverso la vista, ma non riceve segnali corrispondenti dal sistema vestibolare o dai muscoli, generando disorientamento e nausea.

L'approccio si concentra sugli oggetti visibili nella periferia del campo visivo (Peripheral Field of View), che contribuiscono in modo predominante alla percezione del movimento illusorio (*visually induced self-motion* o *vection*). Studi precedenti hanno mostrato come il moto relativo degli elementi presenti nel FOV periferico sia responsabile della maggior parte della sensazione di auto-movimento percepita dall'utente, mentre la visione centrale gioca un ruolo secondario.
La tecnica di Geometry Deformation consiste quindi nel deformare geometricamente la scena virtuale nelle regioni periferiche, riducendo la velocità apparente degli oggetti rispetto alla direzione di navigazione. In questo modo, l'utente percepisce un movimento relativo più lento nella visione periferica, attenuando la sensazione di auto-moto visivo senza alterare l'intenzione di navigazione, i parametri di movimento o il campo visivo complessivo. La porzione centrale del FOV, fondamentale per l'orientamento e l'immersione, viene infatti preservata.

Nel prototipo sviluppato da Ruding, la deformazione geometrica è implementata mediante un approccio di *lattice-based deformation*, introdotto da Sederberg e Parry [10]. Questa tecnica è stata scelta in quanto soddisfa specifici requisiti individuati nello studio: la possibilità di deformare uniformemente tutti gli oggetti presenti nel FOV periferico in funzione della direzione di navigazione e della distanza dall'osservatore, l'indipendenza dalla tessellazione delle mesh e la capacità di deformare simultaneamente più mesh disconnesse.

Il funzionamento dell'algoritmo prevede innanzitutto il calcolo di un bounding box globale che racchiude l'insieme delle mesh da deformare. All'interno di questo volume viene generata una griglia tridimensionale regolare (lattice), i cui vertici fungono da punti di controllo per un solido di Bézier tridimensionale. Le mesh della scena non vengono modificate direttamente; al contrario, esse ereditano la deformazione applicata al solido di Bézier in cui sono contenute.
Durante la navigazione dell'utente nella scena virtuale, la deformazione è guidata direttamente dai parametri di movimento dell'osservatore. In particolare, la direzione e la velocità di navigazione determinano lo spostamento dei punti di controllo del lattice, che vengono mossi nella stessa direzione del movimento dell'utente. L'intensità della deformazione applicata a ciascun punto di controllo dipende dalla sua distanza dall'osservatore lungo la direzione di navigazione: i punti più vicini subiscono uno spostamento maggiore, mentre quelli più distanti risultano progressivamente meno alterati. Questo meccanismo consente di ottenere una deformazione graduale e continua della scena, riducendo il movimento relativo percepito visivamente, soprattutto nella visione periferica, e mantenendo al contempo una geometria stabile e priva di discontinuità.

Nel contesto sperimentale considerato, la scena virtuale è costituita da un ambiente urbano semplificato, composto da edifici disposti lungo una strada. L'utente si muove lungo un percorso rettilineo a velocità costante, mentre la deformazione viene applicata

progressivamente agli edifici ai lati della strada.
La Figura 4.2 mostra come la scena evolva nel tempo, evidenziando la variazione della geometria in relazione allo spostamento dell'osservatore.
La Figura 4.3, invece, mostra la differenza tra la scena normale e quella deformata.

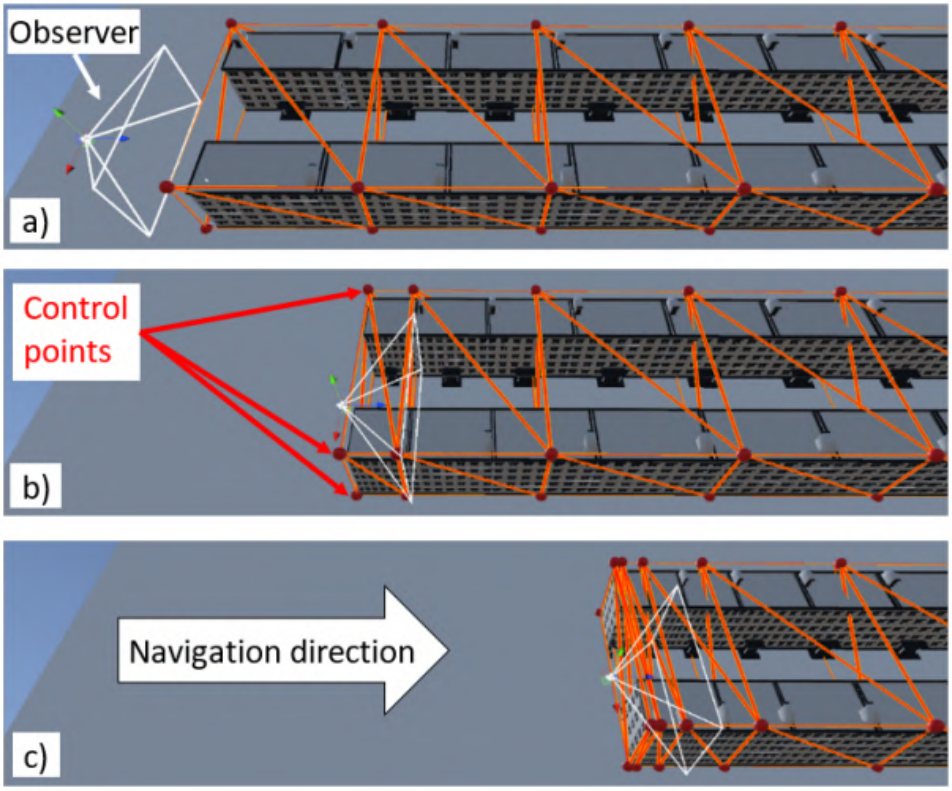

Figura 4.2: Illustrazione della geometry deformation nel tempo (tratto da Ruding, 2019)

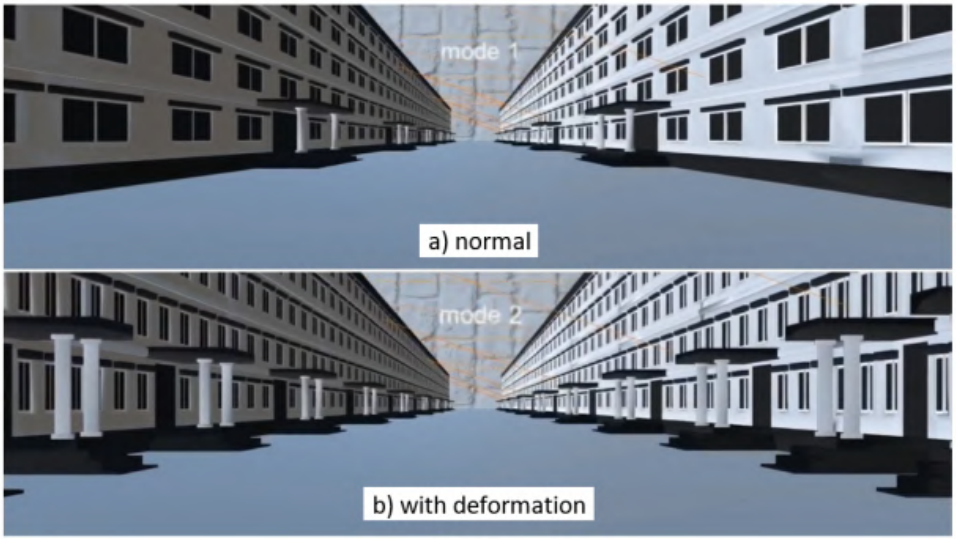

Figura 4.3: Esempio di geometry deformation in una scena virtuale (tratto da Ruding, 2019)

Nonostante i risultati positivi riportati in letteratura, la Geometry Deformation presenta alcuni limiti che ne influenzano l'ambito di applicabilità nei sistemi di realtà virtuale. In particolare, l'efficacia di questa tecnica risulta fortemente dipendente dalla struttura spaziale dell'ambiente virtuale in cui viene applicata.
L'approccio si dimostra particolarmente adatto a scenari caratterizzati da una navigazione prevalentemente direzionale e da geometrie regolari, come corridoi, strade o ambienti canalizzati, in cui la deformazione progressiva degli elementi periferici risulta coerente con il movimento dell'osservatore. Al contrario, in ambienti aperti o con geometrie complesse e irregolari, la deformazione può risultare meno efficace o potenzialmente percepibile dall'utente, soprattutto in presenza di frequenti cambi di direzione o di movimenti rapidi del punto di vista.

Un ulteriore aspetto da considerare riguarda il costo computazionale associato alla deformazione geometrica. Le tecniche basate su lattice richiedono l'aggiornamento continuo dei punti di controllo e il calcolo della deformazione della scena durante la navigazione, introducendo un carico aggiuntivo nella pipeline di rendering. In applicazioni di realtà virtuale, in cui il mantenimento di un frame rate elevato e stabile è un requisito fondamentale per il comfort dell'utente, tale overhead può rappresentare un fattore limitante, in particolare in contesti interattivi complessi.
Infine, la presenza di una deformazione geometrica globale può introdurre una maggiore complessità progettuale e implementativa, richiedendo un'attenta integrazione con gli altri sistemi dell'applicazione al fine di preservare la coerenza percettiva e la qualità dell'esperienza immersiva. Tali considerazioni suggeriscono che la Geometry Deformation, pur rappresentando un approccio efficace alla riduzione della motion sickness in specifici contesti, debba essere valutata attentamente in funzione delle caratteristiche dell'ambiente virtuale e dei requisiti applicativi.

4.1.3 Tunnel Vision

La tecnica di Tunnel Vision (o Tunneling) è stata introdotta dal team di Google Earth VR durante la conferenza *SIGGRAPH 2017* e sviluppata in particolare da Adam Glazier [11]. **Questa soluzione si basa sulla riduzione dinamica del campo visivo (FOV) per contrastare la motion sickness, intervenendo principalmente durante le fasi in cui l'utente è in movimento**.

Il Tunneling limita temporaneamente la visione a un'area circolare centrale, rendendo nuovamente l'ambiente completamente visibile al termine dello spostamento. In pratica, durante la navigazione l'utente osserva il mondo come se fosse "attraverso un oblò", riducendo gli stimoli periferici che maggiormente contribuiscono alla sensazione di nausea.

È possibile considerare il Tunneling come un approccio concettualmente affine alla Geometry Deformation, poiché entrambe le tecniche si basano sull'idea che la percezione di movimento autoindotto è fortemente influenzata dagli stimoli visivi periferici. Mentre la Geometry Deformation riduce la sensazione di self-motion modificando la geometria degli elementi periferici, il Tunneling ottiene un effetto analogo limitando temporaneamente il campo visivo durante la locomozione.

In Google Earth VR, dove l'esperienza è principalmente una navigazione dall'alto, la tecnica viene integrata con un ulteriore elemento visivo: una griglia sottostante al cerchio di Tunneling. Questa griglia serve a fornire un riferimento del "pavimento", aiutando l'utente a percepire la posizione e l'orientamento nello spazio e a stabilizzare ulteriormente la sensazione di movimento, come se stesse osservando il mondo dalla propria stanza.
La Figura 4.4 riporta un estratto della presentazione di Google Earth VR al SIGGRAPH, che evidenzia l'uso della Tunnel Vision e della griglia per ridurre le sensazioni di malessere.

Figura 4.4: Google Earth VR - Tunnel Vision (tratto da presentazione SIGGRAPH, 2017)

Nonostante il Tunnelling si sia dimostrato efficace nel ridurre la motion sickness intervenendo sugli stimoli periferici durante la locomozione, la tecnica presenta alcune limitazioni intrinseche che ne influenzano l'uso in contesti VR più ampi.
In primo luogo, la riduzione temporanea del campo visivo può risultare percepibile dall'utente, soprattutto in presenza di movimenti rapidi della testa o di cambi di direzione frequenti, determinando una sensazione di "finestra" o di visione artificiale. Tale effetto può compromettere la continuità percettiva e, in alcuni casi, aumentare il senso di distacco dall'ambiente virtuale.
Un secondo aspetto riguarda la riduzione della consapevolezza periferica, che può essere svantaggiosa in esperienze in cui è importante mantenere una visione ampia e una pronta reazione a eventi laterali o improvvisi. In applicazioni in cui il gameplay richiede una sorveglianza costante dell'ambiente (ad esempio in scenari competitivi o con elevata densità di elementi dinamici), il Tunnelling potrebbe influire sulla percezione del controllo e sulla reattività dell'utente.
Infine, in applicazioni in cui la percezione di profondità e l'immersione visiva sono elementi centrali, la temporanea riduzione del FOV può risultare meno tollerata rispetto a soluzioni che mantengono un campo visivo completo pur riducendo gli stimoli periferici attraverso modifiche della scena.

Capitolo 5

Gavunia

Gavunia è stato inizialmente sviluppato per piattaforma desktop nell'ambito dell'attività progettuale del corso di *Fondamenti di Computer Graphics M*, utilizzando il motore di gioco **Unreal Engine 5** come ambiente di sviluppo.
Questa versione costituisce la base concettuale e tecnica dell'intero progetto, definendone le meccaniche di gioco, l'architettura software e le scelte estetiche fondamentali, costituendo il punto di partenza per il successivo adattamento in ambiente VR.
Le motivazioni alla base della scelta di Unreal Engine e le sue implicazioni architetturali verranno discusse nella Sezione 5.1, in cui saranno analizzati i punti di forza e le caratteristiche tecnologiche che hanno guidato questa decisione.

La progettazione è stata organizzata secondo una suddivisione in layer tipica dello sviluppo videoludico, adottata al fine di garantire una chiara separazione delle responsabilità e una maggiore modularità del sistema:

- **Modelling Layer**: responsabile della creazione e strutturazione degli asset tridimensionali presenti nella scena, includendo modelli geometrici e relative proprietà spaziali.
- **Rendering Layer**: dedicato alla gestione dell'aspetto visivo della scena, comprendendo illuminazione, ombre, materiali e texturing, con l'obiettivo di garantire coerenza estetica e qualità percettiva.
- **Game Logic Layer**: costituisce il nucleo funzionale del sistema e governa le dinamiche di gioco, incluse regole, meccaniche di interazione, progressione e struttura narrativa. Questo layer è stato progettato in conformità ai principi di Game Design e Human-Computer Interaction (HCI), al fine di assicurare coerenza, usabilità e qualità dell'esperienza utente.
- **Animation Layer**: responsabile della gestione delle animazioni, includendo movimenti dei personaggi, transizioni di stato e feedback dinamici funzionali al rafforzamento dell'esperienza ludica.

La Figura 5.1 rappresenta schematicamente l'architettura a layer adottata, evidenziando le principali componenti del sistema e le loro relazioni.
Ciascun layer verrà analizzato nel dettaglio nelle sezioni successive, con particolare attenzione alle scelte progettuali e implementative adottate.

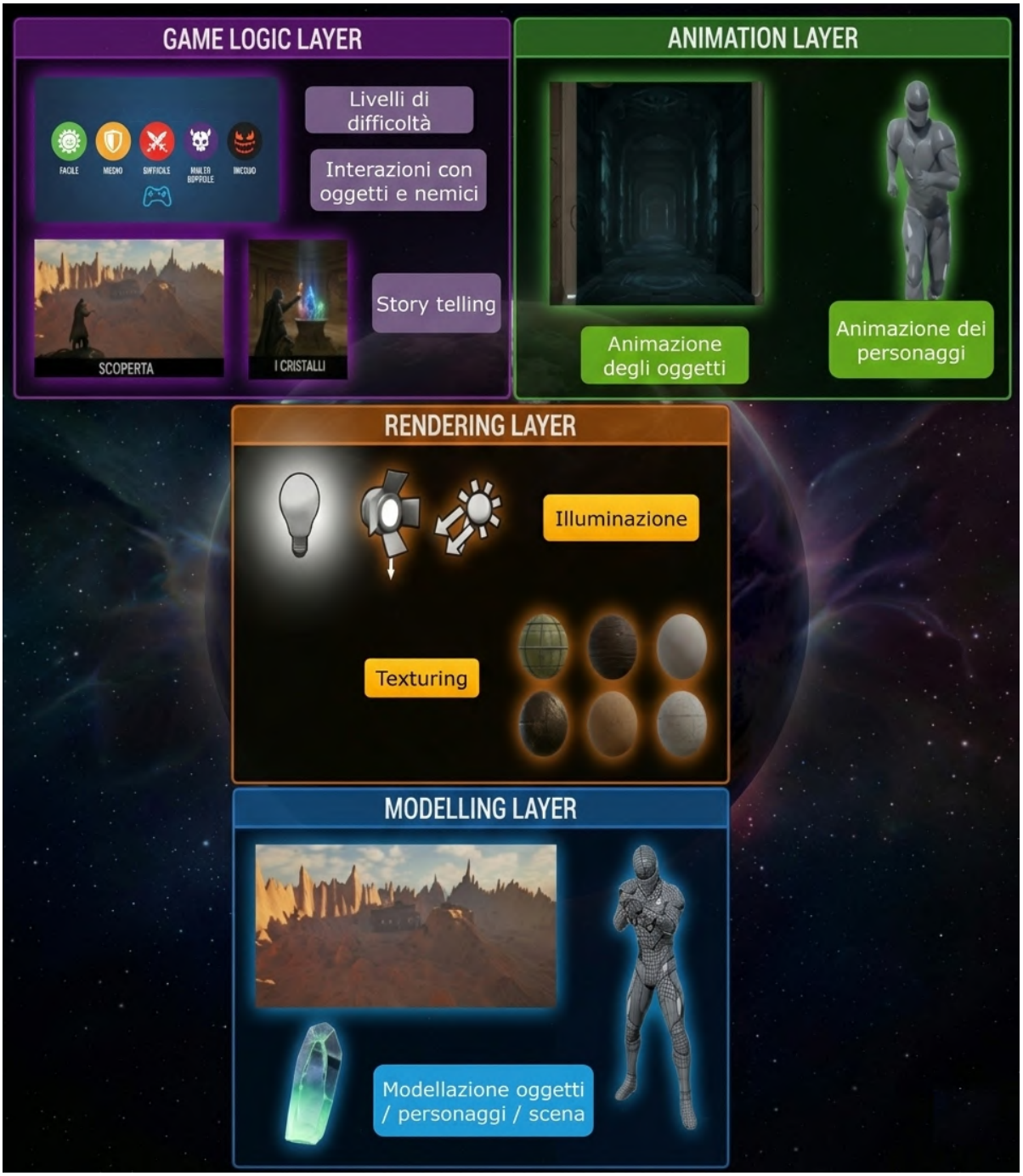

Figura 5.1: Game Layers

5.1 Unreal Engine come motore di gioco

Il motore di gioco Unreal Engine 5 (UE5), sviluppato da Epic Games, rappresenta una piattaforma completa per lo sviluppo di applicazioni 3D interattive, in particolare videogiochi. UE5 copre in modo integrato i principali ambiti di un progetto videoludico, tra cui: il rendering e l'illuminazione delle scene, la gestione delle animazioni di personaggi e oggetti, la fisica e le collisioni, il controllo delle interazioni tramite scripting, e l'organizzazione e l'ottimizzazione degli asset tridimensionali.

All'interno di questi ambiti, UE5 fornisce strumenti specifici per affrontare alcune delle sfide più comuni nello sviluppo. Ad esempio, nella modellazione 3D, il motore consente di trattare mesh con un alto livello di dettaglio senza penalizzare le prestazioni, grazie a tecnologie come **Nanite**, che automatizzano il livello di dettaglio in base alla distanza e alla visibilità degli oggetti. Per quanto riguarda l'illuminazione e i riflessi, UE5 mette a disposizione sistemi dinamici come **Lumen**, in grado di generare effetti realistici in tempo reale, garantendo coerenza visiva anche in scenari complessi.
Il motore supporta inoltre una struttura modulare basata su Actor e Component, che facilita la separazione dei diversi layer del sistema — modellazione, rendering, game logic e animazione — e permette iterazioni rapide durante lo sviluppo. Il sistema di Blueprint, infine, consente di prototipare rapidamente le meccaniche di gioco e testare le interazioni mediante programmazione a scripting visivo.

La scelta di Unreal Engine 5 è quindi stata motivata dalle sue caratteristiche avanzate nel rendering in tempo reale, nella gestione di geometrie complesse e nell'illuminazione dinamica, che risultano particolarmente adatte agli obiettivi progettuali di Gavunia. Altri motori, come Unity, pur offrendo un ecosistema solido e versatile, avrebbero richiesto lo sviluppo di strumenti aggiuntivi per ottenere un livello comparabile di qualità visiva e gestione dei dettagli, comportando tempi di prototipazione più lunghi.

5.2 Modelling Layer

Il Modelling Layer costituisce la componente del sistema dedicata alla creazione e gestione degli asset tridimensionali presenti nel mondo virtuale. Questo layer si occupa non solo della modellazione di singoli oggetti, ma anche della loro disposizione nello spazio, definendo la composizione complessiva della scena, la relazione tra gli oggetti e la struttura ambientale.
Un aspetto fondamentale di questo layer riguarda la preparazione dei modelli per l'applicazione dei materiali e delle texture, come ad esempio il *UV mapping*, che definisce come la superficie tridimensionale di un oggetto viene "srotolata" nello spazio bidimensionale per poterci applicare correttamente le texture.

L'obiettivo principale del Modelling Layer è quindi garantire coerenza visiva, realismo percettivo e corretto posizionamento degli elementi all'interno dell'ambiente di gioco, fornendo una base solida su cui gli altri layer, come rendering, animazione e logica di gioco, possono operare efficacemente.

5.2.1 Ambientazione

Il gioco è ambientato in una landa deserta circondata da montagne, in cui si distinguono due principali punti di interesse: una collina e un edificio, come mostrato in Figura 5.2. La progettazione e modellazione di questo ambiente non si limita alla semplice creazione geometrica, ma è guidata dalla narrazione alla base del gioco, mostrando come i diversi layer interagiscano per costruire un'esperienza coerente e immersiva.
Questa integrazione tra struttura del mondo e storytelling è coerente con la Lens of Elemental Tetrad di Schell (Sezione 2.1.4), che sottolinea come elementi di storia ed estetica si influenzino reciprocamente per rafforzare l'esperienza complessiva del giocatore.
L'ambientazione diventa così non solo uno sfondo, ma un elemento attivo nella trasmissione della narrazione e nella definizione delle dinamiche ludiche.

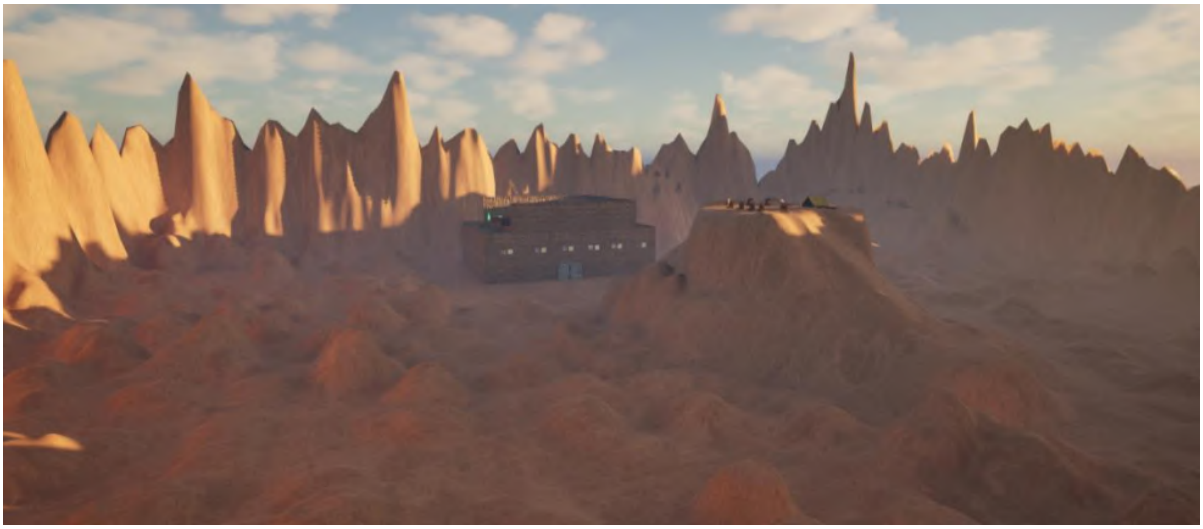

Figura 5.2: Ambientazione di Gavunia

5.2.1.1 Definizione del paesaggio

Il paesaggio esterno di *Gavunia* è stato modellato utilizzando lo strumento **Landscape** [12] integrato in Unreal Engine, che consente di definire la morfologia del terreno attraverso una combinazione di strumenti dedicati. Nel progetto, sono state adottate diverse tecniche per ottenere un ambiente naturale e coerente con l'esperienza di gioco prevista.

In particolare, lo strumento *Sculpt* è stato impiegato per modellare le montagne e creare la rampa di salita verso la collina; *Erase* ha permesso di generare concavità e avvallamenti in alcune aree della mappa, mentre *Smooth* è stato utilizzato per ammorbidire transizioni e superfici, conferendo al terreno un aspetto più realistico. Infine, l'uso dello strumento *Flatten* ha consentito di appiattire la sommità della collina, dove è stato collocato l'accampamento del giocatore.

5.2.1.2 Struttura dell'edificio

La struttura muraria dell'edificio è stata realizzata sfruttando gli strumenti di **Modeling** [13] integrati in Unreal Engine, che consentono di generare mesh a partire da primitive geometriche di base, come box e sphere. I muri esterni e interni sono stati modellati a

partire da oggetti di tipo *Box*, personalizzando parametri quali altezza, larghezza e profondità per adattarli alle specifiche progettuali del gioco. Per creare aperture funzionali come ingressi, varchi, finestre e vani per porte, è stato impiegato lo strumento *PolyCut*, che consente di sottrarre porzioni della geometria tramite operazioni booleane. Questa tecnica è stata adottata anche per realizzare elementi strutturali complessi, come i varchi verticali per il collegamento tra i piani (ad esempio il vano scala), conferendo maggiore articolazione spaziale e coerenza visiva all'architettura dell'edificio.

L'edificio, articolato su tre piani, rappresenta un esempio di come il Modelling Layer si leghi alle esigenze narrative del progetto: la disposizione degli spazi e la struttura dei piani sono progettate per guidare il giocatore attraverso un percorso di esplorazione, combattimento e scoperta.
Il **piano terra** (Figura 5.3) presenta una struttura labirintica in stile rustico, con spazi chiusi che favoriscono l'esplorazione graduale.
Il **primo piano** (Figura 5.4) introduce un cambiamento significativo dal punto di vista estetico, presentando ambienti più eleganti rispetto al piano inferiore. La pavimentazione in marmo e le pareti in pietra antica contribuiscono a trasmettere un senso di maggiore solennità e importanza narrativa.
Il **secondo piano** (Figura 5.5), infine, si apre verso l'esterno e ospita una stanza caratterizzata da un pavimento ricoperto di sabbia rossa, elemento che contribuisce a differenziarne ulteriormente l'atmosfera.
Sebbene i dettagli della narrazione e delle dinamiche di gioco siano trattati nel Game Logic Layer (Sezione 5.4.1), è evidente come le scelte di modellazione spaziale contribuiscano direttamente a supportare la storia e l'esperienza del giocatore.

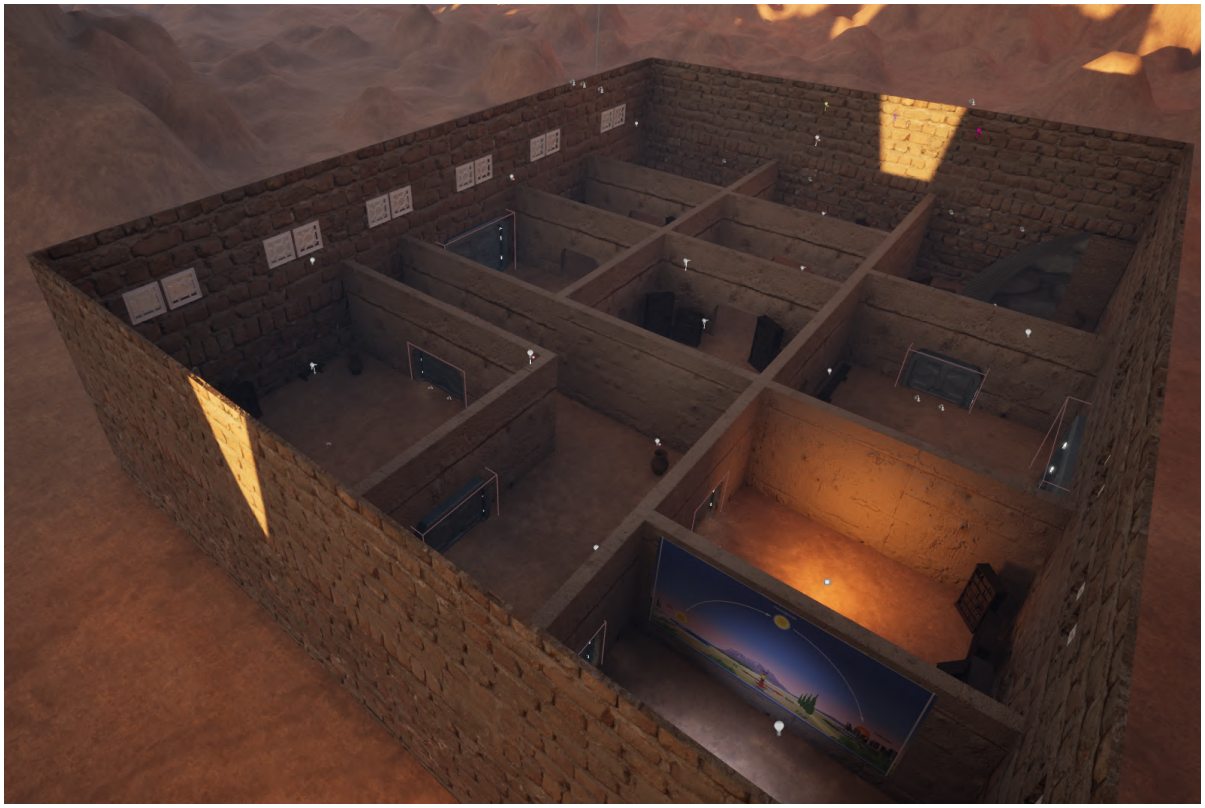

Figura 5.3: Edificio - Piano terra

Figura 5.4: Edificio - Piano primo

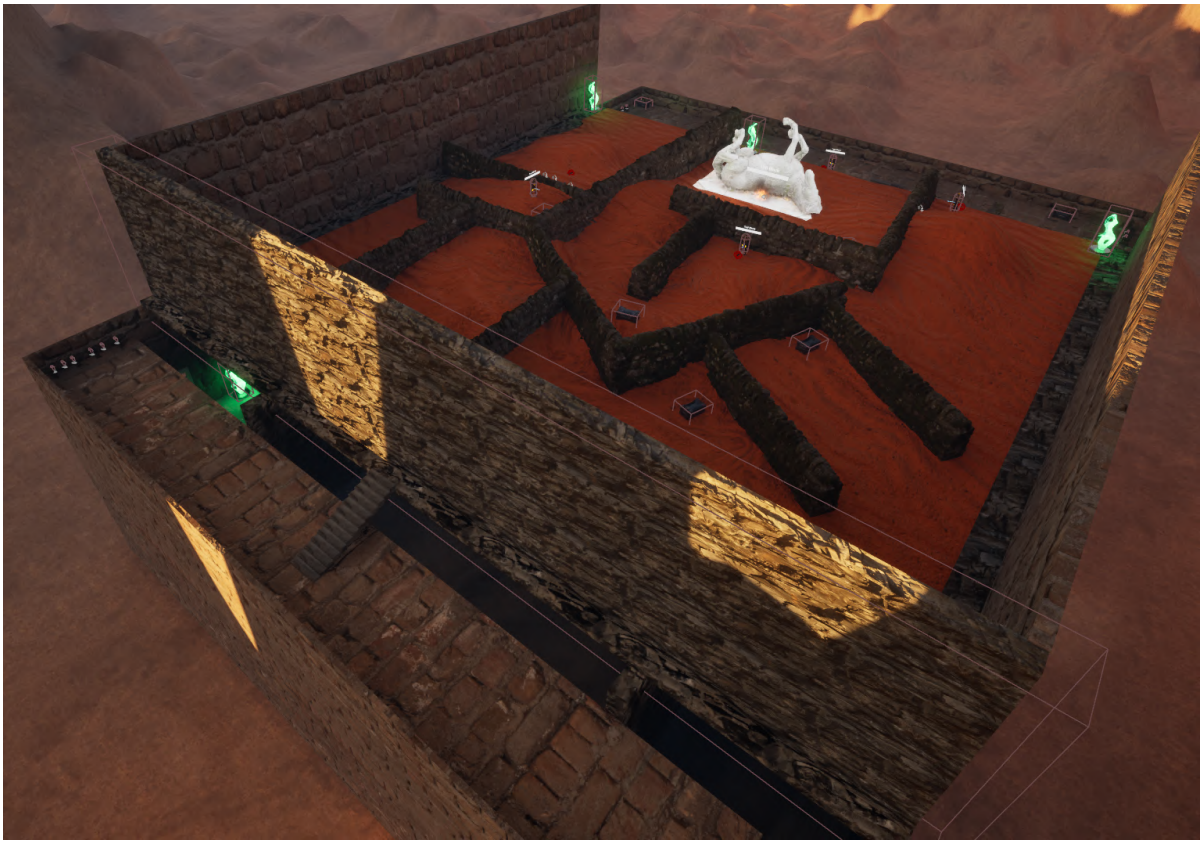

Figura 5.5: Edificio - Piano secondo

5.2.2 Oggetti

In questa sezione gli oggetti presenti nel gioco vengono discussi esclusivamente dal punto di vista della modellazione, attraverso una trattazione di carattere generale relativa alle soluzioni geometriche e alle tecniche di ottimizzazione adottate. L'analisi puntuale dei singoli oggetti, con particolare riferimento alle implicazioni sulla logica di gioco, sarà affrontata separatamente nella Sezione 5.4.2.

