POLITECNICO DI TORINO

Collegio di Informatica, Cinema e Meccatronica

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

Tesi di Laurea Magistrale

Realtà Virtuale: Studio di Interfacce Utente per Configuratori Custom in ambito Automotive

Implementazione di Interfacce Utente Data Driven e Multi-piattaforma per un Plug-in compatibile con Unreal Engine 4.26



Relatore:

Prof. Andrea Bottino

Candidato:

Matteo Saggiani

Correlatore:

Dott. Francesco Strada

APRILE 2021

RINGRAZIAMENTI

Scrivere questa tesi in un periodo così critico e delicato non è stato affatto facile. Proprio per questo vorrei spendere due parole di ringraziamento nei confronti di tutte le persone che sono state al mio fianco, sia fisicamente che col pensiero, in questi mesi.

Prima di tutto, vorrei ringraziare la mia collega di tesi Dalila che mi ha accompagnato durante questi mesi di duro lavoro, senza la quale non avrei potuto raggiungere questo importante traguardo. Grazie al suo spirito d'iniziativa e la sua innata volontà di mettersi costantemente alla prova si è potuto fin da subito instaurare un ottimo rapporto a livello professionale.

Un altro ringraziamento va ai miei relatori, il prof. Andrea Giuseppe Bottino e il dott. Francesco Strada, per la loro disponibilità e i loro preziosi consigli.

Una dedica speciale a tutto il Team di Dead Pixels per averci permesso di entrare a far parte, anche solo per 6 mesi, della loro numerosa famiglia. Ho avuto modo di conoscere persone altamente professionali e sempre pronte ad aiutarci nei vari momenti d'incertezza e difficoltà. Oltre ad averci guidato durante la fase di progettazione, mi avete trasmesso la passione e l'entusiasmo necessari affinché la tesi prendesse forma giorno dopo giorno. In particolare vorrei esprimere un sentito ringraziamento a Maurizio, Luca detto Mario, Michelangelo detto Luigi e a Giacomo detto Giacomo che hanno voluto condividere con noi le loro numerose conoscenze frutto di una lunga esperienza nel settore.

Un pensiero va sicuramente anche a tutta la mia famiglia e a tutti i miei parenti: è grazie a loro sostegno e al loro incoraggiamento se oggi sono riuscito a scrivere queste righe di ringraziamento. L'affetto che mi hanno sempre dimostrato rende questo traguardo ancora più prezioso e in parte lo dedico proprio a loro. Siete stati e rimarrete sempre la mia più grande fonte d'ispirazione.

Per ultimi ma non meno importanti, i miei più cari amici. Mi hanno sempre sostenuto a distanza, nella buona e nella cattiva sorte, sia durante le fatiche e lo sconforto che nei momenti di gioia e soddisfazione che hanno caratterizzato questo percorso. Non potendo menzionarli uno ad uno, mi limito a dire che avete tutti un posto speciale nel mio cuore. Speriamo di poterci vedere e abbracciare presto.

Grazie di tutto.

Matteo Saggiani,

Torino, Aprile 2021

ABSTRACT

Al giorno d'oggi, la nuova tecnologia della Realtà Virtuale (VR), seppur ancora non pienamente accessibile, si sta avvicinando sempre di più ad una fase di maturità e diffusione di massa. È dunque questa la fase in cui si sta cercando di comprendere e definire gli standard di realizzazione e di interazione, soprattutto per quanto riguarda l'Interfaccia Utente (UI). L'obiettivo di questo studio è dunque quello di analizzare quali caratteristiche deve possedere una UI per VR per essere efficace ed intuitiva, tenendo in considerazione lo stato attuale della Realtà Virtuale, la transizione da una logica di UI bidimensionale a una in tre dimensioni che essa impone ed infine anche il differente approccio che possono avere varie tipologie di utenti con essa.

Per fare ciò abbiamo avviato una collaborazione con **Dead Pixels s.r.l.**, azienda torinese che si occupa di Computer Grafica, VR e AR (Realtà Aumentata), per realizzare un plug-in per Unreal Engine, utile alla progettazione di un Product Configurator che sia modulare, multiutente e multi-piattaforma. Nello specifico noi ci siamo occupati della realizzazione delle UI, sia Desktop che VR, partendo da due differenti presupposti: per la Desktop essere meno invasivi possibile; per la VR, in assenza di standard predefiniti, di sfruttare al meglio lo spazio tridimensionale a disposizione, cercando di riprodurre interazioni riconducibili a quelle quotidiane. Per verificare l'efficacia di quanto realizzato e raggiungere l'obiettivo prepostoci, abbiamo svolto una fase di User Testing qualitativo in cui l'applicazione è stata testata da diversi utenti con differenti esperienze, ai quali è stato chiesto in seguito di compilare un questionario riguardo la User Experience (UX).

Per quanto riguarda la VR è stato chiaro che per la visualizzazione di grandi quantità di dati, relativi alle varianti del prodotto da configurare, è ancora efficace una logica desktop, seppur susciti fascino la loro rappresentazione in tre dimensioni. I tester generalmente riescono a portare a termine i compiti assegnati se prevedono interazioni simili a quelle quotidiane, anche se non in tutti i casi è risultata una scelta efficiente. Hanno infatti infastidito compiti che implicassero troppi passaggi o troppi pulsanti da utilizzare in contemporanea per ottenere il risultato richiesto, anche se veniva rispecchiata un'azione reale.

Si può concludere che nella progettazione di una UI per VR sia opportuno perseguire un approccio di fedeltà a gesti comuni, poiché più intuitivi per l'utente, a condizione che l'artificio meccanico imposto dalla tecnologia per riprodurli non ne pregiudichi la semplicità di esecuzione. In questi casi gli utenti preferiscono una interazione meno naturale ma più semplice e pragmatica.

INDICE

RINGRAZIAMENTI I

ABSTRACT II

INDICE 1

INTRODUZIONE 4

OBIETTIVI 4

IL PROBLEMA 5

SOLUZIONE PROPOSTA 6

METODO SEGUITO 7

1 - STATO DELL'ARTE 10

1.1 - REALTÀ VIRTUALE 10

1.1.1 - HYPE CYCLE 10

1.1.2 - LIMITI DELLA VR 15

1.1.3 - ESPERIENZA PERSONALE CON OCULUS RIFT (2016) 20

1.2 - PRESENZA E MULTIPRESENZA 25

1.2.1 - I SENSI COINVOLTI NELLA VR 25

1.2.2 - COLLABORATIVE VIRTUAL ENVIRONMENTS 28

1.3 - COMPETITOR E ANALISI DI DIFFERENTI APPROCCI 31

1.3.1 - FEATURES COMUNI 31

1.3.2 - APPROCCI DIFFERENTI 31

2 - LOGICA DI FUNZIONAMENTO DELL'APPLICATIVO E MEZZI UTILIZZATI 35

2.1 - UNREAL ENGINE E STRUMENTI UTILIZZATI 37

2.1.1 - COLLAB VIEWER TEMPLATE 37

2.1.2 - WIDGET BLUEPRINT E EDITOR UTILITY WIDGET 39

2.1.3 - VARIANT MANAGER 40

2.2 - LOGICA DATA DRIVEN 42

2.2.1 DATA TABLE 42

2.2.2 DATA ASSET 44

2.2.3 CONFRONTO E RAGIONI DI UTILIZZO 46

2.3 - EDITOR UTILITY WIDGET E PERSONALIZZAZIONE DELL'INTERFACCIA 48

2.3.1 INTERFACCIA INTERATTIVA CON EDITOR UTILITY WIDGET 48

2.3.2 LETTURA, SCRITTURA E SALVATAGGIO DEI DATI SU DATA ASSET 54

2.4 - VARIANT MANAGER E VARIANT TEMPLATE 57
2.4.1 - PROCESSO DI CREAZIONE DELLA DATA TABLE 58
2.4.2 - STRUTTURA DELLA DATATABLE 60
2.4.3 - PAYLOAD STRUCT 64
2.4.4 - CUSTOMIZZAZIONE DEL PAYLOAD DA EUW 66
2.5 - BLUEPRINT INTERFACE E HUD 70

3 - INTERFACCIA UTENTE DESKTOP 74
3.1 - FASE DI IDEAZIONE 74
3.1.1 - REFERENCE E DIFFERENTI APPROCCI 74
3.1.2 - MOCKUP E RIUTILIZZO DEGLI SPAZI 79
3.2 - MENU CONFIGURATOR 84
3.2.1 - LAYOUT GRAFICO DEL WIDGET ASSET BUTTON 84
3.2.2 - RECUPERO DEI DATI E CREAZIONE DEI RELATIVI WIDGET 84
3.2.3 - CARICAMENTO ASINCRONO DELLE ICONE 87
3.2.4 - LEAFS BOX E AGGIORNAMENTO DELLE CATEGORIE 88
3.3 - MENU TOOLS 92
3.3.1 - WORKFLOW DI RECUPERO DELLE INFORMAZIONI 94
3.3.2 - POPOLAMENTO DEL MENU 99
3.3.3 - TOOL ENVIRONMENT 102
3.4 - MENU SETTINGS 104
3.5 - UTILIZZO DELLE RISORSE SALVATE SU DATA ASSET 107

4 - INTERFACCIA UTENTE PER REALTÀ VIRTUALE 111
4.1 - FASE DI IDEAZIONE 111
4.1.1 - REFERENCE E DIFFERENTI APPROCCI 111
4.1.2 - BEST & WORST 117
4.1.3 - IDEE, MOCKUP E PRIME IMPLEMENTAZIONI 119
4.1.4 - MAIN MENU E UI MODULARE 125
4.2 - STRUTTURA GENERALE DELL'INTERFACCIA UTENTE VR 130
4.2.1 I MENU E LE INTERAZIONI DI BASE 130
4.2.2 PASSAGGIO IN VR E CREAZIONE DEI MENU 137
4.2.3 SELECT E GRAB INTERFACE 140
4.3 - MENU CONFIGURATOR 144
4.3.1 - STRUTTURA E FUNZIONAMENTO DEL TABLET 145
4.3.2 - TABLET WIDGET 146
4.3.3 - ASSET DASHBOARD 150

4.4 - MENU SETTINGS 156
4.5 - MENU TOOLS 157

5 - TEST 162
5.1 - PERFORMANCE TEST CON DIVERSE TIPOLOGIE DI PRODOTTO 162
5.2 - COSTRUZIONE DEI TEST 166
5.2.1 - TARGET E TEST QUALITATIVI 166
5.2.2 - QUESTIONARIO PRELIMINARE 167
5.2.3 - TASK 169
5.2.4 - QUESTIONARIO POST-ESPERIENZA 170
5.3 - ANALISI DEI DATI 172
5.3.1 - RISULTATI DEL QUESTIONARIO PRELIMINARE 172
5.3.2 - ANALISI DELLE TASK 178
5.3.3 - RISULTATI DEL QUESTIONARIO POST ESPERIENZA 183

CONCLUSIONI 193
RISULTATI OTTENUTI 193
SVILUPPI FUTURI 196

BIBLIOGRAFIA E SITOGRAFIA 198
APPENDICE A - QUESTIONARIO PRELIMINARE 1
APPENDICE B - DEFINIZIONE TASK, SCENARIO E TEMPI DI ESECUZIONE 1
APPENDICE C - QUESTIONARIO POST ESPERIENZA 1

INTRODUZIONE

Tale proposta di tesi ci è stata avanzata dalla società Dead Pixels s.r.l, uno studio torinese specializzato nella realizzazione di contenuti nell'ambito della Computer Grafica, Realtà Virtuale e Realtà Aumentata. Data l'ingente mole di studio e lavoro che interessano questo progetto, il lavoro è stato portato avanti in parallelo dall'azienda e da noi due tesisti, Matteo Saggiani e Dalila Pasquali, motivo per cui in questo documento verrà utilizzata la prima persona plurale.

OBIETTIVI

Con questo progetto di tesi ci prefiggiamo di realizzare uno studio esaustivo delle possibili interfacce Desktop e VR Data Driven che possono essere utilizzate all'interno di un **Plugin** compatibile con il motore di gioco **Unreal Engine,** volto a supportare la progettazione di un Product Configurator multipiattaforma in ambito Automotive e che sia utilizzabile dagli stessi sviluppatori per realizzare, all'interno di un ambiente virtuale, esperienze usufruibili in presenza remota in real time da più utenti.

Il plugin deve poter funzionare su oggetti e prodotti differenti, mantenendo intatte le sue funzionalità. Dovrà dunque essere **modulare** in modo che gli sviluppatori e i designer possano inserire le loro risorse all'interno del nostro plugin e usare i tool messi a disposizione per interagire con esse, a prescindere dal tipo di modello 3D inserito . Il plugin è pensato per essere **multipiattaforma**, dunque deve garantire **compatibilità** tra differenti hardware e interfacce. Inoltre un plugin multi-utente impone ulteriori sfide in relazione alla **User Experience**, per cui intendiamo approfondire lo studio sulle UI Desktop e in particolare VR, poiché quest'ultima risulta ancora poco studiata e sperimentata, quasi completamente priva di standard definiti.

Ci limiteremo dunque allo studio delle Interfacce Utente sopracitate e solo in seguito verranno poi progettate e studiate una versione mobile e una corretta replicazione dei dati per rendere l'applicazione utilizzabile da più utenti in contemporanea, seppur già in questa versione sarà presente una base di gestione delle sessioni.

IL PROBLEMA

Al giorno d'oggi nel mondo lavorativo entrano in conflitto due realtà: la necessità di **lavorare in team** per mettere insieme differenti conoscenze e differenti formazioni specifiche volte ad un unico obiettivo entra in contrasto con un **sistema aziendale basato sulla dislocazione** delle varie sedi, a volte distanti diverse ore di viaggio. A questo, nell'ultimo anno si aggiunge lo stato di emergenza sanitaria dovuto alla pandemia di Covid-19, che impone ulteriore distanziamento sociale in ogni contesto.
La necessità di non perdere tempo e denaro in spostamenti, che potrebbero essere meglio investiti, o di far fronte a sempre più persone che lavorano in smart working a causa del distanziamento sociale, impone di trovare nuovi mezzi per **comunicare efficacemente**, il più possibile alla pari dei tradizionali incontri face to face in termini di efficacia.
Seppur le **Audio Call** e **Video Conference** siano un ottimo mezzo per una comunicazione immediata in real time, non sempre sono sufficienti ed efficaci come un incontro di persona, specie quando si tratta di team che dovrebbero lavorare insieme per ore in un approccio di ***design thinking*** che viene indubbiamente limitato dalla distanza.

E' ormai risaputo che la comunicazione è data per il 7% dal linguaggio verbale, il 38% dal para-verbale e il 55% dal linguaggio non verbale, motivo per cui le Audio Call non sono sempre sufficienti o efficaci nella comunicazione, poiché non permettono di cogliere il linguaggio del corpo o le espressioni della persona che sta parlando, non trasmettendo così i dettagli che l'interlocutore magari sta cercando di trasmettere. Pertanto, le Audio Call possono essere utilizzate principalmente per brevi aggiornamenti, ma non sono particolarmente adatte per discussioni lunghe né tantomeno per scambiarsi risorse.
Nelle Video Conference viene superata parte di questa limitazione, ma non sempre è possibile scambiarsi certi tipi di contenuti, in quanto potrebbero esserci restrizioni sulla dimensione dei file da trasferire dell'applicazione usata per la Video Call, oltre che magari lunghi tempi di trasferimento. Inoltre i partecipanti alla sessione non possono lavorare insieme in tempo reale su un medesimo prodotto ed è quasi impossibile ricreare una situazione che invece si avrebbe in una normale riunione in presenza.

Queste problematiche potrebbero essere risolte con l'utilizzo della VR e in particolare dai **Virtual Meeting**. Questi possono essere sfruttati quando il meeting deve essere condotto per lo scambio di informazioni tecniche, per risolvere problemi complessi, fare brainstorming per

idee innovative, soprattutto in ottica di un approccio di design thinking, e anche per la formazione dei dipendenti.
I Virtual Meeting non solo permettono di risparmiare tempo e denaro, ma forniscono anche altri vantaggi rispetto alle Video Conference e Audio Call: la possibilità di **interagire in uno stesso ambiente virtuale**, potendo lavorare su un unico oggetto o modello seppur a distanza; lo scambio immediato di file pesanti e di difficile gestione; l'opportunità di condurre **Training Session** per la formazione di gruppo dei dipendenti in un contesto simulato e quindi più sicuro, oltre ad offrire un monitoraggio migliore dei risultati. A livello business la VR offre la possibilità di realizzare **tour virtuali** per i clienti, ma anche **virtual showrooms** per mostrare in anticipo i propri prodotti, che siano in fase di progettazione o già pronti per essere lanciati sul mercato.

SOLUZIONE PROPOSTA

Essendo consci della vastità di questo problema non intendiamo peccare di supponenza dicendo di volerlo risolvere nella sua totalità, tuttavia ci siamo preposti di studiare alcune soluzioni a livello di interfaccia utente, che potranno, una volta aggiunta la corretta replicazione dei dati che non sarà oggetto di questo studio, agevolare la realizzazione di un applicativo per poter lavorare simultaneamente sulla configurazione di uno stesso oggetto.

Nei capitoli che seguiranno verrà spiegato nel dettaglio ogni step compiuto ed ogni mezzo utilizzato per realizzare un prototipo di tale applicativo con le soluzioni da noi proposte. Si eseguirà anche una fase di user testing per verificare l'efficacia di quanto implementato, i cui risultati verranno esposti nei capitoli finali di questo documento di tesi.
Come base di partenza è stato utilizzato un template messo a disposizione da Unreal Engine 4, il Collab Viewer. Di esso è stata mantenuta tutta la struttura di base, la logica di gestione delle sessioni e alcuni tool, la cui logica di funzionamento è descritta all'interno del programma, ma non le UI, che sono state completamente riprogettate alla luce delle necessità derivanti dall'implementazione di un configuratore.
Sfruttando inoltre un plugin sviluppato da Dead Pixels, volto a migliorare tutto il sistema di gestione delle informazioni di configurazione di un prodotto, il nostro studio si è incentrato su come rendere comprensibili e facilmente utilizzabili a livello di interfaccia tali informazioni, in modo che si potesse sfruttare al meglio le potenzialità del configuratore, sia nel caso

che l'utente finale decida di utilizzare un PC, ma anche nel caso volesse provare l'esperienza con un HMD per la Realtà Virtuale.
Oltre a ciò, si è deciso di semplificare il processo di creazione del configuratore e di personalizzazione dell'interfaccia utente, risparmiando il tedioso compito allo sviluppatore di riscrivere ogni volta l'intera logica di creazione delle UI e facilitando ad un designer l'approccio al progetto, mettendolo nelle condizioni di scegliere lo stile che più gli aggrada per le interfacce Desktop e VR, senza la necessità di conoscerne per forza le logiche interne di programmazione, oltre che a ricevere dei feedback immediati del risultato finale. In questo modo la costruzione di un configuratore per un preciso cliente si accorcerebbe da settimane o mesi di lavoro a qualche giorno.
Focalizzeremo la nostra attenzione sulla progettazione di adeguate UI, sia per Desktop che per VR, dedicando maggiore studio e attenzione alle interazioni e feedback in Realtà Virtuale, i cui standard di progettazione non sono ancora stati ben definiti.

METODO SEGUITO

Per raggiungere l'obiettivo prepostoci, seguendo i punti chiave descritti, intendiamo realizzare una interfaccia utente **Data Driven**, quindi con una determinata struttura logica, definendo per tale struttura dei valori di default come "placeholder" che possano comunque essere modificati in qualunque momento, attraverso appositi Editor, sia da uno sviluppatore che da un designer, rendendo così il plugin fortemente **modulare**. Vorremmo inoltre porre una particolare attenzione alla **User Experience** (UX): lato software rendendo l'interazione di quanto messo a disposizione il più possibile intuitiva, che sia il configuratore o che siano i tool per agire sul prodotto e l'ambiente circostante; lato hardware offrendo una immediata fruizione di tale esperienza, sia che avvenga da Desktop o da VR, rendendo di fatto **compatibili** le due tipologie di UI e permettendo all'utente di passare rapidamente da una all'altra nel caso ne avesse esigenza, oppure di interagire con utenti in VR anche da Desktop e viceversa.
Per seguire una logica *Data Driven* si sfrutteranno strutture dati offerte da Unreal Engine chiamate **Data Asset** e **Data Table**, delle quali in seguito verranno descritte le proprietà e il funzionamento. Il Product Configurator verrà costruito tramite **Variant Manager**, altro strumento di Unreal Engine, ampliato dal plugin realizzato da Dead Pixels, anch'esso ampiamente descritto nei capitoli successivi, che memorizza i propri dati su una *Data Table*.

Per quanto riguarda la personalizzazione grafica delle UI invece intendiamo sfruttare un **Editor Utility Widget**, costruendo un apposito Editor nel quale il designer potrà inserire i dati di suo interesse, che verranno poi scritti su *Data Asset* e da lì reperiti ove necessario. I Tool messi a disposizione saranno invece, almeno per questa versione, non modificabili, sfruttando alcuni di quelli già presenti nel Collab Viewer, come quello che offre la possibilità di scrivere o disegnare per prendere note, quello per prendere misurazioni, il tool per rendere trasparenti gli oggetti dando modo di potervi guardare attraverso, ma anche quello che permette di spostare il prodotto o parti di esso e così via. Ad essi intendiamo aggiungere la possibilità di cambiare ambientazione o luci all'ambiente, mettendo a disposizione dello sviluppatore e dell'utente la possibilità di vedere il proprio prodotto in contesti e luci differenti.

In merito alla struttura delle **interfacce utente** è chiaro che adotteremo due approcci diversi per la UI Desktop e per quella VR.
Per quanto riguarda l'**interfaccia Desktop** è chiaro che richiederà uno studio meno approfondito rispetto a quella VR, essendo ormai da anni definiti tutti gli standard e le convenzioni e avendo noi molte più informazioni per costruire una struttura efficace, derivanti sia dagli studi precedentemente condotti che da una inevitabile e consolidata esperienza personale. Quindi il nostro studio si focalizzerà sul trovare il miglior compromesso tra una interazione intuitiva, con i feedback necessari per la comprensione del suo utilizzo, e una UI non invasiva, lasciando su schermo più spazio possibile alla visualizzazione del prodotto da configurare e all'utilizzo dei Tool su di esso.
La sfida più ardua è indubbiamente lo studio dell'**interfaccia VR**, sia per mancanza di standard predefiniti, sia per la minore esperienza personale in questo contesto, motivo per cui gran parte dello studio preliminare sullo stato dell'arte è incentrato sulla Realtà Virtuale.
Nel capitolo dedicato a questa parte dello studio verranno descritti i differenti approcci presi in considerazione.
Primo tra questi, l'utilizzo di una interfaccia "a orologio" posta all'altezza del dorso della mano, che permetta l'accesso alle informazioni in modo efficace e immediato senza essere invasiva, seguendo dunque la stessa logica utilizzata per l'interfaccia desktop, seppur mostrandola sotto un aspetto differente, più adatta alla maggiore libertà di movimenti della VR.
Una soluzione differente prevede l'utilizzo di una toolbelt che offra un'interazione tridimensionale sia con il configuratore che con i tool, dove questi, per non risultare invasivi, vengono entrambi disposti all'altezza della vita dell'utente. Con questo approccio la toolbelt dovrà

essere aggiornata ad ogni azione compiuta dall'utente sia che essa si rivolga al configuratore sia che coinvolga i tool.
Infine si è pensato di progettare una interfaccia ibrida tra le due appena descritte, che sfrutti sia l'interfaccia ad orologio per la visualizzazione del menu principale e l'accesso alle diverse sezioni, sia delle interfacce tridimensionali, tra cui la toolbelt, che gestiscano l'interazione con il configuratore e con i vari tool a disposizione.

Implementate le interfacce e scelta una delle soluzioni sopra descritte per la VR, si eseguirà una fase di **User Testing**, sia per valutarne l'effettiva efficacia considerando i possibili problemi o errori di valutazione, che per studiare quali possano essere le modifiche da apportare in ottica di una migliore **User Experience**.

1 - STATO DELL'ARTE

1.1 - REALTÀ VIRTUALE

Con il termine **Realtà Virtuale** (o VR dall'inglese *Virtual Reality*), si intende la **simulazione di ambienti e situazioni reali o verosimili attraverso l'utilizzo di software e computer** che supportino interfacce hardware appositamente progettate. Si tratta, per definizione, di ambienti simulati e interattivi che coinvolgono più sensi, in modo da poter generare nell'utente un **senso di immersione e presenza nell'ambiente fittizio** come se si trovasse in un ambiente reale[1].

Al giorno d'oggi sono principalmente tre i sensi coinvolti in Realtà Virtuale: vista e udito, in "eredità" dal mondo 2D e tatto. Olfatto e gusto al momento non sono considerati più che altro per la difficoltà di replicare tali sensazioni con un device, seppur per il primo dei due si stia lavorando in modo da poterlo coinvolgere con odori sintetici.

1.1.1 - HYPE CYCLE

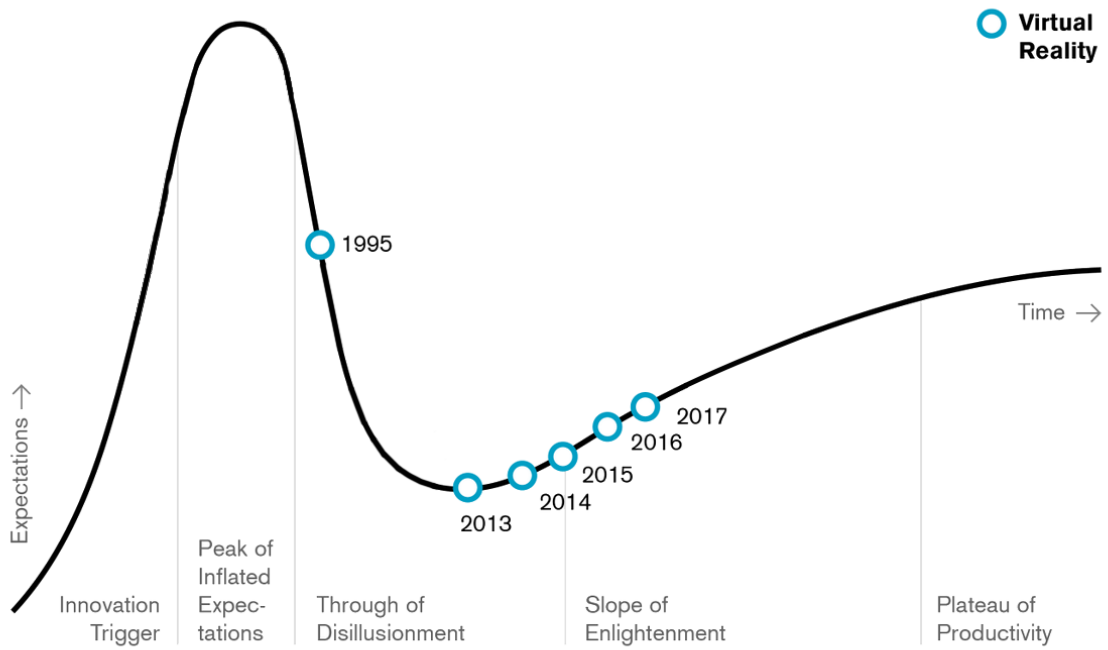


FIGURA 1.1: GRAFICO DELL'HYDE CYCLE DI GARNER RELATIVO ALLA REALTÀ VIRTUALE

La VR, ad oggi, è da considerarsi una tecnologia agli albori della sua fase di maturità. Scomodando l'Hype Cycle di Gartner, che mette in relazione l'interesse del pubblico ad una nuova tecnologia con l'avanzare del tempo, la VR ad oggi sembra aver superato la prima fase

di sovraeccitamento e la fase più critica di collasso dell'interesse ed è quindi prossima alla fase di stabilità, quella che porta all'effettiva produzione e distribuzione su larga scala di questa tecnologia.