Tutti gli oggetti presenti nella scena, come casse di munizioni, tavoli e porte, sono stati importati da **FAB** come asset gratuiti, consentendo di concentrare il lavoro di modellazione su aspetti più legati all'ottimizzazione delle prestazioni, in particolare al numero di triangoli renderizzati per frame. In un approccio tradizionale, questo compito sarebbe stato affidato ai LOD (Level of Detail), ossia diverse versioni della stessa mesh attivate dinamicamente in base alla distanza dalla camera o all'occupazione sullo schermo, in modo da ridurre il carico computazionale degli oggetti lontani o poco visibili.

Per questo progetto, si è scelto di adottare una soluzione più moderna ed efficiente introdotta in Unreal Engine 5, ovvero **Nanite** [14].
Nanite è un sistema di micro-polygon rendering che permette di importare e visualizzare asset 3D ad altissima densità poligonale, come modelli scansionati o creati in ZBrush, senza un impatto significativo sulle prestazioni. Il motore gestisce automaticamente il livello di dettaglio per ogni frame, sfruttando un sistema di streaming e rasterizzazione virtualizzata: ogni oggetto viene suddiviso in micro-cluster di triangoli, e solo quelli visibili e rilevanti rispetto alla camera vengono renderizzati. In questo modo, non è più necessario creare manualmente LOD per ciascuna mesh ad alta risoluzione.
La Figura 5.6 riporta tre immagini esemplificative che illustrano come Nanite adatti dinamicamente la complessità geometrica, modulando il numero di triangoli renderizzati in funzione della distanza dalla camera.
Nanite, pur garantendo prestazioni ottimali, presenta alcune limitazioni: al momento non supporta i Landscape e offre un supporto sperimentale solo per le Skeletal Meshes, come personaggi animati o oggetti deformabili.

La Figura 5.7 mostra un esempio pratico di mesh ottimizzata con Nanite, confrontata con la sua controparte fallback semplificata. La mesh ad alta fedeltà contiene 20.782 triangoli e 11.518 vertici, garantendo dettagli estremamente accurati per superfici complesse come il tavolo presente nell'ambiente. La mesh fallback, utilizzata in scenari in cui Nanite non può essere impiegato, presenta invece solo 588 triangoli e 466 vertici, rappresentando una semplificazione drastica secondo i metodi tradizionali di ottimizzazione tramite LOD.
Questo confronto evidenzia come si sia potuto mantenere un livello di dettaglio geometrico fino a 35 volte superiore rispetto alla mesh semplificata, pur garantendo performance real-time elevate. Grazie a questa tecnologia, si è potuto utilizzare asset estremamente dettagliati nella scena di Gavunia, migliorando la qualità visiva complessiva senza compromettere la fluidità dell'esperienza di gioco.

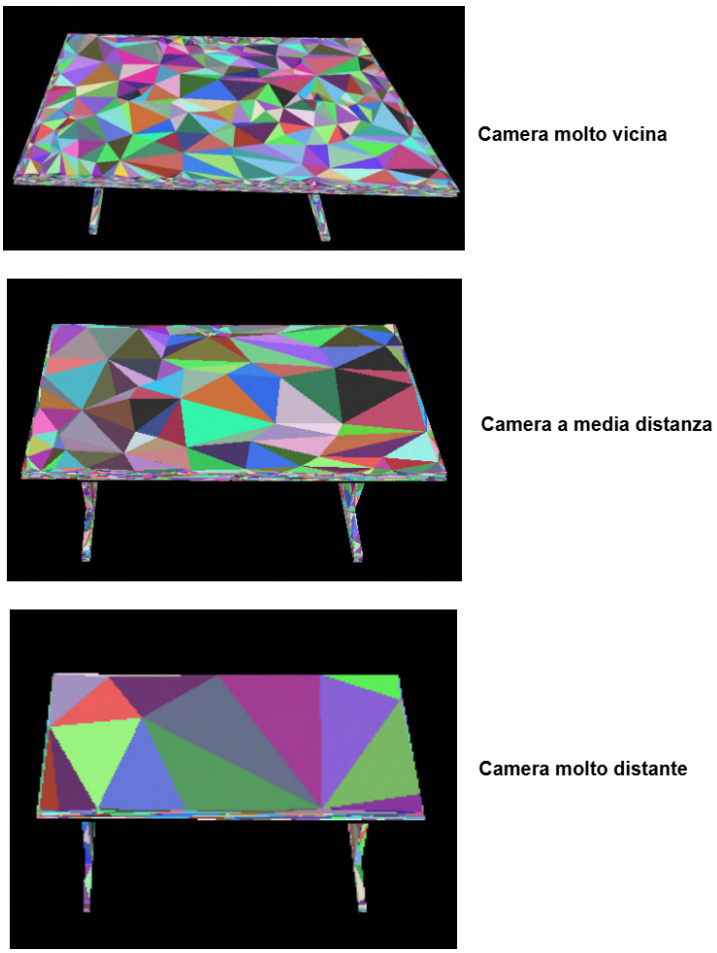

Figura 5.6: Nanite triangle visualization

Figura 5.7: Esempio per confrontare Nanite con LOD

5.2.2.1 Statue

Tra i vari oggetti presenti nel gioco, importati come Static Mesh, le statue meritano una trattazione separata a causa della loro natura parzialmente differente. In particolare, alcune statue distribuite nella scena sono state rese distruttibili. Tale scelta risponde a precise decisioni di game design, che verranno approfondite nella Sezione 5.4.1.

Per implementare oggetti distruttibili in Unreal Engine è stato utilizzato il sistema **Chaos Physics** [15]. Il processo parte da una Static Mesh, dalla quale viene generata una *Geometry Collection*, ovvero una struttura dati che consente di gestire la frammentazione e la simulazione fisica dei frammenti. Successivamente, è stata adottata la modalità *Uniform* per la frattura, suddividendo l'oggetto in pezzi di dimensioni simili. La frammentazione viene attivata tramite un impulso fisico applicato al momento dell'impatto con il proiettile, integrando così il sistema di distruzione con la meccanica di tiro.
La Figura 5.8 illustra il processo di creazione della Geometry Collection.

L'introduzione di oggetti distruttibili ha tuttavia evidenziato alcune criticità prestazionali. Durante la fase di testing è stato osservato un calo significativo degli FPS in presenza di un elevato numero di frammenti attivi nella scena. Questo comportamento è riconducibile al carico computazionale associato alla gestione delle Geometry Collection, in particolare per quanto riguarda l'illuminazione dinamica, il sistema di collisioni e l'aumento delle draw calls necessarie per ciascun frammento.
Per mitigare tali problemi, le Geometry Collection sono state incapsulate all'interno di attori Blueprint, dotati di una logica di rimozione automatica degli oggetti distrutti dopo cinque secondi. Questa strategia ha consentito di ridurre progressivamente il numero di oggetti fisicamente simulati nella scena, alleggerendo il carico sulla GPU e contribuendo al mantenimento di un frame rate stabile, pur preservando l'effetto visivo e l'esperienza di gioco.

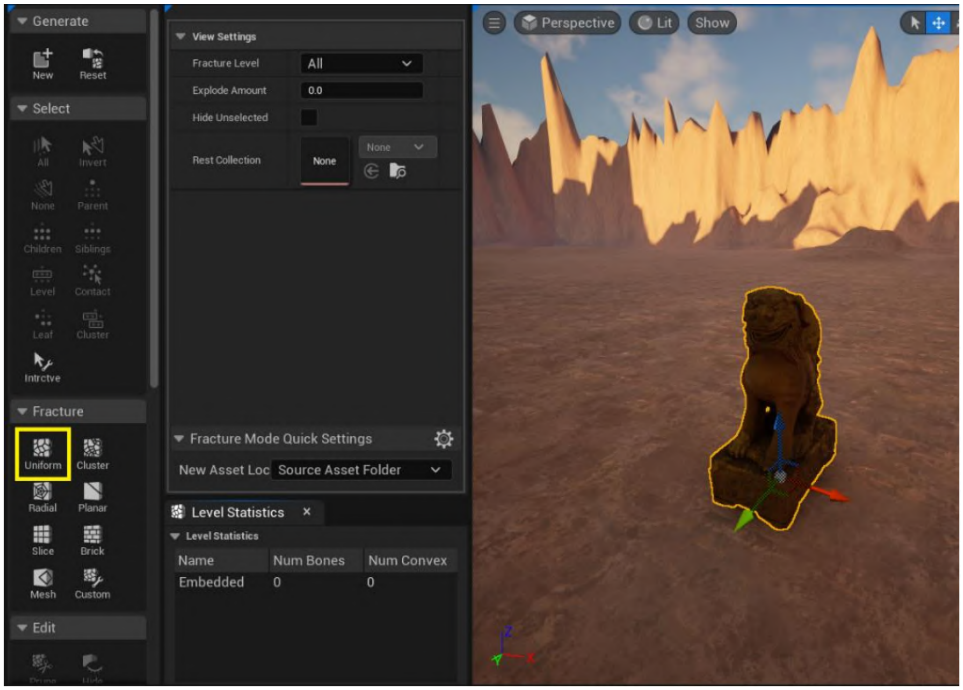

Figura 5.8: Processo per la creazione di Geometry Collection delle statue

5.2.3 Personaggi

Gavunia prevede due tipologie di personaggi: il giocatore e i nemici.
Per quanto riguarda il giocatore, è stata utilizzata la skeletal mesh del manichino Manny standard di Unreal Engine, mentre per i nemici è stata importata da FAB una skeletal mesh di un soldato.
La scelta di non modificare la mesh del giocatore è stata motivata dalla natura del gioco: essendo un titolo in prima persona, senza specchi o superfici riflettenti, il giocatore non visualizza mai il proprio corpo. Per i nemici, invece, è stato fondamentale selezionare una mesh con un livello di dettaglio sufficiente a valorizzare l'aspetto visivo dei personaggi, in quanto direttamente osservabile dal giocatore durante il gameplay.

L'utilizzo di skeletal mesh, rispetto alle static mesh, è motivato dalla presenza di uno scheletro interno che consente di animare singole articolazioni o parti del corpo, a differenza delle static mesh che si comportano come corpi rigidi. Per maggiori dettagli sul sistema di animazione, si rimanda alla Sezione 5.5.1.
La Figura 5.9 mostra la struttura ossea della skeletal mesh utilizzata per i nemici.
Il root bone funge da punto di origine dell'intero scheletro, dal quale si dipartono i principali segmenti corporei come il bacino e la colonna vertebrale. Da questa struttura si diramano le ossa delle braccia e delle gambe, articolate in modo da consentire movimenti naturali e dettagliati.

Un aspetto importante visibile nell'immagine è la presenza di nodi specifici per il controllo tramite *Inverse Kinematics (IK)*, come "ikHandRoot" per le mani e "ikFootRoot" per i piedi. L'IK è una tecnica che consente di animare in modo intuitivo le catene articolari, calcolando automaticamente le posizioni delle articolazioni intermedie a partire dalla posizione finale desiderata, come quella della mano o del piede.
Le linee che collegano le mani e i piedi rappresentano i controller IK, strumenti visivi nell'editor di Unreal Engine utilizzati per sincronizzare e gestire i movimenti degli arti superiori e inferiori. Questi controller facilitano l'animazione di azioni complesse, come l'impugnatura di oggetti con entrambe le mani o il posizionamento realistico dei piedi durante la camminata e le interazioni con il terreno.

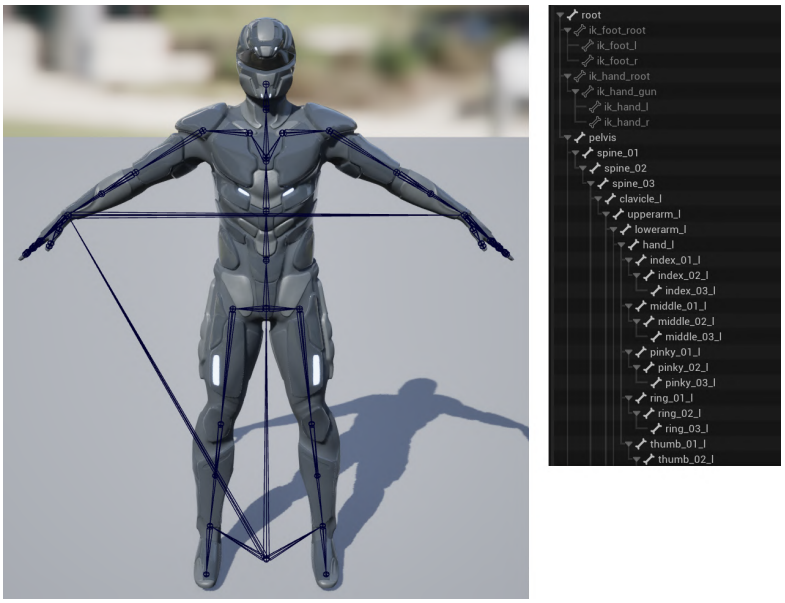

Figura 5.9: Skeletal Mesh dei nemici

5.3 Rendering Layer

Il Rendering Layer costituisce lo strato del sistema dedicato alla gestione dell'illuminazione e dell'aspetto visivo complessivo della scena. Questo layer si occupa della configurazione di luci, ombre e materiali, nonché dell'implementazione di eventuali effetti speciali, come particelle o post-processing.
Questo strato di sviluppo è fondamentale per valorizzare i modelli 3D, conferendo loro realismo e qualità estetica, migliorando così l'esperienza visiva del giocatore.

5.3.1 Illuminazione della scena

Per ottenere un'illuminazione realistica e coerente con l'atmosfera del gioco, è stato adottato il sistema di illuminazione globale **Lumen** [16], introdotto in Unreal Engine 5. Lumen rappresenta una delle innovazioni più significative di questa versione del motore, in quanto permette di simulare in tempo reale l'illuminazione indiretta e le riflessioni, senza necessità di pre-baking o configurazioni complesse.
Questa tecnologia supera i limiti delle soluzioni precedenti, come Lightmass, basato su calcoli statici, e il ray tracing tradizionale, spesso troppo gravoso in termini di prestazioni. Lumen si adatta in modo ottimale a scene dinamiche, reagendo immediatamente ai cambiamenti della luce e della geometria della scena, garantendo un buon compromesso tra qualità visiva e performance.

Nel progetto, l'illuminazione è stata suddivisa in due categorie principali: luci esterne e luci interne agli edifici.

5.3.1.1 Luci esterne

Per definire l'illuminazione generale dello scenario esterno sono stati utilizzati i seguenti attori di Unreal Engine:

- **Directional Light**: simula la luce solare. La rotazione di questo attore permette di modificare l'ora del giorno, influenzando direzione, intensità e colore della luce. In combinazione con Lumen, genera ombre e illuminazione indiretta dinamica.
- **Sky Atmosphere**: simula il comportamento fisico della luce nell'atmosfera terrestre, migliorando il realismo del cielo e la diffusione della luce.
- **Sky Light**: cattura la luce ambientale proveniente dal cielo e la distribuisce sulle superfici, schiarendo le zone in ombra e garantendo un bilanciamento generale dell'illuminazione.
- **Exponential Height Fog**: aggiunge profondità e atmosfera alla scena simulando la foschia in funzione dell'altezza e della distanza, accentuando la percezione di scala e profondità.
- **Volumetric Clouds**: introducono nuvole tridimensionali realistiche, che interagiscono con la luce solare e influenzano l'aspetto del cielo e dell'ambiente sottostante.

La Figura 5.10 illustra l'integrazione congiunta di questi componenti nella configurazione dell'illuminazione della scena esterna.

Figura 5.10: Illuminazione esterna in Gavunia

5.3.1.2 Luci interne

Per l'illuminazione degli ambienti interni sono state impiegate diverse tipologie di sorgenti luminose, selezionate in base alla funzione e alla configurazione spaziale dei singoli ambienti:

- **Point Light**: utilizzate principalmente per garantire un'illuminazione diffusa e omnidirezionale degli spazi interni. Sono stati configurati parametri quali intensità, raggio di attenuazione (Attenuation Radius) e temperatura colore, adattandoli alle specifiche esigenze di ciascun ambiente. Questa tipologia rappresenta la sorgente luminosa più impiegata nel progetto.
- **Spot Light**: impiegate per enfatizzare elementi specifici della scena mediante un fascio luminoso concentrato. In particolare, sono state utilizzate nella "stanza delle statue" per ottenere un effetto scenografico e guidare l'attenzione del giocatore verso punti focali dell'ambiente.
- **Rectangular Light**: adottate per simulare sorgenti luminose con emissione direzionale e superficie estesa, ideali per illuminare pareti o aree ampie in modo uniforme. Questa tipologia consente una distribuzione più morbida e naturale della luce rispetto alle sorgenti puntuali.

Le Figure 5.11, 5.12 e 5.13 mostrano, rispettivamente, un esempio di applicazione delle Point Light, Spot Light e Rectangular Light negli ambienti interni dell'edificio.

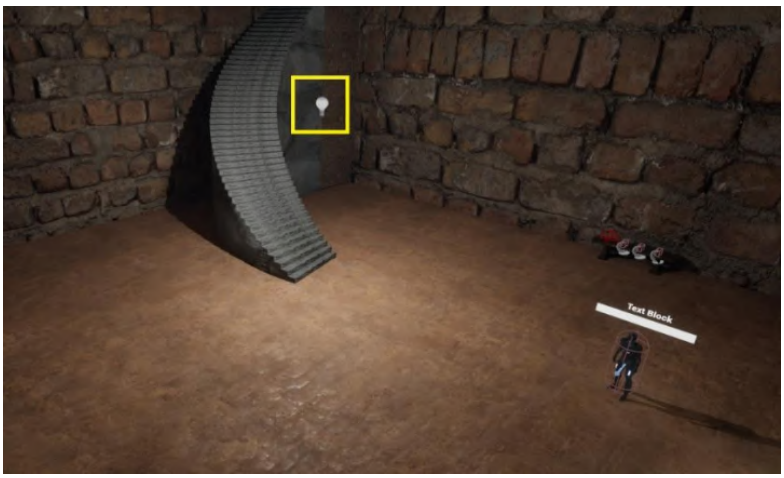

Figura 5.11: Esempio d'uso di Point Light

Figura 5.12: Esempio d'uso di Spot Light

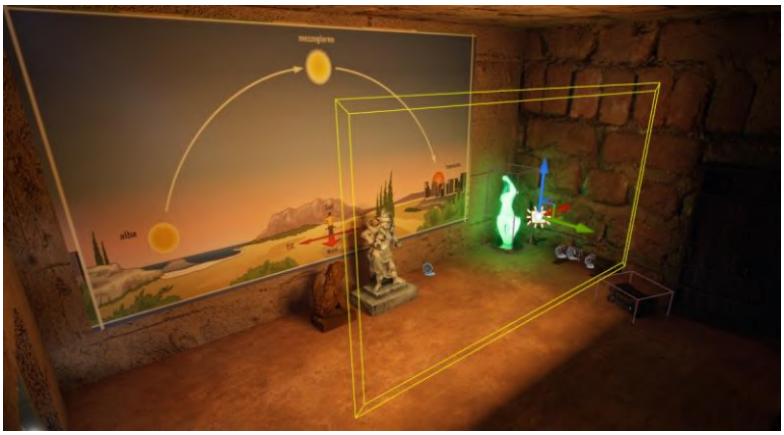

Figura 5.13: Esempio d'uso di Rectangular Light

5.3.2 Materiali

La gestione dei materiali rappresenta un aspetto fondamentale del Rendering Layer, in quanto contribuisce in modo determinante alla resa visiva e al realismo percettivo della scena. In Unreal Engine, i materiali definiscono il comportamento delle superfici rispetto alla luce, regolando proprietà quali colore, riflettanza, rugosità e risposta alle normali.

Una parte significativa degli asset utilizzati nel progetto è stata importata da FAB già completa di materiali e texture associate. In questi casi, le mesh erano dotate del relativo UV mapping, ossia della corretta parametrizzazione della superficie tridimensionale nello spazio bidimensionale, necessaria per l'applicazione coerente delle texture. Ciò ha consentito di integrare rapidamente gli oggetti nella scena mantenendo una buona qualità visiva senza interventi aggiuntivi sulla struttura geometrica.

Per alcuni elementi ambientali è stato invece necessario applicare manualmente materiali esterni reperiti da FAB. In questi casi, dopo l'assegnazione del materiale alla mesh, sono stati regolati specifici parametri per adattarne l'aspetto alla scala e al contesto della scena.
La Figura 5.14 mostra un esempio di tale configurazione, evidenziando il materiale applicato ai muri esterni dell'edificio e i parametri modificati per adattarne l'aspetto alla geometria. In particolare, sono stati regolati:

- *Tiling*: modificato per controllare la ripetizione della texture sulla superficie, evitando distorsioni o pattern visivamente poco credibili sulla superficie del muro.
- *Normal Intensity*: regolata per calibrare l'intensità delle normali, influenzando la percezione dei dettagli superficiali senza aumentare la complessità geometrica.

Tali modifiche hanno contribuito a rendere il muro visivamente più realistico e coerente con l'illuminazione dell'ambiente.

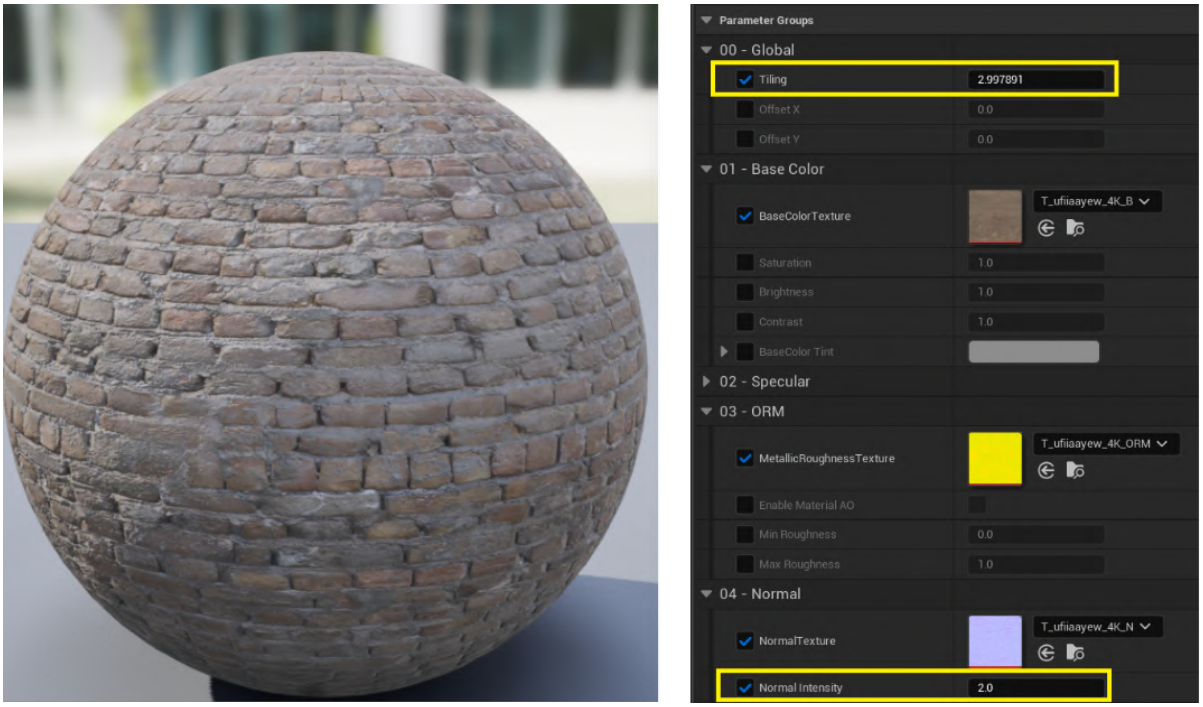

Figura 5.14: Esempio del materiale associato ai muri esterni dell'edificio

5.4 Game Logic Layer

Il Game Logic Layer rappresenta la componente del sistema dedicata alla gestione delle regole, delle interazioni e dei meccanismi che definiscono l'esperienza di gioco. Questo layer si occupa non solo della definizione della storia e degli obiettivi del giocatore, ma anche della gestione degli eventi, delle condizioni di vittoria o sconfitta e delle logiche che governano il comportamento dei personaggi e degli oggetti interattivi.

5.4.1 Arco narrativo e fasi di gioco

Gavunia è un FPS (First Person Shooter) ambientato nella terra fantastica di Gavunia, un mondo incantato avvolto da foreste lussureggianti e abitato da comunità che vivono in armonia con la natura. Tuttavia, anche in questo scenario idilliaco è presente un lato oscuro. Nel profondo di una fortezza dimenticata, sepolta nel tempo e nascosta al mondo, giacciono tre antichi cristalli: da soli non possiedono alcun potere, ma insieme scatenano un'oscurità capace di piegare il mondo alla propria volontà. Questa verità è stata custodita per secoli dai guerrieri di Gavunia, che hanno difeso la fortezza da occhi indiscreti e mani avide.
Il giocatore assume il ruolo di un soldato assetato di potere, che scopre l'esistenza della fortezza e decide di recuperare i cristalli per risvegliare l'antico e malvagio Demone.

Come descritto nel dettaglio nella Sezione 5.2.1, il gioco è ambientato in una landa deserta circondata da montagne, in cui si distinguono due principali punti di interesse: una collina e un edificio.
La progressione del gameplay si sviluppa in tre macro-fasi, che guidano il giocatore attraverso un percorso lineare di apprendimento e sfida:

- La **prima fase** si svolge sulla collina, al di fuori dell'edificio sorvegliato dai soldati nemici. Qui il giocatore può equipaggiarsi e completare un breve tutorial, finalizzato ad acquisire familiarità con i comandi e le meccaniche di base, quali il recupero degli oggetti, lo sparo e la ricarica dell'arma.

- Nella **seconda fase** l'azione si sposta all'interno dell'edificio principale. Il giocatore deve farsi strada tra le guardie nemiche e recuperare i tre cristalli magici nascosti in diverse stanze. Questi cristalli, elementi chiave per sbloccare la porta verso la fase finale, sono occultati all'interno di statue. Alcune statue sono state rese distruttibili per simboleggiare che siano false e indistinguibili da quelle vere, rendendo la ricerca più complessa e stimolante. Tale scelta di design è coerente con le Lenses of Fun (Sezione 2.1.2) e of Flow (Sezione 2.1.6) proposte da Jesse Schell, poiché aumenta il coinvolgimento e mantiene alta la sfida.

- Infine, nella **terza fase**, il giocatore raggiunge l'ultimo piano dell'edificio e deve distruggere la statua di un cavallo che nasconde al suo interno lo Scudo del Demone, obiettivo conclusivo della missione.

L'intera esperienza è accompagnata da una musica di sottofondo che varia in base alla fase di gioco: durante la ricerca dei cristalli predomina un tema più tranquillo e esplorativo, mentre nelle fasi finali di scontro con i nemici la musica assume toni più solenni e intensi, sottolineando il ritmo e il grado di sfida, rafforzando l'immersione del giocatore.

5.4.2 Oggetti interagibili

All'interno del gioco sono stati introdotti diversi oggetti interagibili, progettati seguendo la Lens of Endogenous Value proposta da Schell (Sezione 2.1.3). Secondo tale principio, ogni elemento inserito nel mondo di gioco deve possedere un valore intrinseco per il giocatore, contribuendo in modo significativo al superamento delle sfide proposte e alla progressione dell'esperienza.

In *Gavunia* è possibile distinguere tre principali categorie di oggetti: **oggetti esclusivamente interagibili**, **oggetti inventoriabili** e il **fucile**. I primi permettono al giocatore di interagire direttamente con l'ambiente di gioco, supportando l'esplorazione e la risoluzione di specifiche situazioni. I secondi, invece, possono essere raccolti e conservati all'interno di un inventario virtuale, per essere utilizzati in momenti successivi della partita. Infine, il fucile rappresenta una categoria a sé: una volta raccolto, non può più essere rilasciato e rimane in mano al giocatore fino al termine dell'esperienza, configurandosi come uno strumento centrale e permanente dell'interazione.
L'introduzione di un sistema di inventario risponde a una scelta di game design consolidata nei giochi di avventura e azione, in quanto consente di separare il momento dell'acquisizione degli oggetti da quello del loro utilizzo. Questo approccio favorisce una pianificazione più consapevole delle azioni da parte del giocatore e contribuisce a strutturare la progressione del gioco attorno al raggiungimento di obiettivi intermedi.

5.4.2.1 Oggetti esclusivamente interagibili

Gli oggetti esclusivamente interagibili sono elementi fissi dell'ambiente di gioco che non possono essere né raccolti né spostati. Il loro ruolo principale è quello di influenzare direttamente la progressione e la sopravvivenza del giocatore, agendo sia come ostacoli, quali porte o muri magici, sia come punti di rifornimento, come casse di munizioni o statue vitali.

Porte: Le porte sono presenti nel piano terra dell'edificio e rappresentano un ostacolo alla progressione del giocatore durante la ricerca dei cristalli. L'apertura delle porte è subordinata al possesso di specifiche chiavi, che il giocatore deve recuperare nel corso dell'esplorazione. I nemici, al contrario, non sono soggetti a tale vincolo e possono oltrepassare liberamente le porte, muovendosi tra le diverse stanze dell'edificio. Questa asimmetria rende meno prevedibile la loro presenza nelle aree esplorate dal giocatore, aumentando il livello di tensione e attenzione richiesto.

Muri magici: Analogamente alle porte, anche i muri magici ostacolano l'avanzamento del giocatore nella storia, ma il loro superamento richiede una quantità specifica di cristalli. I due muri magici presenti nel gioco segnano due momenti chiave della progressione narrativa: il primo separa il giocatore dall'ottenimento del terzo cristallo, mentre il secondo introduce la fase finale della missione, precedendo la battaglia per recuperare lo Scudo del Demone. In termini di game design, questi ostacoli fungono da checkpoint cognitivi, fornendo obiettivi chiari e scandendo il ritmo dell'esperienza.

Casse di munizioni: Le casse di munizioni consentono al giocatore di ottenere risorse fondamentali per la sopravvivenza, ovvero proiettili utilizzabili con il fucile. Dal punto

di vista del game design, esse svolgono un ruolo chiave nel bilanciamento della difficoltà e nella gestione delle risorse. Per evitare che il combattimento perda di tensione, il numero di proiettili ottenibili da ciascuna cassa è limitato a un massimo di dieci unità. Questa scelta progettuale mantiene il giocatore in uno stato di attenzione costante, incentivandolo a pianificare l'uso delle munizioni e a valutare con cura gli scontri. Al contempo, la presenza di un numero adeguato di casse distribuite nell'ambiente di gioco riduce il rischio di frustrazione, garantendo un equilibrio tra sfida e accessibilità.

Statue vitali: Le statue vitali rappresentano un importante strumento di supporto nel percorso del giocatore verso il completamento del gioco, in quanto consentono di recuperare una quantità significativa di punti vita. Analogamente alle casse di munizioni, esse sono distribuite all'interno dell'edificio in posizioni strategiche, progettate in relazione alla difficoltà delle aree circostanti.
In particolare, nel piano terra — dove si svolge la fase iniziale di esplorazione e ricerca — la presenza delle statue vitali è più diradata, poiché il livello di minaccia dei nemici è relativamente contenuto rispetto ai combattimenti che caratterizzano il secondo e il terzo piano. Con l'aumentare della difficoltà degli scontri, la loro distribuzione diventa progressivamente più frequente, offrendo al giocatore un supporto maggiore nelle fasi più avanzate dell'esperienza.

La Figura 5.15 presenta una panoramica degli oggetti interagibili discussi, distinguendo quelli che costituiscono ostacoli da quelli che forniscono ricompense al giocatore.

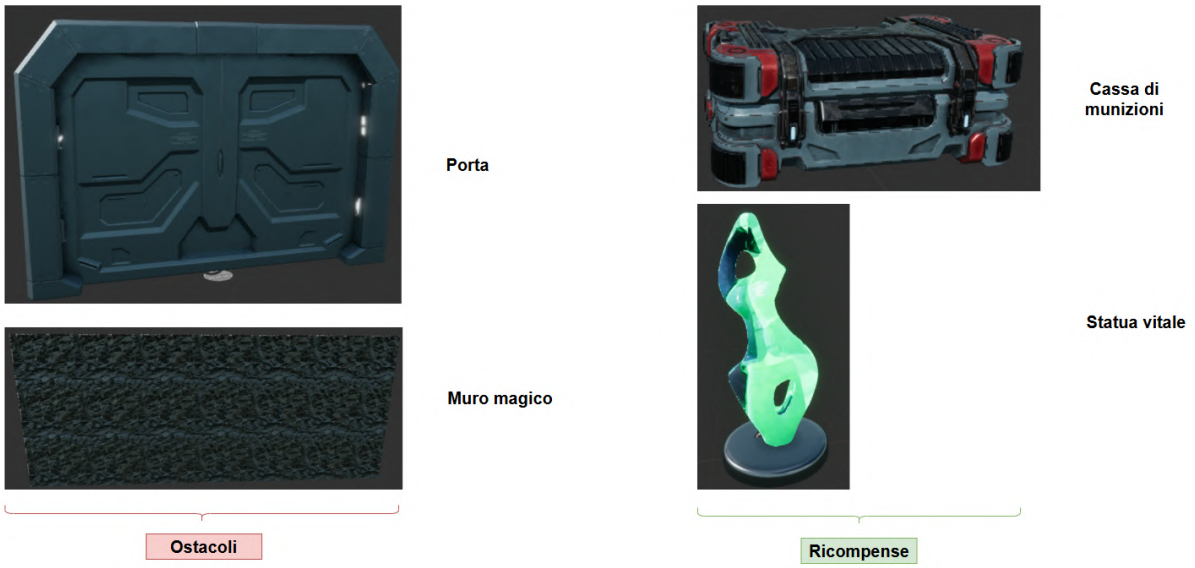

Figura 5.15: Presentazione degli oggetti esclusivamente interagibili

5.4.2.2 Oggetti inventoriabili

Gli oggetti inventoriabili sono elementi mobili che il giocatore può raccogliere, con la raccolta che comporta l'aggiunta diretta all'inventario. In questo modo, l'oggetto sarà disponibile per un utilizzo successivo, secondo le necessità del giocatore. Questi oggetti fungono da ricompense e supporti concreti nella progressione dell'avventura, aiutando il giocatore a superare sfide e a raggiungere gli obiettivi di gioco, incentivando una pianificazione strategica delle azioni.

Munizioni: Le munizioni possono essere trovate all'interno delle casse o sottratte ai nemici una volta sconfitti. Ottenere munizioni dai nemici premia direttamente l'abilità del giocatore, creando un feedback immediato e gratificante per l'azione riuscita. Inoltre, questa dinamica incoraggia il giocatore a pianificare attentamente quali scontri affrontare e quando, introducendo un elemento strategico che si integra con la progressione del gioco. In questo modo, le munizioni contribuiscono anche a scandire il ritmo dei combattimenti e a fornire piccoli obiettivi secondari all'interno delle missioni principali, rafforzando l'esperienza complessiva di esplorazione e sfida.

Cristalli: I cristalli costituiscono l'obiettivo principale del giocatore, in quanto il loro possesso è necessario per accedere alla stanza finale dell'edificio. Essi sono nascosti all'interno di statue magiche distribuite nell'edificio, alcune delle quali false e indistinguibili dalle altre, al fine di aumentare il livello di sfida e il coinvolgimento nella fase di ricerca. La loro collocazione segue una logica di difficoltà incrementale: i primi due cristalli si trovano nel piano terra, mentre il terzo è protetto da una squadra di guardie d'élite in una stanza accessibile solo dopo aver acquisito i primi due, introducendo un momento chiave di progressione narrativa e di sfida per il giocatore.

Chiavi: Le chiavi sono nascoste all'interno dell'edificio e consentono di aprire porte, permettendo l'accesso a nuove stanze. Esse possono essere recuperate anche sconfiggendo i nemici, sebbene non tutti ne siano dotati, introducendo un elemento di casualità che aumenta la sfida per il giocatore.
L'uso di una chiave ne comporta la sua perdita, aggiungendo un ulteriore livello strategico: il giocatore deve decidere se utilizzarla subito per aprire una porta o rimandarne l'uso alla ricerca di un'altra porta, valutando l'esplorazione dell'edificio e pianificando percorsi alternativi per ottimizzare le proprie scelte.

Pozioni: Le pozioni consentono al giocatore di recuperare punti vitali, sebbene in quantità inferiore rispetto alle statue. Possono essere trovate sparse nell'edificio o recuperate sconfiggendo i nemici. Il loro principale vantaggio rispetto alle statue risiede nella possibilità di conservarle nell'inventario e utilizzarle in momenti di difficoltà o in aree dove le statue non sono presenti. Ciò introduce scelte strategiche rilevanti: il giocatore deve valutare attentamente quando e dove impiegare le pozioni, considerando la loro disponibilità limitata e la possibilità, in alcune situazioni, di affidarsi invece alle statue per il recupero della salute.

La Figura 5.16 mostra gli oggetti appena descritti.

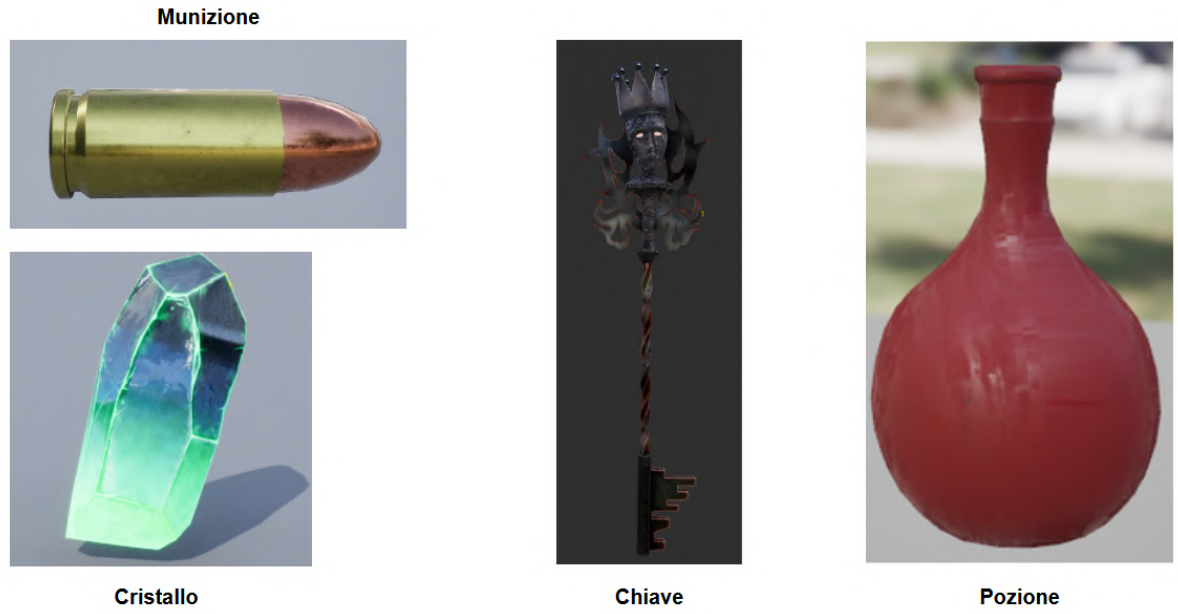

Figura 5.16: Presentazione degli oggetti inventoriabili

5.4.2.3 Fucile

Il fucile rappresenta l'unica arma disponibile nel gioco e, una volta raccolto, rimane in possesso del giocatore fino alla conclusione dell'esperienza. Questa scelta di game design semplifica la gestione dell'inventario e delle risorse, permettendo al giocatore di concentrarsi sulla strategia e sull'esplorazione senza doversi preoccupare di cambiare arma. Allo stesso tempo, la permanenza del fucile garantisce coerenza nell'esperienza di combattimento e lascia aperta la possibilità di introdurre in futuro nuove tipologie di armi senza stravolgere le meccaniche di base.

5.4.2.4 Considerazioni di game design sugli oggetti

Dal punto di vista del game design, l'insieme di ostacoli fissi e punti di rifornimento contribuisce a catturare e mantenere l'attenzione del giocatore attraverso una progressione a difficoltà incrementale. La presenza di obiettivi chiari, come la ricerca delle chiavi o dei cristalli necessari per superare determinate aree, si alterna a momenti di ricompensa rappresentati dall'accesso a nuove zone o dal recupero di risorse, con l'obiettivo di mantenere il giocatore in uno stato di equilibrio tra sfida e competenza, evitando sia la frustrazione sia la perdita di coinvolgimento (Lens of Flow, Sezione 2.1.6).

Allo stesso tempo, la possibilità di esplorare liberamente, raccogliere risorse e pianificare strategicamente l'uso degli oggetti stimola curiosità, senso di libertà e gratificazione personale, favorendo esperienze di scoperta e interazione volontaria che arricchiscono la progressione del gioco (Lense of Fun, Sezione 2.1.2).

Nel loro insieme, queste dinamiche rendono la progettazione degli oggetti coerente, stimolante e capace di coinvolgere attivamente il giocatore in ogni fase dell'esperienza.

5.4.3 Comportamento IA nemica

Il comportamento dell'intelligenza artificiale nemica rappresenta uno degli elementi centrali nella definizione dell'esperienza di gioco.
Dal punto di vista concettuale, ogni nemico può essere visto come un agente autonomo dotato di tre macro-funzionalità principali: percezione dell'ambiente, presa di decisione ed esecuzione delle azioni.

5.4.3.1 Percezione

La percezione dell'ambiente rappresenta la prima fase del comportamento dell'IA nemica e costituisce il punto di ingresso delle informazioni provenienti dal mondo di gioco. Affinché i nemici possano reagire in modo credibile alle azioni del giocatore, è necessario che siano in grado di rilevare eventi significativi e di interpretare correttamente ciò che accade nello spazio circostante.

In *Gavunia*, la percezione dell'IA è basata su stimoli visivi e uditivi, che simulano in maniera semplificata i sensi di vista e udito dei personaggi controllati dalla CPU. La percezione visiva consente ai nemici di individuare il giocatore quando entra nel loro campo visivo, mentre la percezione uditiva permette di reagire a eventi sonori rilevanti, come gli spari o l'impatto dei proiettili. La combinazione di questi sensi permette ai nemici di adattare il proprio comportamento non solo alla posizione del giocatore, ma anche alle sue azioni, aumentando il senso di reattività e tensione durante il gameplay.
Per implementare questi meccanismi, è stato utilizzato il componente **AI Perception** di Unreal Engine 5 [17]. Questo strumento consente di gestire in modo centralizzato i sensi dei nemici, catturando stimoli dall'ambiente e filtrando le informazioni in base a criteri specifici, rilevando solo attori di interesse come il giocatore e evitando interazioni indesiderate, come conflitti tra nemici alleati.

Come mostrato in Figura 5.17, i sensi vengono attivati e aggiornati continuamente in base alla scena circostante. È possibile osservare come i range di vista e di ascolto possano essere configurati separatamente: nel progetto, ad esempio, è stato dato maggior peso alla percezione visiva rispetto a quella uditiva più limitata.

Figura 5.17: Fase di percezione della IA: in verde lo stimolo visivo, mentre in giallo quello uditivo.

5.4.3.2 Processo decisionale

La presa di decisione rappresenta il nucleo centrale del comportamento dell'IA nemica e costituisce la fase in cui le informazioni raccolte durante la percezione dell'ambiente vengono elaborate per determinare l'azione più appropriata da eseguire. Questa fase è gestita attraverso l'utilizzo dei **Behavior Tree** di Unreal Engine 5, che permettono di strutturare la logica comportamentale in modo gerarchico e modulare. [18]
All'interno di questa architettura, il Behavior Tree agisce come il "cervello" dell'IA, coordinando le diverse attività del nemico sulla base dello stato corrente del gioco e degli input percettivi disponibili. I comportamenti sono organizzati in una serie di nodi che rappresentano azioni o condizioni, valutati dinamicamente durante l'esecuzione per selezionare il percorso comportamentale più adatto alla situazione.

Un elemento chiave della presa di decisione è la gestione delle priorità tra le diverse attività. Il comportamento complessivo dell'IA è infatti suddiviso in più azioni caratterizzate da differenti livelli di priorità, come illustrato nella Tabella 5.1. Le attività con valore di priorità più basso risultano predominanti rispetto alle altre: ad esempio, attaccare il giocatore ha priorità maggiore rispetto alla fase di pattugliamento. Di conseguenza, il rilevamento del player comporta l'interruzione delle attività a priorità inferiore, garantendo una risposta immediata e appropriata alla situazione di gioco.
A supporto del processo decisionale è stato impiegato anche l'**Environment Query System (EQS)** [19], che consente ai nemici di raccogliere informazioni contestuali sull'ambiente circostante. Attraverso tale sistema, l'IA è in grado, ad esempio, di individuare posizioni di copertura o punti strategici da cui attaccare, rendendo il comportamento dei nemici coerente con lo spazio di gioco e sensibile alle scelte del giocatore.

Priorità	Nome	Descrizione
1	Take weapon	Recupero dell'arma più vicina
2	Heal	Cura se la vita è troppo bassa
3	Attack player	Attacco se individuato il player
4	Go to ammo box or reload	Ricarica arma o ricerca munizioni quando non si hanno proiettili nel caricatore
5	Location inspection	Ispezione di una location se sentito un rumore o se ultima posizione nota del player
6	Take cover	Ricerca di una posizione di riparo se subito danno
7	Patrol	Pattugliamento generale

Tabella 5.1: Comportamenti principali della IA nemica

5.4.3.3 Esecuzione delle azioni

L'esecuzione delle azioni rappresenta la fase conclusiva del ciclo comportamentale, durante la quale le decisioni prese dall'IA vengono tradotte in operazioni concrete e percepibili dal giocatore, rendendo così visibile ciò che l'intelligenza artificiale "ha deciso di fare". Queste azioni si articolano principalmente in due categorie: movimento e interazione.

Movimento: Per quanto riguarda il movimento, i nemici si affidano alla **NavMesh** [20], una struttura che definisce le aree percorribili della mappa e permette all'IA di pianificare percorsi coerenti e realistici. Tuttavia, poiché non tutte le zone navigabili sono immediatamente connesse, si utilizzano strumenti come **NavLink**, che abilitano la gestione di eventi specifici quali l'apertura di porte o il superamento di ostacoli tramite salti, garantendo così un movimento fluido e credibile (Figura 5.18).

Interazione: Oltre alla semplice locomozione, l'esecuzione comprende anche l'interazione con oggetti o elementi dell'ambiente, come il recupero di risorse, l'utilizzo di coperture o l'attivazione di meccanismi di gioco. In questo contesto, alcune azioni particolarmente rilevanti, come l'attacco diretto al giocatore, vengono implementate tramite task personalizzate all'interno del Behavior Tree, che estendono la logica standard e traducono le decisioni astratte in comportamenti concreti e specifici, garantendo al contempo modularità e riusabilità del codice.

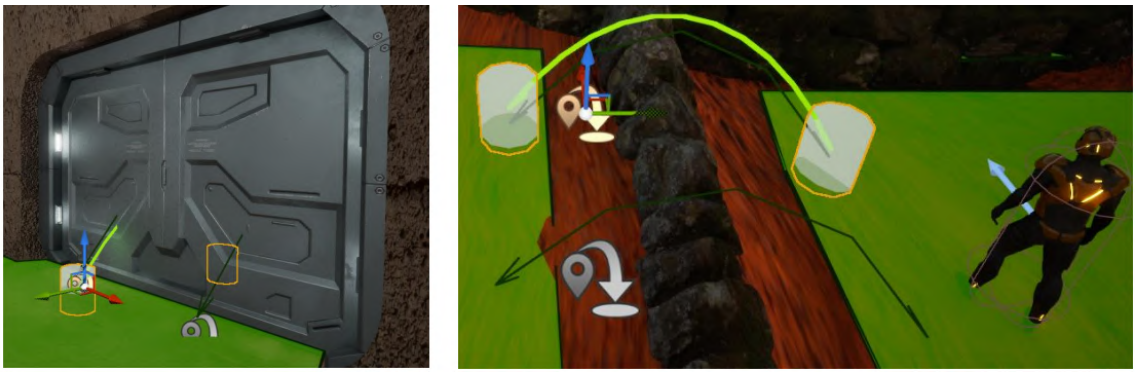

Figura 5.18: Utilizzo di NavLink per superamento di ostacoli nel movimento della IA

5.4.4 Logica di business

Durante lo sviluppo di *Gavunia* desktop è stata seguita una progettazione orientata ai design pattern, con l'obiettivo non solo di scrivere codice funzionante, ma anche ben strutturato, modulare, manutenibile ed estendibile nel tempo. In particolare, è stato adottato il **Component Pattern** [21], sfruttando il sistema degli *ActorComponent* di Unreal Engine 5. Questo approccio consente di delegare specifiche funzionalità a componenti modulari, separando le responsabilità e riducendo la complessità degli attori principali (*AActor o APawn*), senza compromettere la coesione della logica di gioco.
Il pattern può essere paragonato, in maniera intuitiva, alla struttura del corpo umano: ogni componente svolge un compito preciso, ma è l'insieme coordinato di tutte le parti a generare un sistema complesso e funzionante. Allo stesso modo, combinando diversi *ActorComponent*, è possibile costruire attori con comportamenti articolati e coerenti, garantendo modularità, riusabilità e facilità di estensione. Il vantaggio principale di questa strategia è che un singolo componente può essere impiegato da più attori senza duplicazione del codice, semplificando l'inserimento di nuove funzionalità o la creazione di nuovi attori.
In combinazione con i componenti, sono state utilizzate **interfacce Blueprint (BPI)** per permettere la comunicazione tra attori in modo disaccoppiato. Questo approccio evita l'uso eccessivo del casting, che introduce dipendenze rigide e riduce la flessibilità del codice. Le interfacce, al contrario, definiscono un contratto astratto che diversi attori possono implementare indipendentemente dalla loro classe base o dalla struttura interna.

Un ulteriore vantaggio di questo approccio modulare si è rivelato durante il porting dalla versione desktop a quella VR. Grazie alla separazione tra logica di business e interazione con il giocatore, i sistemi fondamentali del gioco sono rimasti intatti, e lo sviluppo si è potuto concentrare quasi esclusivamente sulla ridefinizione delle interazioni (HCI) per il nuovo medium. Così, la logica di gioco sottostante, dalla gestione dei danni a quella dell'inventario, ha continuato a funzionare senza modifiche, mentre l'esperienza immersiva è stata adattata alle esigenze della realtà virtuale.

5.4.4.1 Gestione della salute e sistema di attacco

La gestione della salute e dei danni degli attori è affidata al componente **DamageSystem**, progettato per centralizzare tutta la logica relativa allo stato vitale dei personaggi. In questo sistema, le informazioni fondamentali come la vita massima e la vita corrente vengono monitorate costantemente, così come lo stato dell'attore (vivo o morto).
Per garantire modularità e riusabilità, il componente lavora in combinazione con l'interfaccia personalizzata **Damagable**, che espone i metodi necessari per gestire danni e vita. Qualsiasi attore che deve poter subire o infliggere danni può implementare questa interfaccia, delegando la logica concreta al DamageSystem. In questo modo, il sistema rimane flessibile e facilmente estendibile: nuovi attori possono essere introdotti senza modificare quelli esistenti, mantenendo intatta la coesione della logica di gioco.

Un ulteriore elemento chiave del sistema è il meccanismo di token per attacchi collaborativi, pensato per coordinare il comportamento dei nemici e bilanciare il gameplay. Prima di attaccare il giocatore, un nemico deve acquisire un token: se non sono disponibili, resta in uno stato di attesa pur mantenendo il focus sul giocatore. Terminato

l'attacco, il token viene rilasciato, consentendo ad altri nemici di agire. Nel progetto, il numero massimo di token disponibili è impostato a due, limitando così il numero di nemici che possono attaccare contemporaneamente. Questa logica è integrata all'interno del Behavior Tree, assicurando un comportamento coordinato dell'IA.

5.4.4.2 Gestione dell'inventario

La gestione dell'inventario è affidata al componente **InventorySystem**, progettato come un sistema modulare e facilmente estendibile, in grado di supportare diverse tipologie di oggetti e modalità di utilizzo senza introdurre dipendenze rigide con gli attori che lo utilizzano. L'obiettivo principale è stato quello di separare la logica di gestione degli oggetti dalle modalità di interazione, garantendo una struttura flessibile e riutilizzabile. Il componente rappresenta il nucleo centrale del sistema e si occupa di mantenere lo stato dell'inventario, gestendo l'aggiunta e la rimozione degli oggetti. Gli item sono organizzati in slot logici che associano a ciascun oggetto una quantità, permettendo di gestire sia oggetti unici sia risorse accumulabili, come munizioni o consumabili.
Per favorire il disaccoppiamento tra inventario e attori di gioco, il sistema fa ampio uso di interfacce Blueprint. In particolare, l'interfaccia **Inventory** definisce le operazioni minime che un attore deve esporre per poter possedere un inventario, mentre **Consumable** viene implementata dagli oggetti che possono essere utilizzati o consumati.
A queste si affianca l'interfaccia **Interactable**, che identifica gli oggetti con cui è possibile interagire nel mondo di gioco e rappresenta il punto di ingresso verso il sistema di inventario: nel caso degli oggetti inventariabili, il metodo *Interact* comporta l'aggiunta all'inventario dell'attore. Questo approccio consente al giocatore e ai nemici controllati dall'IA di condividere la stessa logica di inventario senza dipendere da classi specifiche.
Un ruolo chiave è svolto anche dall'uso di **Data Table**, che contiene le informazioni descrittive degli oggetti di gioco. Attraverso questo meccanismo, il sistema è in grado di determinare dinamicamente le proprietà e il comportamento di un oggetto a partire da un identificatore, separando i dati dai comportamenti e semplificando l'introduzione di nuovi contenuti senza modificare la logica esistente.

Dal punto di vista funzionale, il sistema distingue chiaramente tra aggiunta e rimozione degli oggetti. L'aggiunta avviene in seguito all'interazione con elementi raccoglibili presenti nel mondo di gioco, mentre la rimozione può verificarsi in due scenari principali: il consumo dell'oggetto, che produce un effetto diretto sul personaggio (ad esempio il recupero di salute o munizioni), oppure il rilascio dell'oggetto nell'ambiente, rendendolo nuovamente interagibile. Questa distinzione consente di gestire in modo uniforme comportamenti diversi, mantenendo la logica concentrata all'interno del componente di inventario.

Analogamente agli altri sistemi di business logic, anche la gestione dell'inventario è rimasta inalterata nel passaggio dalla versione desktop a quella VR. Grazie alla separazione tra logica e interazione, il porting ha richiesto unicamente l'adattamento delle modalità di accesso e visualizzazione dell'inventario (si veda Sezione 6.3.5.4), mentre il funzionamento interno del sistema ha continuato a operare senza modifiche. Ciò conferma l'efficacia dell'approccio modulare adottato e la sua idoneità a supportare l'evoluzione del progetto verso nuovi medium.

5.4.5 Livelli di difficoltà

La versione desktop di *Gavunia* prevede la possibilità di selezionare il livello di difficoltà all'inizio dell'esperienza di gioco.
In particolare, la difficoltà incide sui punti vita iniziali del giocatore e su quelli dei nemici, secondo una relazione inversamente proporzionale: a livelli di difficoltà più bassi corrisponde una maggiore resistenza del giocatore e una minore resistenza dei nemici, mentre all'aumentare della difficoltà il rapporto si inverte, rendendo gli scontri progressivamente più impegnativi.

Dal punto di vista del game design, questa soluzione permette di modulare la tolleranza all'errore e l'intensità del combattimento senza alterare la struttura delle meccaniche di gioco o il comportamento della IA nemica. Un numero maggiore di punti vita favorisce un approccio più esplorativo e meno punitivo, mentre valori più contenuti richiedono una gestione più attenta degli scontri e delle risorse, incidendo sul ritmo dell'esperienza. In questo modo, la selezione della difficoltà contribuisce al mantenimento dello stato di flow, adattando la sfida alle capacità del giocatore senza introdurre variazioni strutturali nel gameplay.

5.5 Animation Layer

Animation Layer è responsabile della gestione, coordinazione e riproduzione di tutte le animazioni presenti nel sistema. Dal punto di vista architetturale, questo livello riceve input dal Game Logic Layer (ad esempio cambi di stato del personaggio come idle, walk, run, attack) e li mappa in configurazioni animate tramite controller e macchine a stati. In questo modo viene mantenuta una separazione chiara tra comportamento logico e rappresentazione visiva, migliorando modularità e manutenibilità del sistema.

5.5.1 Personaggi

Per garantire un adeguato livello di realismo e coerenza percettiva, il sistema di animazione dei personaggi è stato progettato con l'obiettivo di assicurare movimenti fluidi, naturali e strettamente sincronizzati con gli eventi di gameplay. Particolare attenzione è stata dedicata alle azioni legate al combattimento, come l'utilizzo dell'arma, la ricarica e lo sparo, in quanto elementi centrali dell'esperienza di gioco.
Nel contesto di Unreal Engine, la gestione delle animazioni è demandata a una classe derivata da *AnimInstance*, configurata attraverso un **Animation Blueprint** [22]. Questa soluzione consente di separare in modo netto la logica comportamentale del personaggio dalla sua rappresentazione visiva animata.

Nel progetto sono state impiegate due principali Animation Blueprint:

- **ABPManny**, fornita di default nello starter pack del motore e successivamente modificata per integrare animazioni personalizzate per il personaggio controllato dal player.

- **ABPWarrior**, ottenuta tramite processo di retargeting e adattata alla skeletal mesh del personaggio controllato dall'intelligenza artificiale.

5.5.1.1 Gestione del Anim Graph

L'elemento centrale di una classe derivata da AnimInstance è l'AnimGraph, che definisce il flusso di valutazione delle animazioni e ospita la macchina a stati responsabile della gestione delle transizioni tra differenti configurazioni motorie del personaggio.

Per entrambe le Animation Blueprint adottate nel progetto, il nucleo della macchina a stati è rappresentato dal modulo di Locomotion, incaricato di governare le animazioni fondamentali di movimento (idle, camminata, corsa), sia in condizioni standard sia con arma equipaggiata.
Le transizioni tra stati sono regolate da condizioni logiche valutate a runtime. Ogni arco della macchina a stati è infatti associato a una o più espressioni booleane che determinano il passaggio da una configurazione animata a un'altra. Ad esempio, il passaggio dallo stato "Idle" allo stato "Idle con arma" è subordinato al valore della variabile "hasWeapon", impostata dalla logica di gioco. Questo meccanismo consente di modellare il comportamento animato del personaggio come funzione diretta del suo stato interno, garantendo coerenza tra dinamica di gameplay e rappresentazione visiva.
Inoltre, ogni stato della macchina può definire una singola sequenza animata oppure un blend parametrico di più animazioni, eseguito in loop fino al verificarsi delle condizioni di transizione. Questo approccio permette di ottenere transizioni fluide tra animazioni differenti, come ad esempio il passaggio graduale dalla camminata alla corsa, migliorando la percezione di naturalezza dei movimenti.

5.5.1.2 Animazioni event-driven: Animation Montage

Mentre le animazioni gestite tramite State Machine risultano particolarmente adatte a rappresentare stati persistenti del personaggio, come locomozione, idle o mira con arma, alcune azioni di gameplay richiedono animazioni event-driven, ovvero eseguite una sola volta in risposta a eventi puntuali. Esempi tipici includono lo sparo di un'arma, il lancio di una granata o la reazione a un danno subito. Queste azioni non si prestano a essere integrate nella macchina a stati di base, poiché sono transitorie e devono poter essere attivate in maniera controllata senza interrompere lo stato locomotorio principale del personaggio.
Per gestire questo tipo di animazioni, Unreal Engine 5 mette a disposizione le **Animation Montage** [23], ovvero asset che permettono la riproduzione di sequenze animate su richiesta.

Nel progetto, un'applicazione concreta di questo meccanismo riguarda la reazione ai danni subiti dai personaggi. Quando il personaggio riceve un colpo, la logica di gioco invoca il metodo "Play Montage", che riproduce il montaggio contenente l'animazione "HitReaction". In questo modo, la risposta visiva al danno viene eseguita senza interferire con gli stati di locomozione o di combattimento già in corso, garantendo continuità e coerenza nell'esperienza di gioco.

5.5.1.3 Animazioni personalizzate

Sebbene lo starter pack di Unreal Engine fornisca un set base di animazioni per il personaggio Manny, le specifiche esigenze progettuali hanno reso necessaria l'estensione di tale repertorio, in particolare per quanto riguarda le azioni di combattimento quali ricarica dell'arma, sparo e locomozione in assetto armato.
Le animazioni aggiuntive sono state importate da Fab. In particolare, è stato utilizzato un pacchetto di animazioni sviluppato per il Mannequin di Unreal Engine 4.
Dal momento che il personaggio Manny impiegato nel progetto utilizza lo scheletro aggiornato della versione più recente del motore, si è reso necessario un processo di retargeting per garantire la compatibilità tra strutture scheletriche differenti.
A tal fine, è stato impiegato il sistema di **IK Retargeting** messo a disposizione da Unreal Engine 5, che consente il trasferimento delle animazioni tra scheletri non perfettamente coincidenti preservando coerenza nei movimenti e proporzioni. Tale procedura è stata successivamente applicata anche al personaggio controllato dall'intelligenza artificiale, la cui skeletal mesh presenta differenze rispetto a quella standard.