Dal 2018 la VR non è più stata inserita nell'Hype Cycle di Gartner, quando invece ci si aspettava, essendo stata nel 2016 e 2017 nel "Slope of Enlightenment", che continuasse in questa fase o raggiungesse quella denominata "Plateau of Productivity". Ciò però non significa che la VR non sia più in crescita, poiché, a detta dello stesso Gartner, il motivo della sua scomparsa è dato dal fatto che la realtà virtuale è già quasi matura e non può quindi essere valutata come una nuova tecnologia[2].

Che lo sviluppo della VR non si sia fermato è infatti supportato da una serie di dati ed eventi degli ultimi anni. Per fare affidamento su dati oggettivi e alla portata di tutti, si può osservare il grafico di Google Trends, in merito alle ricerche, e quindi all'interesse, in merito alla VR, notando una costante crescita negli ultimi anni, dopo un calo di interesse.

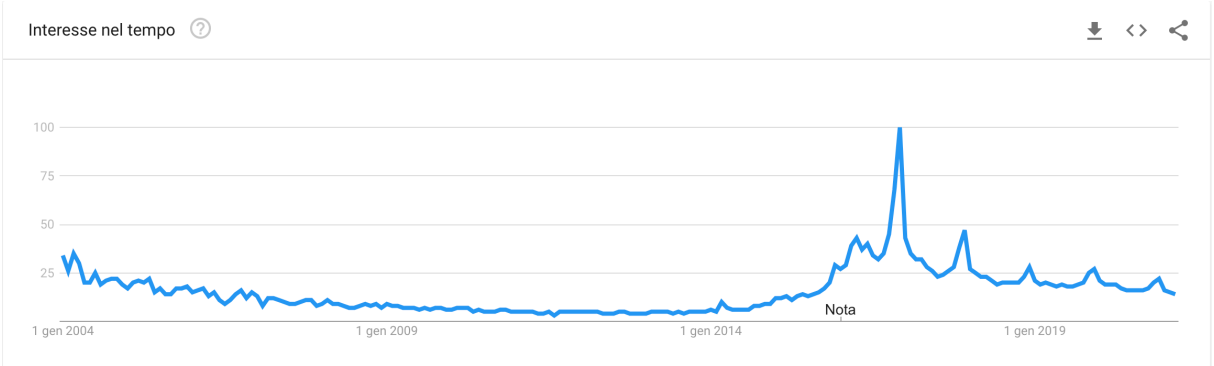


FIGURA 1.2: GRAFICO DI GOOGLE TRENDS (IN DATA 16.03.2021) RIGUARDO LE RICERCHE DI "VIRTUAL REALITY"

Dal grafico è piuttosto evidente un picco d'interesse nel 2016. A cosa è dovuto? Perché si è tanto parlato di VR in quell'anno? Sicuramente uno degli eventi di maggior rilievo nel 2016 è stata la dichiarazione di Zuckerberg a Oculus Connect, evento annuale del celebre colosso VR, acquistato da casa Facebook nel 2014 per 2 miliardi di dollari. Mark Zuckerberg in quell'occasione dichiarò di voler portare in breve tempo la VR in mano a un miliardo di utenti. Nello stesso anno sempre Zuckerberg presenzia a sorpresa a Mobile World Congress dove annuncia una collaborazione con Samsung sempre in merito alla VR.

Seppur i tempi per portare la VR ad un pubblico così vasto non siano brevi, Zuckerberg ha comunque i mezzi per farlo considerato che già ad oggi, con Facebook, Instagram e Wha-

tsapp possiede già la più ampia platea di utenti dalla quale partire per estrapolare i pionieri della realtà virtuale.

Zuckerberg ritiene che il futuro dell'era social risieda nel passaggio da web a VR, infatti è da anni che Facebook sta lavorando ad **Horizon**, un mondo interamente personalizzabile nel quale gli utenti vivono in multipresenza in VR. La chiave è la personalizzazione, dal proprio avatar all'ambiente stesso, una sorta di "User generated world". Horizon intende apparire come un mondo piacevole ed esclusivo che dia la possibilità di evadere dalla realtà. La possibilità di creare un avatar diverso dal proprio aspetto reale, di personalizzare il proprio ambiente, di scappare in un mondo diverso e allo stesso tempo più bello. Citando eventi più recenti, l' 8 Marzo 2021, ospite del podcast "The Information's 411", Zuckerberg ha detto che: "*Entro il 2030 le nuove generazioni di Oculus permetteranno agli utenti di teletrasportarsi da un luogo all'altro senza muoversi dal divano. Non solo per giochi e intrattenimento, ma anche per lavoro*"[3].

Dunque è chiaro come Facebook stia lavorando sia per rendere accessibile alla sua stessa platea l'hardware, avendo acquistato Oculus, che per far provare e soprattutto far tornare gli utenti nel mondo della VR. Horizon è stato rilasciato in versione Beta nell'agosto del 2020[4], ma ad oggi i beta tester possono accedervi esclusivamente su invito, e solo in alcuni Paesi previa compilazione di un questionario.



FIGURA 1.3: PRESENTAZIONE DI FACEBOOK HORIZON

Non è una novità che Zuckerberg sia sempre stato lungimirante, ma non è il solo a puntare sulla realtà virtuale in ottica futura. Partendo dal visore a basso costo **Google Cardboard** ri-

lasciato da Google, seguito dall'annuncio della stessa di voler rilasciare **Daydream**, una piattaforma per la realtà virtuale mobile di alta qualità e dal **Microsoft's Hololens AR development kits** che da pochi anni viene venduto e spedito in tutto il mondo[5]. In tutto questo si deve citare anche **Apple**, che già da due anni supporta l'AR nei suoi dispositivi con iOS e che dal 2022 lancerà il suo head mounted display[6]; oltre alla già sopracitata Samsung, con il suo **Gear VR** e Sony con la sua **PlayStation VR**.

Non è da trascurare il fatto che, stando a quanto dice Jeremy Schifeling, CEO di Break into Tech, ci sono molte opportunità anche al di fuori di colossi come Microsoft e Oculus/Facebook, i quali rappresentano solamente il 2% di tutte le aziende attualmente coinvolte nello sviluppo di questa nuova tecnologia. .

Secondo TechCrunch[7], le Venture Capitalists hanno investito in VR e AR 1,7 miliardi di dollari nei 12 mesi precedenti a marzo 2016, di cui 1,2 miliardi solo nel primo trimestre dell'anno. Alla luce di ciò, il mercato della realtà virtuale è stato valutato a 11,5 miliardi di dollari nel 2019 e si stima raggiungerà circa gli 88 miliardi entro il 2025[8].

Sembra iniziare ora l'interesse del pubblico di massa per la realtà virtuale, dato che nel 2019 le vendite annuali di applicazioni consumer in VR hanno superato per la prima volta il significativo valore di 1 miliardo di dollari.

Sebbene dunque l'interesse da parte del pubblico di massa sia moderato, vi sono invece forti investimenti a livello di business, supportati dal fatto che tutti gli analisti americani più conosciuti, chi più prudentemente come Juniper[9] e Beecham[10], chi più ottimisticamente come Gartner[11], concordano su una costante crescita di questa nuova tecnologia, unanimi sul definirla come *"The Next Big Thing"* in un'era post-web.

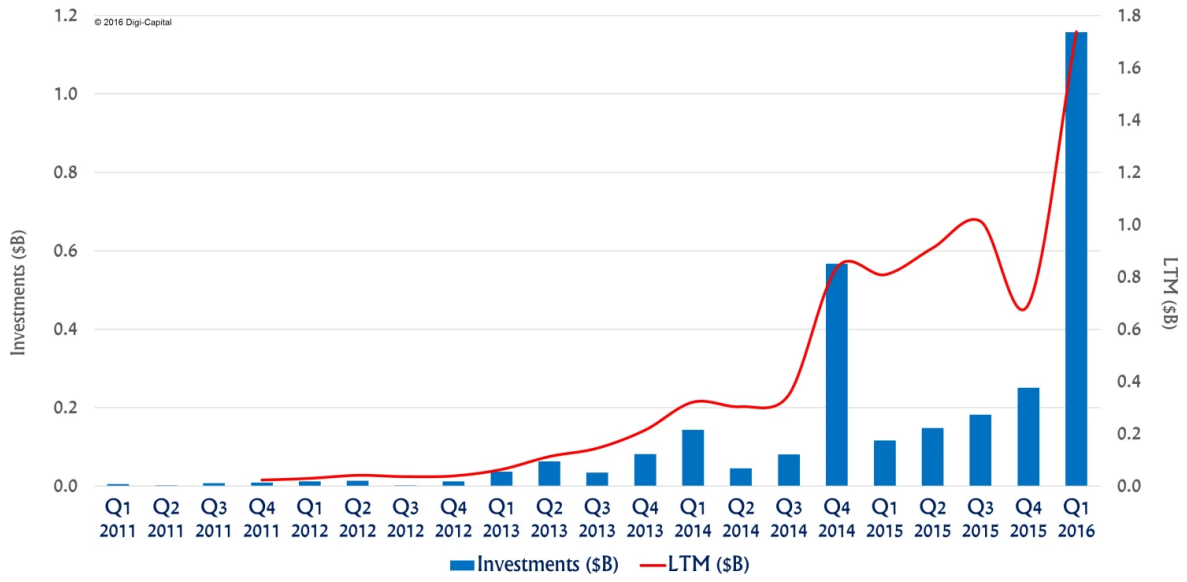


FIGURA 1.4: INVESTIMENTI IN AR E VR DAL 2011 AL 2016 SECONDO DIGI-CAPITAL (GRAFICO DEL 2016)

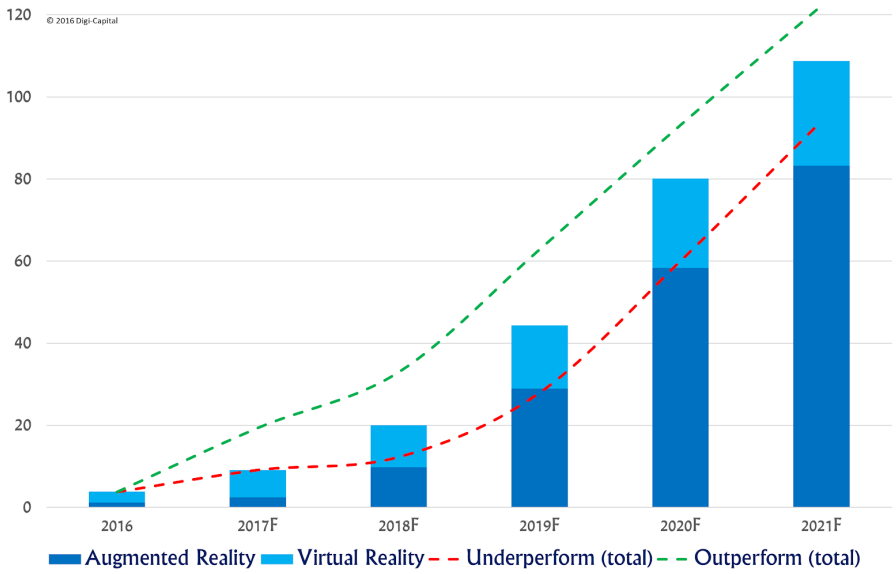


FIGURA 1.5: STIMA DEL RENDIMENTO DI AR E VR DAL 2016 AL 2021 SECONDO DIGI-CAPITAL (GRAFICO DEL 2016)

Per quanto sia alto l'interesse sul lato business e vi siano forti investimenti di varie aziende high tech, si può dire che non si sia ancora raggiunta una maturità, sia hardware che software, di questa tecnologia.

Lo stesso Nate Mitchell, Co-fondatore di Oculus ed ora entrato in Facebook, nel 2016 disse: "*Sappiamo di aver generato molta aspettativa attorno a questo fenomeno, perché crediamo davvero nelle potenzialità della VR [...] ma siamo realisti. Siamo i primi a dire che ci aspetta un viaggio che durerà almeno dieci anni. [L'arrivo di Oculus più economici] coinvolgerà una differente tipologia di utenti e aumenterà il numero complessivo di persone che useranno la VR.*"

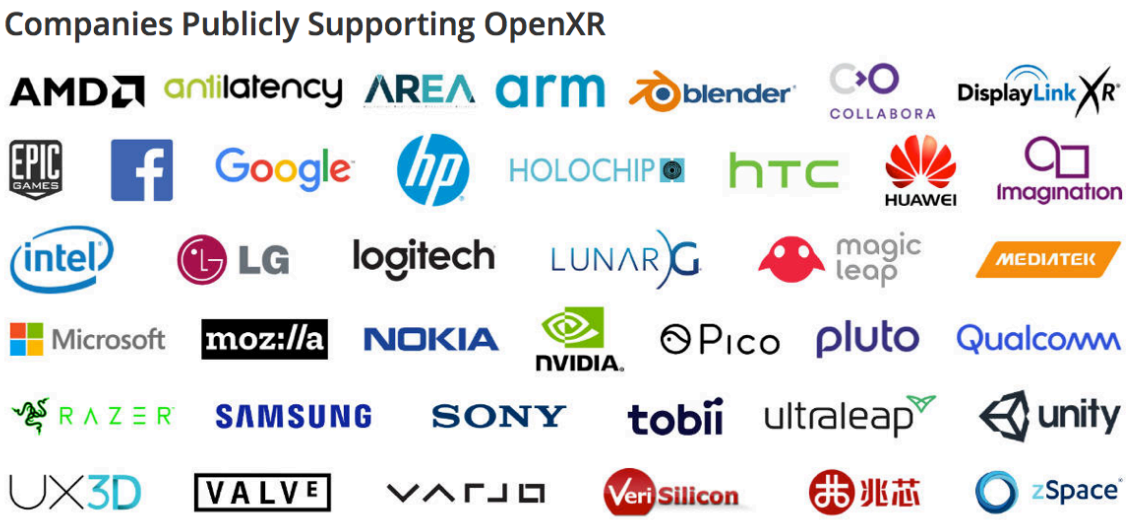


FIGURA 1.7: AZIENDE PARTECIPANTI AL PROGETTO OPENXR

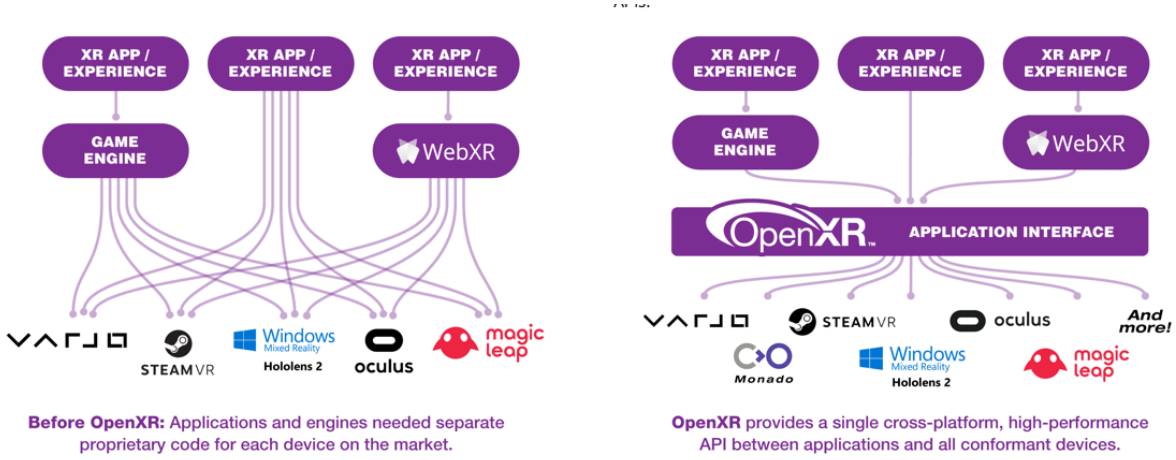


FIGURA 1.6: WORKFLOW SENZA E CON OPENXR

1.1.2 - LIMITI DELLA VR

Al giorno d'oggi vi sono ancora molti dubbi ed incertezze, sia tra i consumatori che tra le stesse aziende, sulle effettive potenzialità della VR e sulla diffusione del suo utilizzo nella società.
Riassumiamo dunque le principali barriere che accomunano sia utenti che gli stessi sviluppatori e imprenditori che vogliono scommettere sulla realtà virtuale per dare una trasformazione digitale alla propria azienda[12].

La prima barriera a cui ci si trova di fronte è l'**assenza di standard di sviluppo consolidati**. Ad oggi si è ancora nella fase in cui i principali attori del settore cercano di imporre i propri standard e facendo ciò impediscono che vi sia qualsiasi tipo di dialogo per avvicinarsi collettivamente ad una linea comune. La principale conseguenza di tutto ciò è la mancanza di una Killer App che renda semplice e immediata la scelta, quindi chi vuole approcciarsi al mondo della VR sin da ora, che sia utente o sviluppatore, si trova a doversi porre domande del tipo: "*Meglio usare Unity o Unreal Engine?*" oppure "*Meglio acquistare un Oculus o Vive?*", con la consapevolezza che ognuno di essi presenterà sia vantaggi che limiti e che, a prescindere dalla scelta compiuta, non ci sarà nessuna compatibilità con gli altri hardware sul mercato.

Nell'aprile del 2017 è iniziato lo sviluppo del primo tentativo di standardizzazione, **OpenXR**[13]. Dal sito di Khronos Group, che ne ha seguito lo sviluppo, si legge che OpenXR è uno standard Open e Royalty-free, che fornisce alte performance di accesso a Realtà Aumentata (AR) e Realtà Virtuale (VR) - collettivamente conosciute come XR - nelle loro piattaforme e dispositivi. Dunque questo progetto ha l'obiettivo di unificare i sistemi di AR e VR, sia lato

hardware che software, i motori di gioco e i loro contenuti, rendendo di fatto l'intero ecosistema più inter-operabile.
La versione 1.0 di OpenXR è stata rilasciata il 29 Luglio 2019 e tutt'ora vi stanno partecipando i maggiori colossi della VR (Oculus, Vive, Unity, Epic Games, Samsung, Magic Leap, Hololens, Google, ecc…) collaborando attivamente per trovare una linea di programmazione comune e cross-platform.

La seconda barriera alla quale ci si trova davanti è quella data dall'hardware per la VR che, per quanto vi siano grandi miglioramenti rispetto ad una decina di anni fa, non ha ancora raggiunto un rapporto qualità prezzo accettabile per una diffusione di massa. Siamo dunque di fronte ad **hardware ancora molto costoso per essere scomodo e limitato**. Nel complesso possiamo dividere i visori in tre categorie: per mobile, standalone e per PC. Quest'ultimi sono coloro che offrono le migliori prestazioni, potendo contare sulla potenza computazionale di un computer, anche se ad oggi per "portare a casa" un'esperienza VR di questa portata l'investimento necessario è di oltre mille euro, considerato sia il costo di un VR Headset che del computer con le caratteristiche minime per poterlo utilizzare al meglio. È logico affermare che pochi sono disposti ad investire una somma simile per una esperienza che comunque resta ancora limitata e di nicchia. In merito di qualità si è fatto un significativo passo avanti sia con **Oculus Rift S** uscito nella primavera del 2019, che elimina definitivamente la necessità di sensori di movimento, che con **HTC Vive Cosmos**, uscito a ottobre 2019, il quale, come il suo principale competitor, aumenta la risoluzione ed elimina la necessità di sensori[14].



FIGURA 1.8: OCULUS RIFT S, HTC VIVE COSMOS

Se all'uscita di Oculus Go, il primo visore standalone di casa Oculus, non si era riusciti a mantenere uno standard qualitativo paragonabile a quello offerto dal Rift, oggi invece si può fare un ragionamento differente. La direzione in cui tutti i più grandi colossi si stanno spin-

gendo è proprio quella di un dispositivo completamente indipendente da mobile, sensori e computer che abbia però prestazioni uguali o superiori agli headset come Rift S e Cosmos. Il 20 Ottobre 2020 è uscito sul mercato **Oculus Quest 2**, erede di Quest dell'omonima casa, entrambi successori di Oculus Go. Oculus Quest 2 è completamente indipendente da qualunque altro dispositivo (se non da smartphone per configurazione iniziale), senza necessità di sensori esterni, con due controller e la possibilità, grazie a Oculus Link, di essere collegato via cavo ad un computer per giocare i titoli disponibili solo per Oculus Rift e Rift S. All'uscita di Oculus Quest 2 hanno cominciato a girare rumors riguardo ad un nuovo headset standalone di HTC per rispondere al nuovo hardware del suo maggior competitor, ma ad oggi non vi sono informazioni certe sulle specifiche del nuovo headset di casa HTC, che potrebbe essere il successore di HTC Vive Focus, come anche un prodotto inedito, chiamato HTC Vive Proton presentato a febbraio 2020 come prototipo[15].



FIGURA 1.9: OCULUS QUEST 2, HTC VIVE PROTON (PROTOTIPO)

Seppur vi siano oggettivamente dei miglioramenti, anche analizzando solamente gli ultimi 5 anni, vi sono ancora problemi intrinseci nell'hardware, che attualmente risulta essere ancora grosso, pesante e scomodo, oltre che ancora poco preciso. L'insieme di questi fattori impedisce di poter usufruire di un head mounted display per diverse ore, che sia per gioco ma soprattutto per lavoro. È indubbio quanto sarebbe enormemente vantaggioso per un designer, ad esempio, lavorare direttamente su un oggetto in 3D, ma ad oggi bisogna fare ancora i conti con una risoluzione insufficiente, una precisione approssimativa e l'impossibilità di sopportare questo tipo di hardware per intere giornate di lavoro.
La soluzione a questo tipo di problema, al momento non ancora attuabile, sembra essere quella di ridurre al minimo l'hardware necessario, sfruttando in futuro la velocità di una rete 5G. In questo modo si potrà tenere in cloud anche buona parte dell'hardware, utilizzando la la potenza computazionale di un computer e rendendo il visore nettamente più leggero, comodo e versatile.

Il terzo ostacolo da tenere in considerazione riguarda la **rivoluzione delle interfacce**.
Chiunque abbia avuto un minimo a che fare con la tecnologia desktop o mobile si è interfacciato con tre tipi di interfacce hardware: mouse, tastiera e più recentemente con touch-screen. Con l'arrivo della VR si ripresenta la situazione in cui si sono trovati coloro che per primi misero mano a tastiera e mouse a metà del secolo scorso. Se per molti oggi usare una tastiera, un mouse o un touch screen è un gesto di uso quotidiano è perché vi è stata, in un qualche momento della sua vita, una fase di apprendimento nell'utilizzo di questi strumenti. La VR rivoluziona completamente il tipo di interazione a cui siamo abituati poiché non si basa più sugli strumenti sopracitati e impone all'utente una nuova fase iniziale di apprendimento nell'utilizzo dei nuovi strumenti, quali visore e controller.
Se quanto descritto risulta essere un ostacolo per il cliente finale, che viene in qualche modo scoraggiato da questa sfida, lo è allo stesso modo per chi si accinge a progettare UI per la realtà virtuale. Cambia decisamente il modo in cui l'utente si approccia al nuovo mondo virtuale, sia per quanto riguarda le interfacce hardware, sia per un radicale cambiamento dello spazio a disposizione, poiché si passa da un approccio in 2D ad una interazione con il mondo circostante su 3 dimensioni e quindi con 6 gradi di libertà di movimento.

Questo passaggio presenta sia degli svantaggi che dei vantaggi. Tra i lati negativi troviamo il fatto che sia una situazione nuova, ancora da comprendere e priva di qualunque standard per la realizzazione. Prendiamo ad esempio il fatto che i mouse, di qualunque marca, ormai presentano tutti un tasto destro, uno sinistro e una rotellina al centro. Questo si può definire come standard, come lo è la posizione delle lettere sui differenti layout di tastiere a seconda della lingua; ciò non esiste, o per lo meno non in modo ben definito, se si parla di dispositivi VR. Questo discorso può essere ripreso anche per la UI, poiché l'interazione con uno schermo, che sia o meno touch-screen, presenta ormai tutta una serie di standard e convenzioni che non esistono nell'interazione in VR. Anche qui possiamo portare l'esempio delle finestre di lavoro delle varie applicazioni o dell'icona di chiusura a forma di croce posta in alto a destra o a sinistra a seconda del sistema operativo. In VR che significa finestra? Che significa "in alto a destra" se il mondo in cui mi muovo non è più bidimensionale? Vi è dunque la necessità di ripensare ogni interazione considerando una dimensione in più nello spazio a disposizione oltre che a maggiore libertà di movimento.
Molti, per comodità o per dare all'utente una parvenza di familiarità di interazione, tendono semplicemente a riportare in 3D ciò che era stato progettato per un interfaccia 2D, creando pannelli che si aprono nello spazio esattamente come le finestre che si aprono sul desktop,

ma per quanto sia familiare è un approccio limitante in una situazione dove il "dover" considerare una terza dimensione può invece essere rivalutato come "possibilità" di non limitarsi a una interazione piatta.
Il lato positivo di questo nuovo tipo di interazione è che essa, pur essendo mediata comunque da interfacce hardware, si avvicina maggiormente all'interazione naturale, di quanto non sia quella con mouse e tastiera. Questo tipo di "avvicinamento" è già stato operato dai dispositivi con touch-screen, poiché viene imposto di toccare direttamente l'oggetto con cui voglio interagire invece di muovere il mouse, e quindi il relativo cursore, sopra quanto desiderato. Con la VR si fa un passo ulteriore, poiché per spostare un oggetto si può "semplicemente" prenderlo e spostarlo, per raccogliere qualcosa da terra si può fisicamente abbassarsi per prenderlo, per muoversi si può camminare o ruotare su se stessi, seppur nei limiti che lo spazio reale impone. Per cui, per quanto le interazioni di questa tecnologia siano ancora da reinventare, può essere più intuitiva e di semplice apprendimento per l'utente, mettendo il developer nelle condizioni di poter ripensare le interazioni chiedendosi come egli agirebbe in una qualsiasi interazione quotidiana nel mondo reale.



FIGURA 1.10: OCULUS RIFT CON TOUCH CONTROLLER E SENSORI

1.1.3 - ESPERIENZA PERSONALE CON OCULUS RIFT (2016)

Lavorando a questo progetto, ci è stata gentilmente concessa da Dead Pixels l'opportunità di poter utilizzare due Oculus Rift del 2016. Abbiamo così approfittato dell'occasione per non limitarci allo studio "passivo" delle caratteristiche di un headset, provando in prima persona esperienze di utilizzo del dispositivo VR.

Per quanto riguarda i problemi riscontrati lato designer e developer vi ci soffermeremo nei successivi capitoli, analizzando approfonditamente ogni singolo ostacolo o limite riscontrato e come abbiamo pensato di risolverlo. Ora intendiamo soffermarci sull'esperienza più generale lato consumer, consapevoli di aver utilizzato un dispositivo del 2016 e che da esso, come già detto, vi sono stati ulteriori miglioramenti. Precisiamo che l'analisi che segue effettuata su Oculus Rift non è finalizzata ad esaltare e/o denigrare in alcun modo tale dispositivo, ma si pone l'obiettivo di analizzarne le funzionalità offerte in modo oggettivo.