La Figura 5.19 illustra il processo di retargeting delle animazioni appena descritto. Tale procedura ha consentito il riuso di animazioni originariamente progettate per strutture scheletriche differenti, riducendo significativamente i tempi di produzione rispetto alla realizzazione ex novo delle stesse.

Nel caso specifico del progetto, il processo è risultato relativamente lineare poiché gli scheletri coinvolti condividono una configurazione antropomorfa analoga. Le principali catene cinematiche (arti superiori, arti inferiori e colonna vertebrale) presentano infatti una corrispondenza strutturale coerente, rendendo la mappatura delle ossa sostanzialmente uno-a-uno, con differenze limitate prevalentemente alle proporzioni e non alla topologia dello scheletro.
In scenari caratterizzati da divergenze morfologiche più marcate — ad esempio nel trasferimento di animazioni tra un modello umanoide e una creatura non antropomorfa — il processo richiederebbe interventi manuali più significativi, sia nella definizione delle pose di riferimento sia nell'adattamento delle catene cinematiche. In tali casi, il retargeting automatico non garantirebbe risultati immediatamente soddisfacenti, rendendo necessario un lavoro di rifinitura o la produzione di animazioni dedicate.

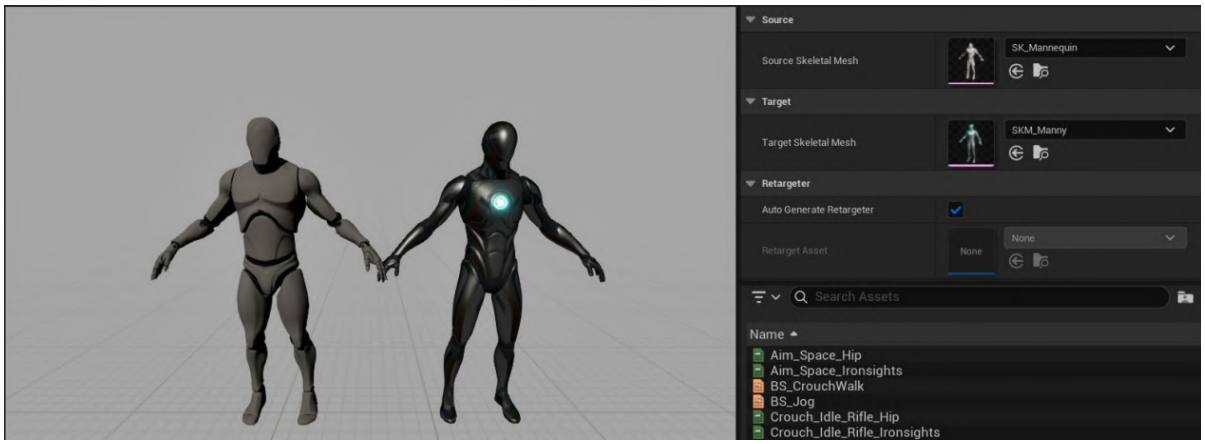

Figura 5.19: Retarget delle animazioni da Mannequin UE4 a Manny UE5

5.5.2 Oggetti

Per quanto riguarda gli oggetti interagibili, il sistema di animazione adottato differisce sostanzialmente da quello impiegato per i personaggi. A differenza di questi ultimi, modellati tramite skeletal mesh e dotati di una gerarchia ossea articolata, gli oggetti sono stati implementati come static mesh, ovvero corpi rigidi privi di articolazioni interne.
Tale distinzione strutturale ha influenzato direttamente la scelta dello strumento di animazione. In questo caso non si è reso necessario l'utilizzo di AnimInstance, bensì di **Timelines** [24], messe a disposizione da Unreal Engine 5, che consentono di definire variazioni temporali di parametri quali posizione, rotazione o scala.

L'impiego di Timeline puntuali si è rivelato adeguato in quanto le animazioni richieste sono di natura deterministica e limitate a trasformazioni rigide, come l'apertura di una porta. In tali casi, l'adozione di un sistema basato su skeletal animation avrebbe introdotto complessità non necessarie, risultando sproporzionata rispetto alla semplicità strutturale dell'oggetto animato.

Nella versione desktop del progetto, tra gli oggetti interagibili descritti nella Sezione 5.4.2.1, solo le porte e i muri magici sono dotati di animazione esplicita. Al contrario, elementi quali statue e casse di munizioni svolgono esclusivamente una funzione di ricompensa: una volta attivata l'interazione tramite pressione di un tasto, l'oggetto eroga il beneficio previsto e viene rimosso dalla scena, senza necessità di una rappresentazione animata intermedia.
Nel Capitolo 6 verrà tuttavia mostrato come il porting in ambiente VR abbia richiesto una revisione di tale approccio. La maggiore enfasi sulla presenza fisica e sull'interazione diretta propria dell'esperienza immersiva ha reso necessaria l'introduzione di animazioni anche per oggetti precedentemente gestiti in modo istantaneo, al fine di preservare coerenza percettiva e credibilità dell'ambiente virtuale.

Capitolo 6

Sviluppo di Gavunia VR

In questo capitolo verranno illustrate le tecniche e i procedimenti che hanno permesso di trasferire *Gavunia* dalla versione desktop alla versione VR.
Il porting non si è limitato a rimappare i comandi da tastiera e mouse ai motion controller, ma ha richiesto un vero e proprio redesign dell'esperienza di gioco, sia a livello di game design sia di interaction design.
L'obiettivo principale è stato sfruttare appieno le potenzialità del medium VR, ridefinendo il modo in cui il giocatore interagisce con l'ambiente, gli oggetti e le sfide del gioco, garantendo al contempo immersione, intuitività e divertimento.

Tenendo a riferimento i quattro layer discussi nel Capitolo 5, il porting alla versione VR ha comportato modifiche specifiche in ciascun ambito:

- **Modelling Layer**: il redesign dell'interazione con gli oggetti (maggiori dettagli nella Sezione 6.3.4) ha reso necessario adattare alcune mesh, modificandone proporzioni e punti di presa, al fine di garantire un'interazione naturale e coerente con la percezione immersiva.

- **Rendering Layer**: il passaggio dalla resa desktop a quella immersiva ha comportato un aumento del carico di rendering, poiché in VR ogni occhio riceve una propria immagine stereoscopica. Si è quindi resa necessaria una fase di ottimizzazione delle performance. Maggiori dettagli sulle strategie di ottimizzazione e sulle valutazioni delle prestazioni sono riportati nel Capitolo 7.

- **Game Logic Layer**: il redesign della HCI ha comportato modifiche alle modalità di interazione con gli oggetti, adattando le logiche di pickup, manipolazione e utilizzo agli input dei motion controller VR, in modo da rendere l'esperienza intuitiva e coerente con le aspettative del giocatore.

- **Animation Layer**: l'interazione immersiva ha reso necessaria la definizione di nuove animazioni, al fine di garantire coerenza tra i movimenti del giocatore e quelli degli oggetti nel mondo virtuale.

6.1 Configurazione hardware

Gavunia VR è stato sviluppato per un contesto PC-VR, e non come applicazione standalone eseguita direttamente sul visore. Tale scelta ha consentito di sfruttare la potenza computazionale di un computer dedicato, permettendo l'impiego di tecniche di resa grafica più avanzate rispetto ai vincoli tipici dei dispositivi autonomi.
Alla luce di questa impostazione, prima di entrare nel dettaglio del redesign per il porting in VR, è opportuno descrivere la configurazione hardware necessaria a garantirne la corretta esecuzione e comprendere le basi tecnologiche su cui si fondano resa grafica e interattività.

Il **visore Meta Quest** rappresenta il dispositivo principale per l'interazione con l'ambiente virtuale, svolgendo una duplice funzione: da un lato, attraverso i suoi sensori e controller, acquisisce i movimenti e i comandi dell'utente, fungendo quindi da dispositivo di input. Dall'altro, riproduce in tempo reale le immagini generate dalla GPU, assumendo di fatto il ruolo di monitor dedicato alla realtà virtuale.

Alla base del funzionamento del sistema vi è la **CPU**, responsabile della gestione della logica di gioco, dell'elaborazione degli input, delle simulazioni fisiche e della preparazione dei dati che descrivono la scena virtuale. È la CPU che organizza e coordina l'esecuzione delle istruzioni, inviando alla scheda grafica (GPU) i comandi necessari per la rappresentazione visiva.

La **GPU** si occupa della parte computazionalmente più intensa, ovvero del rendering della scena tridimensionale. Attraverso una pipeline altamente parallela, la GPU trasforma i dati ricevuti dalla CPU in immagini, applicando trasformazioni geometriche, calcoli di illuminazione e effetti visivi complessi. Per svolgere questo compito, utilizza una memoria dedicata ad alta velocità, detta VRAM, in cui vengono temporaneamente immagazzinati dati fondamentali come texture e shader. Una porzione particolarmente importante della VRAM è il framebuffer, ovvero un contenitore di immagini pronte per essere lette dal visore.

La Figura 6.1 illustra i componenti appena descritti e la loro interazione.

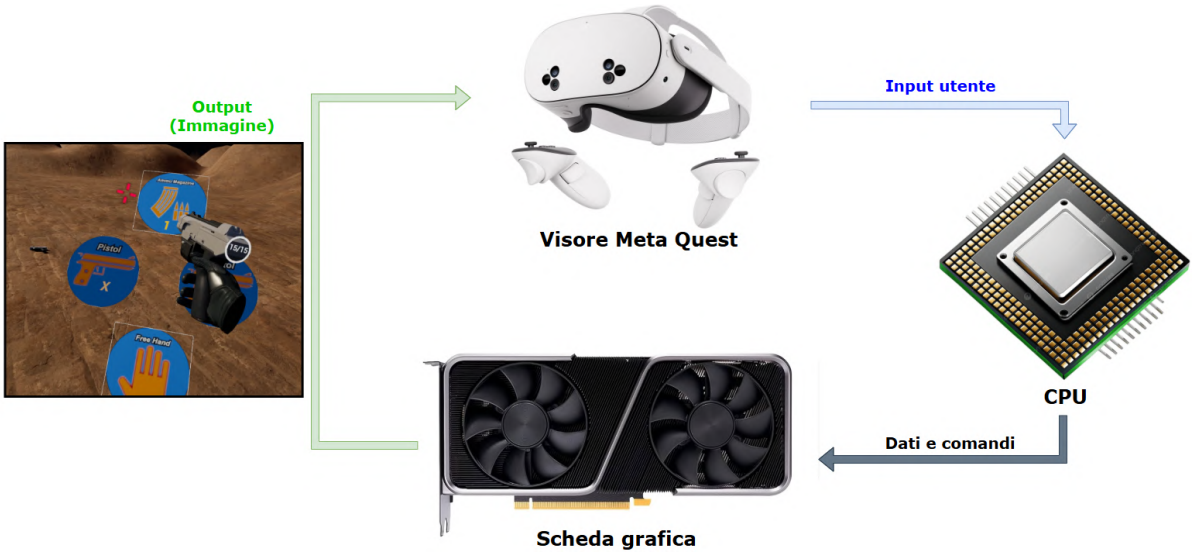

Figura 6.1: Configurazione hardware per esperienza PC-VR

6.1.1 Ambiente di sviluppo e vincoli prestazionali

Per lo sviluppo e il testing dell'applicazione è stato utilizzato un laptop con le seguenti specifiche:

- **Scheda grafica**: GPU NVIDIA GeForce RTX 3070 con 8 GB di memoria dedicata.
- **RAM**: 16 GB di RAM DDR4 a 3200 MHz
- **CPU**: Intel Core i7 di undicesima generazione a 2.30 GHz

Per quanto riguarda il visore, è stato utilizzato un Meta Quest 3S così caratterizzato:

- **Risoluzione**: 1832 x 1920 pixel per occhio, 773 PPI (pixel per inch), 20 PPD (pixel per degree)
- **Frequenza di aggiornamento**: 72 Hz, 90 Hz, 120 Hz
- **RAM**: 8 GB
- **CPU**: Snapdragon XR2 Gen 2

Questa configurazione ha garantito una potenza computazionale adeguata sia alla gestione della logica di gioco sia al rendering in tempo reale richiesto dall'esperienza VR. Tuttavia, l'esperienza immersiva impone requisiti stringenti in termini di stabilità del framerate e latenza, poiché eventuali cali prestazionali incidono direttamente sul comfort dell'utente.
Per questo motivo, accanto allo sfruttamento delle risorse hardware disponibili, è stato necessario adottare un processo sistematico di ottimizzazione delle performance, che verrà approfondito nel Capitolo 7.

6.1.1.1 Configurazione dell'esperienza in base all'hardware disponibile

Al fine di garantire la fruibilità del gioco anche su hardware meno performante, l'applicazione prevede la possibilità di modulare la qualità grafica tramite i **Scalability Settings** di Unreal Engine [25].
Questi parametri, regolabili direttamente dal menu iniziale (Sezione 6.3.5.2), consentono di adattare componenti come ombre, effetti post-processing e qualità delle texture in funzione delle capacità hardware disponibili.

I profili predefiniti (Low, Medium, High, Epic e Cinematic) rappresentano compromessi differenti tra qualità visiva e prestazioni. Nella pratica, tuttavia, i profili Epic e Cinematic in VR vengono raramente utilizzati, poiché le esigenze di framerate elevato e bassa latenza rendono preferibili configurazioni più conservative. Per questo motivo non sono stati previsti come profili selezionabili in *Gavunia VR*.

Per dispositivi di fascia medio-bassa, in particolare privi di scheda grafica dedicata, si raccomanda l'uso del profilo Medium, che riduce significativamente il carico computazionale derivante da effetti come illuminazione globale e gestione delle ombre, mantenendo una qualità visiva accettabile, seppur con illuminazione meno realistica e perdita di profondità in alcune zone.
Per dispositivi di fascia medio-alta con scheda video dedicata, è invece consigliato il profilo High, che consente di ottenere la migliore resa possibile del gioco.

6.2 Stack software - OpenXR

Per garantire la compatibilità con diversi dispositivi, *Gavunia VR* sfrutta il supporto fornito da **OpenXR** [26], uno standard aperto e privo di royalties che definisce un insieme comune di API per lo sviluppo di applicazioni XR.

In particolare, Unreal Engine tramite OpenXR, rende disponibili componenti predefiniti come Camera Rig, MotionController e mesh delle mani, consentendo di leggere lo stato dei pulsanti, rilevare la posizione e la rotazione dei controller e manipolare oggetti nel mondo virtuale senza dover implementare driver o logiche specifiche per ogni visore.
Sul lato del rendering, OpenXR fornisce informazioni sul tracciamento della testa e sugli occhi, permettendo all'engine di generare immagini stereoscopiche corrette e sincronizzare il refresh rate del visore, riducendo latenza e artefatti visivi.
Grazie a questa integrazione, lo sviluppo in VR può concentrarsi sul design dell'interazione e sull'esperienza immersiva, mentre **OpenXR e Unreal Engine si occupano della gestione standardizzata dell'input/output dei dispositivi, garantendo compatibilità multipiattaforma e una base solida per il porting su diversi headset**.

Questa panoramica tecnica si inserisce all'interno di un più ampio contesto architetturale, illustrato in Figura 6.2.
Al centro, l'Application Interface funge da strato intermedio essenziale che connette le esperienze XR — sviluppate tramite game engine o tecnologie come WebXR — con le implementazioni hardware conformi allo standard. Questo livello di astrazione permette alle applicazioni di comunicare in modo uniforme con un ampio spettro di dispositivi prodotti da numerosi vendor, tra cui aziende leader come Acer, Canon, HTC, Meta, Microsoft, Sony e piattaforme software come SteamVR e Varjo.
L'interfaccia OpenXR non solo facilita la portabilità e la compatibilità multipiattaforma, ma crea anche un ecosistema collaborativo in cui hardware e software coesistono secondo un protocollo condiviso, semplificando l'adozione e lo sviluppo di soluzioni XR.

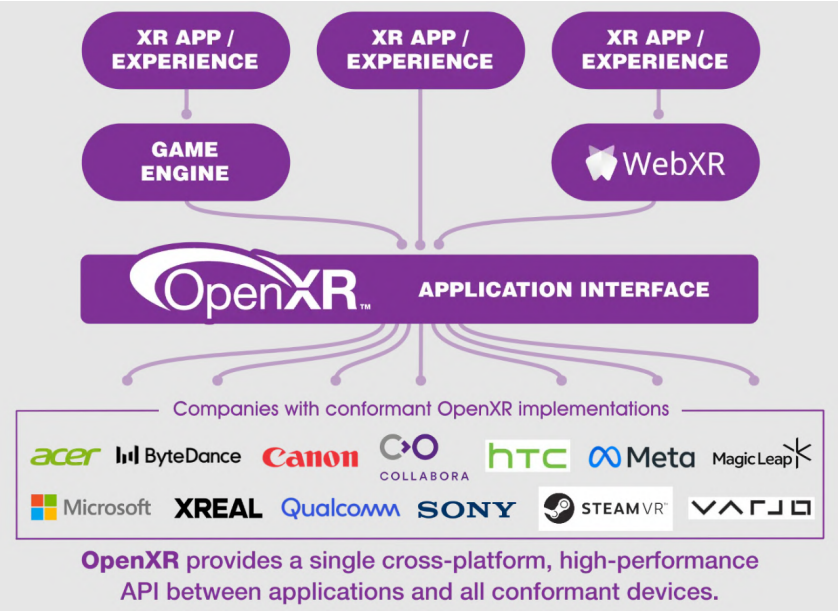

Figura 6.2: Architettura di OpenXR (tratto da Khronos Group)

6.3 Redesign della HCI

Se a livello di logica di business il porting dalla versione desktop a quella VR non ha comportato modifiche sostanziali, grazie all'adozione di un'architettura modulare basata su componenti (Sezione 5.4.4), sul piano della Human-Computer Interaction (HCI) si è resa necessaria una riprogettazione profonda. Il passaggio dalla fruizione su schermo bidimensionale alla realtà virtuale introduce infatti un cambiamento di paradigma che incide in modo diretto sulle modalità di interazione tra giocatore e sistema.
La VR offre possibilità di interazione radicalmente diverse rispetto al desktop tradizionale, consentendo un coinvolgimento diretto del corpo e l'utilizzo delle mani virtuali come principale mezzo di comunicazione con il mondo di gioco. Un semplice remapping degli input, limitato alla trasposizione dei comandi da tastiera e mouse ai motion controller, non sarebbe stato sufficiente a valorizzare tali potenzialità e avrebbe compromesso il senso di immersione e presenza. Per questo motivo, l'interazione è stata ripensata privilegiando gesti naturali e azioni dirette, con l'obiettivo di sfruttare appieno le caratteristiche espressive del medium VR.

In questa sezione vengono pertanto presentate e motivate le principali scelte di Interaction Design adottate per la versione VR del gioco, mostrando come esse abbiano influito non solo sulle modalità di interazione, ma anche, in alcuni casi, su aspetti del Game Design.
La riprogettazione della HCI rappresenta così il principale ambito di intervento nel processo di porting, configurandosi come elemento chiave per l'adattamento dell'esperienza di gioco alla realtà virtuale.

6.3.1 Meccanismi di grabbing

Il grabbing rappresenta una delle operazioni fondamentali nelle applicazioni di realtà virtuale e può essere considerato una naturale estensione del concetto di selezione (Sezione 3.1.3), nella quale l'atto di selezionare un oggetto si traduce direttamente nel suo afferramento.
In *Gavunia VR* sono stati progettati e implementati due differenti meccanismi di grabbing, pensati per rispondere a esigenze di interazione diverse: un sistema basato sull'interazione naturale e uno basato su un'interazione di tipo magico (Sezione 3.1.1).
Il primo approccio privilegia il realismo e riproduce il gesto di afferrare un oggetto solo quando questo si trova entro una distanza fisicamente raggiungibile dalla mano virtuale. Questa modalità è sempre disponibile e costituisce la forma di interazione principale con gli oggetti del mondo di gioco.
Il secondo meccanismo, invece, attivabile nel menu delle impostazioni iniziali (Sezione 6.3.5.2), introduce la possibilità di afferrare oggetti a distanza, superando i vincoli fisici dell'interazione naturale. Pur trattandosi di una soluzione meno realistica, essa risulta particolarmente efficace in contesti di gioco frenetici, come quelli tipici degli sparatutto in prima persona, dove rapidità e accessibilità possono avere la precedenza sulla simulazione fedele del gesto. Tuttavia, si è scelto di non attivare questa modalità di default. In questo modo, gli utenti che desiderano un'interazione basata esclusivamente sull'uso diretto delle mani virtuali possono evitare qualsiasi componente di grabbing a distanza, mantenendo un rapporto fisico e immediato con il mondo di gioco.

6.3.1.1 Grabbing naturale

Il grabbing naturale rappresenta l'afferramento realistico degli oggetti e si applica esclusivamente agli elementi fisicamente raggiungibili dalla mano virtuale. Ogni oggetto afferrabile dispone di un **GrabComponent**, che si occupa di gestire la logica di presa e rilascio, coerente con il principio del Component Pattern adottato per tutta la logica di gioco.

Il fulcro dell'azione di grabbing risiede nella fase di selezione, ossia nel determinare quale oggetto tra quelli presenti nella scena il giocatore stia tentando di afferrare. Tra le varie tecniche di selezione presentate nella Sezione 3.1.3.2, si è scelto di utilizzare il **ray casting**, una strategia semplice ed efficiente già supportata da Unreal Engine tramite la funzione *TraceForObject* [27]. Questo metodo permette di definire un raggio (o primitive alternative come sfera o capsula) a partire dal motion controller, fino a una distanza massima empiricamente stabilita per il grabbing naturale, e di limitare le intersezioni a specifiche categorie di oggetti. Per facilitare questa selezione, è stato creato un canale di collisione personalizzato, chiamato *InteractableObject*, che include esclusivamente gli attori afferrabili. In questo modo, il sistema di picking considera solo gli oggetti di interesse, ottimizzando le prestazioni e riducendo il rischio di interferenze con altri elementi della scena.
L'oggetto individuato dal ray casting viene poi associato visivamente alla mano del giocatore, secondo il tipo di presa definito nel GrabComponent. Maggiori dettagli sulle modalità di presa saranno discussi nella Sezione 6.3.2.

6.3.1.2 Grabbing a distanza

Il grabbing a distanza è stato introdotto come interazione "magica" per superare i limiti del realismo fisico e semplificare la manipolazione di oggetti in situazioni frenetiche, come durante uno scontro a fuoco. Questa scelta sacrifica parte del realismo, ma permette di aumentare il dinamismo dell'esperienza.
L'implementazione si ispira ai Gravity Gloves di *Half-Life: Alyx*, che consentono al giocatore di attirare verso di sé oggetti distanti e afferrarli una volta sufficientemente vicini. A differenza delle tecniche a distanza classiche (Sezione 3.1.3.2), come HOMER o la Go-Go technique, che spostano la mano verso l'oggetto o estendono un "braccio meccanico", il grabbing a distanza porta direttamente l'oggetto verso la mano del giocatore. Quando l'oggetto è abbastanza vicino, l'interazione si comporta come il grabbing naturale descritto nella sezione precedente.

A livello di implementazione, la selezione dell'oggetto viene gestita tramite ray-casting, estendendo la distanza di interazione rispetto al grabbing naturale. Tuttavia, a differenza di quest'ultimo, la selezione non coincide con l'immediato afferramento, ma rappresenta solo la prima fase. Una volta selezionato, l'oggetto riceve un feedback visivo, assumendo un overlay arancione che indica chiaramente quale elemento il giocatore ha selezionato.

La seconda fase rileva l'intenzione del giocatore di portare a sé l'oggetto, tramite un meccanismo di pull-back del braccio. Questo controllo si basa sul vettore direzione calcolato come differenza tra la posizione del Motion Controller al momento della pressione del

tasto di grab e la posizione corrente al rilascio:

$$\vec{v} = P_{\text{rilascio}} - P_{\text{pressione}} \tag{6.1}$$

Il vettore $\vec{v}$ viene poi normalizzato (in modo da avere lunghezza unitaria) per poi essere confrontato con il forward vector $\vec{F}$ della camera mediante il prodotto scalare:

$$\vec{v} \cdot \vec{F} = \|\vec{v}\| \, \|\vec{F}\| \cos \alpha \tag{6.2}$$

dove α rappresenta l'angolo tra i due vettori.
Essendo sia $\vec{v}$ sia $\vec{F}$ dei versori, il prodotto scalare diventa semplicemente $\cos \alpha$. Ne consegue che la condizione di attivazione della terza fase si verifica quando $\cos \alpha < 0$, ossia $\alpha > 90°$, indicando che il movimento del braccio è avvenuto in direzione opposta rispetto al puntamento, come mostrato in Figura 6.3.
Per evitare attivazioni involontarie dovute a piccoli movimenti del braccio, il valore del prodotto scalare deve superare una soglia definita dal parametro *PullBackThreshold*, migliorando precisione e robustezza dell'interazione. Il valore di questo parametro è stato impostato a -0.6, corrispondente ad un angolo α di circa 127°.
Questa scelta rappresenta un buon compromesso tra precisione e naturalezza dell'interazione, poiché è sufficientemente selettiva da filtrare micro-movimenti, ma permette comunque gesti naturali e rapidi, evitando di costringere il giocatore a un movimento troppo estremo per attivare l'azione.
Questo approccio, volto a ridurre gli errori umani, riprende principi simili a quelli applicati da Krompiec e Park nel caso di studio sulla ricarica dell'arma (Sezione 3.2.4).

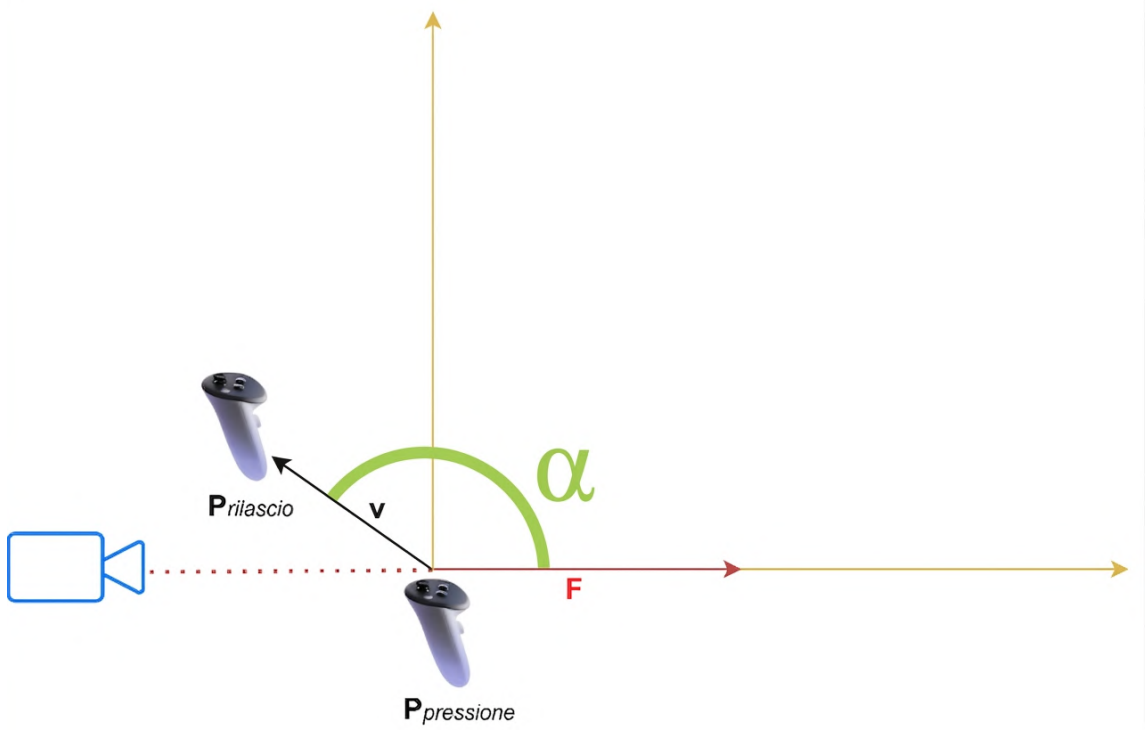

Figura 6.3: Illustrazione della verifica tramite vettori dell'azione di pullBack per grabbing a distanza

Una volta validato il movimento del controller, ha inizio la fase di richiamo dell'oggetto verso la mano del giocatore. La gravità dell'oggetto viene temporaneamente disabilitata per evitare interferenze con la simulazione della traiettoria.
Il movimento viene gestito tramite una Timeline della durata di un secondo, che genera un parametro $\beta \in [0, 1]$ rappresentante la progressione temporale dell'animazione. Le componenti X e Y della posizione dell'oggetto vengono interpolate linearmente tramite una funzione Lerp tra la posizione iniziale dell'oggetto e quella target della mano del giocatore:

$$PosOggetto_{XY}(\beta) = \text{Lerp}(P_{\text{start}}, P_{\text{target}}, \beta) \tag{6.3}$$

mentre la componente Z è calcolata separatamente per simulare una traiettoria parabolica. In particolare, l'offset verticale è definito come:

$$\text{Offset}_Z = \text{ArcHeight} \cdot \sin(\beta \cdot \pi) \tag{6.4}$$

dove *ArcHeight* corrisponde all'altezza massima dell'arco della traiettoria. L'offset verticale viene sommato alla componente Z interpolata, in modo che l'oggetto raggiunga l'elevazione massima a metà della Timeline e ricada progressivamente verso la mano del giocatore. Al termine della Timeline, la gravità viene riattivata, consentendo all'oggetto di essere correttamente afferrato.

La Figura 6.4 mostra il movimento parabolico dell'oggetto dalla sua posizione iniziale alla mano del giocatore.