Abbiamo constatato che la fase di set-up è particolarmente lunga, molto lontana da quella ideale di un approccio "plug and play" generalmente preferito dall'utenza di massa, per quanto si venga seguiti passo a passo in questa fase, con abbondanza di feedback e consigli. Sicuramente l'idea di eliminare i sensori statici, come già attuato nei visori più recenti, è indubbiamente un passo avanti, poiché proprio questi hanno richiesto più tempo per essere correttamente settati e nonostante ciò continuavano saltuariamente a dare problemi.

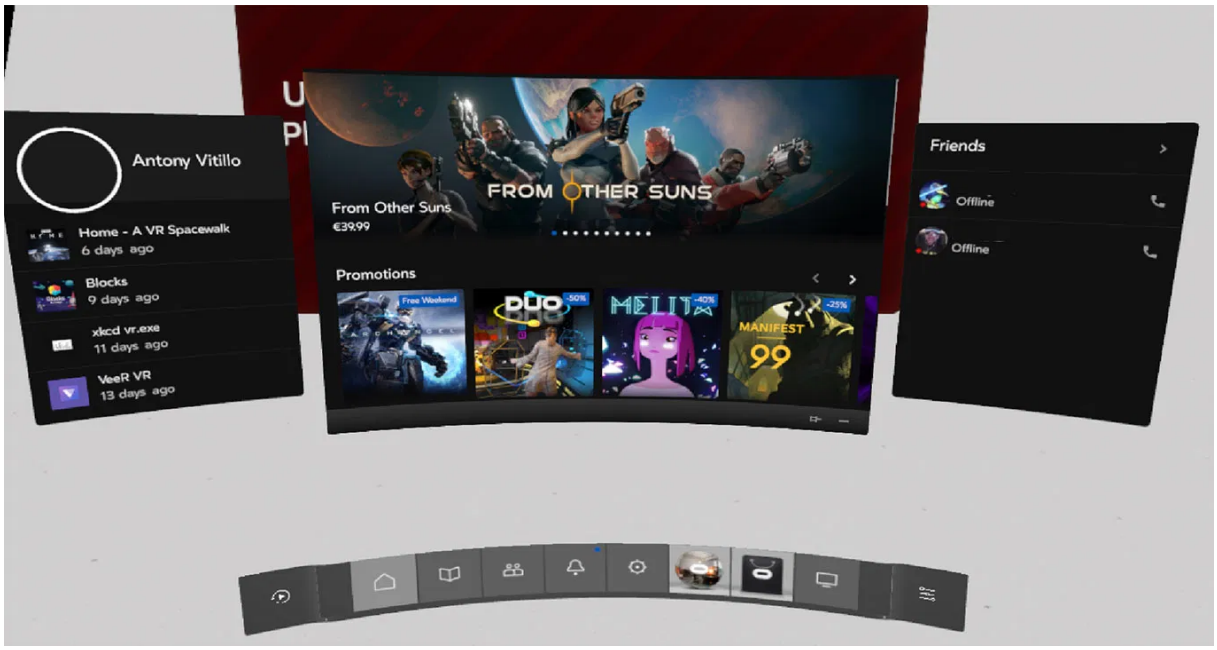


FIGURA 1.11: MENU E DASHBOARD OCULUS

Concluso il settaggio, l'app di Oculus offre un tutorial interattivo che spiega l'interazione con i touch controller e lascia spazio e tempo all'utente per ambientarsi in questa nuova realtà, mediante delle esperienze interattive e dei video a 360 che mostrano il massimo di resa grafica, realismo e immersione raggiungibile con Oculus.

Finito il tutorial si passa all'ambiente dedicato al menu iniziale di Oculus, composto da una dashboard e uno schermo ricurvo, ulteriormente divisibile. È possibile dedicare ad ognuno di questi schermi una finestra di una qualunque app aperta sul computer, avendo la possibilità quindi di lavorare sul proprio computer ma in modalità VR. Se questo idealmente può sembrare molto interessante e pieno di possibilità, in realtà mostra già alcuni evidenti e anche disturbanti limiti della VR. Portare una realtà bidimensionale come quella desktop su tre dimensioni implica il dover ri-adattarne l'interazione; si veda il caso della scrittura di un testo che in VR significa cliccare uno ad uno i tasti di una tastiera virtuale, avendo a disposizione un unico dito e un puntatore la cui precisione dipende molto dai sensori ed è ancora limitata. Ci si riesce e anche abbastanza velocemente, ma è impensabile poter scrivere qualcosa di più lungo di un indirizzo e-mail, o anche solo chattare con qualcuno dovendo continuamente scrivere messaggi. Stessi limiti, seppur meno evidenti, sono legati al dover riprodurre le interazioni tipiche di un mouse e questo rende difficile l'utilizzo di programmi più complessi. Dunque a primo impatto è tutto molto bello, suggestivo, anche se in quasi la totalità dei casi lo si prova per una decina di minuti per poi spegnere tutto e tornare a lavorare da computer nel caso si debbano fare operazioni più complesse.

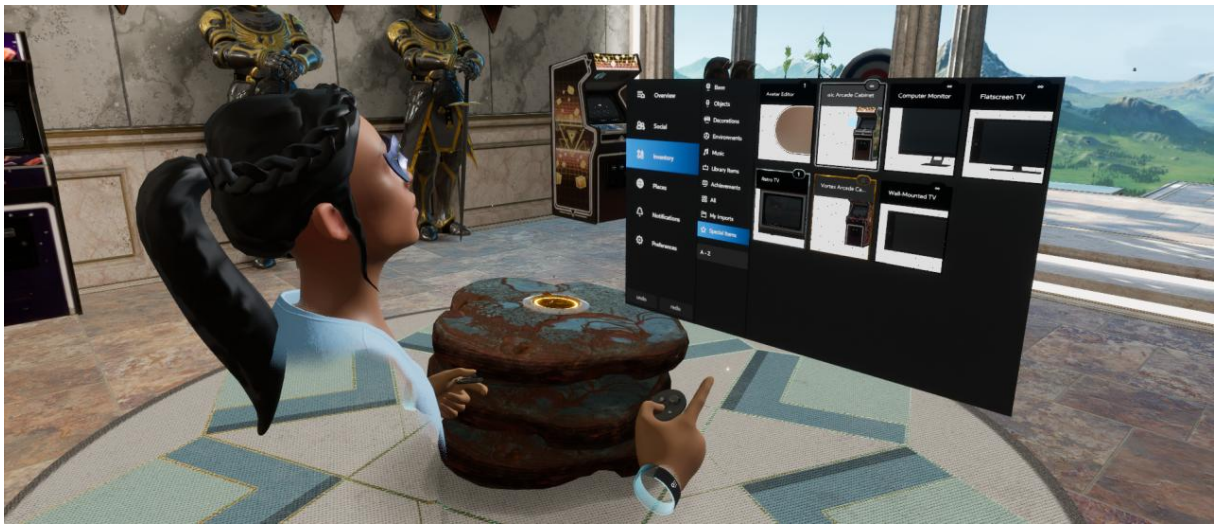


FIGURA 1.12: AVATAR OCULUS DURANTE LA FASE DI PERSONALIZZAZIONE DELLA SUA HOME



FIGURA 1.13: PIÙ UTENTI PRESENTI IN UNA STESSA HOME

Oltre al menu iniziale, Oculus offre la possibilità di creare delle stanze virtuali personali, nelle quali è possibile incontrare e invitare altri utenti. Le potenzialità sono molte, poiché si possono organizzare riunioni in un ambiente virtuale, si può condividere lo schermo del computer e vedere insieme ad altri utenti film, video o presentazioni, ma rimane costante la sensazione che non sia ancora abbastanza, oltre al fatto di essere evidentemente un progetto social in un mondo ancora troppo poco popolato per sfruttare appieno queste potenzialità.
Nello Store di Oculus sono a disposizione molti titoli, ognuno dei quali viene catalogato su vari parametri: la compatibilità o meno con tutti i tipi di visore, la possibilità di essere usufruiti sia stando in piedi che seduti, ma la caratteristica messa maggiormente in evidenza è l'**indice di comfort**. Questo indice è legato alla **cyber-sickness,** effetto simile alla **motion-sickness (o cinetosi)**, ma in particolare ad alcune esperienze in VR e affligge circa l'80% degli utenti che sperimentano questa nuova tecnologia[16].
Le effettive cause della cyber-sickness non sono ancora note, ma come sostengono Gallagher e Ferrè nel loro "*Cybersickness: A Multisensory Integration Perspective*"[17], la causa principale, come nella motion-sickness, sembra legata alla discrepanza di informazioni che il cervello riceve dagli occhi e dal sistema vestibolare (dell'orecchio interno che regola l'equilibrio). In molte applicazioni VR, il flusso ottico suscita un'illusoria sensazione di movimento all'utente comunicandogli che si sta muovendo in una certa direzione con una certa accelerazione, mentre allo stesso tempo gli organi propriocettivi e vestibolari forniscono al cervello l'infor-

mazione di essere fermi. Questi segnali contrastanti possono portare a discrepanze sensoriali e alla fine a sensazioni di nausea, disorientamento, affaticamento degli occhi.

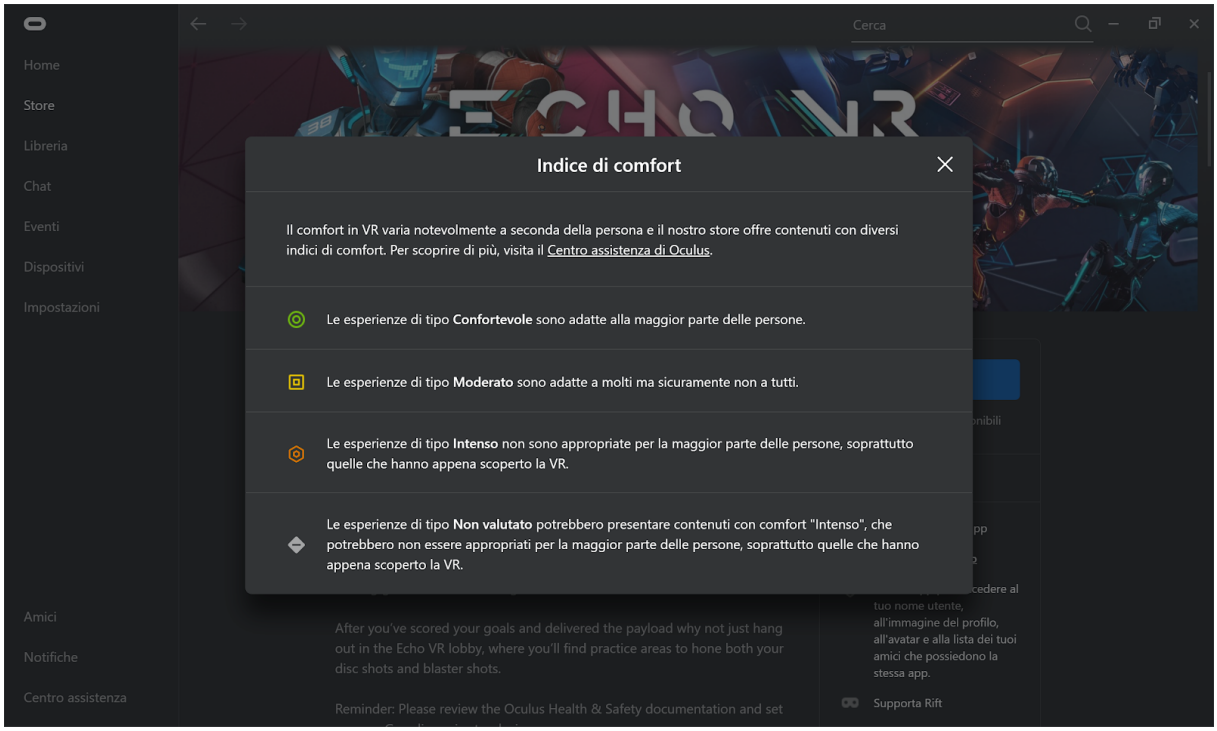


FIGURA 1.14: SPIEGAZIONE DELL'INDICE DI COMFORT DELLE APP PRESENTI NELL'OCULUS STORE

Provando il gioco Echo VR, a disposizione nello store Oculus, abbiamo riscontrato esattamente quanto appena descritto. Se nella propria Home ci si può muovere tramite "teletrasporto", ciò non avviene in questo gioco, dove già nei primi momenti di tutorial un utente inesperto può immediatamente percepire una leggera sensazione di nausea e disorientamento. Per quanto questa situazione sia piuttosto disturbante all'inizio e aumenti con gli spostamenti eseguiti più velocemente, personalmente abbiamo riscontrato un attenuazione di questo effetto man mano che si passava del tempo nel gioco. Siamo consapevoli però che questo è profondamente soggettivo e questa sensazione può sia attenuarsi come invece diventare sempre più pesante a seconda dell'individuo.

Infine intendiamo sottolineare, sempre secondo la nostra esperienza, altri due aspetti maggiormente legati ai problemi di questa tecnologia, secondo noi, ancora immatura.
L'assenza di standard consolidati già citata in precedenza è sicuramente un problema critico anche per gli utenti, in quanto al momento ogni sviluppatore propone differenti soluzioni e approcci per contesti anche simili fra loro. Dunque l'utente si trova a dover passare da un'app dove la direzione di teletrasporto si sceglie ruotando il controller, ad una dove ciò

avviene tramite thumbstick, da una dove gli oggetti si prendono con il Trigger button, ad un'altra dove si prendono con il Grip button, dovendo ogni volta dedicare qualche minuto a ri-apprendere i comandi richiesti per quel contesto.
Dopo una o due ore di utilizzo, è inevitabile constatare quanto l'hardware sia ancora irrimediabilmente pesante e scomodo. Per quanto si faccia attenzione a regolarlo correttamente, risulta ancora impensabile utilizzare questo strumento lo stesso numero di ore consecutive che ad oggi si possono passare di fronte ad un computer per lavoro. Il suo peso a lungo andare può provocare piccoli dolori sulla parte posteriore della testa dovuti ad un eccessivo sbilanciamento del visore sul lato anteriore e la sua risoluzione, ancora troppo bassa, affatica lo sguardo anche dopo solo un'ora di utilizzo. È dunque evidente come anche su questi aspetti siano indispensabili dei miglioramenti.

Per quanto siano ancora tanti i lati negativi la Realtà Virtuale risulta già essere una esperienza fortemente immersiva e coinvolgente, in grado di spingere l'utente a "sopportare" alcuni problemi esaltando le potenzialità e le forti aspettative che questa tecnologia suscita in chiunque abbia occasione di provarla.

Se dunque, dopo quanto analizzato, per l'utenza di massa occorre ancora del tempo, ciò non vale per l'ambito business che, per quanto comunque soggetto alle limitazioni soprascritte, avrebbe sicuramente più vantaggi in un investimento di questo tipo. Magari non tanto per basarci completamente il proprio lavoro, ma come appendice rivoluzionaria da affiancare a quanto già si utilizza, avendo sicuramente delle spese iniziali, ma acquisendo il vantaggio di arrivare per primi, fornendo ai clienti un'esperienza completamente nuova, immersiva e interattiva non ancora proposta dai competitor.

Si ha quindi una situazione in cui la VR affascina, interessa come singola esperienza, ma non riesce ancora a convincere completamente l'utenza e ad offrire quindi una motivazione valida che incentivi il passaggio da un uso sporadico e limitato ad uno continuo e soprattutto piacevole. L'indossare un caschetto non deve essere solo una scomodità necessaria ma piuttosto uno strumento che dia la possibilità di evadere da una realtà per entrare in un nuovo mondo, su cui ancora c'è da lavorare, ma a cui i colossi del settore stanno fortemente puntando.

1.2 - PRESENZA E MULTIPRESENZA

Prima di parlare di multipresenza, è doveroso soffermarsi sul significato della parola "presenza" all'interno di un ambiente virtuale, che sempre più spesso viene confuso con quello della parola "immersione". Se da un lato il senso d'**immersione** misura il livello di fedeltà sensoriale fornito da un sistema VR, il **senso di presenza** rappresenta invece una risposta psicologica soggettiva di un utente che sperimenta un'applicazione in realtà virtuale[18]. L'origine di tale fenomeno psicologico è da attribuirsi all'impressione dell'utente di trovarsi all'interno di un luogo reale.

1.2.1 - I SENSI COINVOLTI NELLA VR

Il senso di presenza è un fattore chiave che deve essere considerato quando si progetta un'ambiente basato sull'**IVR** (Immersive Virtual Reality). Tuttavia, ricreare tale sensazione non è così facile come potrebbe sembrare, poiché, come come analizzato in precedenza, l'hardware disponibile ai giorni nostri non è ancora in grado di fornire un'esperienza sensorialmente credibile in realtà virtuale e di ricreare quindi nell'utente quel senso d'immersione fondamentale al raggiungimento del senso di presenza. Per un completo coinvolgimento, tutti i nostri cinque sensi dovrebbero essere coinvolti, per convincerci che ci troviamo in una nuova realtà[19]. Nonostante questi deficit tecnologici è di vitale importanza che alla base di di ogni progetto vi sia uno studio su come implementare un applicativo che restituisca quel minimo senso di realismo tale da permettere all'utente di abbandonarsi all'esperienza e di seguirne le logiche di funzionamento. Solo in seguito ci si potrà poi focalizzare sulla realizzazione di un sistema multiutente che attraverso interfacce e feedback istantanei riesca a mettere in comunicazione più utenti contemporaneamente, portando quel senso di realismo menzionato in precedenza ad un livello superiore.

Tra i cinque sensi da coinvolgere in questo processo di simulazione della realtà, la vista è sicuramente quella più importante, in quanto da sola può già ingannare il cervello umano a tal punto da fargli credere che tale ambiente artificiale sia effettivamente reale. Proprio per questo si sfrutta la tecnica della **stereoscopia**, largamente già utilizzata nel mondo delle video produzioni e della fotografia per trasmettere un'illusione di tridimensionalità, analoga a quella generata dalla visione binoculare del sistema visivo umano[20]. Nella pratica vengono utilizzate due o più camere all'interno del HMD dedite alla ripresa di un filmato a 360 gradi che vengono poste a livello della posizione occupata rispettivamente dall'occhio sinistro e

destro dell'utente. Queste videocamere registrano continuamente una serie d'immagini da due prospettive leggermente diverse in modo da creare una minima sovrapposizione tra le due riprese, il cosiddetto **effetto di parallasse**. Attraverso algoritmi avanzati di *Image Processing* le due diverse sequenze di immagini vengono poi unite in un'unica ripresa grazie ad un processo denominato **Stitching** che restituisce un'esperienza visiva tridimensionale. Tuttavia, l'effetto di parallasse non viene completamente eliminato, causando così fenomeni di *cyber-sickness*[21].



FIGURA 1.15: CASO ESTREMO DI PARALLASSE

Per queste oggettive problematiche che non permettono un completo coinvolgimento sensoriale, si sfruttano anche i cosiddetti **suoni spazializzati**, ovvero cluster di sorgenti sonore che vengono posizionati all'interno dello spazio tridimensionale in grado di riprodurre suoni realistici e coinvolgenti, tenendo conto di fattori reali come la distanza, il rumore di fondo, la geometria e l'occlusione[22].

Si è cercato poi di simulare questo senso di immersione, aumentando quindi anche quello di presenza, adottando **sistemi aptici**, che forniscono reazioni fisiche come feedback a seguito di un'azione compiuta dall'utente. Dal più semplice **force feedback** di un motion controller, passando per **guanti aptici** che restituiscono la sensazione tattile di tenere in mano un oggetto virtuale, per arrivare infine a delle **tute aptiche** come la **TactSuit X40**[23] della **bHaptics** o la **TeslaSuit**[24]**,** in grado di inviare sia impulsi elettrici che feedback vibro-tattili a tutto il corpo grazie alla presenza di piccoli computer interni alla tuta. Tuttavia, come già accennato nel precedente sottocapitolo, il grosso problema di questi device è la loro accessibilità per un consumatore medio di un prodotto VR. Tralasciando le tute aptiche full body che in commercio sono disponibili solamente a prezzi maggiori di diecimila dollari, anche solo un paio di guanti aptici può arrivare a costare cinquemila dollari e risultare quindi proibitivo per la maggior parte dei suoi possibili consumatori. Senza contare che la maggior parte delle applicazioni presenti sui diversi store online non sono programmate per supportare device aptici più sofisticati di un semplice motion controller.

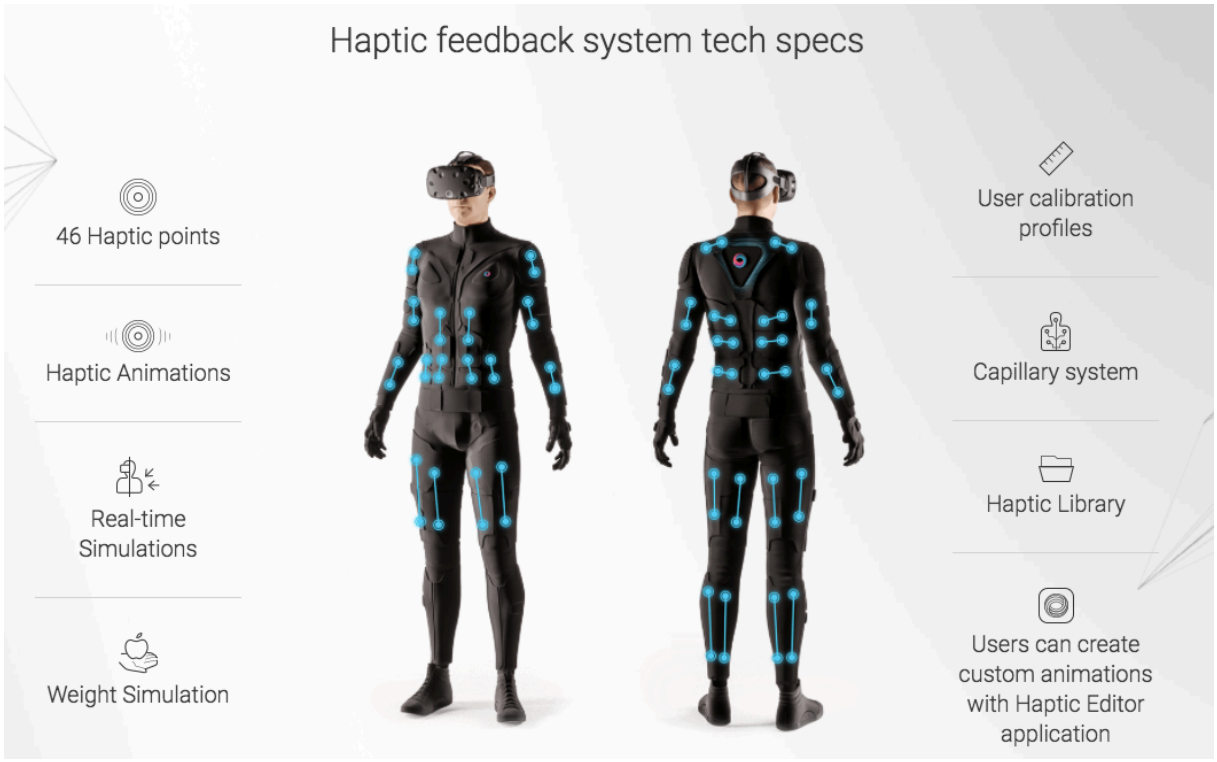


FIGURA 1.16: SPECIFICHE TECNICHE DELLA TUTA APTICA TESLASUIT DELLA TESLA STUDIOS

Oltre a questi, Wonjun Park et al. del Dipartimento Software dell'Università Cattolica di Pusan (Korea) nel loro articolo di ricerca "*A Study on the Presence of Immersive User Interface in Collaborative Virtual Environments Application*"[25], affermano che anche la collaborazione, o **in-**

terdipendenza sociale, che può nascere tra i diversi utenti partecipanti alla medesima sessione di un'applicazione VR, sia un fattore che possa aumentare il loro grado di piacere e soddisfazione, nonché quello d'immersione, che è l'elemento più importante per raggiungere il senso di presenza.
Gli studi condotti da Wonjun Park et al. si focalizzano sugli **ambienti virtuali asimmetrici**, ovvero ambienti in cui sono compresenti sia utenti dotati di HMD (Head-Mounted Display) che non e che interagiscono continuamente tra loro per il raggiungimento di un obiettivo comune. Nello specifico, vengono elencati gli elementi imprescindibili che devono essere presenti in un interfaccia utente (UI), non solo VR, affinché essa possa facilitare l'interazione tra player ed ambiente e possa inoltre mettere in comunicazione più utenti dotati di hardware differente.

Tale interfaccia deve innanzitutto mettere a disposizione una **modalità di interazione diretta** tra utente e ambiente virtuale attraverso VR Motion Controller che simulino il comportamento delle **mani**, la parte del nostro corpo più frequentemente usata per interagire con tutto ciò che fisicamente ci circonda. Così facendo viene suggerito in modo indiretto all'utente che potrà interagire con gli oggetti che gli si presenteranno davanti come farebbe nel mondo reale. Per consentire agli utenti di sperimentare realisticamente azioni e comportamenti all'interno di ambienti virtuali, il movimento delle loro articolazioni deve essere rapidamente rilevato e accuratamente riconosciuto, nonché riflesso nell'ambiente virtuale. Pertanto una **sincronizzazione rapida e accurata** tra movimento compiuto e feedback visuale è indispensabile per prevenire disturbo o qualsiasi fenomeno di cyber-sickness durante l'esperienza.

1.2.2 - COLLABORATIVE VIRTUAL ENVIRONMENTS

Per favorire qualsiasi fenomeno di interdipendenza sociale all'interno di ambienti asimmetrici è necessario incentivare la collaborazione tra i partecipanti. Se ad ogni utente viene assegnato un compito che deve essere svolto con l'aiuto di altri, inevitabilmente aumenta il senso di immersione sperimentato dal singolo utente rispetto alla medesima esperienza in solitaria. Questo solo grazie ad un'**interfaccia di comunicazione** che permetta non solo di condividere istantaneamente informazioni sul proprio status corrente, ma in generale di scambiare con gli altri utenti qualsiasi informazione necessaria al compimento di un'azione.
Osservazioni condivise anche da Stephan Streuber, docente del dipartimento di Computer and Information Science dell'Università di Konstanz (Germania), e Astros Chatziastros, psi-

cologo presso il Max Planck Institute for Biological Cybernetics di Tübingen (Germania), nel loro articolo "*Human Interaction in Multi-User Virtual Reality*"[26]. In aggiunta a quanto detto in precedenza, essi propongono di unire quelle che sono le principali caratteristiche degli ambienti IVR con quelle degli **OMUE** (online multi-user environments) per creare degli **IMUE** (immersive multi-user environments), le cui feature sono state opportunamente riassunte nella tabella sottostante.

	IVR	OMUE	IMUE
Sensory Motor Integration	X		X
Large Field of View	X		X
Ego Perspective	X		
Multi User Interaction		X	X
Interaction Quality (accuracy, realism)	X		X
User is visually represented		X	X
Somatosensory Interaction	X		X

FIGURA 1.17: TABELLA CON CARATTERISTICHE DI UN AMBIENTE IMUE

Tra queste citiamo l'importanza di rappresentare fisicamente un utente nell'ambiente attraverso degli **avatar virtuali** che, come diversi studi precedenti hanno dimostrato, facilitano l'interazione tra utenti e il completamento di alcuni task generali. Ma anche la necessità di **superare i limiti tecnologici** degli HMD, aumentandone la risoluzione e l'ampiezza del campo visivo. Da queste si evince come sia fondamentale fornire costantemente all'utente dei feedback, sia visivi che aptici, sul suo operato e su quello degli altri utenti, in modo da incentivare quella **interazione multi-utente** tipica degli OMUE.

A seguito di ciò, si giustifica il forte legame tra presenza e multipresenza; due obiettivi da raggiungere per chiunque voglia progettare dei **Collaborative Virtual Environments (CVEs)** e quindi anche per questo progetto di tesi.

Durante il processo di costruzione delle nostre interfacce utente abbiamo voluto seguire le linee guide tracciate da Chatziastros e Planck, cercando quindi di creare nel complesso un'esperienza sensorialmente "credibile". Non potendo intervenire a livello tecnologico, ci siamo focalizzati sia sulla realizzazione di opportuni sistemi di feedback per le nostre UI, che, più in generale, sulla progettazione di modalità d'interazione che fossero le più accurate e realistiche possibili, puntando molto sulla **Interaction Quality** del nostro applicativo.

Detto questo, al giorno d'oggi non esistono degli standard di progettazione che siano ottimali per qualunque tipo di applicativo finalizzato alla realizzazione di un ambiente virtuale. Abbiamo quindi analizzato le modalità d'interazione e le interfacce UI di varie applicazioni VR presenti in commercio, cercando di capire se vi fossero comunque pattern comuni da cui potevamo prendere ispirazione e che potevano essere ri-adattati in base alle caratteristiche e al target del nostro progetto.

1.3 - COMPETITOR E ANALISI DI DIFFERENTI APPROCCI

Come richiestoci da Dead Pixels, abbiamo analizzato alcuni competitor che hanno realizzato in questi ultimi anni progetti sperimentali basati sulla **multipresenza in un ambiente VR.** Essi hanno sviluppato differenti applicativi che permettono di **lavorare simultaneamente nello stesso ambiente**, dando la possibilità di muoversi liberamente all'interno dello spazio virtuale e di interagire con gli altri utenti attraverso i propri **avatar**.
I competitor analizzati sono: ***The Wild***[27]***, Insite VR***[28]***, Holodeck***[29]***, BMW***[30]***, All Thing Media***[31]***, CocoVerse***[32] e ***Gravity Sketch***[33]**.**

1.3.1 - FEATURES COMUNI

Dopo un'attenta analisi dei progetti portati avanti dalle suddette aziende, abbiamo riscontrato diverse features comuni a tutti i competitor. Tutti permettono di **interagire con l'ambiente**, cambiando colori e materiali dei vari oggetti presenti in esso, modificando gli oggetti stessi o parti di essi. *CocoVerse* permette inoltre di creare nuovi oggetti che vengono aggiunti alla scena rendendoli visibili e interagibili anche da altri utenti e non solo da chi li ha generati. È immediatamente evidente come quasi tutti i competitor ragionino in ottica di **software Multi-piattaforma**, rendendo l'ambiente virtuale accessibile non solo da VR ma anche da pc desktop e mobile (*The Wild, Coco Verse, Holodeck, All Thing Media*).
Infine abbiamo constatato come sia pressoché comune anche la logica di movimento all'interno dello spazio. Il classico spostamento fisico nell'ambiente è spesso integrato con il **teletrasporto**, in alcuni casi anche su differente scala, potendo "entrare" in un edificio che prima si stava osservando come modello dall'esterno. Il teletrasporto avviene mediante salvataggio di alcune viste accessibili in qualunque momento (*The Wild, Insite VR, Holodeck, CocoVerse*).

1.3.2 - APPROCCI DIFFERENTI

L'approccio riguardo la **comunicazione** tra gli utenti è differente nei vari competitor analizzati. *The Wild* e *CocoVerse* utilizzano un **approccio testuale**, dando la possibilità a chi accede tramite VR di poter scrivere i messaggi tramite uno **speech to text**. Sempre testuale è l'approccio di Insite VR, che permette agli utenti desktop di scrivere **commenti** che vengono **fisicamente aggiunti all'ambiente** e resi visibili a tutti gli utenti. Tuttavia, non è chiaro quanti dei competitor utilizzino le varie feature sopra descritte come aggiunta alla comunicazione vocale o se quelle risultino l'unico mezzo di comunicazione presente.

Anche nel caso dell' l'**interfaccia** si utilizzano approcci differenti, alcuni riconducibili al mondo 2D di pc desktop e mobile, altri che sfruttano maggiormente la nuova realtà 3D della VR.

Holodeck utilizza **pannelli** richiamati ad hoc dall'utente, che si aprono di fronte all'avatar simulando l'utilizzo di tablet che compaiono secondo l'utilità. Per quanto riguarda *BMW* abbiamo sempre un interfaccia richiamata ad hoc, simile ad un **disco diviso in sezioni** e interattivo alle gesture dell'utente.



FIGURA 1.18: BMW RADIAL MENU E HOLODECK NVIDIA MENU A FINESTRA

Per l'interazione con l'ambiente spesso vengono usati dei **puntatori laser**, posto uno per mano, che danno all'utente un feedback immediato di ciò che sta indicando e permettono di agire direttamente sull'oggetto a seconda dello strumento selezionato. Nel caso di Holodeck questi sono sempre visibili e rafforzano anche la comunicazione vocale fungendo da indicatori.

Interessante il caso di CocoVerse, la cui interfaccia è caratterizzata da una **toolbelt**, ovvero una sorta di cintura virtuale posta all'altezza della vita dell'utente, in cui sono disposti vari tool, i quali possono essere afferrati e richiamati facilmente con una gesture simile a quella reale di prendere qualcosa appeso alla cintura. Così facendo non si va ad interferire eccessivamente con la visuale, incoraggiando un approccio hands-free quando non si utilizzano tool specifici.

Ci interessava poi soffermarci brevemente su quelle feature che contraddistinguono un particolare competitor da tutti gli altri, perché proprio da queste nascono le migliori idee per sviluppare qualcosa di completamente nuovo che ampli il panorama delle possibili modalità di interazione scoperte finora.

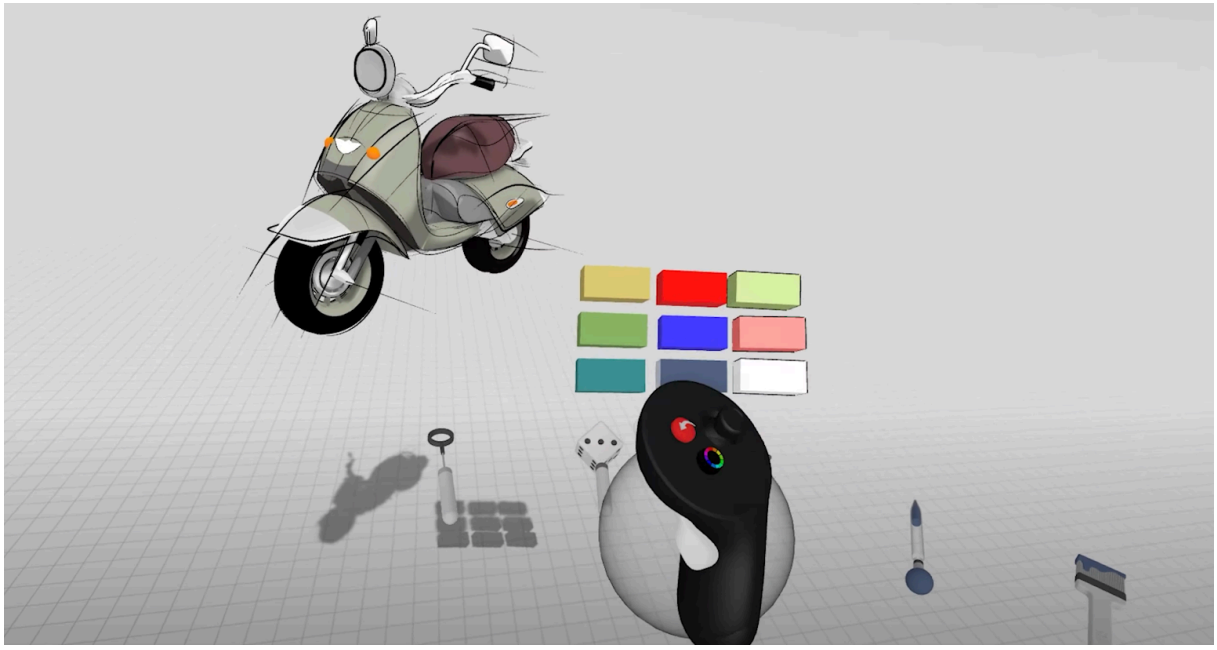


FIGURA 1.19: TOOLBELT E MENU IN 3D DI GRAVITY SKETCH

Insite VR utilizza una sorta di **master player**, con più "potere" all'interno della scena, in grado di poter guidare e supportare gli altri utenti base meno esperti durante l'esperienza
BMW racconta di aver realizzato un **software modulare e scalabile**, in modo che sia facile sostituire un modello di automobile con uno nuovo in qualunque momento in caso vi fosse necessità e con tempi di sviluppo molto ridotti.



FIGURA 1.20: CAMBIO AMBIENTAZIONE DI HOLODECK NVIDIA

Questo tipo di approccio, per quanto non ne siamo certi, sembra sia stato adottato anche da All Thing Media, in quanto vengono mostrate le stesse interazioni in ambienti e con oggetti differenti.

Holodeck è il competitor che mostra più feature in assoluto, aggiungendo a quanto già descritto la possibilità di **disegnare in 3D** all'interno dell'ambiente, sia su appositi pannelli che nello spazio circostante o su degli oggetti. Ha inserito poi una sorta di **osservazione a X-Ray**, che mette l'utente nella condizione di vedere attraverso gli oggetti focalizzando l'interesse su una parte specifica della mesh in questione. Offre poi la possibilità di **suddividere un oggetto** in tutti i suoi componenti con una funzione Explode e di **interagire** singolarmente con ognuno di essi, ma anche quella di **prendere misure** all'interno della scena, sia lineari che angolari, e di **cambiare ambientazione**, osservando l'effetto di luci differenti sull'oggetto.

Nota: Holodeck è stato utilizzato per didattica a distanza per gli studenti di H-International School, in questo periodo di pandemia da Covid-19 (2020).

2 - LOGICA DI FUNZIONAMENTO DELL'APPLICATIVO E MEZZI UTILIZZATI

La struttura del nostro applicativo è divisa in tre blocchi distinti, ognuno adibito alla costruzione di una precisa sezione delle nostre UI. Questi blocchi possiamo definirli come: ***Tool, Configuratore*** e ***Design.***

La struttura di base del progetto, come la gestione della navigazione, la logica delle sessioni, gli avatar e alcuni degli strumenti messi a disposizione nel nostro applicativo, sono presi dal ***Collab Viewer Template*** di Unreal Engine. Come verrà spiegato in seguito, abbiamo cercato, nel limite del possibile, di non apportare alcuna modifica diretta al template in questione, per rendere il nostro programma il più indipendente possibile da esso, pur sfruttando le sue funzionalità. Ragion per cui è stata inserita una classe adibita a comunicare direttamente con esso, denominata ***DP_Interface***, che è anche l'elemento chiave per collegare l'intera struttura del *Collab Viewer* con la classe ***HUD_UI_Logic*** da noi implementata**.** Quest'ultima è al centro della creazione delle UI, della gestione del passaggio da modalità Desktop a VR e viceversa, e infine del funzionamento dei vari ***Tool***, recuperati dal template e di conseguenza disponibili negli omonimi menu delle nostre Interfacce Utente.

Il ***Configuratore*** è la parte di struttura dell'applicazione che gestisce l'omonima sezione delle UI, volta a offrire all'utente i mezzi per customizzare il prodotto in scena. I dati del configuratore devono essere inseriti da uno sviluppatore, o da un designer, in un *Variant Manager,* strumento di Unreal Engine che permette di gestire le fasi di configurazione di un modello e le relative varianti. Considerando la complessità che può raggiungere un Configuratore per veicoli, le funzionalità di base del Variant Manager messe a disposizione da Unreal Engine vengono ampliate da un Plugin creato da Dead Pixels. Esso sfrutta una classe ***VT_Manager*** per riordinare le informazioni inserite nel Variant Manager creando un'apposita *Data Table* con i dati appena prelevati. Essa offre inoltre la possibilità di aggiungere informazioni aggiuntive a quelle presenti nel Variant Manager attraverso una interfaccia utente fornita dal plugin stesso. Sempre attraverso il *VT_Manager* si possono recuperare poi agilmente i dati inseriti in Data Table e passarli adeguatamente alle UI.

Infine, il terzo blocco è quello adibito alla gestione del ***Design grafico*** dell'applicazione, che mette lo sviluppatore o il designer in questione nelle condizioni di poter customizzare a pro-

prio piacimento alcune sue parti. Tutte le risorse personalizzate introdotte vanno a modificare lo stile delle UI e sono inseribili nel programma attraverso un apposito ***Editor UI*** da noi creato, che ha il compito di memorizzarle in due *Data Asset*, uno per ogni UI. In questo modo l'intero design grafico dell'applicazione viene reso facilmente modificabile. I dati all'interno dei *Data Asset* vengono letti da *HUD_UI_Logic* ed ogni sezione delle UI reperisce da essa le informazioni di suo interesse.

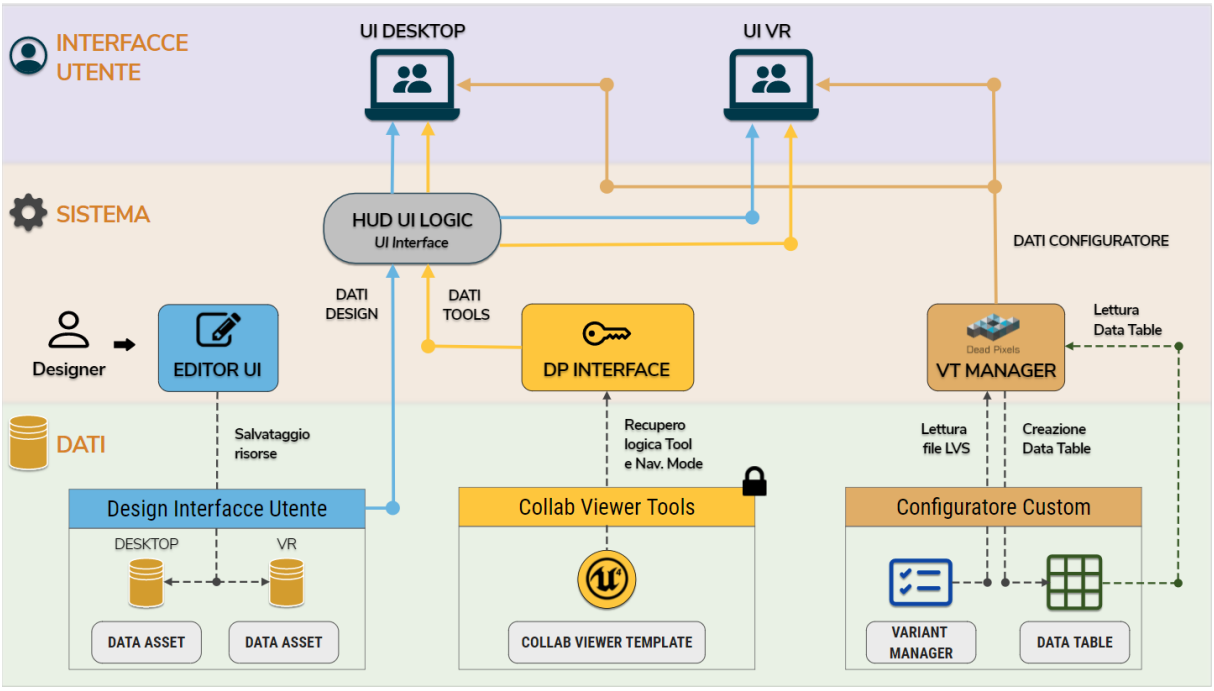


FIGURA 2.1: DIAGRAMMA DI ARCHITETTURA SOFTWARE

Come si può notare dal diagramma in figura **2.1**, ognuno dei tre blocchi agisce sulla costruzione di precise sezioni delle due UI progettate. Sia per la UI Desktop che quella VR, il flusso di dati derivante dal blocco *Design*, passando attraverso l'*HUD_UI_Logic*, va a modificare il design grafico di tutte le componenti di entrambe, conferendo uniformità di stile a tutte le sezioni. D'altro canto gli altri due blocchi forniscono i dati necessari alla corretta costruzione dei vari menu presenti: dal *Collab Viewer* viene ripresa la logica di funzionamento dei Tool e del cambio di navigazione, che vengono entrambe rese disponibili alle UI sempre dalla l'*HUD_UI_Logic*; grazie alla classe *VT_Manager* invece vengono recuperati i dati del configuratore memorizzati nella Data Table da essa creata, in modo da popolare correttamente il menu di configurazione. La struttura delle UI è mostrata nel diagramma in figura **2.2** e tutte le sezioni, con le relative scelte di progettazione, verranno approfondite nel capitolo **3**, riservato alla UI Desktop, e nel capitolo **4**, adibito alla UI VR.

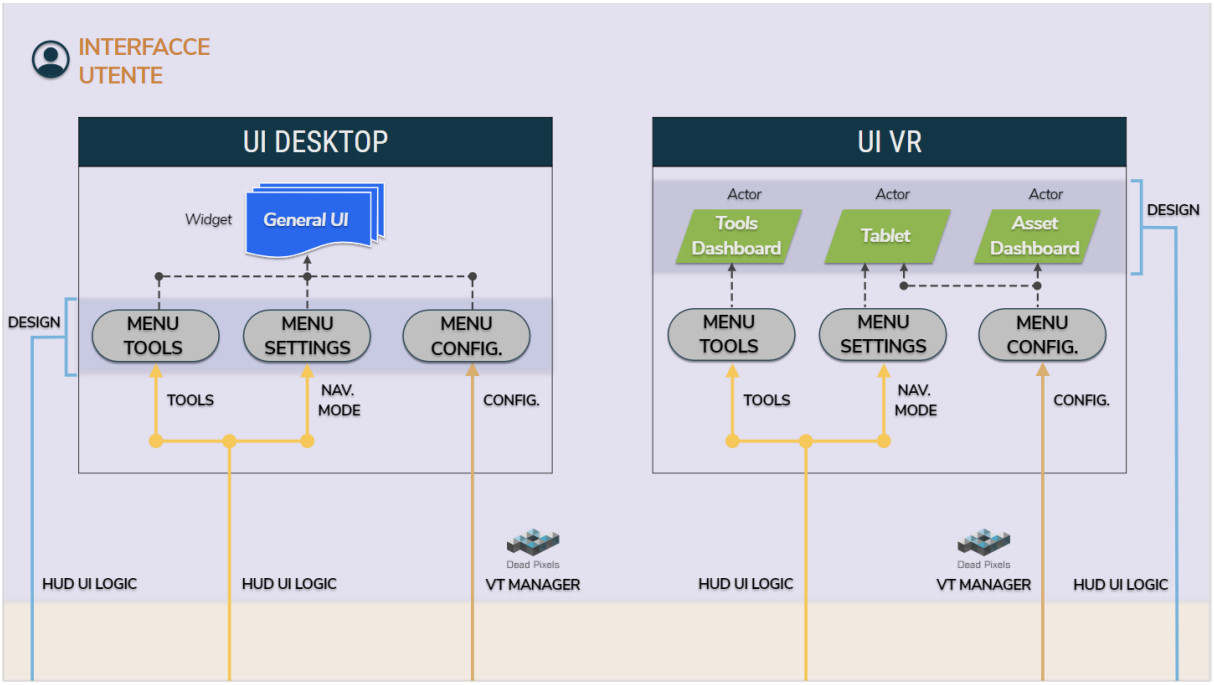


FIGURA 2.2: DIAGRAMMA DELLA STRUTTURA GENERALE DELLE DUE INTERFACCE UTENTE

2.1 - UNREAL ENGINE E STRUMENTI UTILIZZATI

La scelta di utilizzare Unreal Engine, software proprietario della **Epic Games**[34], deriva da molteplici fattori, non solo dal fatto che i membri del team di Dead Pixels lavorano attivamente con questo motore grafico già da diversi anni. Unreal Engine è il motore di gioco più utilizzato per la realizzazione di applicazioni di pre-visualizzazione architettonica e di revisione di un prodotto industriale. È grazie al suo estremo **fotorealismo** e al fatto che tutte le sue funzionalità e potenzialità sono accessibili **gratuitamente** da chiunque, che Unreal Engine, ora arrivato alla versione numero 4, si è costruito nel corso degli anni il proprio successo e la propria fama. La presenza poi di un editor visuale intuitivo, il cosiddetto **Blueprint Editor**, permette anche a coloro che non hanno conoscenze di programmazione in linguaggio C++ di progettare applicativi senza scrivere alcuna riga di codice. Offre alla sua community la possibilità di scaricare una serie pressoché infinita di kit di sviluppo, template e asset di alta qualità dal **Marketplace**, aiutando i suoi utenti meno esperti o i piccoli studios a costruire ambienti qualitativamente impressionanti con il minimo sforzo possibile.

2.1.1 - COLLAB VIEWER TEMPLATE

Ma il motivo principale che ci ha spinti ad utilizzare il suddetto motore grafico è stato la scoperta del template *"Collab Viewer"*, reso disponibile solo di recente all'avvio del UE4 Launcher. Ha rappresentato per noi un ottimo punto di partenza, un suolo fertile su cui costruire

funzione dopo funzione le interfacce del nostro plugin. Ci è stato di particolare aiuto nei primi mesi di sperimentazione nella comprensione delle logiche basilari di funzionamento del software, soprattutto quelle riguardanti la creazione e la gestione di interfacce desktop e VR, e ci ha fornito una struttura base solida che già presentava i componenti base necessari all'implementazione d'interfacce utente.



FIGURA 2.3: LIVELLO BASE DEL COLLAB VIEWER TEMPLATE

Collab Viewer[35] è integrabile in qualsiasi progetto Unreal Engine ed offre una varietà di strumenti allo sviluppatore per creare esperienze virtuali multi-utente all'interno delle quali è possibile interagire in real-time su un medesimo modello 3D. Mette a disposizione di default un livello liberamente navigabile ed interattivo, nonché una serie di strumenti che permettono all'utente di interagire con gli oggetti presenti in scena in real-time. Dopo aver associato ad ogni player un avatar che lo identifica all'interno della sessione, l'utente avrà piena libertà di muovere gli oggetti presenti nell'ambiente; rendere invisibili i materiali di qualunque oggetto aiutandolo ad ispezionare sezioni di esso normalmente non visibili perché occluse; utilizzare una funzionalità denominata "Explode" che separa un oggetto nelle sue componenti singole distribuendole nello spazio circostante per permetterne una interazione più precisa e mirata; generare scie di colore dal VR motion controller come se fosse un pennarello e scrivere quindi note nello spazio circostante, e molto altro ancora.

Ogni player potrà spostarsi nell'ambiente grazie ad una modalità di teletrasporto attivabile da controller e utilizzare un laser pointer incorporato sempre nel motion controller, sia per far comprendere agli altri utenti quale oggetto d'interesse stia visionando un determinato player, ma anche per utilizzare al meglio i tool disponibili indirizzando il laser, quindi il motion controller in sé, direttamente verso l'oggetto (mesh o actor che sia) con cui si vuole interagire.

Il Collab Viewer gestisce la maggior parte dei problemi inerenti alle esperienze multi-utente, tra cui l'impostazione delle connessioni tra server e client e la trasmissione delle informazioni sulla presenza di più utenti tra più computer. Funge da punto di partenza per tutti quei progetti che si prefiggono di realizzare esperienze di revisione o customizzazione del design di un prodotto, in modo da dar la possibilità agli sviluppatori di poter dedicare meno tempo alla configurazione della rete e più tempo invece alla customizzazione dell'esperienza.

2.1.2 - WIDGET BLUEPRINT E EDITOR UTILITY WIDGET

Partendo dal template Collab Viewer, abbiamo deciso di riutilizzare alcuni tool già in esso implementati, ri-adattandoli in base alle nostre esigenze, e di implementare nuove feature che siano personalizzabili dallo sviluppatore o dal designer in fase di creazione del progetto. Uno degli strumenti che permetterà al designer di customizzare a proprio piacimento la grafica e lo stile delle UI sarà un'apposita interfaccia, richiamabile da editor, da noi creata sfruttando uno dei nuovi tool messi a disposizione da Unreal Engine 4.
Le User Interface in Unreal Engine, sono basate su **Unreal Motion Graphics** (UMG) e ogni interfaccia o parte di essa può essere costruita attraverso **Widget Blueprints** (abbreviato Widget o WB). I Widget servono a costruire le UI da utilizzare runtime, ma di recente Unreal ha messo a disposizione un nuovo strumento, chiamato **Editor Utility Widget** (abbreviato EUW) che invece serve a modificare la UI stessa del programma, attraverso la creazione di apposite finestre con interfacce personalizzate. Gli EUW si basano anch'essi su UMG per cui la loro creazione e utilizzo è pressoché identica a qualunque altro Widget. Una volta create, queste finestre sono facilmente richiamabili e posizionabili ovunque nella UI come qualunque altra finestra presente di default.
Intendiamo sfruttare le potenzialità di queste interfacce per consentire allo sviluppatore o al designer di modificare rapidamente la grafica di quasi tutti gli oggetti in scena, senza andare a modificare direttamente le *Blueprint Class*: classi predefinite contenenti diverse funzionalità

che descrivono tutta la logica del plugin. Evitando allo sviluppatore, ma soprattutto al designer, il tedioso lavoro di comprensione della struttura di base del plugin, nonché della logica che governa le varie Blueprint Class, il processo di customizzazione risulterà immediato e di più facile comprensione, sia per chi non conosce la logica di funzionamento del progetto, ma soprattutto per chi non ha conoscenze di programmazione di alcun genere.

2.1.3 - VARIANT MANAGER

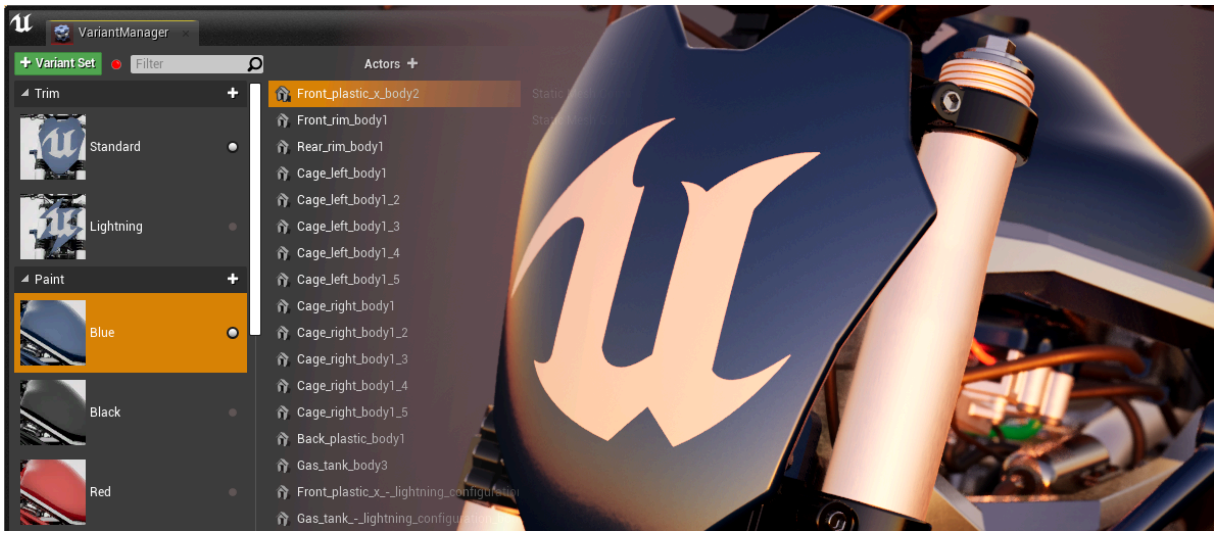


FIGURA 2.4: UE4 VARIANT MANAGER

Come dice il nome stesso dell'interfaccia, il compito del Variant Manager è di aiutare lo sviluppatore a creare e gestire diverse configurazioni, o *varianti*, di uno o più Attori (o un gruppo di mesh) all'interno di un Livello. È suddiviso in **Variant Set** *(colonna 1 dell'immagine)* che raggruppano le varianti in categorie in base al cambiamento di una o più proprietà (*colonna 3)* scelte dallo sviluppatore (materiale, geometria, dimensione, etc...). Ogni variante controlla a suo modo una versione differente di un insieme di proprietà di uno o più Attori *(colonna 2)* ed è attivabile o disattivabile dinamicamente sia da codice che da Blueprint in qualunque istante. Di fatto permette di cambiare velocemente la configurazione di una mesh o di un Attore togliendo e/o aggiungendo la visibilità di una variante in base alle scelte dell'utente.