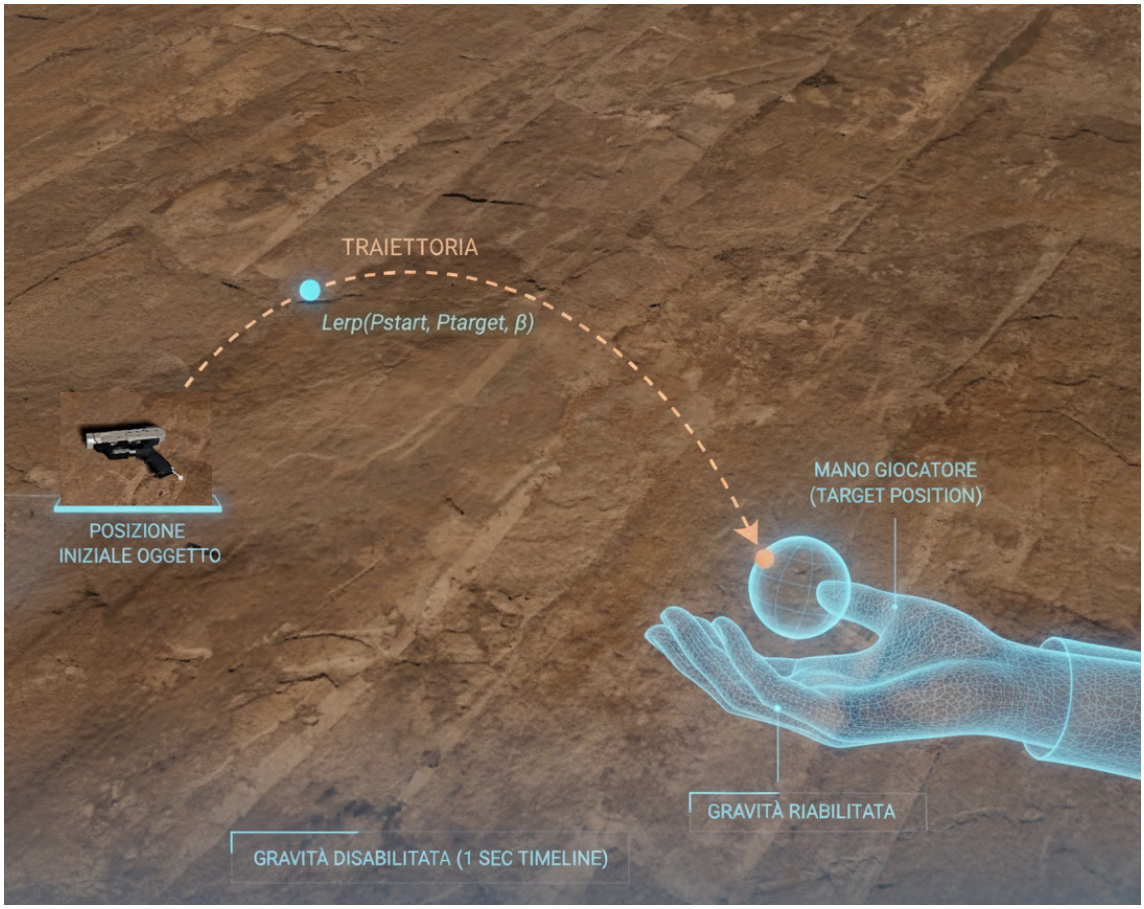

Figura 6.4: Illustrazione del movimento dell'oggetto nel grabbing a distanza

6.3.1.3 Parentesi sul sistema di tracing in Unreal Engine

Come descritto precedentemente, entrambe le tecniche di grabbing si basano sulla selezione degli oggetti tramite la funzione *TraceForObject*, che sfrutta il sistema di Traces nativo di Unreal Engine. Questo sistema realizza query di tipo fisico basate sulle collisioni, consentendo di verificare l'intersezione tra una primitiva (linea, sfera, capsula) e gli oggetti presenti nella scena.

Affinché un attore possa essere rilevato da una trace, è necessario che disponga di una o più primitive di collisione associate. In Unreal Engine, le collisioni possono essere configurate secondo due modalità principali: collisioni semplici e collisioni complesse.
Le **collisioni semplici** si basano su primitive geometriche elementari (ad esempio box, sfere o capsule) che approssimano il volume dell'oggetto. Questa modalità, utilizzata di default ma modificabile nelle impostazioni di progetto, presenta un costo computazionale ridotto: il calcolo dell'intersezione con primitive matematiche è infatti molto più efficiente rispetto alla valutazione diretta sulle geometrie dettagliate della mesh.
Le **collisioni complesse**, invece, utilizzano direttamente la geometria della static mesh, valutando l'intersezione sulle singole facce triangolari che la compongono. Questo approccio garantisce maggiore precisione, ma comporta un costo computazionale significativamente superiore, risultando meno adatto a operazioni frequenti come le trace eseguite ad ogni frame, in particolare nel contesto VR.

Nel progetto sviluppato, si è scelto di utilizzare collisioni semplici opportunamente configurate per gli oggetti interagibili, in modo da mantenere elevate le prestazioni e garantire la stabilità del frame rate, requisito cruciale nelle applicazioni di realtà virtuale.

La Figura 6.5 illustra la differenza tra collisioni semplici e complesse, evidenziando come, già per una mesh relativamente semplice (4K facce), l'uso delle simple collision riduca significativamente il carico computazionale, pur mantenendo una buona precisione attraverso una configurazione accurata delle primitive di collisione.

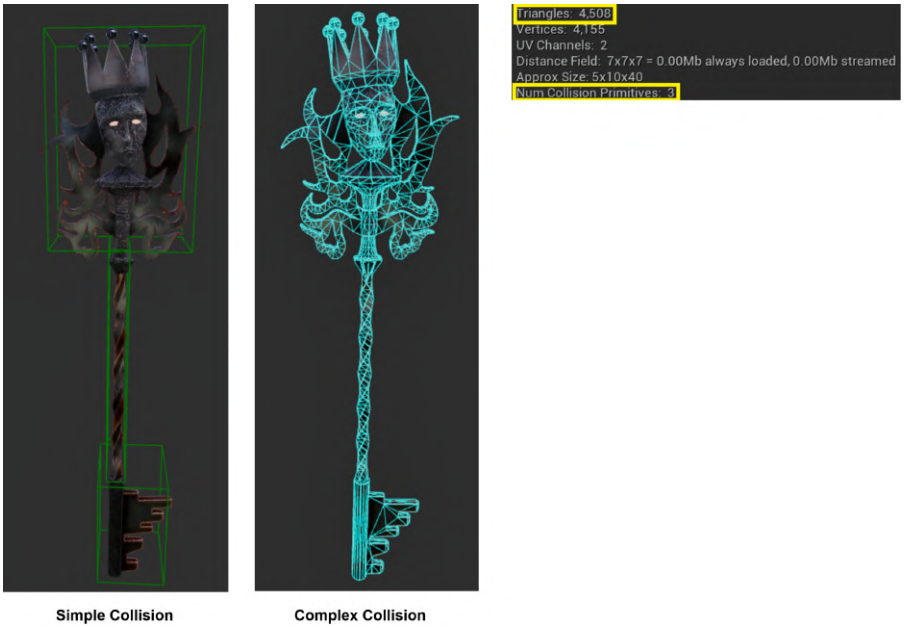

Figura 6.5: Confronto tra i tipi di collisioni in Unreal Engine 5

6.3.2 Tipologie di prese

Le modalità di presa riguardano la fase finale del processo di grabbing, ossia il modo in cui la mano virtuale posiziona le dita sull'oggetto.
Nel prototipo iniziale era stata definita un'animazione standard unica, applicata a tutti gli oggetti. Tuttavia, già dai primi test si evidenziava una perdita significativa di realismo, poiché la posa fissa non si adattava alla forma e alle caratteristiche dei diversi oggetti.
Per migliorare l'interazione, sono state quindi implementate due tecniche di grabbing complementari, selezionate dinamicamente in base al tipo di oggetto tramite il parametro *grabType* del Grabbing Component:

- **Grabbing procedurale** (Sezione 6.3.2.1): applicato agli oggetti con grabType = Free, consente alla mano virtuale di adattare le dita in maniera dinamica a oggetti manipolabili di forma variabile, garantendo flessibilità e naturalezza nella presa.

- **Grabbing basato su pose specifiche** (Sezione 6.3.2.2): applicato agli oggetti con grabType = InteractableObject, prevede un'animazione predefinita ottimizzata per ciascun tipo di oggetto legato al gameplay, garantendo precisione e realismo per gli oggetti chiave.

6.3.2.1 Grabbing procedurale

Il grabbing procedurale [28] si basa sulla generazione in tempo reale della configurazione della mano virtuale, superando il paradigma delle animazioni preimpostate. La postura finale non è quindi determinata da una sequenza registrata, bensì calcolata dinamicamente sulla base delle informazioni geometriche dell'oggetto target.
Nel dettaglio, la posizione e la flessione di ciascun segmento vengono determinate mediante un algoritmo che valuta la distanza e l'eventuale intersezione tra le falangi e la superficie dell'oggetto. Tale processo consente di modulare i pesi di rotazione delle articolazioni in funzione della geometria locale, garantendo un adattamento progressivo e coerente durante la fase di chiusura della mano.
La modalità procedurale, associata nel sistema agli oggetti classificati con *grabType = Free*, risulta **particolarmente adatta a elementi a forma libera e privi di vincoli di orientamento specifici**. Essa consente l'afferramento da angolazioni differenti e la rotazione libera nello spazio, mantenendo una configurazione delle dita coerente con la geometria dell'oggetto durante l'interazione.

A titolo esemplificativo, si consideri l'interazione con un oggetto sferico, come mostrato in Figura 6.6:

- **Presa standard fissa**: l'animazione predefinita prevede una configurazione generica (presa tra pollice e indice) che non tiene conto delle dimensioni specifiche della sfera. La chiusura delle dita segue una traiettoria rigida e predeterminata, con possibili compenetrazioni o disallineamenti rispetto alla superficie curva dell'oggetto.

- **Presa procedurale**: il sistema procedurale calcola in tempo reale la rotazione di ciascun segmento digitale, adattando progressivamente la curvatura delle dita alla superficie sferica.

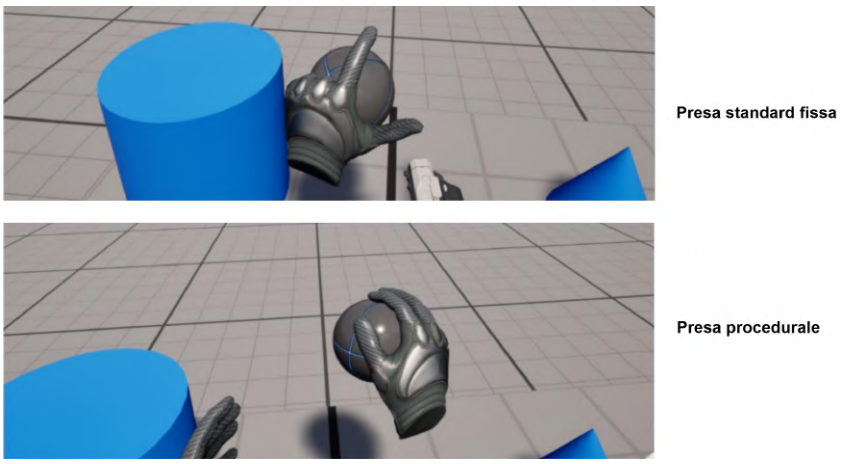

Figura 6.6: Confronto tra presa fissa e presa procedurale

Per configurare la presa procedurale, a ciascun dito è stata associata una spline che rappresenta la traiettoria naturale di chiusura durante la presa.
Per determinare l'influenza di ciascun dito sull'animazione finale viene calcolato un **peso proporzionale** basato sull'intersezione tra la spline del dito e la superficie dell'oggetto da afferrare. In particolare, per ciascun dito si valuta la porzione della spline che effettivamente entra in contatto con l'oggetto e si confronta con la lunghezza totale della spline. Il peso viene quindi determinato come il rapporto tra la lunghezza della parte in intersezione e la lunghezza complessiva della spline: se gran parte della spline interseca l'oggetto, il peso risulta elevato (vicino a 1), indicando che il dito deve chiudersi completamente nella presa. Viceversa, se solo una piccola porzione della spline entra in contatto con l'oggetto, il peso è basso (vicino a 0), e il dito rimane in posizione più rilassata o distesa.
Il peso calcolato per ciascun dito viene poi utilizzato dal sistema di animazione per modulare la chiusura delle falangi e determinare la posa finale della mano.

La Figura 6.7 illustra il procedimento di calcolo dei pesi. La porzione evidenziata in rosso indica il tratto di spline che interseca l'oggetto; nel caso analizzato tale porzione costituisce la maggior parte della spline, determinando un peso prossimo a 1 e, di conseguenza, la completa chiusura del dito nella presa.

Figura 6.7: Illustrazione del calcolo dei pesi

6.3.2.2 Grabbing basato su pose specifiche

Il grabbing basato su pose specifiche, associato nel sistema agli oggetti classificati con *grabType = InteractableObject*, è utilizzato per gli oggetti funzionali del gameplay, come pozioni, chiavi, cristalli o armi, che **richiedono un orientamento definito una volta afferrati**. Sebbene tale orientamento possa essere ottenuto anche tramite il grabbing procedurale, quest'ultimo richiede un intervento attivo da parte dell'utente: se l'oggetto viene afferrato da un orientamento non ottimale, l'utente deve ripetere l'azione di presa per ottenere la posizione desiderata. Questo livello di realismo è interessante per oggetti a forma libera, ma risulta meno appropriato per oggetti funzionali al gameplay.
Per ridurre possibili frustrazioni e facilitare l'esperienza utente, si è scelto di definire pose specifiche per questi oggetti. In questo modo, le dita vengono automaticamente allineate e l'oggetto viene ruotato nella posizione ottimale per il suo utilizzo.
La Figura 6.8 mostra una serie di esempi di queste pose fisse per gli oggetti.

A livello implementativo, nella skeletal mesh della mano virtuale sono state definite delle **sockets** [29], ossia punti di ancoraggio collegati alle ossa della mesh, ai quali è possibile attaccare componenti o altre mesh in modo che seguano correttamente il movimento dell'animazione. Ciascuna socket può essere configurata, modificandone posizione, rotazione e scala, per garantire un allineamento preciso dell'oggetto rispetto alla mano.
In particolare, è stata definita una socket distinta per ciascun oggetto funzionale, in modo da ancorarlo nella posizione e rotazione ottimali. Parallelamente, sono state impostate le pose fisse della mano, ovvero le rotazioni delle articolazioni necessarie a posizionare correttamente le dita sulla mesh dell'oggetto.

A differenza del grabbing procedurale, in cui la configurazione della mano viene calcolata dinamicamente in base alla geometria e all'intersezione con l'oggetto, in questo caso la posa è predefinita e immutabile. Questo approccio è stato necessario per garantire che gli oggetti chiave del gameplay fossero immediatamente utilizzabili nella posizione corretta, evitando il rischio che il giocatore dovesse ri-orientare manualmente l'oggetto. Di conseguenza, non è stato possibile combinare grab procedurale e pose specifiche, poiché il blending dinamico con un oggetto rigidamente orientato avrebbe potuto produrre posizioni incoerenti della mano e compromettere la funzionalità immediata richiesta dal gameplay.

Figura 6.8: Esempi di grabbing basato su pose specifiche per una serie di oggetti

6.3.3 Fisica delle mani

L'introduzione del grabbing procedurale e del grabbing basato su pose specifiche ha migliorato significativamente il realismo delle animazioni delle mani. Tuttavia, entrambe le modalità agivano esclusivamente a livello cinematico, senza integrare una simulazione fisica delle interazioni. Di conseguenza, le mani e gli oggetti potevano penetrare altri elementi della scena, compromettendo la coerenza fisica e l'immersività dell'esperienza.
Per ovviare a questo limite, è stato implementato un sistema che distingue la mano logica, associata al Motion Controller e indicata come **HandTracking**, dalla mano fisica, indicata come **Hand** [30].
La connessione tra la mano logica e quella fisica è realizzata tramite un vincolo fisico (**Physics Constraint** [31]), che limita il movimento lineare e rotazionale della mano fisica rispetto alla mano logica. Questo vincolo consente alla mano fisica di propagare correttamente le forze esterne e di seguire con precisione la posizione e la rotazione della mano logica attraverso motori lineari e angolari, garantendo una simulazione stabile e naturale. In questo modo, la mano logica continua a guidare il tracking e le interazioni, mentre la mano fisica impedisce la penetrazione degli oggetti e mantiene coerente la simulazione fisica. La mesh visibile e animata è associata alla mano fisica, che si adatta alle posizioni e rotazioni della mano logica tramite il vincolo, mentre la mano logica gestisce esclusivamente il tracking e le query, senza essere direttamente renderizzata.

6.3.3.1 Definizione dei Physics Asset e Constraint per le mani

Per consentire alle mani fisiche di interagire in modo stabile e credibile con l'ambiente virtuale, è stato configurato un **Physics Asset** [32] dedicato associato alla skeletal mesh della mano. In particolare, sono stati definiti corpi di collisione distinti per il palmo e per ciascun dito, allineati alle rispettive ossa dello scheletro, come mostrato in Figura 6.9.
Questa suddivisione consente interazioni più precise rispetto a una singola collisione globale, permettendo alle dita di entrare in contatto con gli oggetti in modo indipendente e coerente con la posa assunta durante l'animazione.

Figura 6.9: Physics Asset della mano. In viola i corpi di collisione dedicati per ciascun dito e il palmo

Per garantire un comportamento articolare plausibile, sono stati inoltre definiti Physics Constraint specifici per le falangi. Tali vincoli sono stati configurati con l'obiettivo di limitare i gradi di libertà in modo coerente con la biomeccanica reale delle dita, prevenendo rotazioni o configurazioni non realistiche durante la simulazione.

Dal punto di vista lineare, tutti i gradi di libertà traslazionali lungo gli assi X, Y e Z sono stati completamente bloccati per le articolazioni delle falangi intermedie e distali. Questa scelta garantisce che la relazione tra un osso e il successivo rimanga puramente rotazionale, evitando separazioni o micro-spostamenti numericamente instabili, che non sarebbero fisicamente plausibili.

Per quanto riguarda la componente angolare, nelle stesse articolazioni sono stati applicati vincoli differenziati: le oscillazioni laterali (Swing 1 e Swing 2) sono completamente bloccate, mentre la torsione lungo l'asse longitudinale del dito (Twist) è limitata a un intervallo ridotto. Questa configurazione riflette accuratamente il comportamento biomeccanico delle falangi intermedie e distali, che consentono prevalentemente movimenti di flessione ed estensione, con torsione minima e trascurabile libertà laterale.

Diversamente, per le articolazioni delle falangi prossimali è stata mantenuta una maggiore libertà di movimento, lasciando l'asse Swing 2 con un'apertura di 30°, per consentire una rotazione laterale più ampia in linea con la biomeccanica naturale della mano.

La Figura 6.10 illustra i vincoli fisici impostati che impediscono movimenti o rotazioni non naturali delle dita.

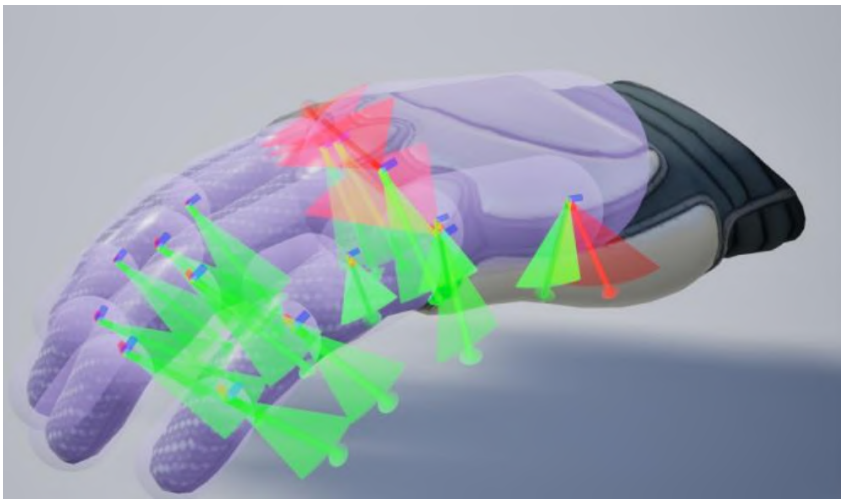

Figura 6.10: Vincoli fisici configurati per le dita. In verde la torsione assiale (Twist) permessa per le falangi intermedie

6.3.4 Interazione con gli oggetti

La riprogettazione dell'interazione con gli oggetti ha rappresentato la parte più sostanziale dello sviluppo.
Nella versione desktop, le interazioni sono generalmente limitate alla pressione di un tasto (ad esempio, premere "E" per aprire una porta). Nel passaggio al VR, invece, è stato necessario implementare meccanismi di interazione specifici per ciascun oggetto, in modo da sfruttare appieno le potenzialità del nuovo sistema.
La seguente sezione mostrerà quindi come sono state ridefinite le interazioni dei singoli oggetti, evidenziando come il redesign del HCI abbia richiesto modifiche ai layers di Game Logic, Modelling e Animation.

6.3.4.1 Strategie di riconoscimento robusto delle interazioni

Prima di procedere con l'analisi delle singole interazioni implementate per gli oggetti di gioco, è opportuno descrivere le due principali strategie progettuali adottate al fine di garantire robustezza rispetto all'errore umano.

La prima strategia si fonda sul medesimo principio geometrico introdotto per il grabbing a distanza (Sezione 6.3.1.2), **basato sull'analisi del prodotto scalare tra versori**. Tale approccio consente di valutare l'orientamento relativo tra due entità nello spazio tridimensionale attraverso il coseno dell'angolo compreso tra esse. Definendo una soglia angolare di accettazione, è possibile introdurre una tolleranza controllata che permette di riconoscere come valida un'interazione anche in presenza di lievi scostamenti rispetto alla configurazione ideale.
Questo criterio è stato applicato nelle interazioni che richiedono una specifica configurazione angolare tra oggetto e direzione di vista o tra oggetto e superficie di destinazione. In particolare, è stato impiegato per verificare la corretta rotazione della chiave nella serratura delle porte e riconoscere l'inclinazione della pozione verso la camera del giocatore durante il gesto di bevuta.

La seconda strategia adottata si basa invece su un criterio di prossimità spaziale tra l'oggetto manipolato e il punto di interazione previsto. In questo caso, l'attivazione dell'effetto dipende dal superamento di una soglia di distanza predefinita, garantendo una gestione semplice ma efficace delle interazioni che non richiedono vincoli di orientamento stringenti. Tale approccio è stato utilizzato per i muri magici, le statue vitali, le casse di munizioni e per la procedura di ricarica dell'arma.

L'integrazione di soglie angolari e soglie di distanza non risponde esclusivamente a esigenze implementative, ma costituisce una precisa scelta di game design volta a rendere le interazioni più resilienti, intuitive e coerenti con le aspettative percettive del giocatore, riducendo la frustrazione e mantenendo elevato il senso di controllo all'interno dell'esperienza immersiva.

6.3.4.2 Apertura delle porte

Nella versione desktop, l'apertura delle porte avviene tramite un'interazione contestuale basata sulla prossimità: quando il giocatore si avvicina a una porta e possiede una chiave nell'inventario, è sufficiente premere il tasto "E" per consumarne una e attivare il meccanismo di apertura. L'animazione di apertura è gestita tramite una Timeline, che modifica la posizione delle ante per simulare lo scorrimento laterale e consentire il passaggio. Pur introducendo un minimo di realismo visivo, questa interazione non mostra effettivamente l'utilizzo della chiave come parte integrante del processo di apertura.

Nella versione VR si è scelto di rendere l'interazione più immersiva e concreta intervenendo sulla mesh della porta, alla quale sono stati aggiunti una fessura per l'inserimento della chiave e un pulsante dedicato all'apertura.
L'inserimento della chiave genera un feedback audio immediato, confermando al giocatore l'azione corretta. Per sbloccare la porta è necessario ruotare la chiave, riproducendo il gesto tipico della vita reale. Al completamento di questa azione, la chiave scompare dalla scena, indicando che la porta è stata sbloccata. Da quel momento, la porta può essere aperta premendo il pulsante, che cambia colore in verde durante l'animazione, fornendo un ulteriore feedback visivo.
La Figura 6.11 riassume i passaggi appena raccontati per interagire con le porte.

Dal punto di vista implementativo, l'inserimento della chiave è determinato tramite il criterio di prossimità spaziale rispetto al punto di interazione nella serratura. La validazione della rotazione, invece, si fonda sulla soglia angolare calcolata mediante l'analisi del prodotto scalare tra il versore associato all'asse della chiave e quello della capsula nella serratura. La soglia è stata fissata per imporre sia un'elevata coerenza direzionale sia la concordanza del verso di rotazione. In questo modo l'interazione viene riconosciuta esclusivamente quando la chiave risulta orientata nello stesso verso dell'asse previsto, escludendo configurazioni speculari o rotazioni inverse.

Questa soluzione risulta coerente con le Lenses di controllo e interfaccia (Sezione 2.1.7), in particolare con la Lens of Physical Interface, poiché il mapping tra il gesto fisico — inserimento della chiave, rotazione e pressione del pulsante — e l'azione risultante appare naturale e congruente rispetto all'obiettivo dell'interazione.

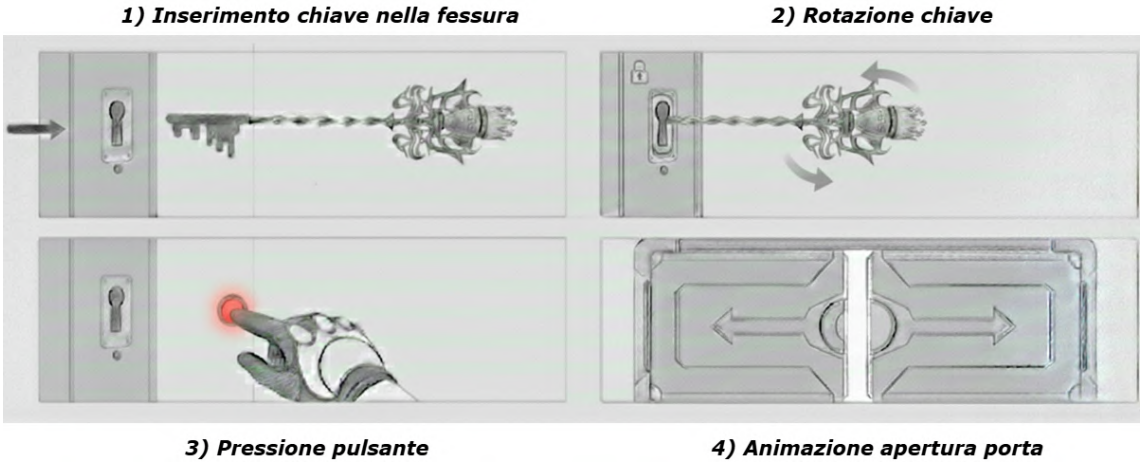

Figura 6.11: Illustrazione delle fasi nell'interazione con le porte

6.3.4.3 Muri magici

Nella versione desktop, l'apertura è subordinata alla presenza nell'inventario del giocatore dei cristalli richiesti. Al verificarsi della condizione, il muro si abbassa automaticamente, liberando il passaggio. L'interazione mantiene un minimo di realismo visivo attraverso un'animazione contestuale, ma i cristalli rimangono elementi puramente simbolici: il giocatore non li utilizza direttamente nello spazio di gioco, bensì come semplice requisito logico verificato dal sistema.

Nella versione VR, l'interazione è stata riprogettata per sfruttare le potenzialità dell'ambiente immersivo. La mesh del muro è stata modificata introducendo un cilindro dedicato per ciascun cristallo richiesto. L'apertura non dipende più da una verifica astratta dell'inventario, bensì da un'azione fisica esplicita: il giocatore deve afferrare il cristallo e posizionarlo nell'alloggiamento corrispondente, basandosi sul colore. Ad esempio, il cristallo blu deve essere appoggiato sopra il cilindro blu.
La Figura 6.12 riassume i passaggi appena raccontati per interagire con i muri magici.

Dal punto di vista implementativo, l'interazione tra cristallo e cilindro si basa sul criterio di prossimità spaziale: quando un cristallo è sufficientemente vicino al punto di interazione di un cilindro, si attiva la logica di verifica della corrispondenza sul colore. In questo caso non è necessaria la strategia della soglia angolare, poiché non è richiesta una rotazione specifica del cristallo rispetto al cilindro.

Per quanto riguarda l'aspetto progettuale, questo cambiamento segna il passaggio da un'interazione di tipo simbolico, mediata dall'interfaccia, a un'interazione spaziale. Il requisito non è più soltanto "possedere" l'oggetto, ma utilizzarlo attivamente nello spazio tridimensionale. In questo modo, il cristallo assume una funzione diegetica, diventando parte integrante del mondo di gioco e contribuendo ad aumentare il senso di presenza e coinvolgimento del giocatore.

Figura 6.12: Illustrazione delle fasi nell'interazione con i muri magici

6.3.4.4 Casse di munizioni

Nella versione desktop, l'interazione con le casse di munizioni è estremamente semplificata: alla pressione del tasto "E" viene aggiunta all'inventario una quantità variabile di proiettili. Il feedback fornito al giocatore è esclusivamente di tipo testuale, tramite un messaggio temporaneo visualizzato nella viewport che indica il numero di munizioni ottenute. L'azione si configura quindi come un'interazione astratta e mediata dall'interfaccia, priva di una rappresentazione fisica dell'atto di recupero.

Nella versione VR, l'interazione è stata riprogettata per introdurre un maggiore grado di coerenza spaziale e realismo percettivo. La cassa è stata modificata a livello strutturale separando la base dal coperchio, così da poter animare l'apertura di quest'ultimo. L'attivazione avviene quando il giocatore avvicina entrambe le mani ai meccanismi di apertura posizionati lateralmente: il coperchio ruota verso l'alto, simulando l'effettiva apertura. Le munizioni, collocate fisicamente all'interno della cassa, devono essere recuperate tramite afferramento diretto, trasformando l'azione in un gesto esplorativo e aumentando il senso di coinvolgimento rispetto alla controparte desktop.
La Figura 6.13 riassume i passaggi appena raccontati per interagire con le casse di munizioni.

A livello implementativo, l'apertura delle casse si basa sul criterio di prossimità spaziale: quando entrambe le mani si trovano sufficientemente vicine ai punti di interazione sui pulsanti laterali, si attiva l'animazione di apertura tramite Timeline, che modifica la rotazione del coperchio. Non è necessaria la strategia della soglia angolare, poiché non è richiesta una rotazione specifica delle mani rispetto ai pulsanti.

Dal punto di vista del game design, come motivato nella Sezione 6.3.4.6, i proiettili singoli sono stati sostituiti da caricatori completi. Tale scelta ha comportato una riprogettazione del sistema di distribuzione delle risorse. Per mantenere un bilanciamento analogo alla versione desktop, si è passati da un massimo di dieci proiettili per cassa a un massimo di due caricatori, riducendo contestualmente il numero complessivo di casse presenti nell'ambiente di gioco. Questa proporzione è motivata dal fatto che un caricatore completo contiene quindici proiettili, garantendo così una quantità di munizioni potenzialmente superiore per singolo ritrovamento ma meno frequente nello spazio di gioco.

Figura 6.13: Illustrazione delle fasi nell'interazione con le casse di munizioni

6.3.4.5 Statue vitali

Nella versione desktop, l'interazione con le statue vitali è attivata mediante la pressione prolungata del tasto "E" per alcuni secondi. Al termine dell'azione, la salute del giocatore viene ripristinata di cinquanta punti in maniera istantanea. Il feedback visivo è minimale: la statua scompare una volta utilizzata e l'incremento dei punti vita è visibile esclusivamente attraverso l'aggiornamento della HUD. L'azione si configura quindi come un evento discreto, mediato dall'interfaccia e privo di una rappresentazione progressiva del processo di cura.

Nella versione VR, l'interazione è stata riprogettata per enfatizzare la dimensione percettiva e la continuità dell'azione. Il giocatore deve appoggiare una mano sulla statua per attivare il processo di guarigione; durante tale fase viene riprodotto un effetto speciale di colore verde che avvolge l'oggetto, suggerendo visivamente l'energia curativa in atto. La cura non avviene più in modo istantaneo, ma come flusso progressivo: nell'arco di cinque secondi il giocatore recupera dieci punti vita al secondo, per un totale di cinquanta punti. La Figura 6.14 riassume i passaggi appena raccontati per interagire con le statue vitali.

A livello implementativo, l'interazione con le statue si basa sul criterio di prossimità spaziale. Quando una o entrambe le mani del giocatore si trovano sufficientemente vicine alla mesh della statua, viene attivato il sistema particellare responsabile dell'effetto visivo di cura e del contestuale ripristino dei punti vita. Non è necessaria l'applicazione della strategia della soglia angolare, poiché l'interazione non richiede una rotazione specifica delle mani rispetto alla statua.

Per quanto riguarda l'aspetto progettuale, questa modifica introduce una differenza significativa dal punto di vista dell'esperienza: l'atto di guarigione non è più un semplice comando eseguito e risolto dal sistema, ma un processo temporale percepibile, che richiede immobilità e attenzione da parte del giocatore. Ciò contribuisce a rafforzare il senso di presenza e a rendere l'azione maggiormente integrata nel contesto VR.

Figura 6.14: Illustrazione delle fasi nell'interazione con le statue vitali

6.3.4.6 Gestione delle munizioni e ricarica dell'arma

Nella versione desktop, il sistema di gestione delle munizioni e della ricarica delle armi è piuttosto semplice: il giocatore raccoglie le munizioni premendo il tasto "E" e ricarica l'arma utilizzando il "middle button" del mouse. In questa modalità, le munizioni vengono automaticamente consumate dall'inventario e l'animazione di ricarica del fucile viene riprodotta. Sebbene questo approccio introduca un minimo realismo visivo, le munizioni non sono considerate come oggetti di scena interattivi.

Nella versione VR, per aumentare l'immersione, è stato introdotto un oggetto rappresentativo del caricatore, consentendo di simulare l'azione di ricarica reale, cioè lo sgancio del caricatore esaurito e la sua sostituzione con uno nuovo. In una prima fase si era progettato un sistema ancora più realistico, in cui il giocatore poteva inserire i proiettili uno a uno nel caricatore. Tuttavia, i test hanno evidenziato che tale interazione risultava frustrante e ripetitiva, soprattutto con l'aumentare delle munizioni reperibili nelle casse o rilasciate dai nemici. Questo riscontro ha richiamato la Lense of Essential Experience di Schell (Sezione 2.1.1), che sottolinea come non sia opportuno introdurre dettagli realistici che non apportano un valore significativo all'esperienza dell'utente.
Di conseguenza, **si è deciso di rimuovere la gestione manuale dei proiettili, mantenendo soltanto i caricatori come unità di ricarica**. I caricatori continuano a rappresentare contenitori di proiettili, ma si trovano già completi nelle casse e non richiedono più un caricamento manuale.
La Figura 6.15 riassume i passaggi appena raccontati per ricaricare l'arma.

A livello implementativo, il meccanismo di ricarica si basa sul criterio di prossimità spaziale: quando il caricatore è sufficientemente vicino al punto di interazione sul calcio dell'arma, viene associato alla pistola e rimosso dalla scena. Non è necessaria la strategia della soglia angolare, poiché la presa specifica (Sezione 6.3.2.2) del caricatore nella mano garantisce l'orientamento corretto nell'inserimento.
Lo sgancio del caricatore, invece, è gestito tramite lo spawn di un nuovo oggetto nel mondo: il caricatore parte dalla socket del calcio e, grazie alla gravità simulata, cade verso il basso, riproducendo l'azione classica delle pistole senza ulteriori controlli di orientamento.

Passaggio da fucile a pistola: Nel processo di progettazione delle armi per il giocatore, si è deciso di sostituire temporaneamente il fucile con una pistola. Questa scelta è stata motivata principalmente dalla necessità di rendere le interazioni più semplici e naturali in VR: la pistola può essere impugnata e utilizzata con una singola mano, garantendo un controllo immediato e intuitivo. Inoltre, la configurazione della pistola consente al giocatore di sparare con entrambe le mani, offrendo flessibilità nell'impiego delle armi e aumentando l'immersività.

Il fucile non è stato escluso definitivamente: in una versione futura del gioco è previsto il suo reinserimento come arma a due mani, coerente con il comportamento realistico di impugnatura e sparo. La decisione attuale di utilizzare la pistola rappresenta quindi un compromesso tra semplicità d'uso e immersione, permettendo al giocatore di familiarizzare con le meccaniche di sparo in VR senza introdurre complessità aggiuntive.

Questo caso d'uso dimostra come una riprogettazione orientata alla HCI possa condurre a un game redesign significativo, integrando modifiche come la sostituzione del fucile con la pistola e la semplificazione della gestione delle munizioni da proiettili singoli a caricatori completi, evidenziando la stretta interdipendenza tra interazione utente e meccaniche di gioco.

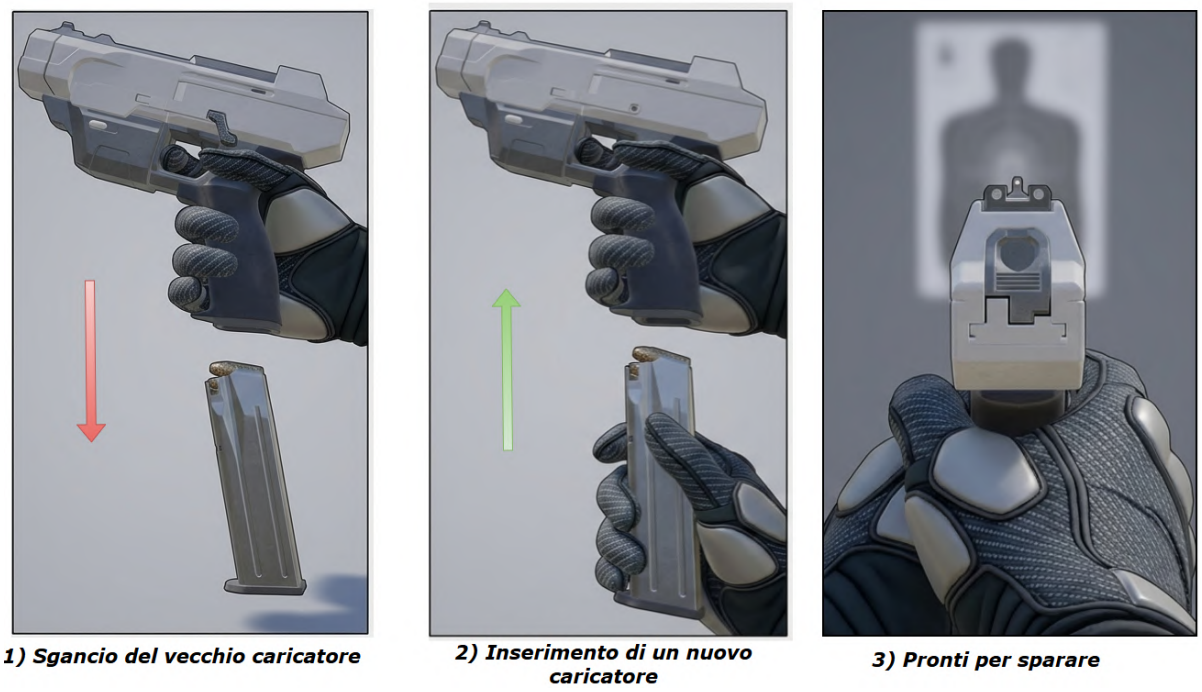

Figura 6.15: Illustrazione delle fasi nell'interazione per la ricarica dell'arma

6.3.4.7 Pozioni

Nella versione desktop, l'assunzione della pozione è attivata mediante la pressione del tasto "P". L'effetto avviene immediatamente e l'unico riscontro fornito al giocatore è l'aggiornamento dei valori nella HUD. Non viene riprodotta alcuna animazione né rappresentazione visiva dell'atto di bere, configurando l'azione come un comando puramente funzionale.

Nella versione VR, l'interazione è stata riprogettata con l'obiettivo di aumentare il realismo e la coerenza gestuale dell'azione. Il giocatore deve rimuovere il tappo della pozione e portare fisicamente la bottiglia alla bocca per attivarne l'effetto. A tal fine, la mesh dell'oggetto è stata modificata separando il corpo della bottiglia dal tappo, così da consentire un'interazione indipendente tra le due parti.
La Figura 6.16 riassume i passaggi appena raccontati per bere le pozioni.

Dal punto di vista implementativo, l'attore Potion è stato strutturato per supportare due punti distinti di afferramento, uno associato alla base della bottiglia e uno al tappo, permettendo una manipolazione più articolata rispetto agli altri oggetti presenti nel gioco.

Per il gesto di bevuta, il meccanismo utilizza il criterio della soglia angolare. In particolare, si calcola il prodotto scalare tra il versore Up della pozione e il Forward vector della camera. La soglia è impostata negativa, richiedendo quindi che l'Up vector della pozione sia orientato opposto al Forward della camera, cioè con il beccuccio rivolto verso la bocca virtuale del giocatore. Grazie a questo controllo, si verifica implicitamente anche che la pozione si trovi a un'altezza compatibile con la bocca: se fosse troppo bassa o troppo alta, l'angolo formato non soddisferebbe la soglia e l'interazione non verrebbe attivata.

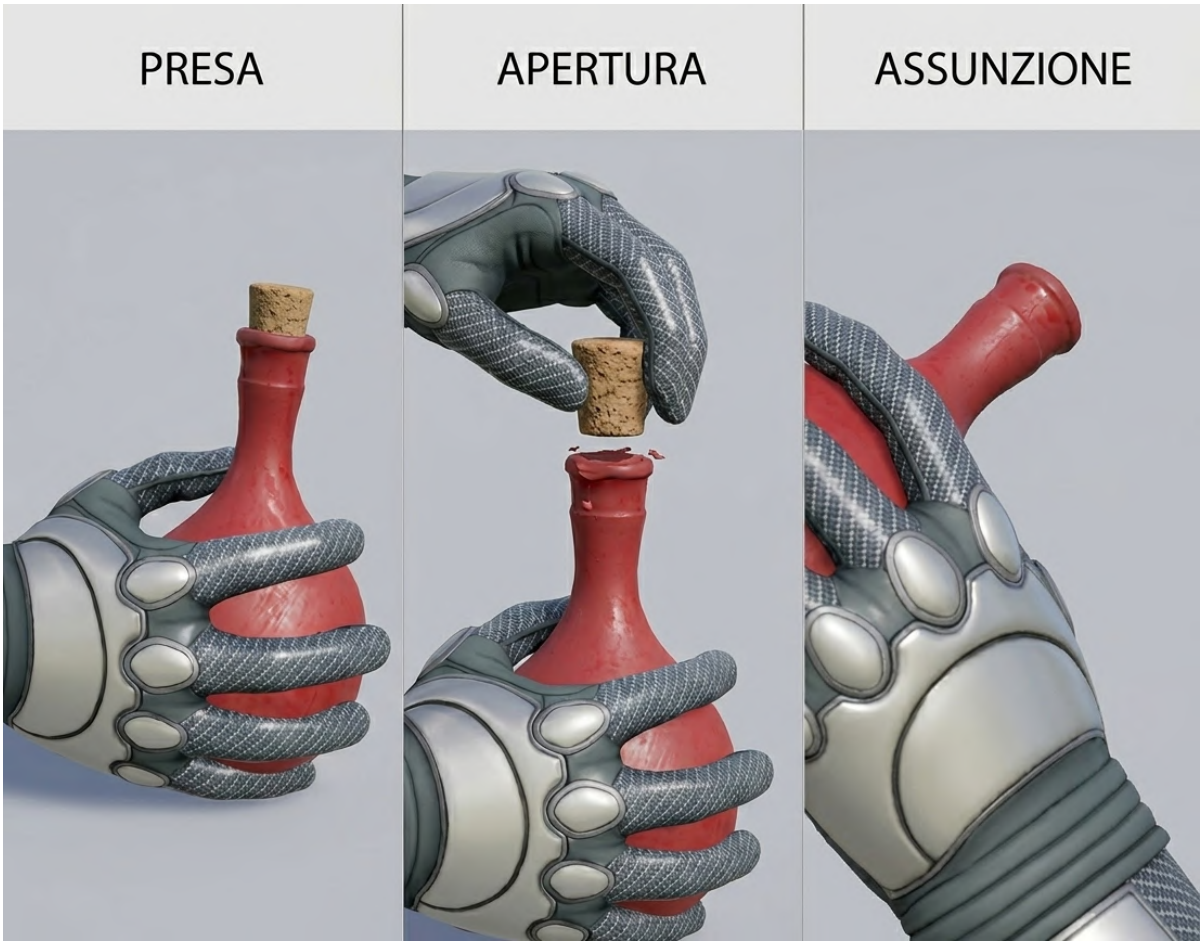

Figura 6.16: Illustrazione delle fasi nell'interazione per la bevuta della pozione

6.3.5 Menu

La progettazione di un menu efficace rientra nell'ambito degli studi di Human-Computer Interaction relativi al System Control (Sezione 3.1.2), che analizza le modalità attraverso cui l'utente interagisce non tanto con gli oggetti virtuali dell'ambiente di gioco, quanto con le configurazioni del sistema.

Il redesign dei menu per la versione VR è stato motivato dal passaggio da un paradigma tipico delle applicazioni desktop, in cui gli elementi vengono semplicemente "aggiunti alla viewport", a un sistema in cui i menu sono oggetti tridimensionali integrati nella scena. Questo approccio consente di mantenere il senso di presenza e l'immersione, favorendo interazioni più naturali e coerenti con il contesto virtuale, ma richiede soluzioni progettuali specifiche.

Nel caso di *Gavunia VR*, il sistema di menu si articola in due macro-categorie: il menu di configurazione iniziale pre-esperienza, che consente di impostare parametri quali livello di difficoltà, lingua e preferenze specifiche per la fruizione in VR; e i menu contestuali disponibili durante la partita, comprendenti la HUD e gli hand-attached menu.
In entrambe le categorie, il design dei menu punta a garantire non-intrusività e continuità dell'esperienza immersiva, permettendo all'utente di accedere alle informazioni o modificare impostazioni senza interrompere il flusso del gioco.

6.3.5.1 Interagire con i menu

Prima di procedere con l'analisi dei menu di *Gavunia VR*, è opportuno descrivere le modalità attraverso cui viene gestita l'interazione con gli elementi di interfaccia in Unreal Engine.
Dal punto di vista implementativo, i menu tridimensionali sono stati realizzati tramite il sistema **UMG (Unreal Motion Graphics)**, integrato all'interno della scena mediante componenti di tipo *Widget Component* [33].
A differenza della versione desktop, in cui i widget vengono renderizzati direttamente nella viewport, in ambiente VR essi sono istanziati come superfici bidimensionali applicate a oggetti tridimensionali presenti nello spazio virtuale.

L'interazione con tali elementi avviene attraverso il *Widget Interaction Component* [34], che simula eventi di input (hover, click, pressione) tramite raycast proiettati dai controller del visore. In questo modo, l'input proveniente dai motion controller (ad esempio la pressione del trigger) viene rimappato su eventi equivalenti a quelli di un dispositivo di puntamento tradizionale (come il click sinistro del mouse), consentendo di riutilizzare la logica di interazione tipica dei sistemi desktop in un contesto immersivo.

6.3.5.2 Impostazioni iniziali

Il menu iniziale, visualizzato all'avvio dell'applicazione, consente al giocatore di configurare l'esperienza in base alle proprie preferenze. Le impostazioni si distinguono in parametri generali, già presenti nella versione desktop, e parametri specifici per la fruizione in VR, come illustrato in Figura 6.17.
Tra i parametri generali rientrano:

- **Difficoltà di gioco**: selezionabile tramite un menu a tendina con cinque livelli (Facile, Medio, Difficile, Molto Difficile, Incubo). Le differenze tra i livelli riguardano principalmente i punti vita del giocatore e degli avversari (maggiori dettagli nella Sezione 5.4.5).

- **Lingua**: permette di selezionare la lingua dell'interfaccia testuale, sia nei menu sia negli elementi informativi presenti durante il gameplay. Attualmente sono supportate Italiano e Inglese. La gestione delle traduzioni avviene tramite il sistema *Localization Dashboard* [35] di Unreal Engine, che consente la centralizzazione delle stringhe e la loro organizzazione per cultura di riferimento, facilitando l'estendibilità futura ad ulteriori lingue.

- **Impostazioni grafiche**: consentono di selezionare il livello di qualità visiva del mondo virtuale. La configurazione è gestita attraverso i *Scalability Settings* [25] di Unreal Engine, che permettono di modulare parametri quali qualità delle ombre, dettaglio delle texture, effetti post-processing e distanza di rendering.

Viceversa, tra i parametri specifici per la fruizione in realtà virtuale rientrano:

- **Abilitazione del ray grabbing**: consente di attivare o disattivare la modalità di interazione a distanza già descritta nella Sezione 6.3.1.2.

- **Tipo di movimento**: permette di selezionare la modalità di navigazione tra smooth locomotion e teleportation, entrambe approfondite nella Sezione 6.3.6.

- **Modalità di esperienza (in piedi/seduto)**: consente di indicare la postura di utilizzo prevista. In base alla selezione, viene adeguata l'origine del sistema di riferimento del Pawn del giocatore, e di conseguenza della camera, garantendo coerenza tra configurazione fisica e spazio virtuale.

- **Abilitazione della tunnel vision**: permette di attivare l'effetto di riduzione dinamica del campo visivo durante la locomozione, tecnica illustrata nella Sezione 6.4.2.

Nel complesso, l'insieme dei parametri configurabili contribuisce a migliorare l'accessibilità e l'adattabilità del prodotto sotto diversi profili: ludico, tecnico e percettivo.
La regolazione della difficoltà consente di modulare il livello di sfida in funzione delle competenze del giocatore, riducendo il rischio di frustrazione nei profili meno esperti e, al contempo, evitando fenomeni di disinteresse negli utenti più avanzati. La possibilità di selezionare la lingua amplia il bacino di utenza e favorisce un'esperienza maggiormente inclusiva. Le impostazioni grafiche permettono invece di adattare il carico computazionale alle capacità dell'hardware disponibile, garantendo un compromesso equilibrato

tra qualità visiva e prestazioni, aspetto particolarmente rilevante in ambito VR, dove la stabilità del frame rate incide direttamente sul comfort dell'utente.

I parametri specifici per la realtà virtuale introducono un ulteriore livello di personalizzazione, legato non solo alle preferenze individuali ma anche alle caratteristiche fisiche e fisiologiche del giocatore. La possibilità di scegliere la modalità di locomozione, attivare o meno la tunnel vision e abilitare tecniche di interazione come il ray grabbing, consente di adattare l'esperienza al proprio livello di tolleranza al movimento virtuale e allo spazio fisico disponibile. Analogamente, la selezione della modalità in piedi o seduto garantisce coerenza tra postura reale e configurazione virtuale, preservando immersione e comfort. Nel loro insieme, tali configurazioni evidenziano una progettazione orientata all'utente, in cui la personalizzazione non rappresenta un elemento accessorio, ma una componente strutturale dell'esperienza VR.

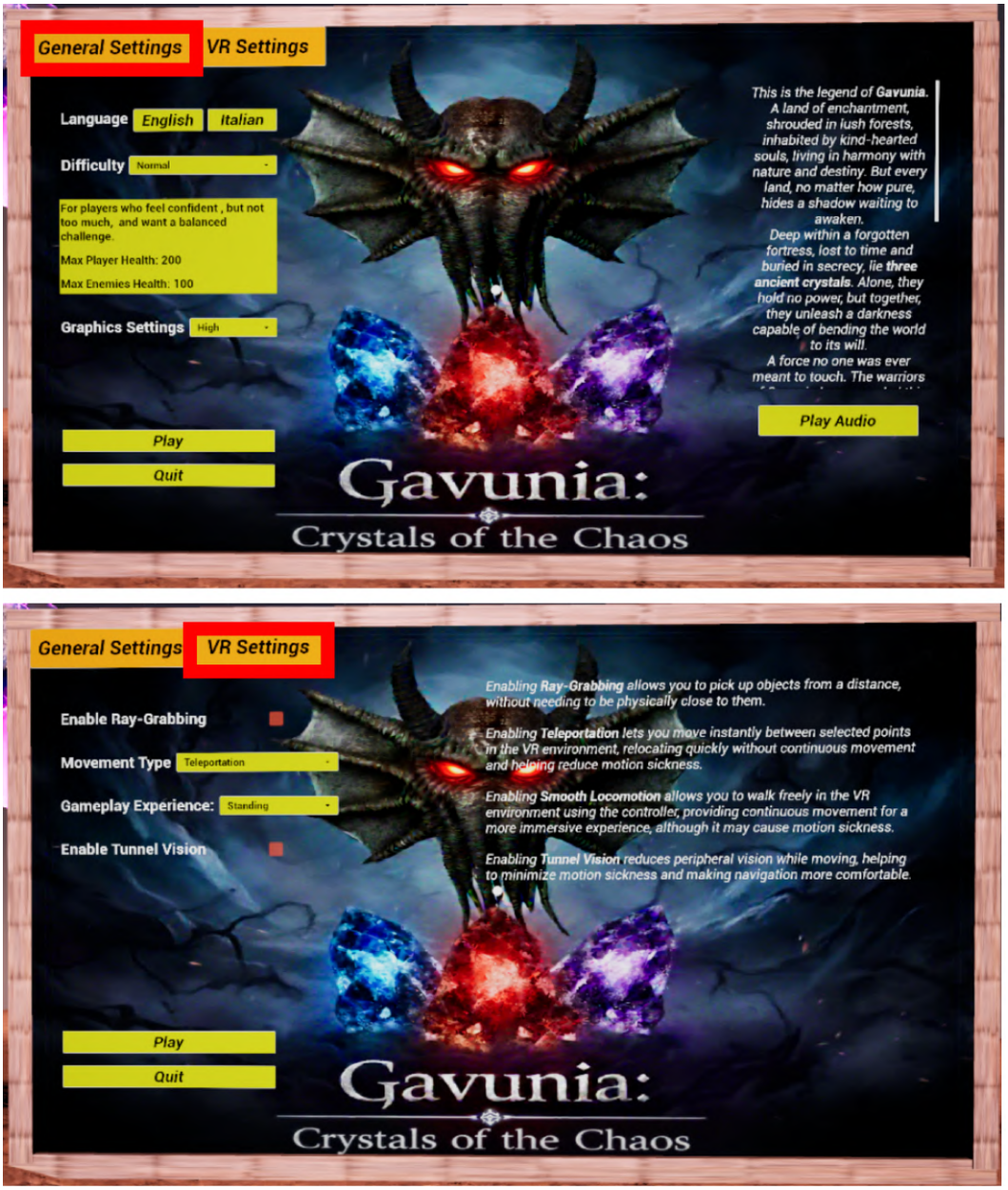

Figura 6.17: Menu iniziale di Gavunia VR

6.3.5.3 HUD

La HUD (Heads-Up Display) rappresenta la componente dell'interfaccia costantemente accessibile al giocatore e deputata alla visualizzazione delle informazioni essenziali per l'esperienza di gioco.
Nella versione desktop di *Gavunia*, la HUD include i punti vita del giocatore e l'indicazione degli obiettivi correnti. Nel primo prototipo della versione VR, tale impostazione è stata replicata mediante un Widget Component ancorato alla camera, sempre visibile nel campo visivo dell'utente. Tuttavia, i test preliminari hanno evidenziato come questa soluzione risultasse eccessivamente invasiva: la presenza permanente di un elemento bidimensionale sovrapposto alla scena occupava una porzione significativa della visuale, interferendo con la percezione dell'ambiente tridimensionale.

Alla luce di tali osservazioni, si è proceduto a una riorganizzazione delle informazioni secondo un criterio di priorità e frequenza di consultazione. **Sono stati mantenuti nella HUD esclusivamente i punti vita**, in quanto informazione critica e di immediata rilevanza, specialmente in un contesto FPS. Gli obiettivi di gioco, invece, caratterizzati da una consultazione meno frequente, sono stati trasferiti all'interno dell'hand-attached menu, descritto nella sezione successiva.
Per ridurre ulteriormente l'intrusività degli indicatori di vita, si è adottata una soluzione ispirata al gioco *Half-Life: Alyx*, nel quale le informazioni vitali sono integrate in un orologio virtuale posizionato sul polso del giocatore. Questa scelta consente di rendere l'informazione rapidamente accessibile mediante un gesto naturale (rotazione del polso), evitando al contempo la presenza costante di overlay nel campo visivo e preservando così continuità percettiva e senso di presenza.
La Figura 6.18 mostra la HUD "a polso" appena descritta.

Figura 6.18: HUD di Gavunia VR

6.3.5.4 Hand-attached menu

Gli hand-attached menu sono interfacce che vengono visualizzate su richiesta dell'utente e risultano ancorate virtualmente alle mani del giocatore.
A differenza della HUD, che fornisce informazioni passive in sola visualizzazione, gli hand-attached menu sono pienamente interattivi. L'interazione è gestita tramite un Widget Interaction Component associato a ciascun motion controller, che consente al giocatore di selezionare e attivare direttamente i pulsanti presenti generando eventi di tipo *Press* e *Release* analoghi a quelli di un dispositivo di puntamento tradizionale.

Menu Gestione: Il menu associato alla mano sinistra, dedicato alla gestione della partita, include gli obiettivi correnti e una serie di comandi di sistema, quali il ritorno al menu iniziale, il riavvio del livello e la chiusura dell'applicazione.
L'associazione del menu di gestione alla mano sinistra risponde a un criterio di separazione funzionale rispetto al menu inventario, collocato sulla mano destra. Tale scelta consente di distinguere chiaramente le operazioni legate alla progressione della partita da quelle relative alla manipolazione degli oggetti, riducendo il rischio di attivazioni involontarie.
La Figura 6.19 illustra un caso d'uso del menu di gestione, evidenziando il sistema di puntamento e selezione dei pulsanti.

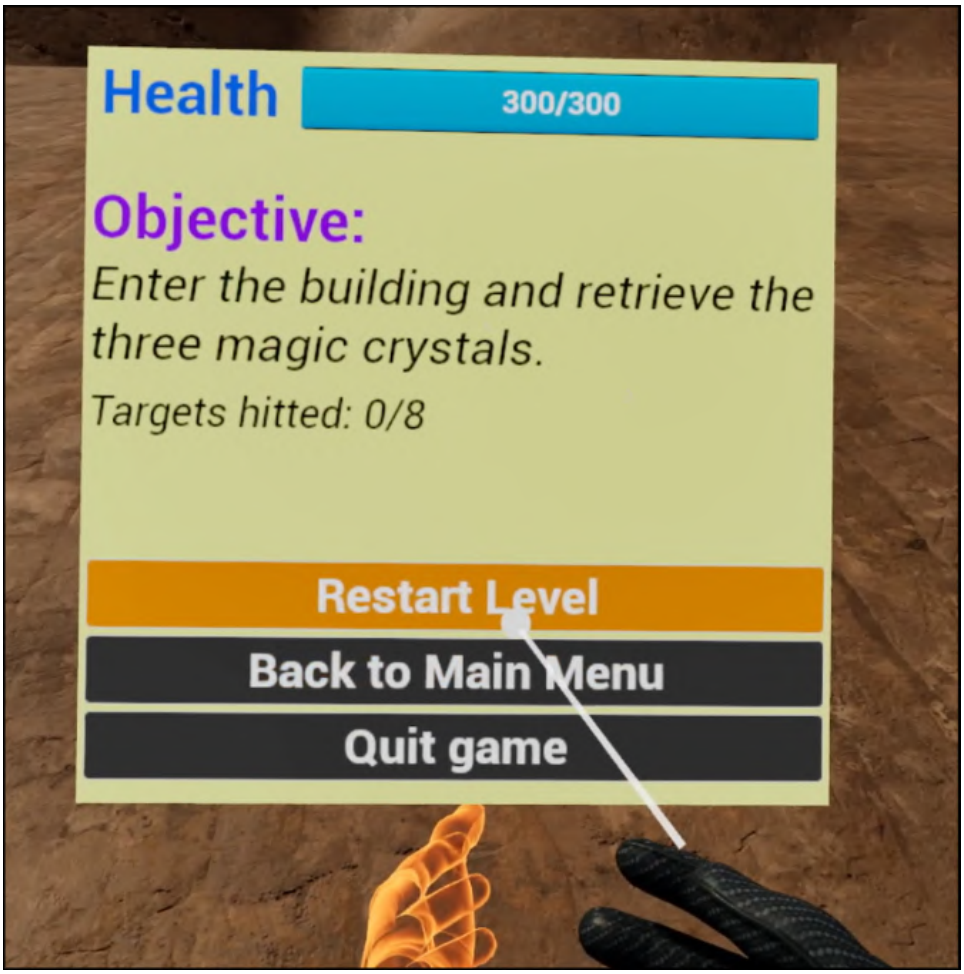

Figura 6.19: Hand-attached menu sinistro in Gavunia VR

Menu Inventario: Il menu associato alla mano destra, dedicato all'inventario, combina l'interazione tramite Widget Interaction Component con la manipolazione fisica degli oggetti presenti nella mano del giocatore.
Nella versione desktop, l'aggiunta di un oggetto all'inventario avviene mediante la pressione del tasto "E" in prossimità dell'elemento selezionato. In VR, tale meccanismo è stato riprogettato secondo un paradigma di interazione spaziale, sostituendo l'input simbolico con un'azione fisica coerente con il contesto immersivo. L'inserimento nell'inventario avviene infatti tramite un sistema di "drag-and-drop": il giocatore afferra l'oggetto e lo rilascia nell'area dedicata del menu.
L'estrazione dall'inventario avviene, invece, puntando l'icona corrispondente dell'oggetto tramite il motion controller e premendo il pulsante "X". Tale azione riduce di un'unità la quantità nell'inventario e determina lo spawn di una nuova istanza dell'oggetto nel mondo di gioco, accanto alla mano del giocatore. La scelta di non posizionare direttamente l'oggetto nella mano è stata motivata dalla necessità di consentire al giocatore di estrarre più oggetti consecutivamente: se l'oggetto fosse stato spawnato nella mano, la limitazione a un solo oggetto per mano avrebbe impedito questa operazione, compromettendo la fluidità dell'interazione.
La Figura 6.20 illustra i due casi d'uso del menu inventario, evidenziando il meccanismo di inserimento tramite drag-and-drop (Caso A) e quello di estrazione con spawn dell'oggetto nel mondo di gioco (Caso B).

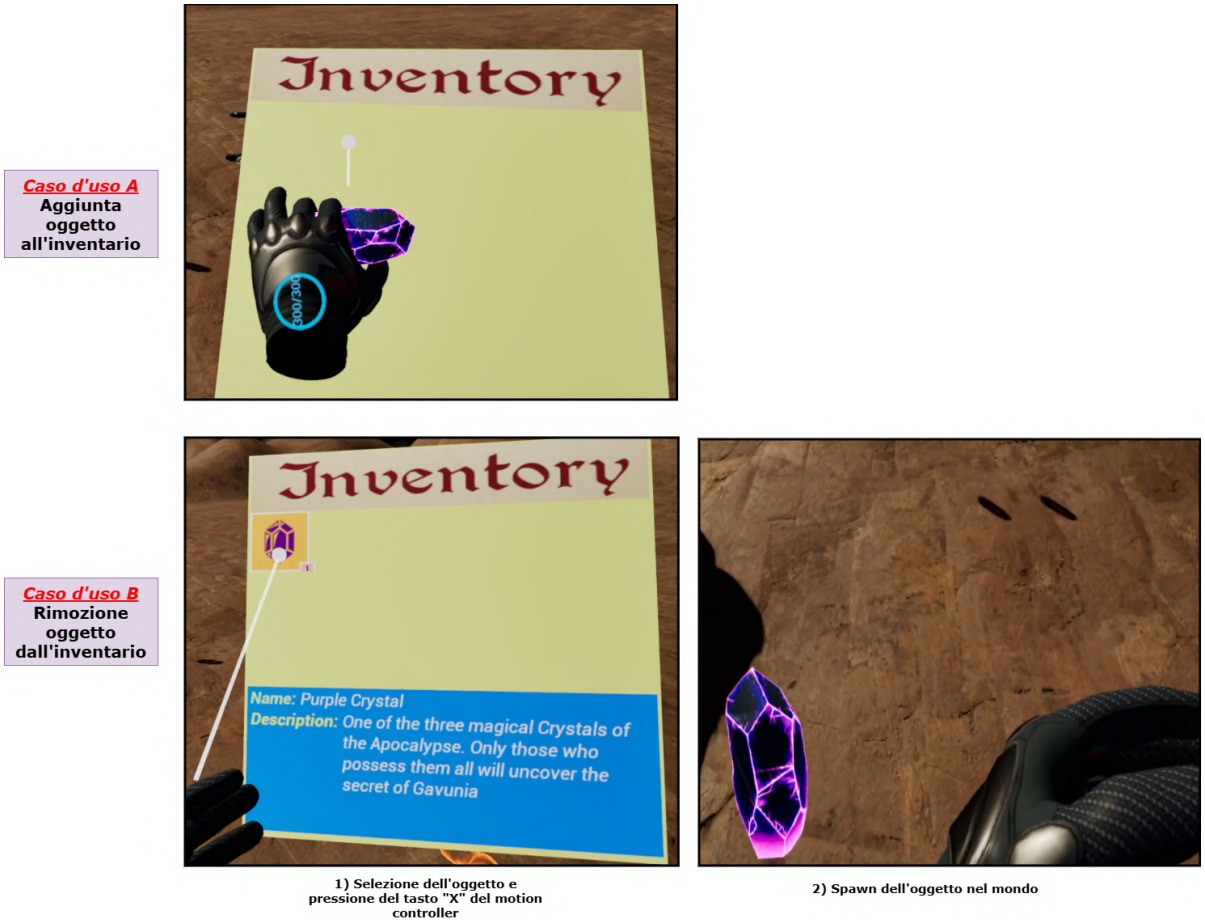

Figura 6.20: Hand-attached menu destro in Gavunia VR

6.3.5.5 Ruota delle armi

La ruota delle armi è un menu circolare che consente al giocatore di selezionare rapidamente le armi disponibili. Essa viene attivata mantenendo premuto il thumbstick del motion controller e rimane visibile per tutta la durata della pressione, scomparendo al rilascio. Questo meccanismo *hold-to-open* garantisce un accesso rapido e contestuale senza interrompere il flusso di gioco.
La ruota è accessibile con entrambe le mani, permettendo al giocatore di gestire simultaneamente due armi e favorendo una maggiore flessibilità strategica durante il gameplay.