FIGURA 2.5: STRUTTURA DELL'INTERFACCIA DEL VARIANT MANAGER

2.2 - LOGICA DATA DRIVEN

Ci siamo preposti, tra i vari obiettivi, di rendere **modulari** le interfacce che costruiremo, in modo da essere facilmente personalizzabili a seconda della necessità o del cliente per cui si vorrà realizzare il configuratore. Quindi le interfacce saranno costruite in modo da definire al meglio la struttura, che funga da "contenitore" per immagini, testi, colori, materiali e mesh che il designer andrà ad inserire.
Per cui diventa logico, in questo contesto, costruire una applicazione, e dunque anche le relative interfacce, che sia **data-driven**[36]. L'approccio dunque sarà quello non di costruire logiche personalizzate per ogni parte dell'applicazione, con valori codificati, ma di strutturare l'intera logica delle UI in modo tale da poter accogliere immagini, colori e stili differenti, pur continuando a mantenere un comportamento coerente.
Risulta quindi necessario memorizzare correttamente i dati inseriti dal designer per poterli recuperare correttamente al momento opportuno. Per fare ciò, Unreal Engine mette a disposizione due strumenti, differenti tra loro per alcune caratteristiche, che li rendono più o meno adatti a seconda del contesto: Data Table e Data Asset.

2.2.1 DATA TABLE

Le **Data Table** (abbreviato DT), come si evince dal nome stesso, sono delle strutture dati sotto forma tabellare. Esse hanno la peculiarità di poter accogliere al loro interno dati di diverso tipo (String, Integer, Boolean, Array, Struct, ecc...) e di poterli raggruppare sotto una stessa logica mettendoli in relazione tra loro in modo utile e significativo.
Possono essere popolate manualmente o partendo da file e strutture esterne, come CSV o JSON, ma è fondamentale che prima di compiere questa operazione venga definita la ***RowStruct***, ovvero la *Struct* che definisce la composizione di ogni singola riga che andrà a popolare la tabella. Essa è fondamentale poiché informa il programma su come vanno interpretati i dati che saranno posti all'interno della DT.
La prima colonna deve essere nominata *ROW NAME* e contenere i nomi che fungono da chiavi attraverso le quali, mediante la funzione apposita *DataTableRowHandle,* si può accedere in modo univoco a ciascuna riga. Conseguentemente a quanto descritto, le colonne avranno come nome l'heading definito nella Struct e sotto, nella stessa colonna, il dato relativo all'intersezione della stessa con la riga opportuna.

Le DT sono progettate per poter gestire e confrontare grandi quantità di dati simili, tuttavia sono strutture statiche che possono essere modificate esclusivamente in fase di programmazione ma non in run-time e che non possono essere gestite con gerarchie e relative ereditarietà.

Per facilitare l'inserimento o la modifica manuale dei campi presenti nella tabella, che possono risultare anche piuttosto complessi nel caso si tratti, ad esempio, di *Array* o *Struct*, cliccando su ogni riga, che sia nuova o già presente, si apre nella parte della finestra sottostante la DT il relativo **Row Editor** che riporta, per la riga selezionata, i vari campi della *Row Struct* sotto forma di elenco, con a sinistra il nome esplicativo di quel campo e a destra il valore dello stesso, con possibilità di modificarlo in qualunque momento.

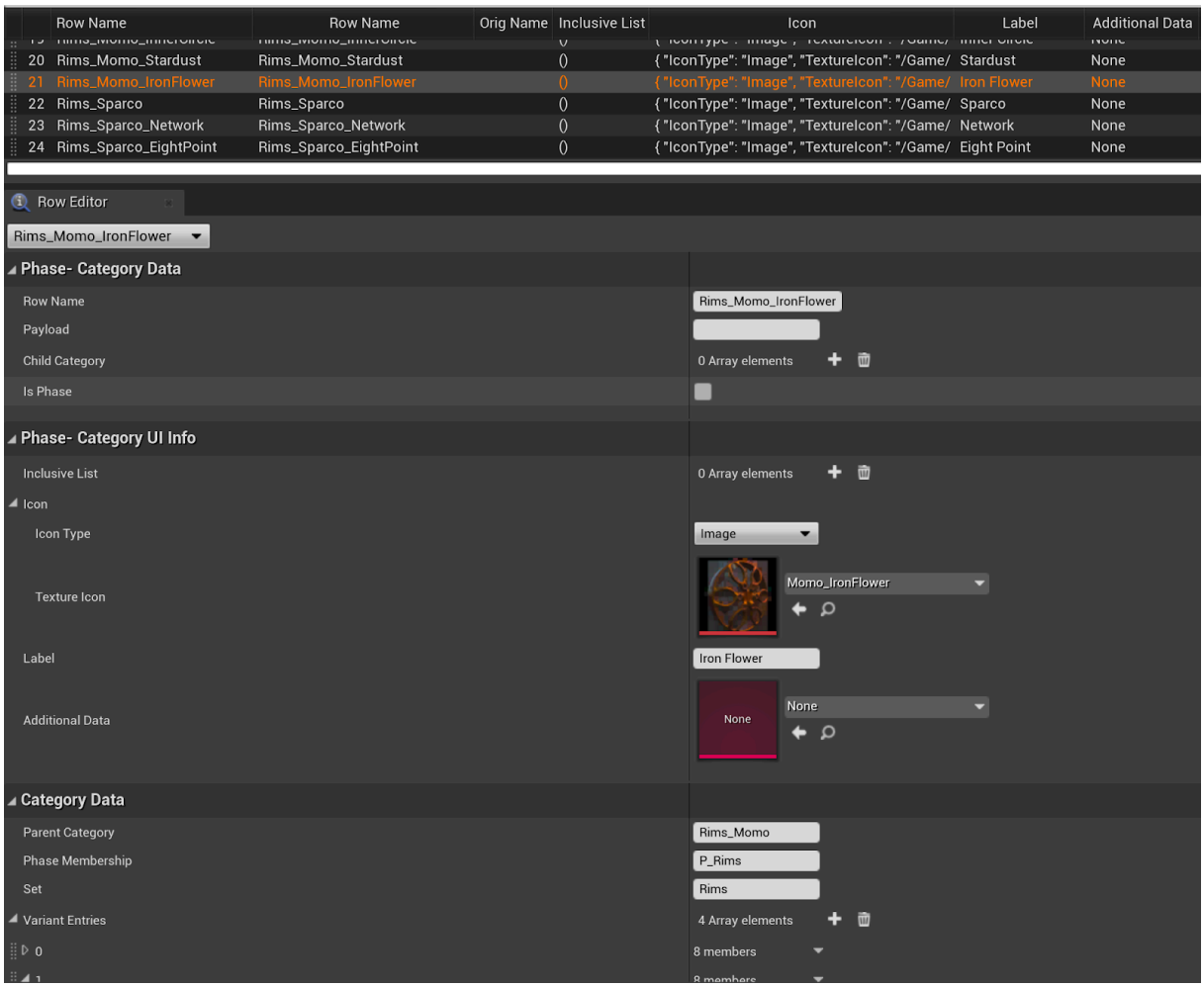
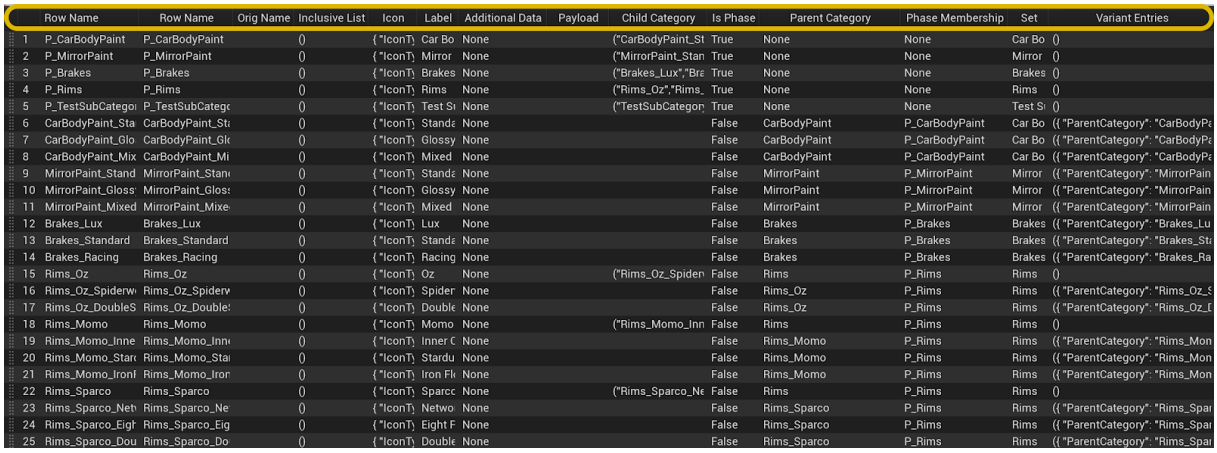


FIGURA 2.6: ROW EDITOR DELLA DT "LAMBORGHINI_DATATABLE_NEW" UTILIZZATA COME ESEMPIO

I *Row Editor* possono essere di lettura molto semplice o, al contrario, di difficile comprensione se contenenti una gran moltitudine di dati, come quello in figura **2.6**; dipende tutto quindi da come è costruita la DT.

Nel nostro caso in particolare, come verrà spiegato meglio in seguito, la DT viene generata automaticamente dal Plug-in a partire dal Variant Manager e utilizzata poi per gestire tutte le informazioni riguardanti il configuratore, con tutte le relative varianti e categorie. I campi della *RowStruct* della nostra DT sono quelli visibili nell'immagine **2.7**, dei quali rimandiamo una approfondita spiegazione nel sottocapitolo **2.4.2**.



	Row Name	Row Name	Orig Name	Inclusive List	Icon	Label	Additional Data	Payload	Child Category	Is Phase	Parent Category	Phase Membership	Set	Variant Entries
1	P_CarBodyPaint	P_CarBodyPaint		()	{"IconTy	Car Bo	None		("CarBodyPaint_St	True	None	None	Car Bo	()
2	P_MirrorPaint	P_MirrorPaint		()	{"IconTy	Mirror	None		("MirrorPaint_Stan	True	None	None	Mirror	()
3	P_Brakes	P_Brakes		()	{"IconTy	Brakes	None		("Brakes_Lux","Bra	True	None	None	Brakes	()
4	P_Rims	P_Rims		()	{"IconTy	Rims	None		("Rims_Oz","Rims_	True	None	None	Rims	()
5	P_TestSubCategor	P_TestSubCategc		()	{"IconTy	Test St	None		("TestSubCategon	True	None	None	Test St	()
6	CarBodyPaint_Sta	CarBodyPaint_St		()	{"IconTy	Standa	None			False	CarBodyPaint	P_CarBodyPaint	Car Bo	{{ "ParentCategory": "CarBodyPa
7	CarBodyPaint_Glo	CarBodyPaint_Gl		()	{"IconTy	Glossy	None			False	CarBodyPaint	P_CarBodyPaint	Car Bo	{{ "ParentCategory": "CarBodyPa
8	CarBodyPaint_Mix	CarBodyPaint_Mi		()	{"IconTy	Mixed	None			False	CarBodyPaint	P_CarBodyPaint	Car Bo	{{ "ParentCategory": "CarBodyPa
9	MirrorPaint_Stand	MirrorPaint_Stan		()	{"IconTy	Standa	None			False	MirrorPaint	P_MirrorPaint	Mirror	{{ "ParentCategory": "MirrorPain
10	MirrorPaint_Gloss	MirrorPaint_Glos		()	{"IconTy	Glossy	None			False	MirrorPaint	P_MirrorPaint	Mirror	{{ "ParentCategory": "MirrorPain
11	MirrorPaint_Mixed	MirrorPaint_Mixe		()	{"IconTy	Mixed	None			False	MirrorPaint	P_MirrorPaint	Mirror	{{ "ParentCategory": "MirrorPain
12	Brakes_Lux	Brakes_Lux		()	{"IconTy	Lux	None			False	Brakes	P_Brakes	Brakes	{{ "ParentCategory": "Brakes_Lu
13	Brakes_Standard	Brakes_Standard		()	{"IconTy	Standa	None			False	Brakes	P_Brakes	Brakes	{{ "ParentCategory": "Brakes_Sta
14	Brakes_Racing	Brakes_Racing		()	{"IconTy	Racing	None			False	Brakes	P_Brakes	Brakes	{{ "ParentCategory": "Brakes_Ra
15	Rims_Oz	Rims_Oz		()	{"IconTy	Oz	None		("Rims_Oz_Spiden	False	Rims	P_Rims	Rims	()
16	Rims_Oz_Spiderw	Rims_Oz_Spiderv		()	{"IconTy	Spider	None			False	Rims_Oz	P_Rims	Rims	{{ "ParentCategory": "Rims_Oz_S
17	Rims_Oz_DoubleS	Rims_Oz_Double!		()	{"IconTy	Double	None			False	Rims_Oz	P_Rims	Rims	{{ "ParentCategory": "Rims_Oz_D
18	Rims_Momo	Rims_Momo		()	{"IconTy	Momo	None		("Rims_Momo_Inn	False	Rims	P_Rims	Rims	()
19	Rims_Momo_Inne	Rims_Momo_Inn		()	{"IconTy	Inner C	None			False	Rims_Momo	P_Rims	Rims	{{ "ParentCategory": "Rims_Mon
20	Rims_Momo_Star	Rims_Momo_Sta		()	{"IconTy	Stardu	None			False	Rims_Momo	P_Rims	Rims	{{ "ParentCategory": "Rims_Mon
21	Rims_Momo_Ironf	Rims_Momo_Iron		()	{"IconTy	Iron Fl	None			False	Rims_Momo	P_Rims	Rims	{{ "ParentCategory": "Rims_Mon
22	Rims_Sparco	Rims_Sparco		()	{"IconTy	Sparcc	None		("Rims_Sparco_Ne	False	Rims	P_Rims	Rims	()
23	Rims_Sparco_Net	Rims_Sparco_Ne		()	{"IconTy	Netwo	None			False	Rims_Sparco	P_Rims	Rims	{{ "ParentCategory": "Rims_Spar
24	Rims_Sparco_Eigh	Rims_Sparco_Eig		()	{"IconTy	Eight F	None			False	Rims_Sparco	P_Rims	Rims	{{ "ParentCategory": "Rims_Spar
25	Rims_Sparco_Dou	Rims_Sparco_Do		()	{"IconTy	Double	None			False	Rims_Sparco	P_Rims	Rims	{{ "ParentCategory": "Rims_Spar

FIGURA 2.7: DT DI ESEMPIO CON *ROWSTRUCT* COME DEFINITA DAL PLUGIN

2.2.2 DATA ASSET

I **Data Asset** (abbreviato DA) sono istanze di classi appositamente definite, chiamate *Primary Data Asset Class*, al cui interno si definiscono i tipi di dati che le varie istanze della stessa potranno contenere. La struttura dunque è fissa e dichiarata nella classe di base, mentre i dati all'interno dei vari Data Asset sono modificabili, sia durante la programmazione che in runtime. I DA dunque sono a tutti gli effetti degli oggetti definiti da una classe, ognuno di loro potrebbe rappresentare una singola riga di una Data Table, ma a differenza di una *Struct* essi hanno tutte le proprietà che può avere una classe, come la possibilità di costruire gerarchie e definire un sistema di ereditarietà tra classi Data Asset, ma anche definire metodi al proprio interno che siano richiamabili esternamente alla classe e che ne definiscano il comportamen-

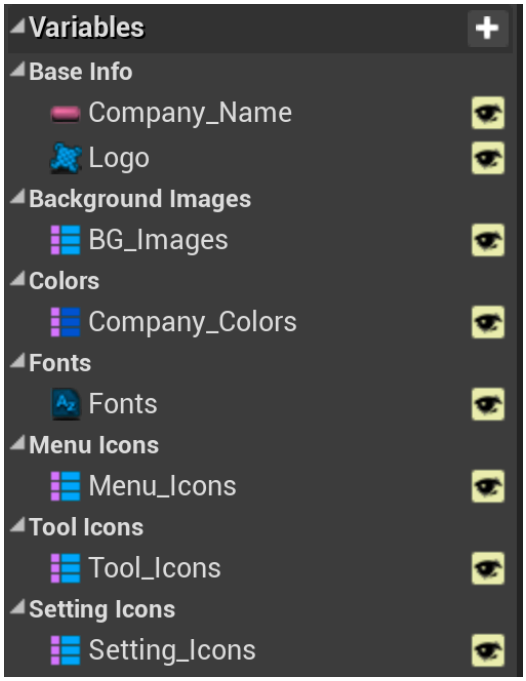


FIGURA 2.8: VARIABILI DEFINITE NELLA *PRIMARY DATA ASSET CLASS* DEL DA USATO NELLA UI DESKTOP

to. In particolare, oltre a poter richiamare i dati al suo interno con semplici funzioni *Get*, anche la scrittura su di essi mediante funzioni *Set* risulta molto semplice.
I Data Asset possono contenere qualunque tipologia di dato, dal più semplice *Integer* o *String*, fino ai più complessi *Array*, *Struct* o *Map*.

Una *Primary Data Asset Class* deve essere creata ogni qualvolta si voglia raggruppare un insieme di dati e da questa poi di solito seguono altri DA che la implementano. Facendo riferimento al nostro progetto, per raggruppare le informazioni grafiche che sarebbero andate a definire le varie interfacce utente, abbiamo definito una *Primary Data Asset Class* per la UI Desktop e una per la UI VR, le quali sono state poi implementate da altri due DA, ***DA_UI_Desktop*** e ***DA_UI_VR***, con tutte le informazioni necessarie per personalizzare l'interfaccia interessata. I DA possono essere popolati attraverso l'apposita interfaccia Editor, molto simile al Row Editor delle DT.

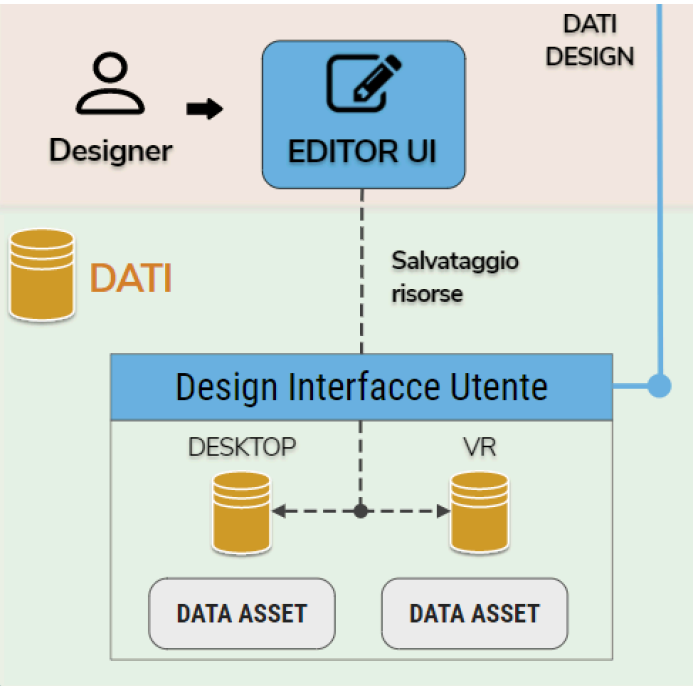


FIGURA 2.9: PORZIONE DEL DIAGRAMMA DI ARCHITETTURA SOFTWARE CON FOCUS SULLA SEZIONE DESIGN.

Come anticipato nella figura **2.8**, la struttura della *Primary Data Asset Class* della UI Desktop, e di conseguenza anche del *DA_UI_Desktop* che la eredita, è composta da:

- una variabile **Company_Name** di tipo *Text* ed una **Logo** di tipo *Texture2D* riservate al nome e al logo dell'azienda;
- una *Map* **BG_Images** composta da variabili *Name* come Key e *Texture2D* come Values che contiene le immagini destinate allo sfondo dell'interfaccia di Login alla sessione;
- una *Map* **Company_Colors** contenente i colori aziendali che verranno ripresi nelle interfacce;
- **Fonts** di tipo *Font Asset* con il/i font aziendali utilizzati nell'interfaccia;
- quattro variabili *Map* **Menu Icons, Tool Icons, Setting Icons** uguali per struttura a *BG_Images* che contengono tutte le icone che popolano il menu.

Come accennato in precedenza, i DA saranno utilizzati dal designer come database dei dati riguardanti lo stile delle UI. Non verrà richiesto però al designer di modificare direttamente i DA, in quanto è stata creata una apposita interfaccia che gli permetterà di inserire facilmente i dati di suo interesse, visualizzando immediatamente una preview di come essi si presenteranno nella UI.

La struttura del *Primary Data Asset Class* dell'interfaccia utente VR presenta invece 5 diverse *Map*: una **Company_Colors** molto simile a quella Desktop, con la sola differenza che sono state inserite anche delle variabili *Linear Color* per customizzare il layout grafico di alcune sezioni dei Menu, e quattro relative al contenimento di tutte le icone dei vari menu disponibili, denominate rispettivamente **Menu Icons**, **Tools Icons**, **Settings Icons** e **Configurator Icons**.

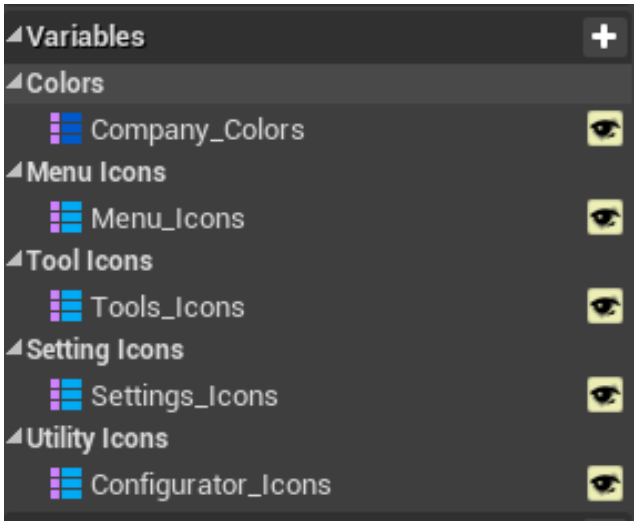


FIGURA 2.10: VARIABILI DEFINITE NELLA *PRIMARY DATA ASSET CLASS* DEL DA USATO NELLA UI VR

2.2.3 CONFRONTO E RAGIONI DI UTILIZZO

Alla luce di tutto ciò è chiaro come le DT siano comode per immagazzinare grandi quantità di dati, ma sono anche strutture statiche, accessibili in lettura in qualsiasi momento ma assolutamente non modificabili. D'altro canto i DA sono istanze di una classe, per cui per quanto siano generalmente scomodi per catalogare grandi quantità di dati, hanno il vantaggio di essere facilmente modificabili e possono sfruttare tutte quelle logiche di ereditarietà di norma precluse alle DT.
Per cui, nel nostro caso specifico, l'utilizzo di una DT risulta ottimo per immagazzinare i dati riguardanti il configuratore, poiché le informazioni legate ad ogni singola categoria o variante non verranno mai modificate in runtime. La DT può essere aggiornata dal plugin in fase di programmazione grazie ad apposite funzioni definite e scritte in C++ all'interno di esso, che di fatto cancellano la tabella precedente sostituendola con una nuova, permettendo da interfaccia editor un aggiornamento della stessa; azione non eseguibile da Blueprint e in ogni caso non permessa in runtime. In questo caso non si è scelto di usare i DA poiché le informazioni

di configurazione non sono quantificabili a priori, potendo potenzialmente creare anche centinaia di righe di tabella. Caso differente riguarda invece le interfacce che, nel caso peggiore, possono occupare al massimo tre DA, contando anche una versione mobile non studiata in questo progetto. Per cui le interfacce possono contenere si una grande moltitudine di dati, ma questi sarebbero allo stesso tempo gestibili in due, massimo tre DA, indubbiamente più comodi sia in lettura che in scrittura.

2.3 - EDITOR UTILITY WIDGET E PERSONALIZZAZIONE DELL'INTERFACCIA

Come anticipato nel sottocapitolo **2.1.2**, per semplificare il processo di personalizzazione delle interfacce abbiamo utilizzato un nuovo strumento messo a disposizione di recente da Unreal: l'***Editor Utility Widget***.

2.3.1 INTERFACCIA INTERATTIVA CON EDITOR UTILITY WIDGET

L'**Editor UI** è stato costruito in modo che il plugin possa essere usato anche da chi non ha conoscenze di programmazione in Unreal Engine, o anche solo di Unreal Motion Graphic, e in modo particolare dei campi necessari all'inserimento e/o modifica delle icone, immagini e colori nei Widget che descrivono l'interfaccia utente. Questo editor permette al designer di inserire facilmente i dati necessari alla costruzione delle varie interfacce che verranno salvati automaticamente in un Data Asset per poi essere ripresi dal programma da noi scritto per costruire l'interfaccia utente. Tutto ciò sarà fattibile senza che il designer debba preoccuparsi di come reperire il DA corretto o addirittura di come andare a modificare i Widget della UI Desktop o VR. L'ulteriore vantaggio che porta la presenza di questo Editor è che i dati inseriti possono essere riutilizzati in più sezioni delle varie UI, in Widget diversi, dando una maggiore uniformità di stile all'interfaccia utente. Se il logo viene riportato in N punti diversi del programma basterà modificarlo una sola volta nell'editor, allo stesso modo per il font o i colori scelti. Ciò è possibile poiché i dati vengono salvati in precise variabili dei DA a cui tutte le UI fanno riferimento.
L'editor da noi creato è suddiviso in tre parti.

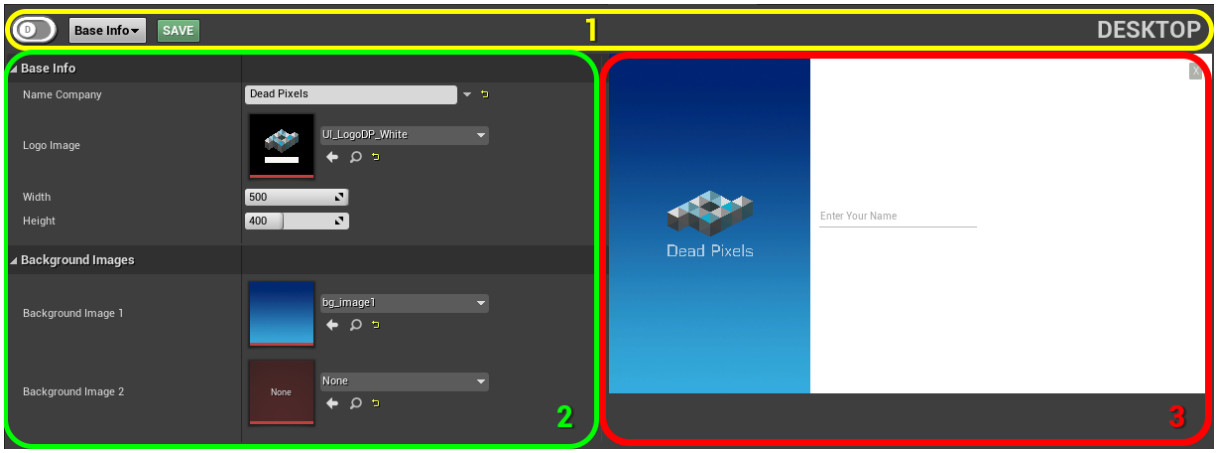


FIGURA 2.11: FINESTRA DELL'EDITOR UI IN ESECUZIONE

Nella parte superiore dell'Editor UI è presente un header (figura **2.11**, parte 1) con all'interno uno o più menu a tendina che danno la possibilità di accedere a diversi campi dell'editor, suddivisi per tipo, che nel complesso definiscono il layout grafico delle interfacce e accompagnano l'utente step by step, dividendo la personalizzazione in varie fasi logiche, senza presentare semplicemente una lunga lista di variabili a cui assegnare un valore. In questo modo riusciamo a sfruttare meglio lo spazio limitato a disposizione ed aiutare allo stesso tempo il designer durante la fase di customizzazione a visualizzare meglio la moltitudine di dati presentati, invitandolo a concentrarsi su una sezione per volta. A lato di questi è stato aggiunto uno *Switch Button* che permette di passare dalla sezione Desktop dell'Editor, di default visibile, a quella relativa alla UI VR e viceversa.
Le varie sezioni della UI Desktop sono state così nominate in modo da definire raggruppamenti secondo una logica comune: **Base Info** riguarda le informazioni base che caratterizzano un'azienda; **Company Style** sono le informazioni di stile come Font e Colori; **Icons** invece si suddivide a sua volta in altre tre sottocategorie di icone a seconda del menu in cui esse sono coinvolte, che sia il **Main Menu**, il **Tools Menu** o il **Settings Menu**.

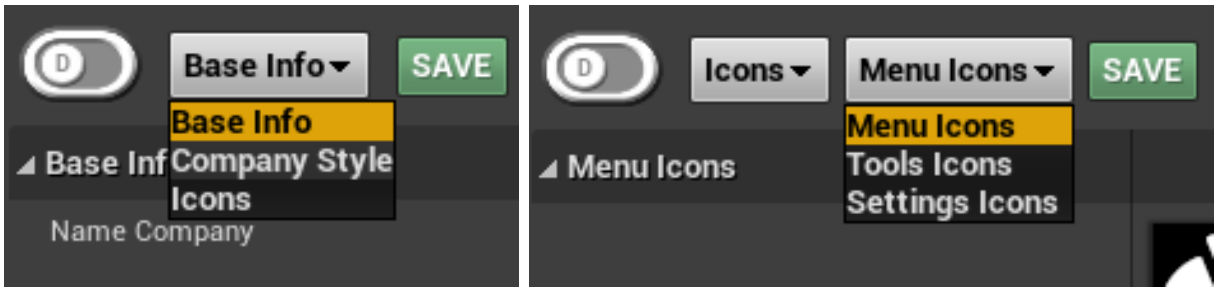


FIGURA 2.12: MENU A TENDINA NELLA TOOL BAR DELL'EDITOR UI

Nel riquadro n.2 verrà visualizzata la lista di tutte le variabili personalizzabili relative al campo scelto dal menu a tendina. Queste variabili, definite via Blueprint, vengono rese visibili, e quindi anche modificabili, nell'editor grazie all'utilizzo di un **Details View**, strumento messo a disposizione da Unreal e ampiamente usato nella sua stessa UI. Questo tool riporta il nome della variabile a sinistra, affiancando a destra un campo editabile a seconda del tipo di variabile. Come si può vedere nella figura 2.11 d'esempio, **Name Company** è una variabile di tipo *Text* e presenta infatti a destra un campo testuale editabile, così come a **Logo Image,** di tipo *Texture2D*, è associato un campo per inserimento di immagini, e così per tutte le altre.
Infine nel riquadro n.3 abbiamo deciso di mettere una preview dell'interfaccia. Questa scelta è data dalla necessità di capire dove andranno posti i vari dati inseriti e soprattutto la resa

grafica degli stessi una volta inseriti nella UI. Questo già è possibile in parte nella costruzione di un Widget con Unreal, seppur riteniamo non sia così efficace, dal momento che la zona dell'Editor Widget detta *Visual Designer* funge sia da preview che da strumento per definire la struttura della UI.

Inizialmente si erano inserite delle immagini con placeholder esplicativi della posizione in cui sarebbero stati posti i dati inseriti dal riquadro 2, ma seppur questa soluzione potesse bastare per informare in modo esaustivo il designer sulla posizione di questi nell'interfaccia utente, non dava alcuna informazione in merito di estetica. Poiché questo editor ha principalmente la funzione di guidare il designer nella definizione dello stile delle UI del plugin ci è presto parso chiaro che l'approccio scelto non era sufficiente.

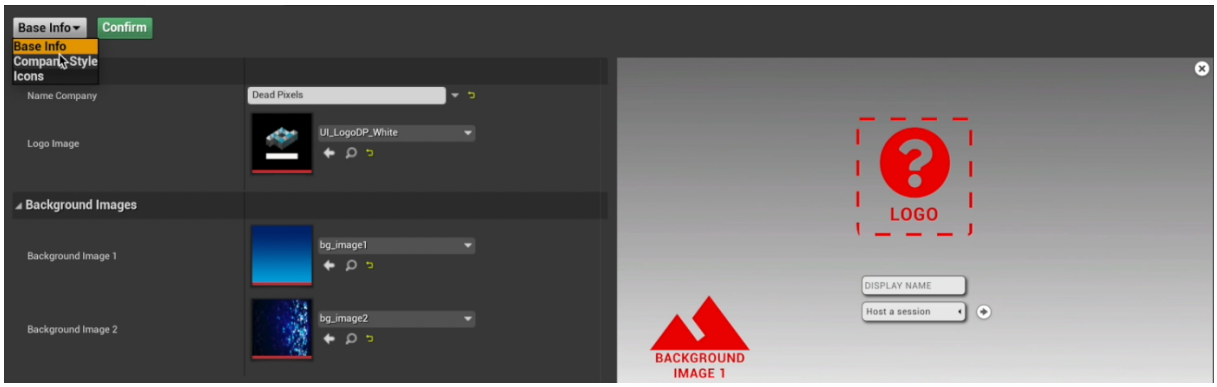


FIGURA 2.13: IMMAGINI DI PLACEHOLDER INSERITE NEL VECCHIO EDITOR

Poiché l'EUW possiede le stesse funzionalità dei Widget con cui abbiamo costruito le interfacce, sarebbe stato relativamente semplice per noi creare una preview interattiva con il layout grafico di tutte le interfacce disponibili. Infatti, riprendendo parte della struttura già realizzata per le UI, abbiamo realizzato tre preview interattive per la UI desktop, una per ogni gruppo di variabili da completare, e cinque per la UI VR.

Per quanto riguarda la sezione **Base Info** la preview riprende la struttura del *Widget_MainMenu*, adibito al menu iniziale di login per creazione e accesso alle sessioni, così come è stato definito all'interno del Collab Viewer (figura 2.11). Modificando i valori delle variabili nel Details View a sinistra, è possibile cambiare il logo e l'immagine di sfondo nella preview e quindi ovviamente anche nel menu iniziale a cui la preview fa riferimento. L'interattività con tale preview si limita all'inserimento di un nome nell'apposito campo e alla scelta, dal menu a tendina, della modalità di accesso alla sessione, con conseguente aggiornamento della preview a seconda della voce scelta. Ovviamente non è stata riportata tutta la logica di funzio-

namento sottostante, qui inutile, ma tutto ciò che riguarda l'aspetto grafico è totalmente fedele a ciò che si otterrà mandando in esecuzione il programma.
Come detto in precedenza, scegliendo il menu **Icons** si ha accesso a tre nuove sottocategorie di icone, ognuna relativa alla sezione dell'interfaccia in cui queste icone sono collocate. Cambiando sottocategoria dall'apposito menu, è subito visibile come la struttura della preview a destra rimanga pressoché invariata, se non per le differenti icone presenti nella nuova categoria scelta e la relativa scomparsa delle presenti in precedenza.

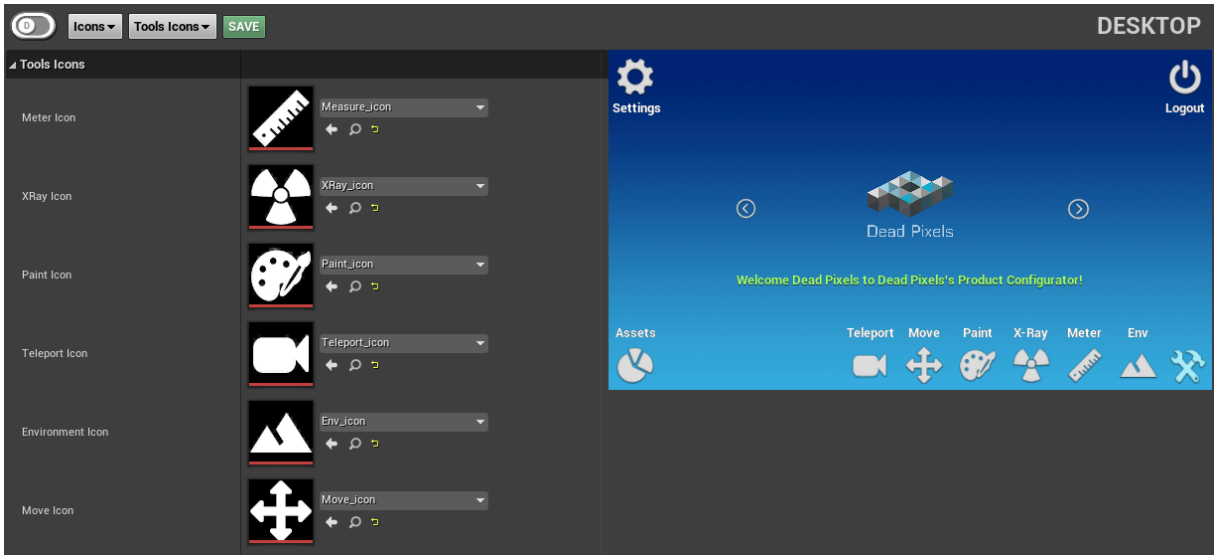


FIGURA 2.14: EUW SU MENU TOOLS ICONS E RELATIVA PREVIEW

In questo caso abbiamo volutamente aumentato la dimensione delle icone, poiché trattandosi di una Preview in spazio ristretto abbiamo voluto dare maggior risalto alla resa grafica e alla fedeltà di posizionamento piuttosto che alle effettive dimensioni. Per dare la possibilità al designer di verificare la resa grafica di quanto inserito in diverse situazioni d'illuminazione, abbiamo pensato di inserire la possibilità di cambiare sfondo, sottoponendo la UI a contesti limite di luce o buio. Inoltre, per comodità, l'interazione è stata progettata per essere bidirezionale, dunque se nella preview si clicca sulle icone dei Tool o dei Settings si aprirà la relativa sezione nel Details View a sinistra con tutte le variabili aggiornate, coerentemente con quanto modificato fino a quel punto.
Per le informazioni relative al **Company Style** si è deciso di adottare un approccio differente. Poiché queste sono informazioni che si andranno a ripercuotere ovunque nelle UI non ci sembrava sensato riportare la medesima struttura della preview vista per le altre due sezioni, ma viene mostrato solamente un bottone, con immagine di placeholder generica. Questo bottone reagisce esattamente come ogni altro bottone nella UI, mostrando i corretti colori a se-

conda delle interazioni. Ad esempio si è scelto di definire **Primary Color** il colore che assumono di default tutte le icone, **Secondary Color** il colore che assumono sull'evento *OnClicked* e l'**Accent Color** il colore che deve mettere in risalto l'oggetto, ad esempio su evento *OnHovered*. Il bottone nella preview reagisce all'interazione con il cambiamento di tali colori così come definiti nel rispettivo Details View e fornisce costantemente all'utente un feedback visivo, dato dal cambiamento degli slider, ma anche testuale, sotto l'icona, con il nome del colore relativo allo stato in cui si trova il bottone in quel preciso istante (Normal, Hovered, Clicked).

Poiché la scelta dei **Company Colors** può essere definita sia da standard aziendali come anche presentare necessità di modifica a seconda del contesto, abbiamo voluto dare la possibilità di modificare i tre principali colori sopradescritti non solo da Details View ma anche dalla preview stessa, scegliendoli dall'apposito menu a tendina.

Nel caso di colori aziendali predefiniti si ha la possibilità di riportare il valore esadecimale direttamente nell'apposito campo e dal bottone sottostante ad esso è possibile salvarsi una palette di colori, così che il designer possa averli sempre a disposizione per ogni evenienza.

Si è comunque sempre liberi di scegliere e creare i propri colori, grazie alla presenza di *Slider* che controllano una specifica proprietà di essi e che nel complesso ne definiscono il valore RGB o HSV. Tutti gli slider presenti sono interattivi e riflettono il cambiamento della proprietà controllata sia nella preview che nel Details View, dando al designer piena libertà di scegliere l'approccio per lui più opportuno per scegliere e modificare i colori.

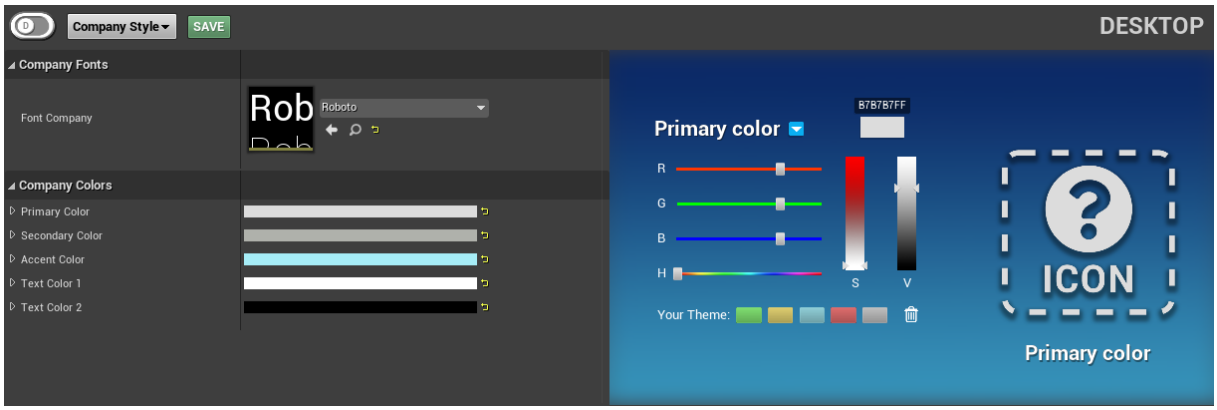


FIGURA 2.15: EUW SU MENU COMPANY STYLE E RELATIVA PREVIEW

Le modalità con cui è possibile customizzare l'interfaccia VR sono pressoché le medesime della UI Desktop, con la possibilità quindi di personalizzare ogni icona presente nei vari menu mediante gli appositi Details View. Anche in questo caso sono state inserite delle preview interattive che notificano visivamente al designer ogni cambiamento apportato nell'Edi-

tor, che si rifletteranno poi sui widget e gli Attori utilizzati per ospitare i vari menu. In aggiunta a quanto visto per la versione Desktop, nella sezione dedicata alla customizzazione dei colori dell'interfaccia VR , è possibile personalizzare il layout grafico di alcuni Attori impiegati dal nostro applicativo che, per ovvie ragioni, non potevano essere presenti nella UI Desktop.

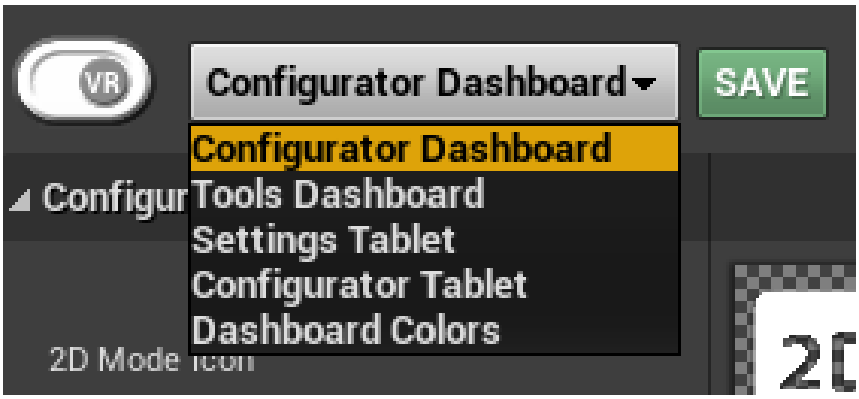


FIGURA 2.16: MENU A TENDINA DELL'EDITOR UI PER LA VR

Nel menu a tendina dell'header sono disponibili cinque diverse sezioni della UI VR, le quali abilitano differenti Details View e preview a seconda del contenuto mostrato.
La funzione di ogni singolo Attore rappresentato nelle preview verrà poi spiegata nel dettaglio nel sottocapitolo **4.2** sulla struttura generale dell'interfaccia VR. Ciò che ci premeva sottolineare qui è il fatto che, a prescindere che debba essere personalizzato un widget per la UI Desktop o un Attore per la UI VR, la logica di customizzazione rimane invariata.

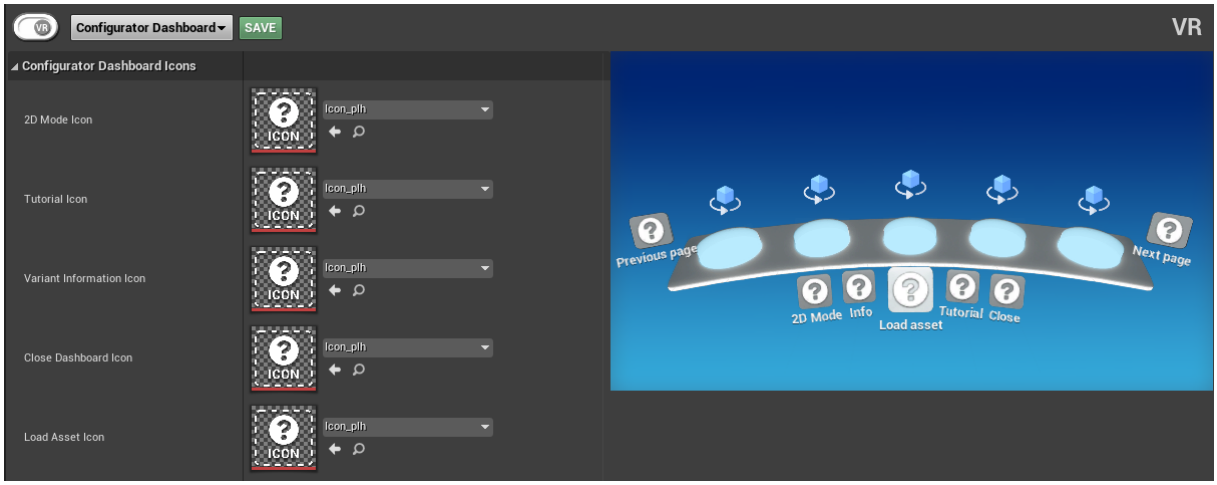


FIGURA 2.17: VERSIONE VR DELL'EUW CON LA SEZIONE CONFIGURATOR DASHBOARD

2.3.2 LETTURA, SCRITTURA E SALVATAGGIO DEI DATI SU DATA ASSET

Come accennato in precedenza, questo Editor UI è stato costruito per facilitare l'inserimento dei dati relativi alle UI nell'apposito Data Asset.
Se il Data Asset di riferimento risultasse vuoto anche tutti i campi presenti dei vari Details View sarebbero completamente vuoti. Per guidare ulteriormente l'utente nella comprensione dell'interfaccia e nell'inserimento nei dati, oltre a tutti i feedback precedentemente descritti riguardanti soprattutto le preview interattive, si è scelto di inserire dei valori di default nei Data Asset, contenenti valori base o icone placeholder a seconda del tipo di dato, in modo da fornire già al designer delle informazioni preliminari relative alla posizione occupata da essi nella UI.

In caso fossero già stati inseriti e salvati dei dati in precedenza, l'Editor provvederà a recuperarli e visualizzarli opportunamente, in modo che si possano apportare facilmente delle modifiche senza dover ripopolare da zero tutta l'interfaccia utente ogni volta che viene avviato Unreal.

Recuperare le informazioni contenute in un Data Asset è piuttosto semplice. Ogni variabile presente in esso è associata ad un metodo *Get*, per cui, recuperato il DA corretto e salvato in una variabile, basta richiamare da esso tutte le variabili, una per una, salvandole nel modo che si ritiene più opportuno.

Nel nostro caso è stata creata una apposita funzione chiamata *Get Data Asset Values* e richiamata sull'*Event Pre Construct* nel *Graph* dell'EUW che descrive la struttura dell'Editor UI. Essendo note a priori le variabili ma non i corrispettivi valori semplicemente andiamo a richiamare ognuna di esse, salvandole nelle rispettive variabili già dichiarate nell'EUW e che vengono esposte all'utente tramite Details View. Nel caso di valori semplici basterà scriverli nell'apposita variabile, nel caso invece di valori più complessi, come le *Map*, si è scritta una funzione che va a recuperare i *Values* corretti a seconda delle *Keys* di riferimento.

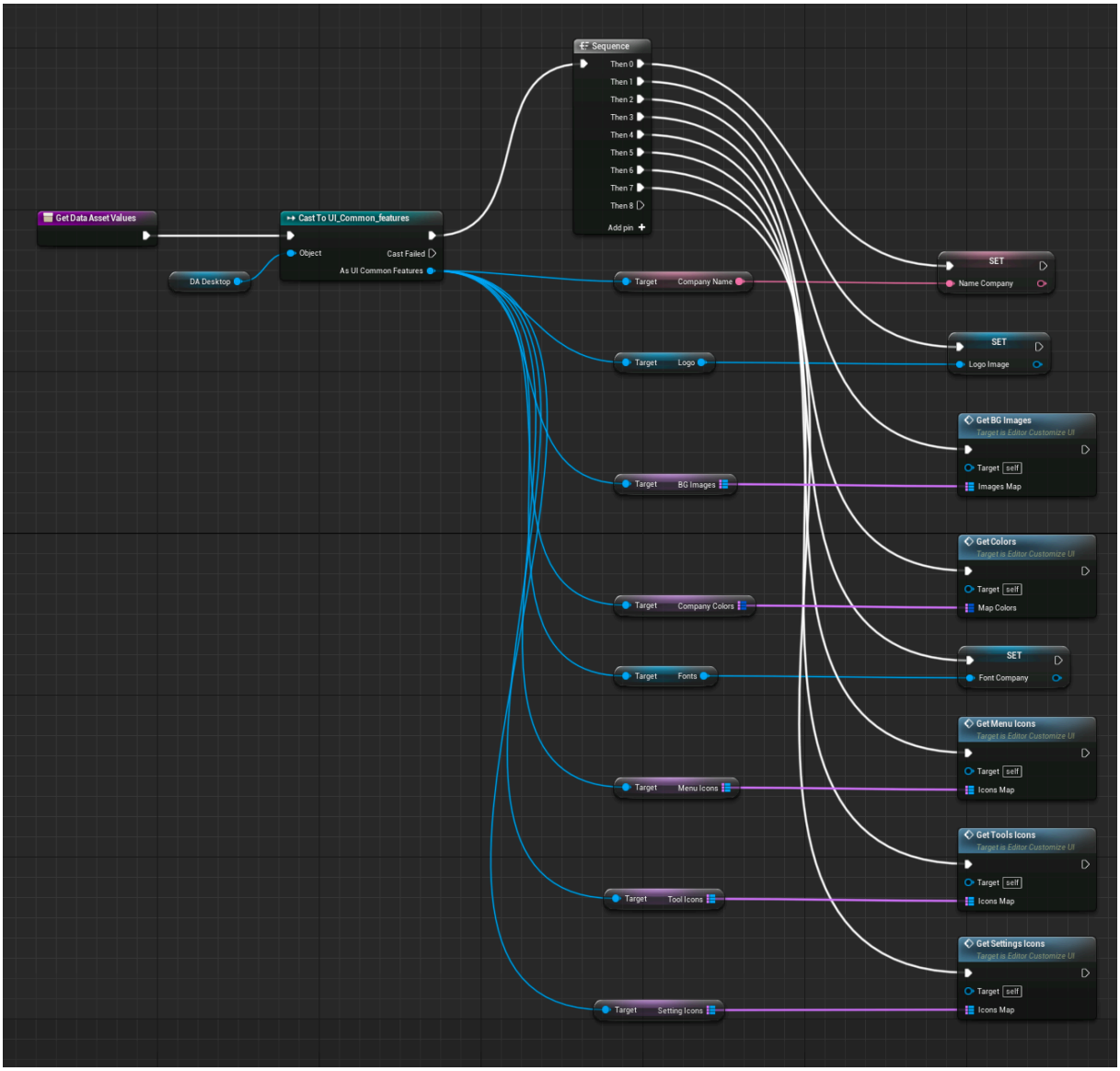


FIGURA 2.18: FUNZIONE *GET DATA ASSET VALUES* CHE RECUPERA I DATI SCRITTI NEL DA DESKTOP E LI SETTA NELLE APPOSITE VARIABILI ALL'INTERNO DELL'EDITOR UI

La modifica dei dati all'interno del DA è altrettanto semplice e si tratta del processo esattamente inverso a quello appena descritto. Se per leggere si recuperano i dati con gli appositi metodi *Get* e li si scrive in variabili con metodi *Set*, nella scrittura avviene l'esatto opposto, recuperando i dati dalle variabili interne dell'Editor UI con metodi *Get* e scrivendoli poi all'interno dell'apposito DA con i relativi metodi *Set*.

Similmente alla lettura, la scrittura è stata strutturata in una apposita funzione chiamata *Set Data Asset Values* e richiamata sull'evento *OnClicked* del bottone *Save* posto nella Tool Bar. È importante sottolineare che questo metodo procede alla scrittura dei dati ma non al loro salvataggio; per fare questo è necessario aggiungere il nodo denominato *Save Loaded Asset*, richiamato sulla variabile del DA che si intende salvare, che esegue l'azione di salvataggio.

Nel Graph dell'Editor UI questo nodo è posto dopo la funzione di scrittura, per cui all'*OnClicked* del bottone *Save* verranno svolte sia le operazioni di scrittura che quelle di salvataggio.