La ruota è stata introdotta come soluzione progettuale per due motivi principali. In primo luogo, permette al giocatore di liberare la mano dall'arma nelle situazioni in cui non è necessaria. In secondo luogo, prepara il sistema a versioni future del gioco in cui il giocatore potrà possedere più tipologie di armi differenti, implementando così un inventario dedicato esclusivamente alle armi.

La scelta di non rendere le armi inventoriabili tramite il menu inventario è motivata dal dinamismo del gameplay: in contesti frenetici, lo switch delle armi deve avvenire nel modo più rapido possibile. L'apertura dell'inventario, la selezione tramite puntamento, la pressione del pulsante per spawnare l'arma e il suo successivo afferramento avrebbero reso l'operazione troppo lenta e macchinosa.
Con la ruota delle armi, invece, la selezione è immediata: ciascuna arma disponibile è associata a un pulsante del menu, e trascinando la mano virtuale verso il pulsante e rilasciando, l'arma viene automaticamente equipaggiata.

Dopo un'attenta analisi, sempre in ottica di rapidità e dinamismo, si è deciso di includere anche i caricatori come oggetti presenti nella ruota delle armi, in quanto strettamente legati ad esse. In situazioni in cui è necessario recuperare un caricatore velocemente, il sistema di menu di inventario sarebbe risultato troppo lento.

La Figura 6.21 illustra i principali scenari operativi della ruota delle armi, evidenziando le diverse modalità di interazione: configurazione iniziale (Caso A), memorizzazione di un'arma nella ruota (Caso B), equipaggiamento di un'arma selezionata (Caso C) e cambio rapido tra arma e caricatore (Caso D).

Figura 6.21: Casi d'uso differenti per la ruota delle armi in Gavunia VR

6.3.6 Meccanismi di navigazione

La progettazione dei meccanismi di navigazione ha richiesto un'attenta analisi, in quanto costituisce un elemento di raccordo tra le dinamiche di gameplay e il comfort dell'utente, in particolare rispetto al rischio di motion sickness.
Tra le tecniche descritte nella Sezione 3.1.4.1 sono state considerate le due soluzioni maggiormente diffuse in ambito videoludico, ovvero smooth locomotion e teleportation. La tecnica di eye-directed control è stata esclusa poiché, nei videogiochi FPS, è frequente dissociare direzione dello sguardo e direzione del movimento, ad esempio per mantenere la mira su un bersaglio mentre ci si sposta verso una posizione di riparo.
Le due tecniche rimanenti sono state entrambe implementate per le seguenti motivazioni: la smooth locomotion garantisce un'esperienza di navigazione più continua e immersiva, ma può aumentare l'insorgenza di motion sickness; la teleportation riduce significativamente il rischio di discomfort, a fronte di un minor livello di realismo percepito.

Considerata la validità di entrambe le soluzioni, è stata lasciata all'utente la possibilità di selezionare il sistema di navigazione preferito (tramite menu di impostazioni), in linea con le buone pratiche di Human-Computer Interaction descritte nella Sezione 3.2.3.

6.3.6.1 Teleportation

Il movimento tramite Teleportation è uno dei metodi più diffusi per la navigazione in ambienti di realtà virtuale, in quanto permette di spostarsi istantaneamente tra punti della scena. Questa tecnica è molto utilizzata nei giochi VR poiché riduce significativamente il rischio di motion sickness.

Per garantire un teletrasporto sicuro, è necessario che il punto di destinazione sia situato all'interno di una zona navigabile del mondo virtuale. In Unreal Engine, ciò corrisponde alla presenza di una **NavMesh** [20], che impedisce al giocatore di teletrasportarsi in aree irraggiungibili, come sopra muri, spazi vuoti o zone bloccate da ostacoli. In questo modo, il sistema di Teleportation non solo rende il movimento sicuro, ma minimizza anche problemi di disorientamento o errori tecnici durante l'azione.

A livello di implementazione, il processo di teletrasporto nel VRPawn si articola in tre fasi principali:

1. **Avvio dell'azione di movimento**: Quando l'utente attiva l'input di movimento, viene visualizzata una traiettoria indicativa tramite un effetto particellare e un marcatore della destinazione, consentendo di selezionare con precisione il punto di arrivo.

2. **Calcolo della posizione di arrivo**: Viene calcolata la posizione finale di arrivo del teletrasporto utilizzando una traiettoria del proiettile, con velocità iniziale e angolo impostati a valori predefiniti. Il punto calcolato viene quindi proiettato sulla NavMesh, in modo da garantire che la destinazione sia navigabile e prevenire che il giocatore finisca in punti irraggiungibili.

3. **Completamento dell'azione**: Al rilascio del comando, il sistema sposta il giocatore nella posizione selezionata e rimuove gli indicatori temporanei, completando l'azione di teletrasporto.

6.3.6.2 Smooth locomotion

Il movimento tramite Smooth Locomotion consente uno spostamento continuo e naturale all'interno della scena virtuale. Tale soluzione favorisce una maggiore immersione rispetto alla teleportation, ma può aumentare l'insorgenza di motion sickness, fenomeno frequentemente associato al movimento continuo in VR.

A livello implementativo, lo spostamento del VRPawn è gestito dal componente **CharacterMovement**, che applica velocità e direzione in risposta agli input del controller. L'intensità del movimento è proporzionale alla pressione esercitata sul thumbstick, mentre la direzione viene calcolata nello spazio locale del personaggio: per l'asse orizzontale si utilizza il Right Vector, per quello verticale il Forward Vector. La velocità massima di movimento è stata impostata a un valore calibrato come compromesso tra dinamismo del gameplay e comfort dell'utente, evitando spostamenti eccessivamente rapidi che avrebbero potuto incrementare sintomi di malessere.

A differenza del teletrasporto, questa modalità non richiede la definizione di una NavMesh, poiché il controllo delle collisioni è demandato alla **Capsule Collision**, che definisce il volume fisico dell'avatar e ne garantisce il corretto comportamento rispetto all'ambiente circostante.

Salto: Per completare le possibilità di movimento offerte dalla Smooth Locomotion, è stata introdotta anche l'azione di salto, necessaria per raggiungere aree sopraelevate non accessibili tramite la sola camminata. Questa funzionalità è inoltre necessaria per garantire al giocatore l'accesso a tutte le aree della mappa raggiungibili tramite Teleportation, preservando la libertà di esplorazione indipendentemente dal sistema di navigazione scelto.

Dal punto di vista implementativo, il salto è gestito direttamente dal componente CharacterMovement, che fornisce le funzioni *Jump* e *StopJumping* per l'attivazione e l'interruzione dell'azione. L'altezza raggiunta non è definita da un valore fisso, ma deriva dalla combinazione di parametri fisici quali la Jump Z Velocity, che determina la velocità verticale iniziale, e la Gravity Scale, che regola l'intensità della gravità applicata al personaggio.
La traiettoria risultante è influenzata anche dalla velocità orizzontale al momento dell'impulso: un salto eseguito durante la corsa produce infatti un arco più esteso rispetto a uno effettuato da fermo. Inoltre, il parametro Air Control consente di mantenere un controllo parziale della direzione anche in fase aerea, contribuendo a rendere il movimento più naturale e coerente con l'intenzione del giocatore.

6.4 Tecniche per riduzione della motion sickness

Durante lo sviluppo del gioco, particolare attenzione è stata dedicata alla riduzione della motion sickness, aspetto fondamentale per garantire il comfort dell'utente senza compromettere la qualità dell'esperienza ludica.
Tra le tecniche analizzate nella Sezione 4.1 sono state implementate la Teleportation (già descritta in Sezione 6.3.6.1) e la Tunnel Vision, mentre la Geometry Deformation è stata esclusa. Questa scelta è motivata dalle caratteristiche strutturali del mondo virtuale di Gavunia, che non presenta configurazioni spaziali prevalentemente direzionali o geometrie regolari — come corridoi o ambienti canalizzati — nelle quali la Geometry Deformation risulta maggiormente efficace. In contesti più aperti e irregolari, infatti, la deformazione progressiva degli elementi periferici può risultare meno coerente con il movimento dell'osservatore. Inoltre, l'alterazione dinamica e continua della scena avrebbe comportato un incremento del costo computazionale in fase di rendering, con il rischio di ridurre il frame rate e compromettere ulteriormente il comfort visivo dell'utente.

Al fine di offrire un'esperienza personalizzabile, le tecniche implementate non sono attive di default e possono essere selezionate dal menu delle impostazioni iniziali, permettendo a ciascun utente di attivarle secondo le proprie esigenze.

6.4.1 Snap turn

Tra le tecniche adottate per la riduzione della motion sickness è stato implementato lo Snap Turn, una modalità di rotazione discreta della visuale che sostituisce la rotazione continua lungo l'asse verticale.
Nella rotazione tradizionale (smooth rotation), lo spostamento angolare avviene in modo progressivo in risposta all'input del controller, generando un flusso visivo continuo che può causare disallineamento tra percezione visiva e sistema vestibolare, aumentando il rischio di discomfort.
Lo Snap Turn, invece, suddivide la rotazione in incrementi angolari prefissati (ad esempio 30° o 45°), attivati tramite una pressione del thumbstick. Il cambiamento di orientamento avviene istantaneamente, eliminando la fase intermedia di movimento continuo. Tale discontinuità riduce significativamente il flusso ottico periferico, principale responsabile della sensazione di nausea durante la rotazione in VR.

A differenza di altre tecniche di mitigazione attivabili opzionalmente, lo Snap Turn è sempre disponibile durante l'esperienza di gioco, in quanto rappresenta un approccio consolidato per la gestione della rotazione della visuale in ambienti VR. La sua integrazione permanente consente di garantire un controllo dell'orientamento efficace e al contempo compatibile con le esigenze di comfort della maggior parte degli utenti.

6.4.2 Tunnel Vision

La decisione di implementare la Tunnel Vision come tecnica per la riduzione della motion sickness è stata supportata dal suo impiego in un'applicazione di grande successo come *Google Earth VR*.
Sebbene *Gavunia* e *Google Earth VR* siano esperienze concettualmente diverse, la tecnica della Tunnel Vision può essere efficacemente riadattata a *Gavunia* con un semplice accorgimento. In particolare, essendo *Google Earth VR* un'applicazione di esplorazione aerea, ai piedi del tunnel è stata inserita una griglia per fornire all'utente un chiaro riferimento del suolo. Nel caso di *Gavunia*, invece, il giocatore dispone già di un riferimento visivo naturale del terreno, rendendo superflua l'aggiunta di tale griglia.

A livello implementativo in Unreal Engine, l'effetto di Tunnel Vision è stato simulato posizionando quattro piani attorno alla camera del giocatore: tre dei quali completamente neri, mentre il piano frontale utilizza un materiale specificamente configurato per riprodurre l'effetto tunnel.

Il materiale si basa su una texture in scala di grigi caratterizzata da un punto nero al centro e da un'area bianca verso i bordi. La texture viene collegata al canale Opacity del materiale, che è configurato con modalità di blending Translucent e modello di shading Unlit.
Nel contesto del canale Opacity, le aree nere della texture corrispondono a valori di opacità bassi (cioè trasparenza elevata), mentre le aree bianche indicano massima opacità. Pertanto, il centro nero della texture genera un "foro" trasparente nel materiale, permettendo una visione chiara e non ostacolata della scena sottostante. Progressivamente, verso i bordi, la tonalità bianca aumenta l'opacità, oscurando gradualmente la visione periferica dell'utente. L'effetto risultante è una finestra visiva circolare che limita la percezione periferica, contribuendo a ridurre il flusso ottico esterno e mitigare i sintomi di motion sickness tipici della realtà virtuale. Il modello di shading Unlit assicura che il filtro rimanga costante e indipendente dall'illuminazione della scena, mantenendo l'efficacia visiva dell'effetto in ogni condizione ambientale.

Come in *Google Earth VR*, il tunnel non è visibile in maniera costante, ma si attiva soltanto durante il movimento, in modo da non limitare permanentemente la visuale dell'utente. In *Gavunia*, il filtro viene attivato solo quando la pressione in avanti sul thumbstick supera una soglia predeterminata, evitando l'attivazione durante piccoli spostamenti che normalmente non generano sintomi di motion sickness. L'intensità della pressione del thumbstick viene fornita direttamente da Unreal Engine, che restituisce un valore continuo nel range [-1, 1], dove -1 corrisponde al movimento completamente indietro e 1 alla spinta massima in avanti.

Capitolo 7

Ottimizzazioni per Gavunia VR

Nel capitolo precedente sono state discusse le principali scelte progettuali adottate per il porting di *Gavunia* dalla versione desktop alla versione VR, con particolare riferimento ai layer di Modelling, Game Logic e Animation.

Il presente capitolo affronta invece il processo di ottimizzazione delle prestazioni, elemento cruciale nello sviluppo di applicazioni in realtà virtuale. A differenza del contesto desktop, l'ambiente VR impone vincoli significativamente più stringenti in termini di frame rate, latenza e stabilità del rendering, poiché eventuali cali prestazionali possono compromettere non solo la qualità visiva, ma anche il comfort e il senso di presenza dell'utente.

L'analisi si concentra in particolare sulle modifiche apportate al Rendering Layer, con attenzione alla gestione dell'illuminazione e alle tecniche adottate per ridurre il carico computazionale mantenendo un adeguato livello qualitativo.
Il processo di ottimizzazione è stato condotto attraverso una fase iterativa di sperimentazione e profiling, confrontando diverse soluzioni tecniche e valutandone l'impatto sulle prestazioni complessive del sistema, fino a definire la configurazione finale del progetto.
La Figura 7.1 illustra il processo effettuato, le cui singole fasi verranno dettagliate nelle sezione successive.

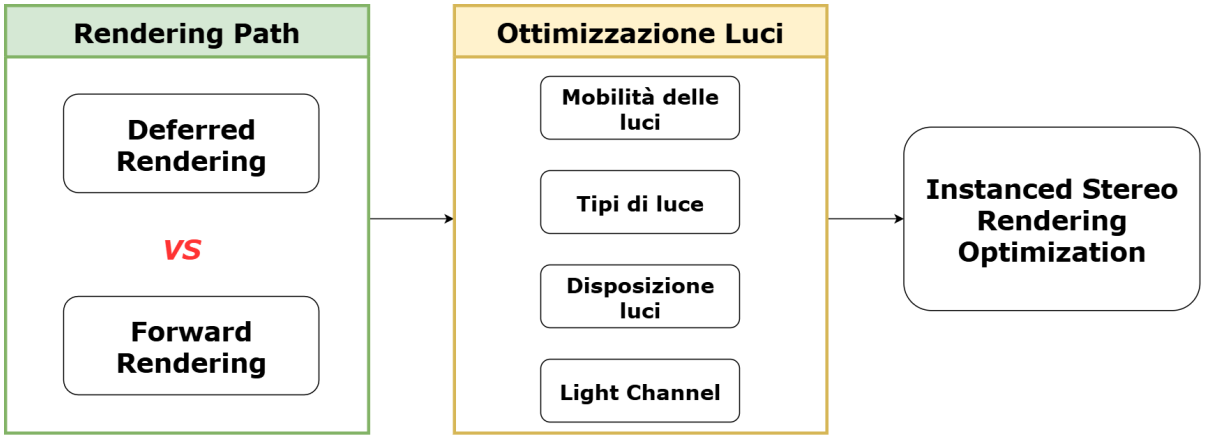

Figura 7.1: Schema del processo di ottimizzazione di Gavunia VR

7.1 Scelta del Rendering Path

Una delle prime scelte affrontate nel processo di ottimizzazione ha riguardato la selezione del rendering path più adatto al contesto VR.
In Unreal Engine, le due principali modalità di rendering disponibili sono il **Deferred Rendering** e il **Forward Rendering**, ciascuna caratterizzata da vantaggi e compromessi differenti.

Deferred Rendering: Nel rendering differito, la scena viene inizialmente renderizzata in una serie di buffer intermedi chiamati *G-Buffer*, nei quali vengono memorizzate per ciascun pixel le informazioni di materiale e geometria (normali, albedo, parametri di roughness e metallic, profondità). L'illuminazione viene successivamente calcolata in uno o più passaggi separati, operando in screen space sui dati precedentemente salvati. Questo approccio risulta particolarmente efficiente nella gestione di numerose luci dinamiche e consente tecniche avanzate di shading e illuminazione globale, come il sistema Lumen. Tuttavia, comporta un maggiore consumo di memoria e una pipeline più articolata, con conseguente incremento del carico sulla GPU.

Forward Rendering: Nel rendering diretto, il calcolo dell'illuminazione avviene direttamente durante la resa degli oggetti, senza l'utilizzo di G-Buffer intermedi. Geometria, materiale e illuminazione vengono processati nello stesso passaggio di shading, riducendo il numero di operazioni intermedie e la pressione sulla memoria. Questo modello risulta generalmente più leggero dal punto di vista computazionale, soprattutto in presenza di un numero limitato di sorgenti luminose, risultando adatto in contesti VR dove il frame rate deve rimanere stabile.

La Figura 7.2 mostra uno schema delle pipeline di Deferred Rendering e Forward Rendering, evidenziandone le principali differenze operative.

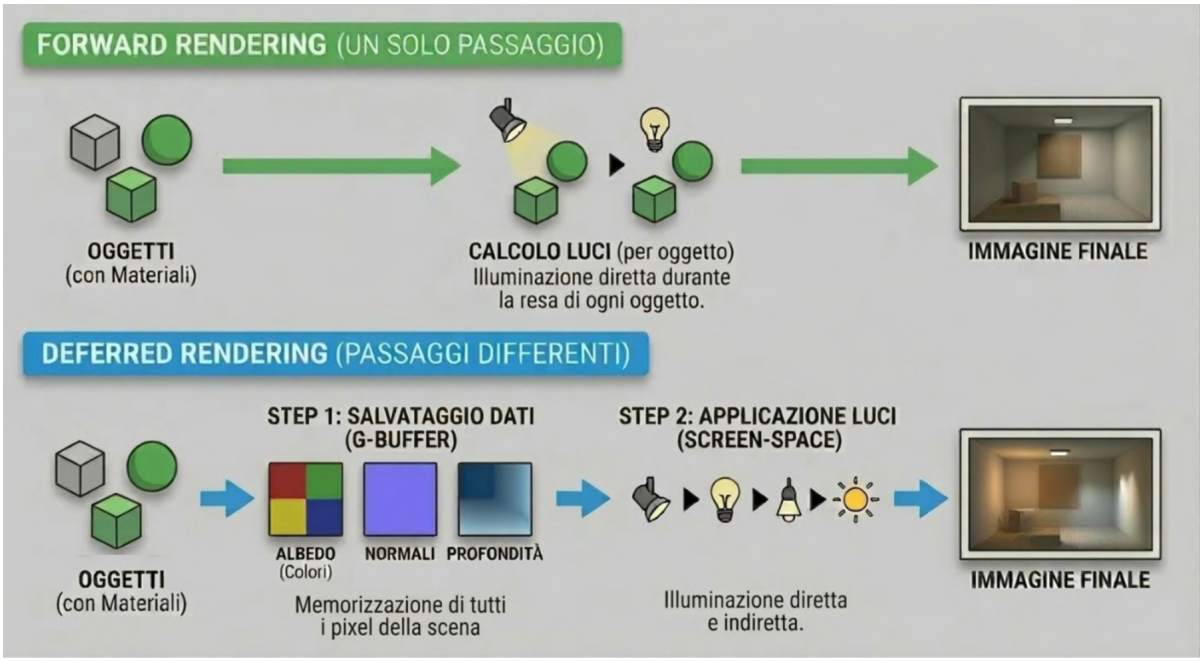

Figura 7.2: Differenze tra Deferred e Forward Rendering

7.1.1 Esperimenti e scelta finale

Nella versione desktop di *Gavunia*, la scelta della modalità di rendering è stata determinata dall'impiego di Lumen, che è compatibile esclusivamente con il Deferred Rendering. Lumen richiede infatti un utilizzo estensivo del G-Buffer, il che ne impedisce l'adozione nel Forward Rendering. Di conseguenza, **optare per il Forward Shading implica la rinuncia a sistemi di illuminazione globale dinamica come Lumen**, rendendo necessaria l'adozione di soluzioni alternative quali il light baking, le reflection captures e altre tecniche di illuminazione statica o ibrida.

Nonostante queste limitazioni, la documentazione ufficiale di Unreal Engine [36] suggerisce spesso l'uso del Forward Shading per applicazioni VR, grazie alla sua maggiore efficienza e velocità di rendering. È stato pertanto sperimentato anche questo approccio per *Gavunia VR*, al fine di valutarne l'impatto sia in termini prestazionali sia qualitativi rispetto al Deferred Rendering.

Per quanto riguarda gli ambienti interni, come evidenziato in Figura 7.3, il passaggio a Forward non ha introdotto variazioni sostanziali.
Diversamente, per l'illuminazione esterna (Figura 7.4), l'attivazione del Forward Shading ha comportato la perdita del sistema dinamico di Sky Atmosphere, che non risultava correttamente integrato nel sistema di illuminazione della scena.
Il cielo appariva pertanto privo di scattering atmosferico e visivamente nero. Per ovviare a tale problematica è stato adottato lo SkyBox fornito nel VR Template di Unreal Engine, costituito da un attore che simula il cielo in funzione dell'inclinazione della Directional Light. Questa soluzione, tuttavia, ha comportato una significativa perdita di realismo sia nell'atmosfera solare sia nelle ombre, comportando inoltre una scena esterna meno luminosa.

Nonostante questa perdita di realismo, in particolare nell'ambiente esterno, **i benefici derivanti dall'adozione del Forward Shading in termini di prestazioni hanno portato alla scelta finale di questa modalità di rendering**.
In particolare, il confronto tra le prestazioni delle due pipeline ha evidenziato una differenza significativa nel frame rate. Utilizzando il Forward Shading è stato possibile mantenere 72 FPS stabili, valore corrispondente al limite massimo imposto dal refresh rate del visore Meta Quest 3S utilizzato per i test, configurato a 72 Hz.
Al contrario, con il Deferred Rendering il frame rate scendeva fino a circa 30 FPS, principalmente a causa del costo computazionale associato al sistema di illuminazione globale dinamica Lumen.

Il raggiungimento di 72 FPS stabili non è stato tuttavia determinato esclusivamente dall'adozione del Forward Shading, ma anche da un'attenta riprogettazione del sistema di illuminazione della scena, che verrà analizzata nel dettaglio nella Sezione 7.2.

Figura 7.3: Confronto tra Deferred Rendering e Forward Rendering negli interni dell'edificio

Figura 7.4: Confronto tra Deferred Rendering e Forward Rendering negli esterni

7.2 Gestione ottimale delle luci

Come discusso nella Sezione 7.1.1, l'adozione del Forward Shading ha comportato la rinuncia ai sistemi di illuminazione globale dinamica offerti da Lumen, rendendo necessario il ricorso a tecniche di illuminazione basate su light baking [37].
In questo contesto, la progettazione del sistema di illuminazione della scena assume un ruolo particolarmente importante, poiché influisce direttamente sia sulla qualità visiva sia sulle prestazioni dell'applicazione in realtà virtuale.
Per questo motivo è stato necessario adottare un processo di ottimizzazione dell'illuminazione, articolato in quattro fasi principali:

- **Studio della mobilità delle luci**, che determinano il modo in cui l'illuminazione viene calcolata e il relativo costo computazionale a runtime.
- **Analisi dei tipi di luce** da utilizzare, privilegiando quelle più efficienti in termini di costo computazionale.
- **Posizionamento strategico delle luci**, in modo da garantire un'illuminazione adeguata in ciascun ambiente senza sovraccaricare la scena con fonti luminose sovrapposte.
- **Utilizzo dei Light Channels**, per limitare l'influenza di alcune luci a specifici oggetti o gruppi di oggetti, riducendo il numero di calcoli necessari per frame.

7.2.1 Studio della mobilità delle luci

In Unreal Engine esistono tre diverse tipologie di mobilità per le luci e per gli oggetti: Static, Stationary e Movable.

Le luci **Static** vengono completamente pre-calcolate tramite il processo di light baking e quindi non comportano alcun costo computazionale a runtime durante l'esecuzione dell'applicazione. Tuttavia, questa tipologia di luce presenta alcune limitazioni: non può contribuire all'illuminazione dinamica degli oggetti con mobilità Movable e non consente la generazione di ombre dinamiche.

Le luci **Stationary** rappresentano una soluzione intermedia tra illuminazione statica e dinamica. In questo caso, la componente indiretta dell'illuminazione viene pre-calcolata durante il baking e memorizzata nelle lightmap associate alle superfici degli oggetti, mentre la componente diretta della luce e le ombre dinamiche vengono calcolate a runtime. Questo consente di mantenere un buon compromesso tra qualità visiva e prestazioni.
Per ragioni di performance, Unreal Engine impone tuttavia una limitazione al numero di Stationary Lights che possono influenzare simultaneamente un singolo oggetto. In particolare, ogni oggetto può essere interessato al massimo da quattro luci Stationary sovrapposte, oltre le quali il motore segnala una situazione di overdraw che può comportare un aumento significativo del costo computazionale.

Infine, le luci **Movable** sono completamente dinamiche e vengono calcolate interamente a runtime, senza alcun contributo del light baking. Questa tipologia di luce offre la massima flessibilità, permettendo ad esempio di modificare posizione, intensità o direzione della luce durante l'esperienza di gioco. Tuttavia, il loro costo computazionale è

significativamente più elevato rispetto alle altre tipologie di mobilità, rendendole poco adatte a contesti in cui è necessario mantenere prestazioni elevate, come nel caso delle applicazioni di realtà virtuale.

Alla luce di queste considerazioni, **nella scena di *Gavunia* è stata adottata principalmente una configurazione basata su luci Stationary**. Questa scelta è stata determinata dalla necessità di mantenere un buon compromesso tra qualità visiva e prestazioni, consentendo al tempo stesso la presenza di ombre dinamiche per gli oggetti mobili della scena. In particolare, diversi elementi dell'ambiente — come le porte e le casse che si aprono durante l'interazione — devono poter muoversi, mentre i nemici necessitano di proiettare ombre dinamiche per evitare di apparire visivamente sospesi nello spazio.

Le luci Stationary sono quindi state opportunamente distribuite nello scena al fine di evitare eccessive sovrapposizioni e mantenere il numero di luci influenti per oggetto entro il limite massimo di quattro luci Stationary supportato dal motore.
Le strategie adottate per la loro disposizione verranno discusse nella Sezione 7.2.3.

7.2.2 Analisi dei tipi di luce

L'analisi sulla tipologia di luci da utilizzare si è concentrata principalmente negli interni dell'edificio, poiché la configurazione esterna della scena è gestita tramite una Directional Light che simula la luce solare.

Nella versione desktop di *Gavunia*, come descritto nella Sezione 5.3.1.2, la scelta della tipologia di luce era meno vincolata da considerazioni prestazionali: avendo a disposizione più margine per il calcolo dell'illuminazione e non dovendo renderizzare la stessa scena due volte (una per occhio) come in VR, la scelta della tipologia di luce è stata fatta semplicemente per gusto estetico, andando a privilegiare le Point Light che emanano luce in tutte le direzioni, illuminando l'ambiente in modo più uniforme.

Tuttavia, in ambito VR questa soluzione risultava eccessivamente costosa. Le Point Light, generando luce a 360°, richiedono più calcoli e tendono facilmente a sovrapporsi con altre luci, aumentando significativamente il carico computazionale. Per questo motivo, è stato necessario rivedere l'illuminazione interna, **privilegiando l'impiego di Spot Light**, che emettono luce lungo un fascio concentrato, consentendo di illuminare in maniera mirata specifiche aree senza introdurre sovrapposizioni eccessive.

Per mantenere un livello di illuminazione complessivo simile alla versione desktop, in alcune zone una singola Point Light è stata sostituita da più Spot Light, mentre le Point Light rimaste sono state riorganizzate diminuendo il raggio d'azione (Attenuation Radius) per ridurre le sovrapposizioni.

Durante questa fase di ottimizzazione, uno strumento particolarmente utile è stato la vista **"Stationary Light Overlap"**, che permette di individuare rapidamente le aree in cui più luci Stationary influenzano lo stesso oggetto, facilitando la redistribuzione delle fonti luminose per ottenere un bilanciamento ottimale tra qualità visiva e prestazioni.
La Figura 7.5 mostra un caso d'uso dello strumento. La barra dei colori, che va dal verde chiaro al bianco, indica il grado di sovrapposizione delle luci nella scena: valori vicini al verde corrispondono a un basso numero di luci sovrapposte (uno o due), mentre

valori verso il rosso segnalano un numero elevato di luci sovrapposte (quattro o più), con conseguente aumento del costo computazionale.