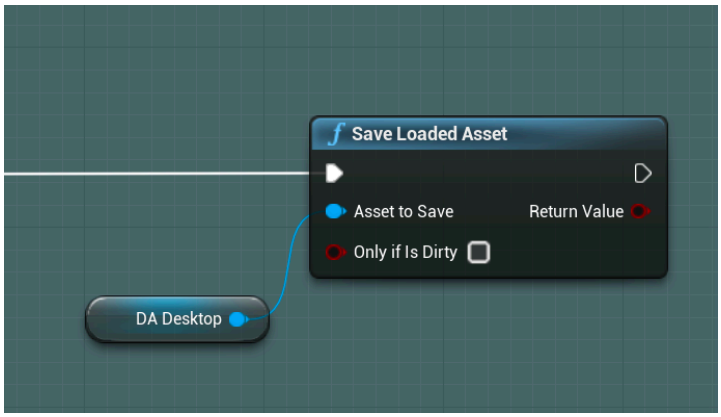


FIGURA 2.19: NODO SAVE LOADED ASSET

Per semplificare ulteriormente questa operazione, dalla prospettiva dell'utente che per errore potrebbe chiudere senza salvare, si intende automatizzare il salvataggio, richiamando queste funzioni e noti su ogni evento *OnPropertyChanged* dell'Editor UI. In questo modo ogni singola modifica sarebbe automaticamente scritta e salvata nei DA.

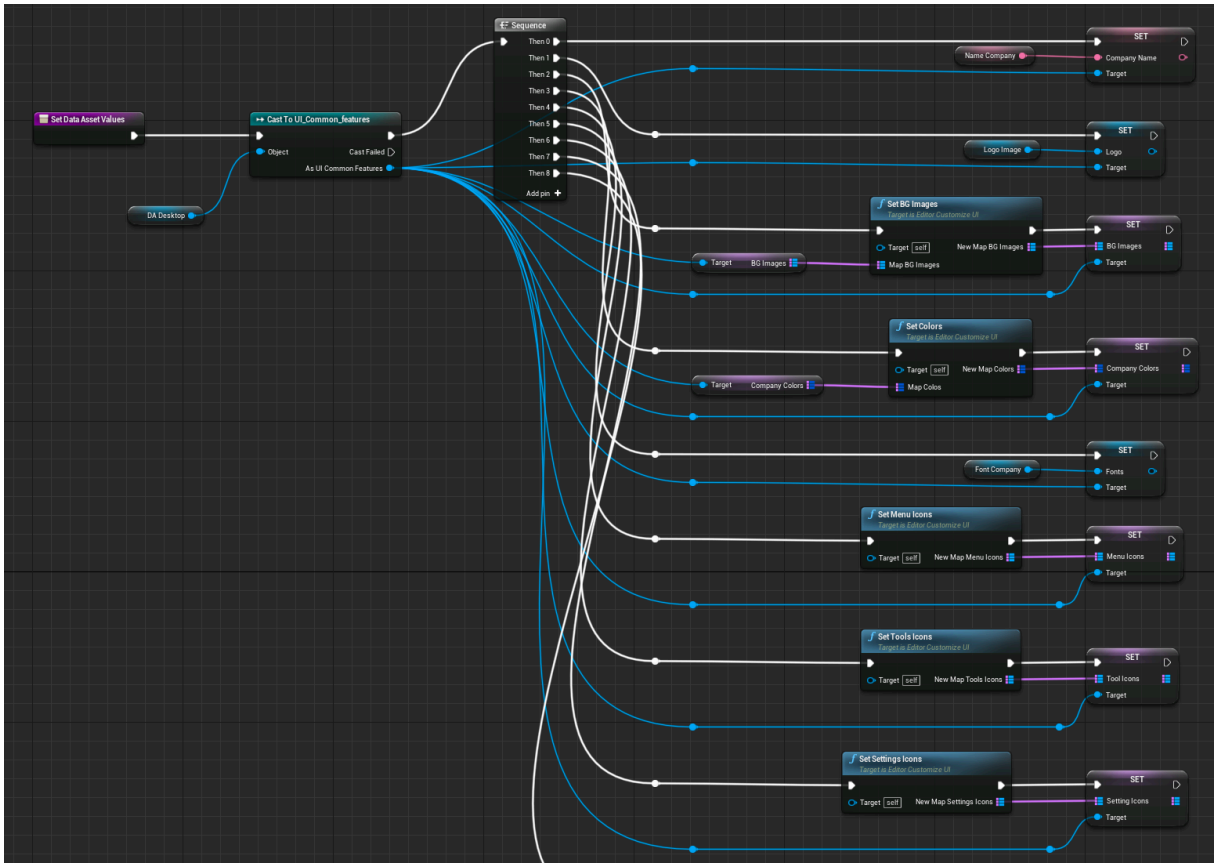


FIGURA 2.20: FUNZIONE *SET DATA ASSET VALUES* CHE RECUPERA I DATI DALLE VARIABILI ALL'INTERNO DELL'EDITOR UI E LI SCRIVE NEL DA DESKTOP

2.4 - VARIANT MANAGER E VARIANT TEMPLATE

La gestione, nonché la customizzazione, delle informazioni relative ad ogni variante definita nel Variant Manager è delegata al **Variant Template**, un modulo chiave del plugin che era già stato sviluppato da Dead Pixels dal quale noi siamo partiti per progettare la struttura e le sezioni delle varie Interfacce Utente. Ne illustreremo ora le caratteristiche principali per far comprendere al meglio le motivazioni di alcune nostre scelte di layout riguardo le interfacce descritte nei capitoli **3** e **4**.

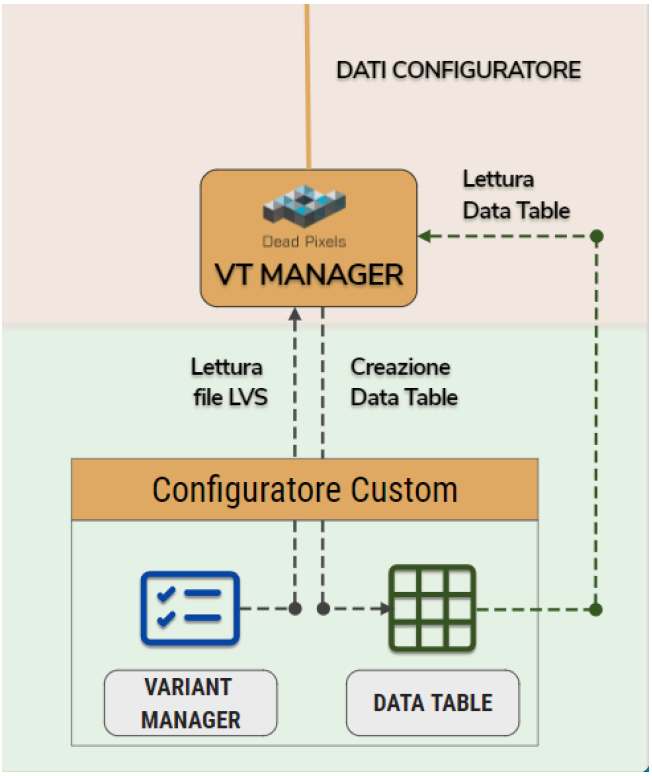


FIGURA 2.21: PORZIONE DEL DIAGRAMMA DI ARCHITETTURA SOFTWARE CON FOCUS SULLA SEZIONE CONFIGURATORE

Uno dei primi passi da compiere per realizzare un product configurator aziendale, dopo aver recuperato tutti i modelli necessari, è quello di creare un database di informazioni centralizzato che raccolga tutte le specifiche tecniche per ogni configurazione del prodotto in questione. Queste saranno fondamentali per l'utente poiché lo guideranno nella scelta delle varie componenti di cui molto spesso non ne conosce le caratteristiche. Da qui si spiega la scelta di implementare un plugin data-driven in cui tutti i dati relativi al prodotto vengano memorizzati su una struttura esterna al *Variant Manager*, affinchè siano facilmente reperibili dall'applicazione a seconda delle esigenze ma anche dallo sviluppatore in fase di progettazione.
Dal momento che il *Variant Manager*, per come è costruito, non permette di immagazzinare informazioni per ogni variante, si è deciso di sfruttare una **Editor Utility Blueprint** (abbreviata EUB) che permetta di creare dinamicamente una *Data Table* che riporta per iscritto quanto contenuto nel file di tipo **Level Variant Sets** (abbreviato LVS) .

Questo presuppone che il designer o sviluppatore abbia già precedentemente creato un file *Level Variant Sets* per ospitare tutte le mesh necessarie alla realizzazione del configuratore e lo abbia popolato sfruttando le interfacce messe a disposizione dal *Variant Manager*. Una volta

creati tutti i *Variant Set* necessari, il designer deve definire tutte le varianti per ogni set, nominandole in modo tale per cui il plugin riesca ad indicizzarle correttamente all'interno della DT. Nello specifico è stata aggiunta, già prima del nostro intervento su questo progetto, l'opzione di suddividere internamente le varianti di ogni *Variant Set* per sottocategorie, questo per dare modo al designer di organizzarle al meglio in base alle proprietà definite.

2.4.1 - PROCESSO DI CREAZIONE DELLA DATA TABLE

Questa più complessa categorizzazione delle varianti non è di default fattibile con un semplice *Variant Manager*, per questo si è deciso di utilizzare un sistema di renaming delle varianti in modo da delegare tale nuova suddivisione ad una classe scritta in C++ denominata **VTE_VariantInfoToDT**, al cui interno sono state implementate delle funzioni che hanno il compito di costruire la relativa DT. Per creare sottocategorie all'interno dei *Variant Set* è necessario che il nome della variante sia preceduto dal nome della sottocategoria padre seguito dal carattere "_". Se il designer ha poi necessità di reiterare tale suddivisione basterà ripetere tale procedimento sul nome della variante come mostrato nell'immagine 2.22 d'esempio sottostante.



FIGURA 2.22: SOTTOCATEGORIA "IRON FLOWER" DELLA CATEGORIA "MOMO" DEI CERCHIONI

La EUB **VT_VariantInfoToDT** è incaricata di richiamare da Blueprint le funzioni della classe *VTE_VariantInfoToDT* da cui la EUB eredita tutte le funzioni e le variabili. Dunque, se i nomi delle varianti sono stati definiti come descritto sopra, ogniqualvolta essa viene richiamata su un LVS provvederà ad inviare correttamente tutti i dati alla classe sopracitata e costruire quindi la DT.

Per invocare la EUB, o più precisamente le funzione ***Variant Info to DTBP*** dichiarata in essa, basterà cliccare prima col tasto destro del mouse sull'icona del LVS nel Content Browser e poi dal menu a comparsa, sotto la sezione *Scripted Actions*, sulla voce *Variant Info to DTBP* come illustrato in figura 2.23 su un LVS d'esempio nominato *Lamborghini_Variant*.

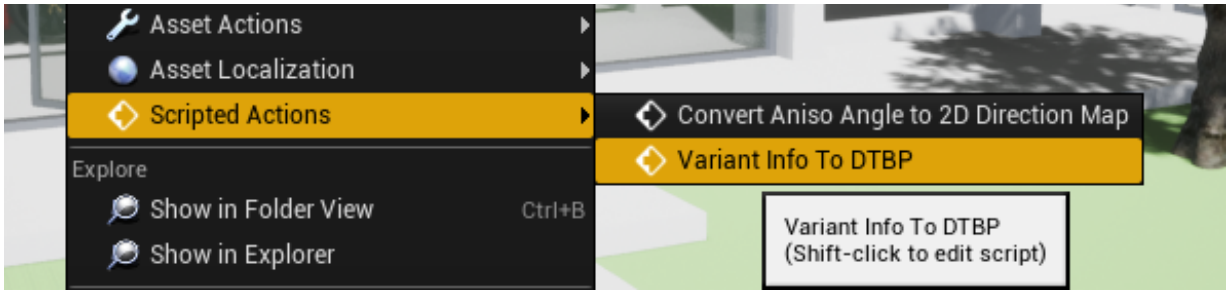


FIGURA 2.23: OPZIONE "VARIANT INFO TO DTBP" DEL MENU A COMPARSA

A questo punto si aprirà **VT_UITableUtility**, un EUW che offre sia la possibilità di creare la DT relativa al LVS in un certo path specificato dall'utente che l'opzione di aggiornare una datatable esistente, da specificare nel campo "Data Table", nel caso fosse stata già associata una DT ma il designer abbia deciso in seguito di apportare qualche modifica al LVS.

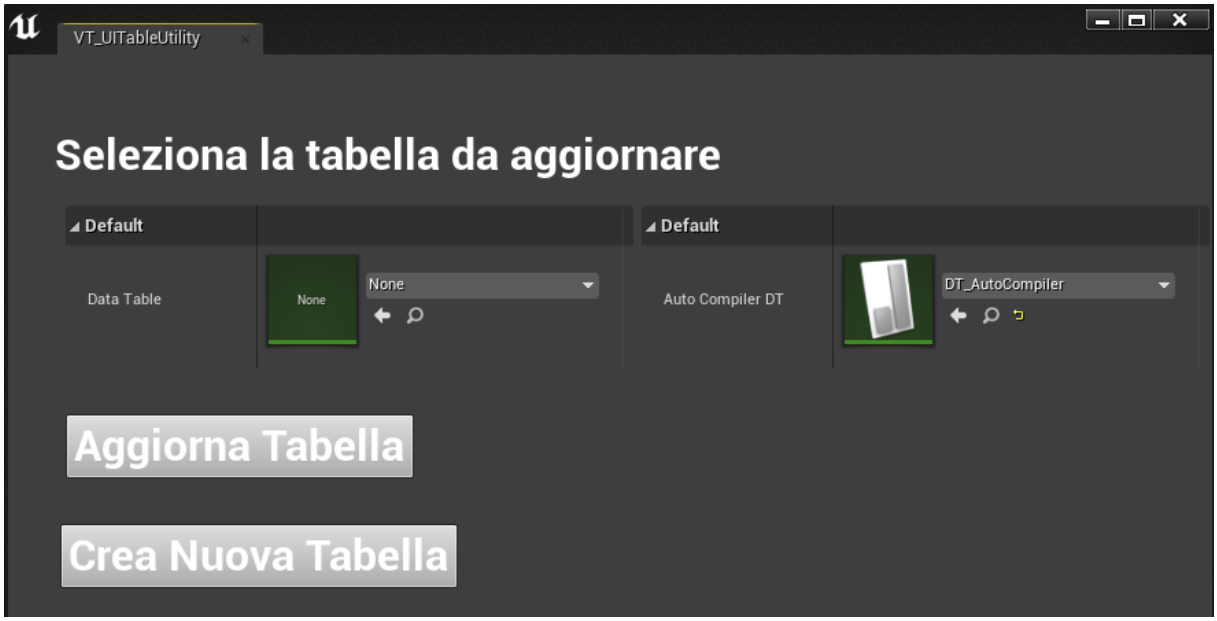


FIGURA 2.24: VT_UITABLEUTILITY WIDGET

Nel caso si scelga di cliccare su *Crea Nuova Tabella* verrà chiamato in background un ***Event Dispatcher*** che notificherà alla EUB *VT_VariantInfoToDT* l'azione compiuta e quindi la necessità di creare la DT nel path scelto a partire dalla *RowStruct Variant Category* da noi definita. Se tutto il processo è stato eseguito correttamente e nell'ordine sopra descritto si presenterà una situazione simile a quella illustrata qui sotto.

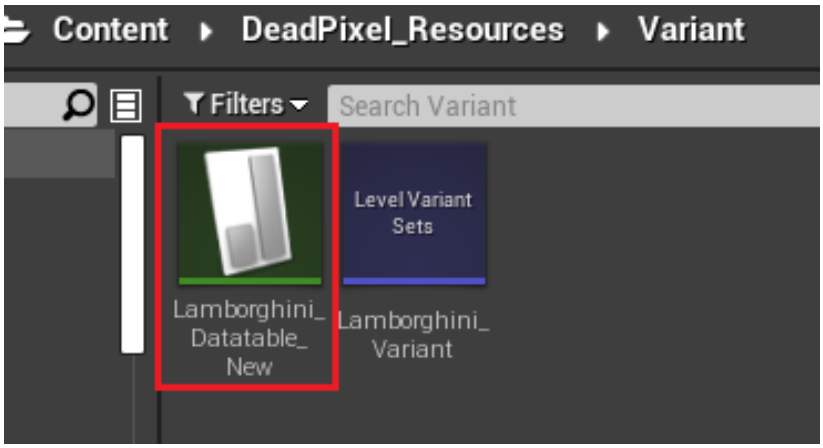


FIGURA 2.25: DATA TABLE LAMBORGHINI_DATATABLE_NEW CREATA DAL LVS LAMBORGINI_VARIANT

2.4.2 - STRUTTURA DELLA DATATABLE

Arrivati a questo punto, possiamo analizzare nel dettaglio la struttura della DT creata per comprendere come sono state catalogate le varie categorie, sottocategorie e varianti in base al modello di renaming esposto sopra. Seppur la DT di esempio sia stata creata da un LVS corposo ma non eccessivamente complesso, la mole di dati è considerevole, per questo per non creare confusioni al lettore si è scelto di suddividere lo studio della DT in piccole sezioni in modo da esporre più chiaramente la funzione di ognuna di esse. Al termine della spiegazione allegheremo comunque una foto della tabella nella sua completezza per permettere al lettore di ricollegare tutte le nozioni ricevute ed avere quindi una visione più completa di tutta la struttura.

	Row Name
1	P_CarBodyPaint
2	P_MirrorPaint
3	P_Brakes
4	P_Rims
5	P_TestSubCategory
6	CarBodyPaint_Standard
7	CarBodyPaint_Glossy
8	CarBodyPaint_Mixed
9	MirrorPaint_Standard
10	MirrorPaint_Glossy
11	MirrorPaint_Mixed
12	Brakes_Lux
13	Brakes_Standard
14	Brakes_Racing
15	Rims_Oz
16	Rims_Oz_Spiderweb
17	Rims_Oz_DoubleSun

FIGURA 2.26: COLONNA ROW NAME DELLA DT DI ESEMPIO

La *RowStruct* ***Variant Category*** presenta come primo campo il **Row Name**, all'interno del quale è presente una variabile *Name* che, a seconda della riga, può contenere o il nome di una fase, che corrisponde al nome del rispettivo *Variant Set* nel *Variant Manager*, oppure il nome di una sottocategoria preceduto dal nome della categoria padre e il carattere "_". Nel caso di più sottocategorie annidate tra loro si avrà il nome dell'ultima sottocategoria preceduto da quello di tutte le altre categorie di cui è figlia. Si pensi alla struttura informatica di un albero, in cui le cate-

gorie principali rappresentano i nodi di livello 1, mentre tutte le altre sottocategorie a nodi di livello 2, 3, 4 a seconda dei casi. Le foglie di questo grafo ad albero saranno quindi le nostre varianti, che qui in figura non sono visibili poiché visualizzabili in un'altra sezione della DT che spiegheremo successivamente.

Nell'esempio qui sopra le "categorie principali" sono indicate come *P_NomeFase*, questo poiché più che di categoria principale si è preferito parlare di **Fase** di configurazione (in inglese *Phase*).

Di fatto le prime cinque righe rappresentano le cinque fasi di configurazione principali derivanti dai *Variant Set* del VM, mentre tutte le altre sono le vere e proprie categorie e sottocategorie interne ai *Variant Set* che vengono rispettivamente indicate come *NomeCategoriaPadre_NomeSottocategoria.* Con questo metodo riusciamo ad associare ad una singola variante del VM tante righe della DT quante sono le occorrenze del carattere "_" nel nome della variante, aiutandoci a capire meglio di quante, e soprattutto quali, categorie è composto il nostro configuratore, a prescindere dal tipo di LVS creato dal designer.

Dal *Row Editor* possiamo poi vedere, a seconda della riga scelta, in che modo sono stati compilati gli altri campi della *RowStruct*.

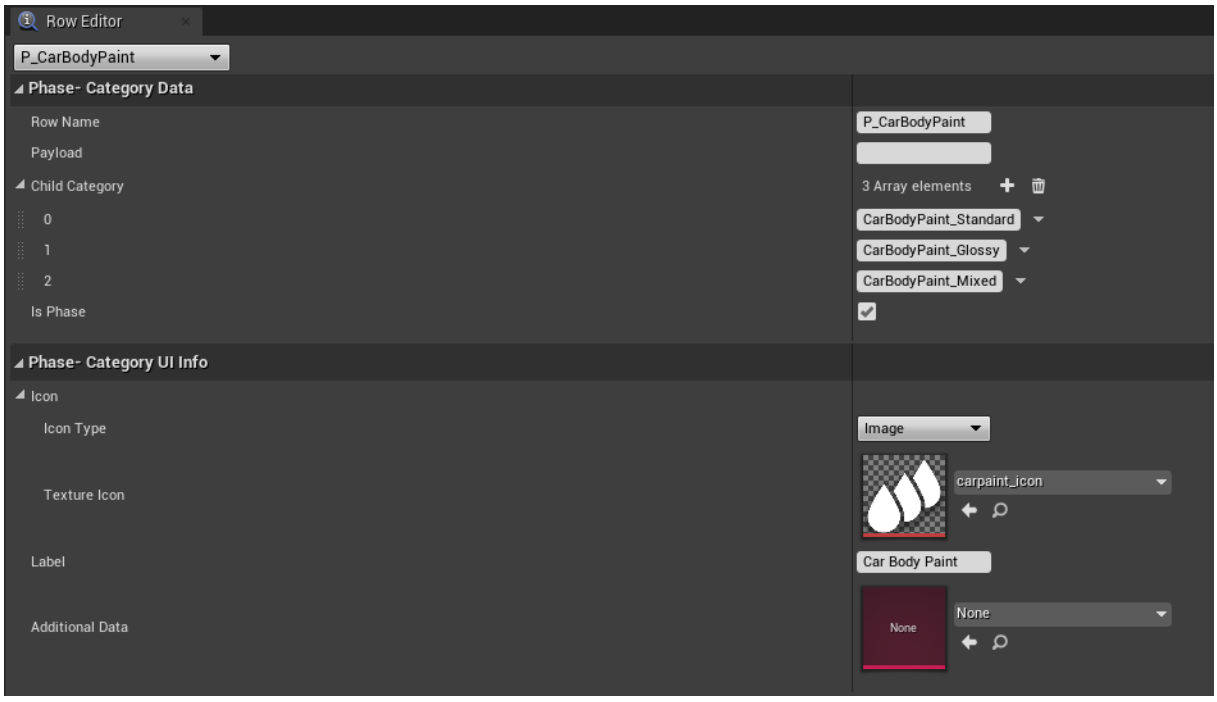


FIGURA 2.27: ROW EDITOR DELLA FASE P_CARBODYPAINT

Prendendo in esame la struttura della riga di una qualunque fase, per esempio *P_CarBodyPaint*, si può notare come siano presenti due macro sezioni: ***Phase-Category Data*** e ***Phase-Category UI Info***. Come è ben visibile in figura 2.27, nel *Phase-Category Data* è presente: il

Row Name; un campo ***Payload,*** la cui funzione verrà spiegata dettagliatamente in seguito; un Array ***Child Category,*** di tipo *Name, c*he racchiude il nome di tutte le prime sottocategorie della suddetta fase; una variabile *Boolean* ***isPhase***.

Nella sezione *Phase-Category UI Info* si ha invece: un campo ***Icon*** di tipo *Struct,* al cui interno vi è una variabile *Enum* ***Icon Type,*** che specifica il tipo di icona che il designer può associare a tale fase, e un campo ***Texture Icon,*** il cui tipo cambia a seconda dell'opzione scelta per l'Icon Type; una ***Label*** di tipo *String* con il nome della fase privo di quei caratteri che inizialmente erano necessari alla categorizzazione (_, P); un campo *Data Asset Soft Object* ***Additional Data*** per l'aggiunta opzionale di un Data Asset con informazioni aggiuntive su una fase.

La struttura del *Row Editor*, se prendiamo in considerazione una categoria di una qualunque fase, è pressoché uguale a quella della fase stessa, fatta eccezione per la presenza di una una nuova sezione *Category-Data* che gestisce le features riguardanti le varianti di tale categoria.

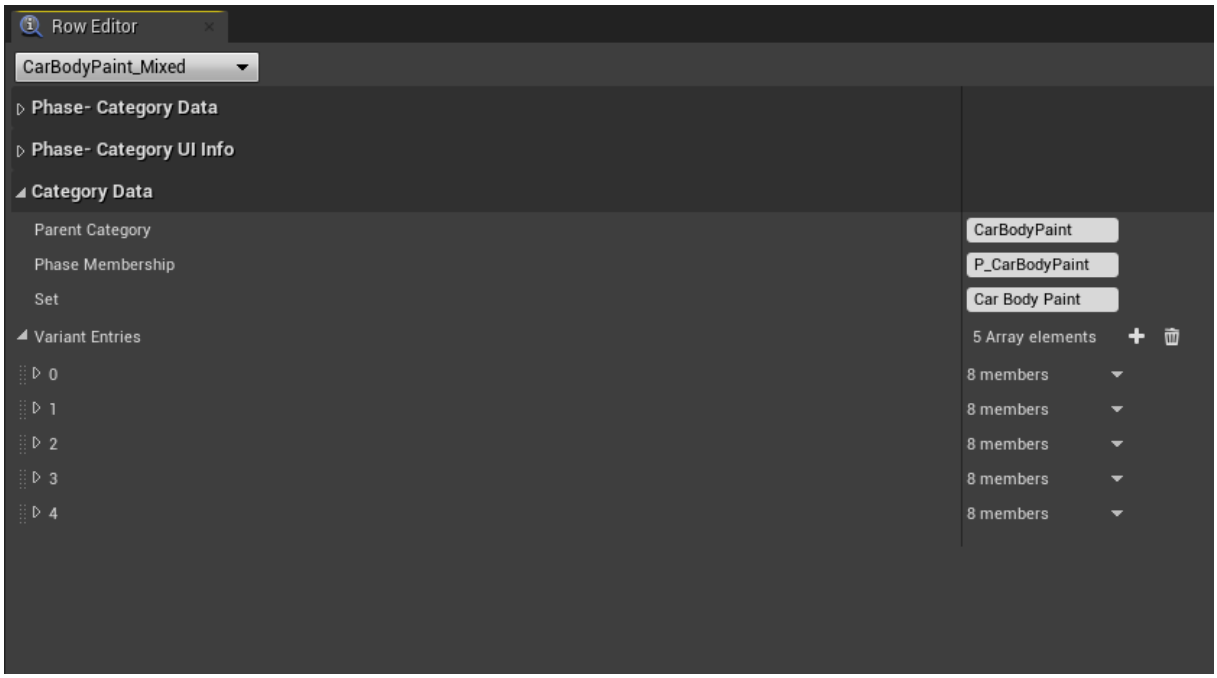


FIGURA 2.28: ROW EDITOR DELLA CATEGORIA CARBODYPAINT_MIXED

Per la categoria *Mixed*, della fase *Car Body Paint,* la sezione *Category-Data* si presenta come in figura 2.28. Abbiamo una variabile **Parent Category** di tipo *Name* con il nome della categoria o fase che gerarchicamente la precede, una **Phase Membership** che indica il *Row Name* della fase a cui tale sottocategoria appartiene ed una **Set** che riporta il nome del *Variant Set* del VM in cui tale sottocategoria è stata definita.