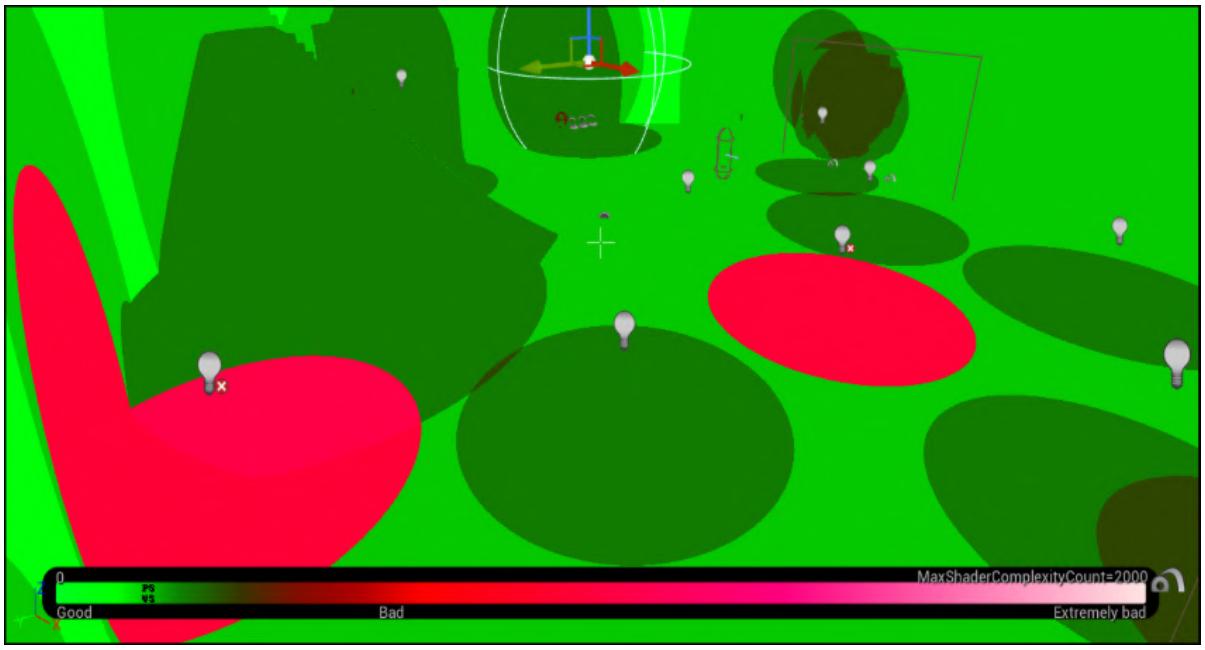

Figura 7.5: Esempio d'uso di "Stationary Light Overlap"

7.2.3 Posizionamento strategico delle luci

A seguito dell'analisi sulla mobilità delle luci, che ha portato alla scelta predominante delle Stationary Light per gestire oggetti e ombre in modo dinamico, e sulla tipologia di luci, che ha indicato un maggior utilizzo di Spot Light rispetto alle Point Light, si è resa necessaria una pianificazione strategica del loro posizionamento.
Tale posizionamento ha l'obiettivo di evitare sovrapposizioni eccessive, garantendo al contempo un'illuminazione bilanciata della scena. In particolare, si è voluto evitare sia ambienti eccessivamente illuminati, con conseguente aumento del carico computazionale, sia zone troppo buie, che comprometterebbero la facilità di esplorazione da parte dell'utente.

La disposizione delle luci è stata comunque progettata in coerenza con la narrazione ambientale di *Gavunia*. Il piano terra presenta un'illuminazione volutamente più cupa e misteriosa, accentuando l'atmosfera labirintica dell'ambiente. Al contrario, il piano superiore è caratterizzato da una luce più intensa, favorita dalle fessure lasciate appositamente per far filtrare la luce solare generata dalla Directional Light esterna.

Per il piano terra, la strategia di illuminazione ha privilegiato una buona visibilità nelle stanze principali, mentre nelle aree di passaggio è stato adottato un approccio più discreto: le Spot Light illuminano lievemente il centro delle stanze, mentre le Point Light posizionate vicino ai passaggi murari guidano il giocatore nell'esplorazione degli ambienti.
Questa configurazione permette di mantenere un'esperienza immersiva e coerente con i principi di game design, ottimizzando al contempo le prestazioni.

La Figura 7.6 mostra un esempio delle luci del piano terra: le Spot Light (in rosso) illuminano il centro delle stanze, mentre le Point Light (in giallo) evidenziano i passaggi tra le stanze.

Nel piano superiore, invece, l'illuminazione complessiva è più elevata grazie anche al contributo della luce solare esterna. A ciò si aggiunge l'uso di luci puntuali di tipo Static, impiegate come una luce ambientale. Queste non influenzano il calcolo delle ombre dinamiche degli oggetti e dei nemici, ma contribuiscono a creare un ambiente più luminoso e facilmente esplorabile, senza aumentare il carico computazionale a runtime.
La Figura 7.7 mostra l'uso delle Static Light (in viola) per una stanza del primo piano, evidenziando come esse permettano di ottenere un'illuminazione maggiore mantenendo prestazioni ottimali.

Figura 7.6: Esempio di configurazione delle luci nel piano terra

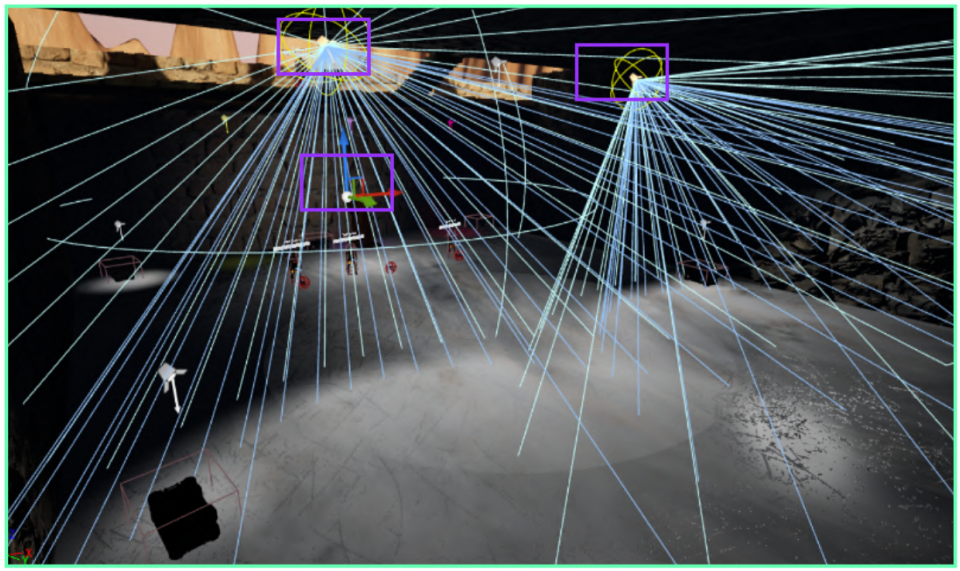

Figura 7.7: Esempio di configurazione delle luci nel piano primo

7.2.4 Utilizzo dei Light Channels

La progettazione del posizionamento delle luci, descritta nella sezione precedente, ha contribuito in modo significativo alla riduzione del carico computazionale. Tuttavia, non è sempre stato possibile evitare completamente la sovrapposizione tra più luci, soprattutto nel caso di sorgenti luminose collocate su piani differenti dell'edificio ma fisicamente vicine nello spazio.
Per affrontare questo problema è stato utilizzato il sistema dei **Light Channels** di Unreal Engine [38], che consente di ridurre il numero di fonti luminose che agiscono contemporaneamente su un singolo oggetto, contribuendo così a diminuire il costo computazionale a runtime.
In Unreal Engine sono disponibili tre canali di luce, denominati 0, 1 e 2. Un oggetto viene illuminato da una luce solo se entrambi condividono almeno un canale attivo. Di conseguenza, un oggetto può ricevere illuminazione da nessun canale, da uno solo oppure da tutti e tre. Per impostazione predefinita, sia gli oggetti sia le luci sono assegnati al canale 0.

Nella versione VR di *Gavunia*, la struttura dell'ambiente ha permesso di sfruttare i Light Channels in modo efficace. **A ciascun piano dell'edificio è stato assegnato un canale specifico**, in modo da separare logicamente le luci dei diversi livelli. In questo modo, luci appartenenti a piani differenti non contribuiscono reciprocamente all'illuminazione degli oggetti, anche quando risultano spazialmente vicine. Di conseguenza, per garantire il corretto funzionamento del sistema, gli oggetti della scena sono stati configurati in modo da ricevere illuminazione esclusivamente dal canale corrispondente al piano su cui sono posizionati.

La Figura 7.8 mostra un esempio esemplificativo dell'efficacia dei Light Channels nella gestione delle sovrapposizioni luminose.
Nel Caso A, entrambe le luci sono assegnate al canale di default 0. Di conseguenza, la zona di intersezione delle loro aree di influenza comporta un aumento del costo computazionale, evidenziato dal colore verde scuro.
Nel Caso B, invece, le luci sono assegnate a canali differenti e pertanto la loro intersezione non viene considerata dal motore. Il pavimento della scena, assegnato al canale di default 0, è illuminato esclusivamente dalla luce appartenente allo stesso canale, mentre la luce di destra, associata a un canale differente, non ha alcun effetto su di esso.

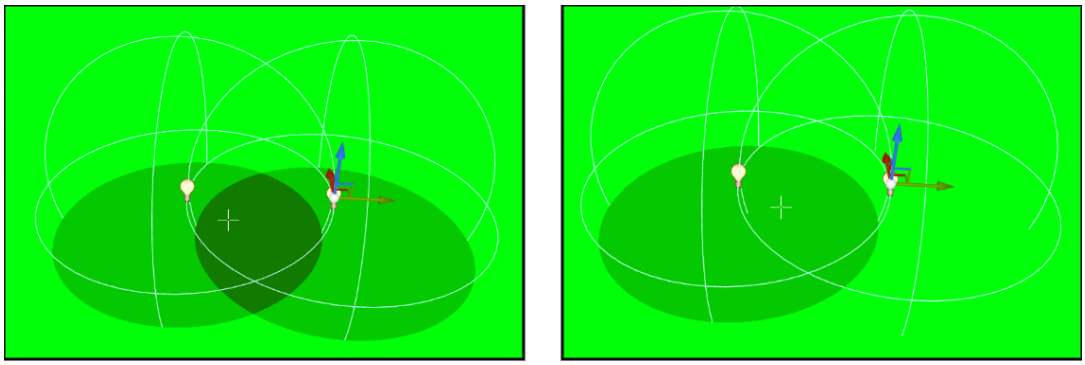

Figura 7.8: Esempio d'uso dei Light Channels

7.3 Instanced Stereo

Nel rendering per realtà virtuale è necessario generare due immagini distinte della scena, una per ciascuna lente del visore. Le due viste sono leggermente differenti, poiché simulano la distanza interpupillare dell'osservatore, consentendo la percezione della profondità attraverso la stereoscopia. Questo comporta un aumento significativo del carico computazionale, poiché gran parte delle operazioni di rendering deve essere eseguita due volte.
Per ridurre tale overhead, Unreal Engine introduce la tecnica di **Instanced Stereo Rendering** [39], progettata per ottimizzare il rendering stereoscopico nelle applicazioni VR.

In un approccio tradizionale, il motore grafico esegue due passaggi di rendering separati per la scena: uno per l'occhio sinistro e uno per l'occhio destro. Ciò implica che molte operazioni sulla CPU, come la preparazione delle draw call e la gestione degli oggetti visibili nella scena, vengano duplicate.
L'Instanced Stereo Rendering consente invece di eseguire gran parte di queste operazioni una sola volta. In particolare, **la scena viene inviata alla GPU tramite un singolo insieme di draw call, mentre la generazione delle due viste stereoscopiche avviene utilizzando meccanismi di instancing a livello di GPU**.
In questo approccio, la CPU prepara una sola volta i dati necessari al rendering, come la geometria degli oggetti e le relative draw call. La GPU utilizza poi questi stessi dati per generare due istanze della scena, ciascuna associata alla matrice di vista corrispondente all'occhio sinistro o destro. In questo modo, entrambe le immagini vengono prodotte nello stesso passaggio di rendering, condividendo gran parte dei dati geometrici e delle operazioni di preparazione della scena.

La Figura 7.9 mostra la differenza tra il rendering Standard Stereo e Instanced Stereo, evidenziando come nel primo caso vengano eseguiti due passaggi distinti, mentre nel secondo entrambi gli occhi vengono renderizzati simultaneamente in un unico passaggio. In particolare, le immagini riportate rappresentano un preciso istante del processo di rendering e sono utilizzate per evidenziare il diverso comportamento delle due tecniche. Nel caso dello Standard Stereo, l'immagine destinata all'occhio destro viene generata soltanto dopo il completamento del rendering dell'occhio sinistro. Al contrario, con Instanced Stereo le due immagini vengono prodotte in parallelo: come si può osservare, nella scena di sinistra l'immagine non è ancora completamente renderizzata, poiché il processo di generazione dei due punti di vista procede simultaneamente.

Questo approccio riduce significativamente il carico sulla CPU, in particolare diminuendo il numero di draw call che devono essere preparate per ciascun frame, e consente di migliorare l'efficienza complessiva della pipeline di rendering.
Il risultato è una riduzione del tempo necessario per il rendering di ogni frame e, di conseguenza, una maggiore probabilità di mantenere il framerate richiesto dal visore.

Per i vantaggi appena discussi, in *Gavunia VR* è stata abilitata questa modalità di rendering. Essa contribuisce, insieme alle tecniche di ottimizzazione descritte nelle sezioni precedenti, al mantenimento di un framerate stabile e adeguato ai requisiti dell'esperienza in realtà virtuale.

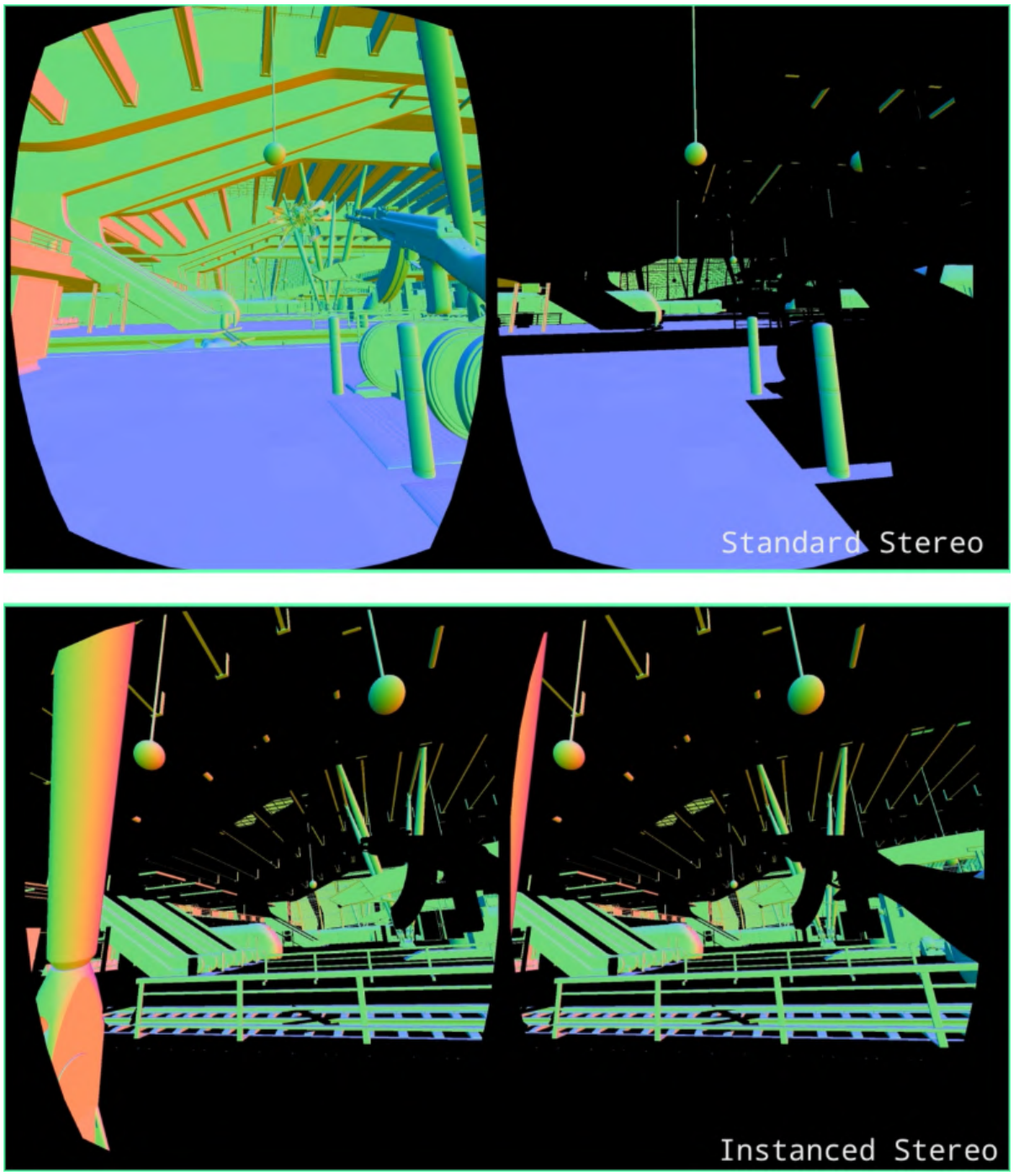

Figura 7.9: Confronto tra Standard Stereo e Instanced Stereo (tratto da "Unreal Engine Documentation" [39])

7.4 Da Nanite a LOD

Il passaggio al Forward Shading ha comportato un notevole aumento delle prestazioni, permettendo di mantenere framerate elevati, ma ha imposto alcune rinunce significative: in particolare, l'impossibilità di utilizzare Lumen per l'illuminazione dinamica (sostituito dal baking delle luci descritto nella Sezione 7.2) e Nanite per la gestione automatica dei dettagli della geometria.

Nella versione desktop di *Gavunia*, come illustrato in Sezione 5.2.2, tutti gli oggetti della scena sono renderizzati tramite Nanite, che garantisce un controllo automatico e moderno dei dettagli geometrici senza la necessità di creare manualmente livelli di dettaglio.

Con la transizione alla versione VR, la **pipeline Forward ha reso necessario il passaggio alla gestione manuale dei LOD, poiché Nanite non è compatibile con questo tipo di rendering**.
A supporto di questo processo, Unreal Engine mette a disposizione strumenti integrati per la generazione automatica dei LOD, basati su preset configurabili che definiscono il grado di semplificazione geometrica. Nel caso di *Gavunia VR* è stato utilizzato il preset Large Props, progettato per oggetti di dimensioni medio-grandi e caratterizzati da una complessità geometrica moderata.

Questo preset genera automaticamente una serie di livelli di dettaglio in cui il numero di poligoni viene progressivamente ridotto in funzione della dimensione apparente dell'oggetto sullo schermo. In Unreal Engine, infatti, la selezione del LOD non è determinata direttamente dalla distanza della camera, ma dal parametro Screen Size, che rappresenta la percentuale dello schermo occupata dalla mesh. Quando l'oggetto occupa una porzione significativa dell'immagine viene utilizzato un livello di dettaglio più elevato, mentre al diminuire della sua dimensione sullo schermo vengono selezionate versioni progressivamente più semplificate.
La Figura 7.10 mostra un esempio di configurazione dei livelli di dettaglio. Nell'immagine a sinistra è visibile la mesh al livello di dettaglio massimo (LOD 0), corrispondente alla versione originale non semplificata. Nell'immagine a destra è invece mostrata la mesh al livello di dettaglio minimo (LOD 3), che rappresenta la versione più semplificata, con circa il 12.5% della complessità geometrica originale, utilizzata quando l'oggetto si trova a grande distanza dall'osservatore.

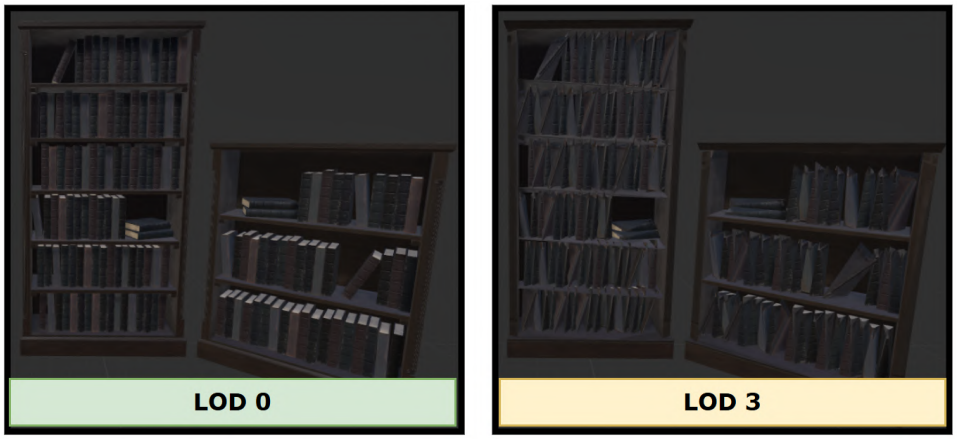

Figura 7.10: Esempio di configurazione dei LOD

Conclusioni

Questo lavoro ha dimostrato come lo sviluppo di giochi in VR offra opportunità significativamente più ampie rispetto alle applicazioni desktop, in particolare nell'ambito delle interazioni uomo-macchina, grazie alla natura immersiva della tecnologia.
Le scelte di Game Design assumono un ruolo cruciale, poiché influenzano direttamente la fruibilità e il divertimento dell'esperienza. In particolare, la progettazione delle meccaniche di movimento deve essere attentamente valutata per ridurre al minimo i fenomeni di motion sickness.

Un altro aspetto cruciale nello sviluppo VR riguarda le prestazioni, che richiedono un'analisi più approfondita e ottimizzazioni mirate rispetto alla controparte desktop. L'ambiente VR impone infatti vincoli più stringenti in termini di frame rate, latenza e stabilità del rendering, poiché cali prestazionali possono compromettere non solo la qualità visiva, ma anche il comfort e il senso di presenza dell'utente. Inoltre, la necessità di renderizzare la scena due volte, una per occhio, comporta costi computazionali maggiori rispetto al desktop.

In sintesi, *Gavunia VR* ha rappresentato una sfida su tre fronti principali:

1. **Game Design e HCI**: è stata progettata un'avventura breve ma variegata, con scenari differenti e attività di ricerca, combattimento e interazioni con oggetti. Le interazioni seguono i principi consolidati di HCI, garantendo robustezza rispetto all'errore umano e una corrispondenza naturale tra gesti fisici e azioni virtuali.

2. **Mitigazione della motion sickness**: sono state adottate tecniche consolidate come Teleportation e Snap Turn, integrate con soluzioni innovative come la Tunnel Vision, ispirata a Google Earth VR.

3. **Bilanciamento tra qualità grafica e prestazioni**: il passaggio dal Deferred Rendering al Forward Rendering ha richiesto una revisione del sistema di illuminazione, sostituendo Lumen con il baking delle luci, e una riprogettazione generale delle luci per garantire un frame rate stabile.

Nel complesso, lo sviluppo di *Gavunia VR* ha quindi permesso di analizzare in modo pratico le principali sfide legate alla progettazione di applicazioni videoludiche in realtà virtuale, evidenziando come aspetti di game design, interazione uomo-macchina e ottimizzazione del rendering siano strettamente interconnessi.

Gavunia VR presenta comunque diversi possibili sviluppi futuri, sia nell'ambito del game design e dell'interazione uomo-macchina, sia dal punto di vista delle prestazioni e delle tecniche di rendering.

Dal punto di vista del game design, un'estensione naturale del progetto consisterebbe

nell'implementazione della parte ancora non mostrata nel mondo di *Gavunia*, caratterizzata da foreste lussureggianti (Sezione 5.4.1), dove vivono gli abitanti della società. La scena presentata nella demo attuale rappresenta infatti soltanto una regione remota del mondo di gioco, concepita come punto di partenza dell'avventura. L'introduzione di nuovi ambienti consentirebbe quindi di ampliare l'esplorazione e di arricchire la varietà delle situazioni di gioco.

Un ulteriore sviluppo riguarda la missione finale di recupero del cristallo. In particolare, potrebbe essere introdotto uno scontro con un boss più articolato, dotato di diverse strategie di combattimento e non limitato all'utilizzo di armi da fuoco. Questo scenario aprirebbe inoltre la possibilità di sperimentare tecniche di modellazione più avanzate, come la simulazione realistica di abiti, aumentando il livello di dettaglio visivo dei personaggi.

Dal punto di vista dell'HCI, l'introduzione di nuove armi e oggetti manipolabili con entrambe le mani incrementerebbe ulteriormente il senso di immersione e la complessità delle interazioni. Inoltre, alcune azioni potrebbero essere rese più realistiche vincolando i movimenti lungo traiettorie o assi specifici; ad esempio, l'inserimento di una chiave all'interno di una serratura potrebbe seguire un percorso guidato, migliorando la coerenza tra i gesti fisici dell'utente e le corrispondenti azioni nel mondo virtuale.

Infine, per quanto riguarda le prestazioni, l'introduzione di scenari più complessi e ricchi di dettagli renderebbe opportuno esplorare tecniche di ottimizzazione più avanzate.
In particolare, sarebbe interessante sperimentare soluzioni basate su approcci di intelligenza artificiale, come alcune tecnologie sviluppate da Nvidia e disponibili come plugin per Unreal Engine [40]. Tali strumenti consentono di migliorare sia la qualità grafica sia le prestazioni attraverso tecniche di upscaling e ottimizzazione basate su reti neurali.
Sebbene queste tecnologie siano attualmente impiegate principalmente in contesti desktop, la loro applicazione in ambienti di realtà virtuale potrebbe aprire nuove prospettive di ricerca e sperimentazione.

Bibliografia

[1] Ralf Doerner, Wolfgang Broll, Paul Grimm e Bernhard Jung. *Virtual and Augmented Reality (VR/AR)*. Springer Cham, 2022.

[2] Meta Horizon. *Save GPU with Eye Tracked Foveated Rendering*. URL: https://developers.meta.com/horizon/blog/save-gpu-with-eye-tracked-foveated-rendering/.

[3] Jesse Schell. *The Art of Game Design: A book of lenses*. CRC press, 2008.

[4] D. Bowman, E. Kruijff, J.J. LaViola e I.P. Poupyrev. *3D User Interfaces: Theory and Practice*. Pearson Education, 2004. ISBN: 9780133390568. URL: https://books.google.de/books?id=JYzmCkf7yNcC.

[5] Przemysław Krompiec e Kyoungju Park. "Enhanced Player Interaction Using Motion Controllers for First-Person Shooting Games in Virtual Reality". In: *IEEE Access* 7 (2019), pp. 124548–124557. DOI: 10.1109/ACCESS.2019.2937937.

[6] Michel Treisman. "Motion Sickness: An Evolutionary Hypothesis". In: *Science* 197.4302 (1977), pp. 493–495. DOI: 10.1126/science.301659. eprint: https://www.science.org/doi/pdf/10.1126/science.301659. URL: https://www.science.org/doi/abs/10.1126/science.301659.

[7] Gary E. Riccio e Thomas A. Stoffregen. "An ecological Theory of Motion Sickness and Postural Instability". In: *Ecological Psychology* 3.3 (1991), pp. 195–240. DOI: 10.1207/s15326969eco0303_2. eprint: https://doi.org/10.1207/s15326969eco0303_2. URL: https://doi.org/10.1207/s15326969eco0303_2.

[8] Diego Monteiro, Hao Chen, Hai-Ning Liang, Huawei Tu e Henry Dub. "Evaluating Performance and Gameplay of Virtual Reality Sickness Techniques in a First-Person Shooter Game". In: *2021 IEEE Conference on Games (CoG)*. 2021, pp. 1–8. DOI: 10.1109/CoG52621.2021.9619145.

[9] Ruding Lou. "Geometry deformation for reducing cybersickness in VR". In: *Journées Françaises d'Informatique Graphique et de Réalité Virtuelle*. France: Recueil des publications Journées Françaises d'Informatique Graphique et de Réalité Virtuelle, nov. 2019.

[10] Thomas W. Sederberg e Scott R. Parry. "Free-form deformation of solid geometric models". In: *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '86. New York, NY, USA: Association for Computing Machinery, 1986, pp. 151–160. ISBN: 0897911962. DOI: 10.1145/15922.15903. URL: https://doi.org/10.1145/15922.15903.

[11] Adam G. *EarthVR*. Ago. 2017. URL: https://adamg.io/earthvr.

[12] Unreal Engine Documentation. *Landscape Quick Start Guide*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/landscape-quick-start-guide-in-unreal-engine.

[13] Unreal Engine Documentation. *Modeling Mode Overview*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/modeling-mode-in-unreal-engine.

[14] Unreal Engine Documentation. *Nanite Virtualized Geometry Overview*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/nanite-virtualized-geometry-in-unreal-engine.

[15] Unreal Engine Documentation. *Physics*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/physics-in-unreal-engine.

[16] Unreal Engine Documentation. *Lumen Global Illumination and Reflections*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/lumen-global-illumination-and-reflections-in-unreal-engine.

[17] Epic Education. *Introduction to AI with Blueprints - AI Perception Theory*. Set. 2022. URL: https://dev.epicgames.com/community/learning/courses/67R/unreal-engine-introduction-to-ai-with-blueprints/586p/ai-perception-theory.

[18] Epic Education. *Introduction to AI with Blueprints - Behavior Tree Theory*. Set. 2022. URL: https://dev.epicgames.com/community/learning/courses/67R/unreal-engine-introduction-to-ai-with-blueprints/qzZ2/unreal-engine-behavior-tree-theory.

[19] Unreal Engine Documentation. *Environment Query System Overview*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/environment-query-system-overview-in-unreal-engine.

[20] Epic Education. *Introduction to AI with Blueprints - Navigation Theory*. Set. 2022. URL: https://dev.epicgames.com/community/learning/courses/67R/unreal-engine-introduction-to-ai-with-blueprints/DYXe/navigation-theory.

[21] Ali Elzoheiry. *Understanding Components in Unreal Engine - UE5 Explained*. Ago. 2024. URL: https://www.youtube.com/watch?v=xo0sbSeWKe4.

[22] Unreal Engine Documentation. *Animation Blueprints*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/animation-blueprints-in-unreal-engine.

[23] Unreal Engine Documentation. *Animation Montage*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/animation-montage-in-unreal-engine.

[24] Unreal Engine Documentation. *Timelines*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/timelines-in-unreal-engine.

[25] Unreal Engine Documentation. *Scalability Reference*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/scalability-reference-for-unreal-engine.

[26] Khronos Group. *OpenXR*. URL: https://www.khronos.org/openxr/.

[27] Unreal Engine Documentation. *Traces Overview*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/traces-in-unreal-engine---overview.

[28] Quinn Kuslich. *VR Procedural Grip Poses in Unreal 5.4 — How to Create Realistic Grip Animations for Any Object*. Lug. 2024. URL: https://youtu.be/m3xanGZ5ETI?si=Mt43bXYhiqNPXgdy.

[29] Unreal Engine Documentation. *Skeletal Mesh Sockets*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/skeletal-mesh-sockets-in-unreal-engine.

[30] Quinn Kuslich. *Advanced VR Hand Physics for in Unreal Engine 5 — Full Tutorial*. Gen. 2025. URL: https://youtu.be/ZSNKRrX9UkE?si=1UnxZcTyBOzK5hW5.

[31] Unreal Engine Documentation. *Physics Constraint Component User Guide*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/physics-constraint-component-user-guide-in-unreal-engine.

[32] Unreal Engine Documentation. *Physics Asset Editor*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/physics-asset-editor-in-unreal-engine.

[33] Unreal Engine Documentation. *Widget Components*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/widget-components-in-unreal-engine.

[34] Unreal Engine Documentation. *Widget Interaction Component*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/umg-widget-interaction-components-in-unreal-engine.

[35] Unreal Gems. *Localization In-Depth*. Nov. 2022. URL: https://dev.epicgames.com/community/learning/tutorials/zwPJ/unreal-engine-localization-in-depth.

[36] Unreal Engine Documentation. *XR Best Practices*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/xr-best-practices-in-unreal-engine.

[37] ICVR. *Light Baking and Lightmaps: A Step-by-Step UE Guide*. Nov. 2024. URL: https://dev.epicgames.com/community/learning/tutorials/KPOx/unreal-engine-light-baking-and-lightmaps-a-step-by-step-ue-guide.

[38] Unreal Engine Documentation. *Lighting Channels*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/using-lighting-channels-in-unreal-engine.

[39] Unreal Engine Documentation. *XR Performance Features*. URL: https://dev.epicgames.com/documentation/en-us/unreal-engine/xr-performance-features-in-unreal-engine.

[40] Eric Phan e Ike Nnoli. *Get Started with Neural Rendering Using NVIDIA RTX Kit*. Feb. 2025. URL: https://developer.nvidia.com/blog/get-started-with-neural-rendering-using-nvidia-rtx-kit/.