Nel caso poi non vi siano altre sottocategorie direttamente discendenti da quest'ultima, sarà visibile anche un campo **Variant Entries** che non è altro che un Array di *Struct* di tipo *Variant Entry,* la cui grandezza dipende dal numero di varianti definite nel VM.
Una singola *Variant Entry* è strutturata come segue:

- Campo **Parent Category** con il *Row Name* della categoria a cui tale variante appartiene;
- Campo **Variant** con il nome della variante completo anche di sottocategoria padre;
- Una Mappa **Requested List** ed un Array di *Inclusive Data* **Inclusive List** che possono essere riempite opzionalmente dallo sviluppatore e che, in breve, servono a impostare delle regole di visibilità di una variante a seconda del contenuto delle due liste**.** Per esempio, il designer potrebbe decidere che tale variante di colore può essere utilizzata solo se il modello della macchina è uguale a "Huracan EVO Spyder" oppure a "Huracan STO", così fosse gli basterebbe inserire il nome di tali categorie/modelli all'interno della *Requested List* e impostare poi la regola d'inclusività *OR* nella *Inclusive List*;
- Campo **Payload** in formato JSON che può essere riempito con informazioni relative al variante, utili per l'utente durante la configurazione del prodotto;
- Campo **Additional Data** simile a quanto descritto sopra per le fasi;
- Campo **Icon** uguale a quello delle categorie;
- Campo **Label** con il nome della variante in formato *String;*

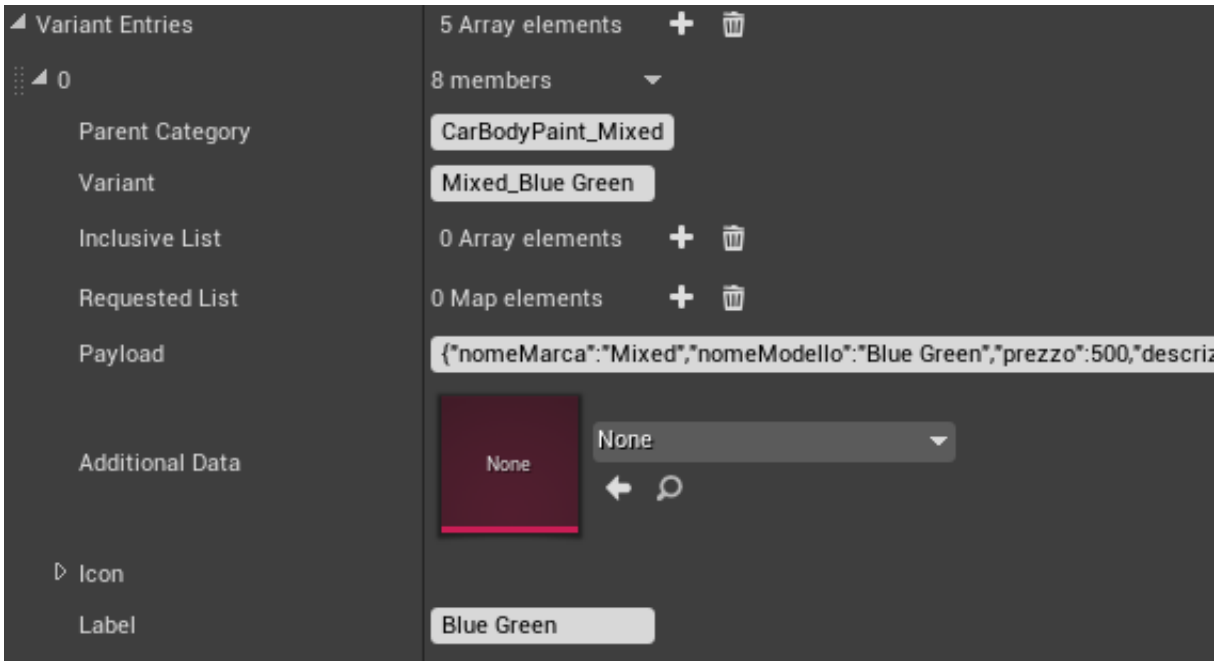


FIGURA 2.29: CAMPO VARIANT ENTRIES

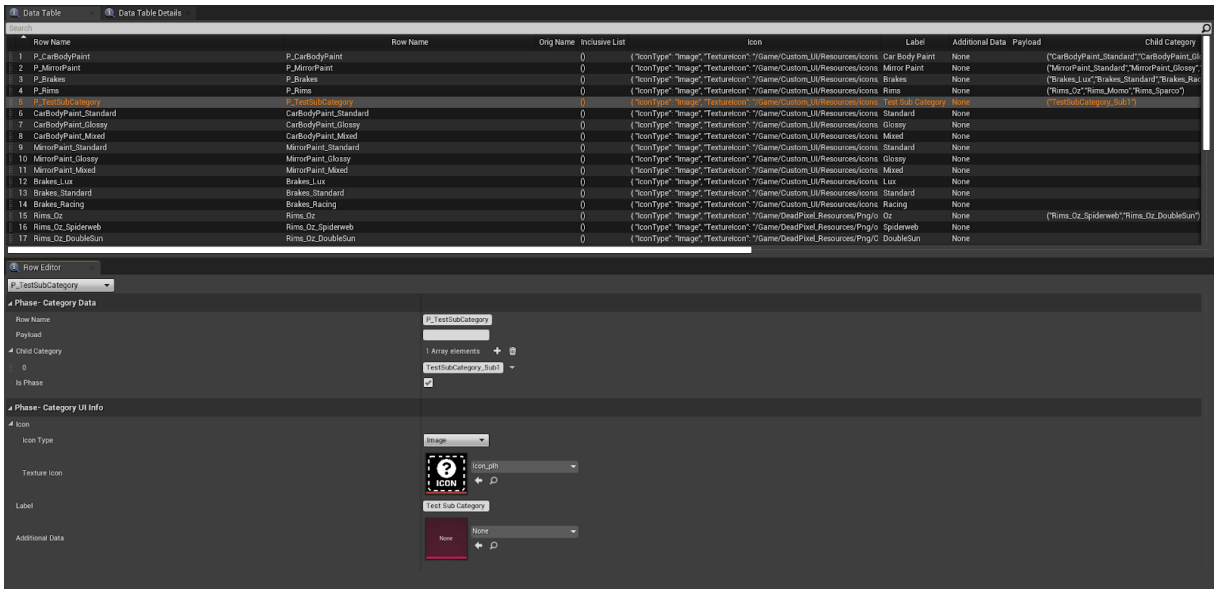


FIGURA 2.30: STRUTTURA COMPLETA DT LAMBORGHINI_DATATABLE_NEW

2.4.3 - PAYLOAD STRUCT

Il campo *Payload* menzionato in precedenza è a disposizione del designer nel caso voglia aggiungere delle informazioni riguardanti il singolo variante, che ne esplichino per esempio le caratteristiche piuttosto che il prezzo o la marca, e che verranno visualizzate dall'utente durante l'esecuzione del programma.
Tuttavia, dal momento che un campo JSON, seppur versatile e comodo in questo contesto, non è propriamente di immediata lettura o scrittura, e per dare modo a noi di standardizzare il tipo di informazioni che possono essere inserite, potendole così visualizzare correttamente su schermo, si è scelto di utilizzare un tool esterno alla datatable per compilare i vari campi del *Payload*.

Qui entra in gioco l'EUB **BP_TablePayloadFiller,** al cui interno vi è una funzione **ShowPayloadUtility** richiamabile da editor dal menu a comparsa alla voce *Scripted Actions* dopo aver cliccato col tasto destro sulla DT creata. Questa aprirà una finestra su schermo che chiede allo sviluppatore/designer di scegliere dai due menu a tendina due *Blueprint Class* che abbiano come *parent class* rispettivamente **BP_Variant Factory** e **BP_Category Factory**.
Queste due classi contengono le funzioni necessarie a reperire i vari campi *Payload* di una DT, sia quelli relativi ad una categoria che ad una variante, e, data una *Payload Struct* (che può essere modificata manualmente dallo sviluppatore) trasformare il contenuto dei suddetti campi in una stringa JSON e scriverla nel campo *Payload* della DT.

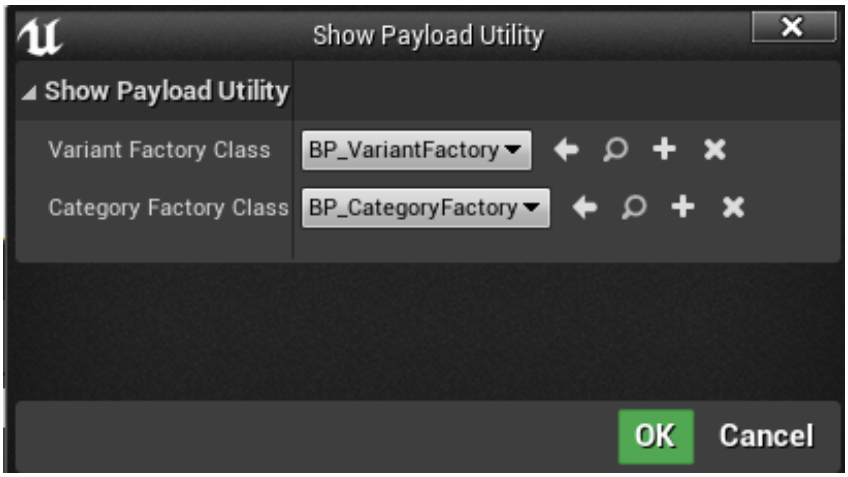


FIGURA 2.31: FINESTRA ATTIVABILE DA SHOWPAYLOADUTILITY

Questa *Payload Struct* di default non è presente nelle due classi *Factory*, per questo lo sviluppatore dovrà: creare due *Blueprint Class* che ereditino da *BP_Variant Factory* e *BP_Category Factory;* creare una *Payload Struct* con tutti i campi che vuole facciano parte della stringa JSON e che vengano visualizzati nella UI del programma; all'interno delle classi create eseguire l'**override** delle funzioni ereditate e aggiungere tra le variabili la *Payload Struct* creata. Una volta inserite le classi create nei due dropdown menu della finestra come illustrato in figura 2.31, si aprirà il widget **W_PayloadFiller_TreeView** che permette di scegliere qualunque categoria o variante definita nella DT e di compilare i campi del Payload, precedentemente dichiarati nella *Payload Struct*.

Consapevoli che questo passaggio possa essere ostico per chiunque non abbia conoscenze di programmazione con *Unreal Engine*, si è deciso di lasciare la funzione *ShowPayloadUtility* disponibile per gli sviluppatori, ma di aggiungere anche una funzione parallela denominata **Payload Fill** che automatizza alcuni passaggi semplificando il processo. Un designer, ad esempio, potrà interfacciarsi solamente col widget *W_PayloadFiller_TreeView* e verranno impostate come *Variant Factory Class* e *Category Factory Class* due classi presenti di default nelle directory del plugin con una struttura standard di *Payload Struct*.

Si illustreranno ora brevemente i vari passaggi necessari a costruirsi una propria *Payload Struct* e renderla compatibile con quanto scritto nella classe *BP_Variant Factory* (il procedimento è il medesimo anche per *BP_Category Factory*). Per coloro che non fossero interessati a seguire tutto il procedimento suggeriamo di saltare i prossimi paragrafi e passare direttamente al sottocapitolo **2.4.4**.

Innanzitutto va creata una Structure, definendo al proprio interno i campi che costituiranno il Payload per le varianti e che verrà inserita nella DT come stringa JSON.
Definiti i campi e i valori predefiniti di ognuno di essi, si procede creando una *Blueprint Class* che erediti da *BP_Variant Factory*, in modo da poterne ereditare anche tutte le funzioni scritte internamente. Si esegue quindi l'override delle funzioni **SetPayloadVariant** e **GetPayloadVariant** che permettono rispettivamente di concatenare il contenuto dei campi della *Structure* in una stringa formato JSON e di suddividere, viceversa, il contenuto di una stringa JSON nei vari campi della *Structure,* in modo da poter leggere da widget direttamente il contenuto dei campi che compongono il *Payload* della DT.

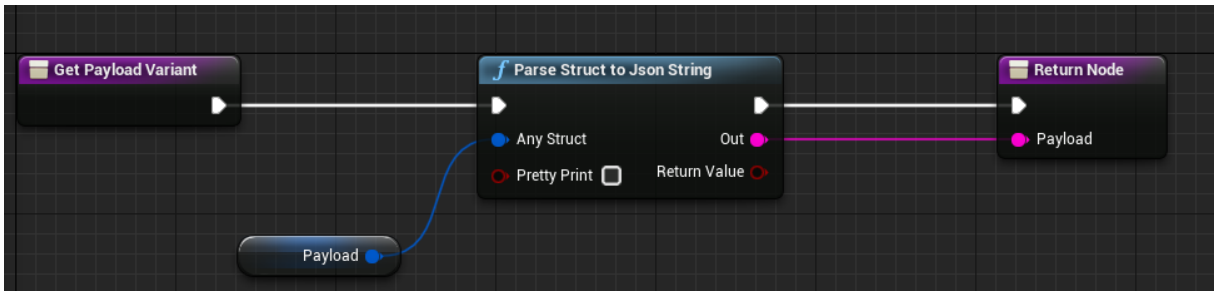


FIGURA 2.32: FUNZIONE GETPAYLOADVARIANT

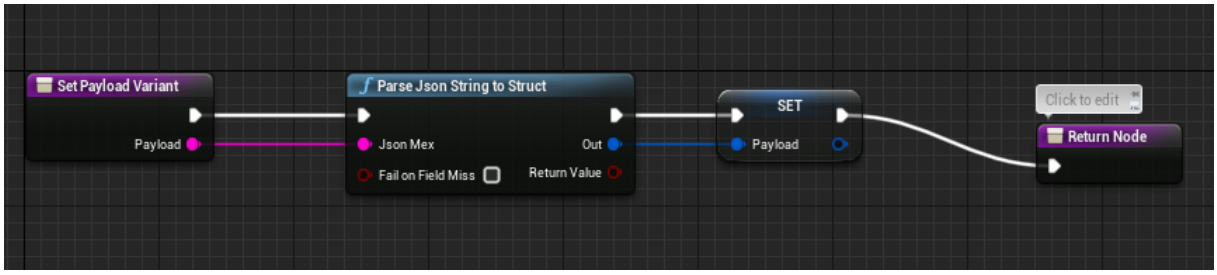


FIGURA 2.33: FUNZIONE SETPAYLOADVARIANT

Basterà inserire poi il nome di tale classe nel primo dropdown menu della finestra *ShowPayloadUtility* per poter interagire con il widget *W_PayloadFiller_TreeView* e customizzare il contenuto dei vari *Payload* con la struttura dati da noi creata.

2.4.4 - CUSTOMIZZAZIONE DEL PAYLOAD DA EUW

Una volta inserite le due classi *Factory*, l'EUB *BP_TablePayloadFiller* creerà due istanze di tali classi e le utilizzerà per aprire correttamente il widget *W_PayloadFiller_TreeView*. Questo si presenta come in figura 2.34, con a sinistra la lista di tutte le fasi della DT, che se cliccate visualizzano anche le categorie e sottocategorie interne, mentre a destra abbiamo la lista di tutte le varianti della categoria selezionata. I bottoni collocati a lato di ogni fase/categoria/va-

riante permettono di definire quelle regole d'inclusività di tipo *OR* o *AND* menzionate in precedenza nel sottocapitolo **2.3.2** e di aggiungere inoltre tale categoria o variante alla *Requested List*.

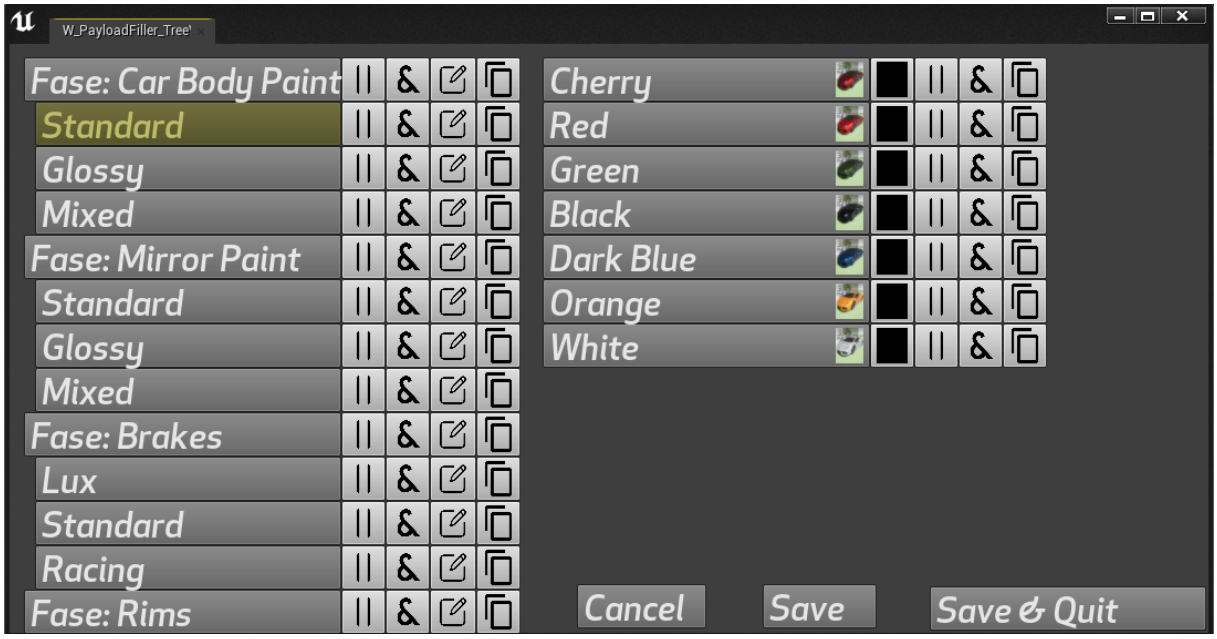


FIGURA 2.34: INTERFACCIA WIDGET W_PAYLOADFILLER_TREEVIEW

Cliccando direttamente sul nome di una variante, questa verrà abilitata all'editing e verrà visualizzata una terza colonna sulla destra con due macro sezioni: *Base Variant Detail* e *Payload Data*.

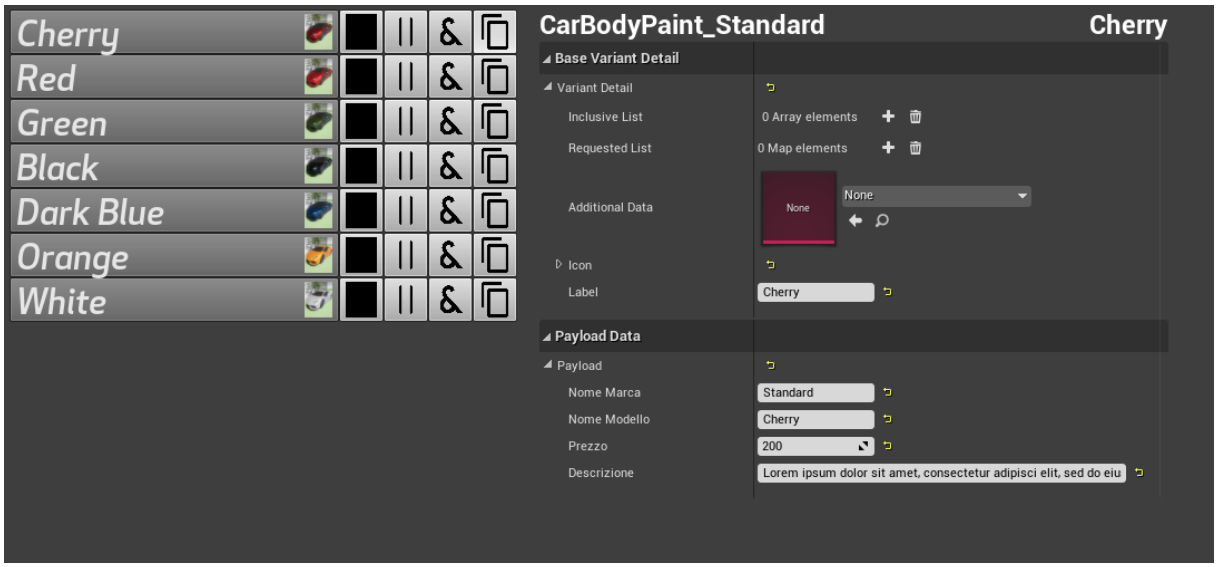


FIGURA 2.35: MACRO SEZIONI BASE VARIANT DETAIL E PAYLOAD DATA DELLA VARIANTE "CHERRY"

In *Base Variant Detail* ritroviamo gli stessi campi che avevamo già analizzato nel *Row Editor* della DT, ad esclusione ovviamente del campo *Payload* che è stato spostato nella sezione *Payload Data* in cui ritroviamo i campi definiti nella *Payload Struct*, ora facilmente editabili. Una volta terminato, e cliccato sul bottone *Save & Quit*, le informazioni inserite in quest'ultima sezione verranno convertite in stringa JSON e scritte nella DT.

Terminata anche questa fase non resta che trasferire tutte le informazioni della DT del configuratore nella UI, in modo da renderle visibili durante l'esecuzione dell'applicativo. Questo processo viene preso in carico da un Attore già collocato di default in scena della classe **VT_Manager,** scritta in C++.

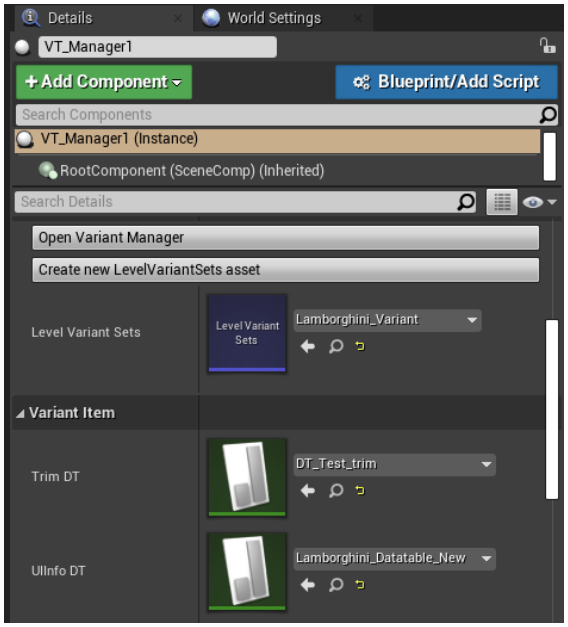


FIGURA 2.36: DETAILS DI VT MANAGER1

Nella sezione *Details* di tale Attore va specificato con quale LVS e con quale DT la classe *VT_Manager* dovrà interfacciarsi per recuperare tutte le informazioni. come si può vedere in figura 2.36. La Trim DT presente nella sezione Variant Item serve alla classe VT_Manager per recuperare il path di tali file all'interno del programma e quindi per sapere dove andare a recuperare i dati richiesti. Questa classe serve allo sviluppatore per recuperare, le varianti dal LVS e le relative caratteristiche da DT.

Nel nostro caso il *VT_Manager*, o meglio le funzioni messe a disposizione da esso, sono state utilizzate per dare possibilità all'utente di cambiare dinamicamente, attraverso la User Interface, le varianti a seconda delle categorie o fasi scelte. Queste funzioni, già scritte prima del nostro intervento sul progetto, sono visibili dal *Blueprint Editor,* una volta creata una variabile di tipo *VT_Manager*, nel menu delle azioni disponibili per tale nodo alla voce *Variant Manager*.

Tali funzioni sono state scritte in C++ in VT_Manager.cpp e nel relativo *header file* VT_Manager.h.Quest'ultimo viene processato dall'**UnrealHeaderTool** che rende accessibili da Editor le funzioni scritte in C++**.** Di fatto, per capire come utilizzare al meglio il VT_Manager, ci è bastato leggere quali funzioni erano presenti nell'omonimo header file e richiamarle all'occorrenza da Editor, sotto forma di nodi.

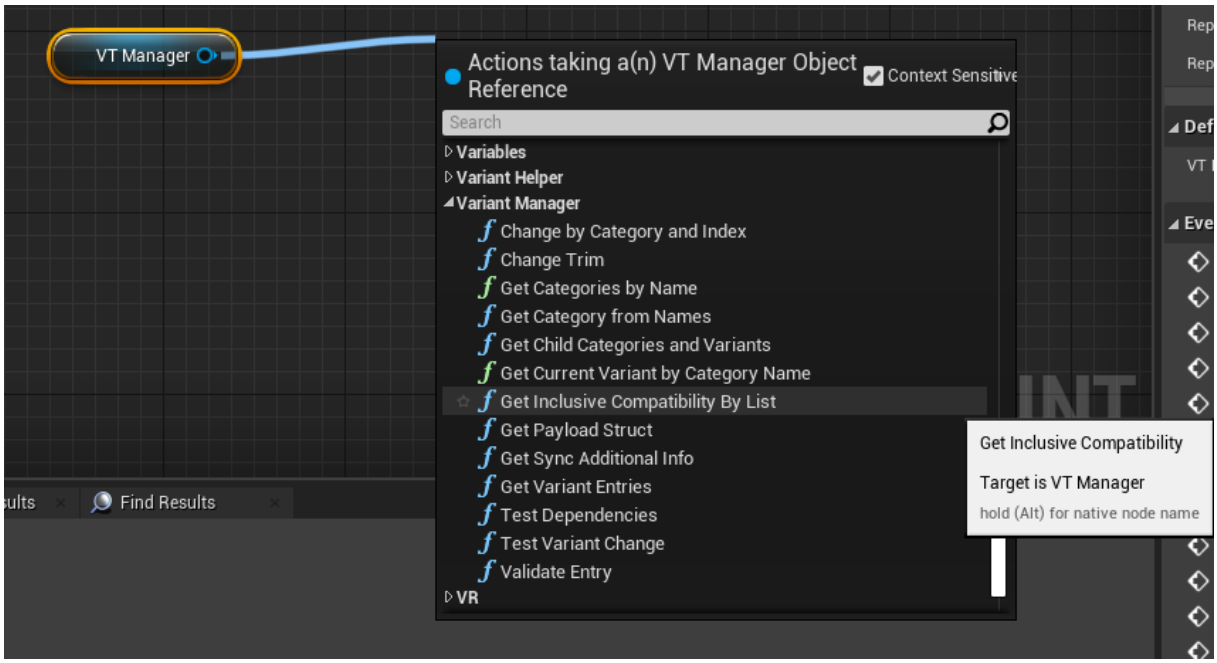


FIGURA 2.37: FUNZIONI DEL PLUGIN DISPONIBILI PER LA VARIABILE VT MANAGER

2.5 - BLUEPRINT INTERFACE E HUD

Una **Blueprint Interface**[37] (abbreviato BI) è una collezione che contiene esclusivamente la dichiarazione di una o più funzioni che possono essere aggiunte ad altre Blueprint. Ogni Blueprint a cui è stata aggiunta una Interface garantisce non solo di avere queste funzioni, ma anche di implementarle e dunque dotarle di funzionalità specifiche per il suo caso.
Le Blueprint Interface non si discostano dal comune concetto di Interface noto in programmazione, che permette l'accesso a tipi diversi di Object che la implementano.
All'interno delle BI è possibile esclusivamente dichiarare delle funzioni, con i relativi valori in input, ma non si possono aggiungere variabili, Component o editare il Graph Editor che è Read Only.
Dunque una qualunque classe che implementa una Interface possiede delle specifiche funzionalità che sono note a tutti e dunque richiamabili ovunque.

Per quanto riguarda le nostre UI, le BI sono state usate in due occasioni: la definizione di comportamenti comuni negli Actor presenti nella UI VR, come ad esempio la gestione degli eventi di Grabbed o Selected, che spiegheremo approfonditamente nel capitolo **4**; la gestione della comunicazione tra tutta la logica del Collab Viewer Template e le nostre UI.

Come spiegato in precedenza si è scelto di utilizzare il Collab Viewer offerto da Unreal per tutte le funzionalità già descritte in esso, utili come base di partenza per il nostro studio, dandoci la possibilità di concentrarci esclusivamente sulle User Interface. Per garantire una maggiore modularità di quanto costruito si è scelto, nel limite del possibile, di non modificare il codice che descrive il suddetto Template, tenendolo completamente separato dalla costruzione delle UI. Questo tipo di scelta è data sia dal fatto che, nel caso di un futuro aggiornamento del Collab Viewer, esso sia facilmente aggiornabile anche nel nostro applicativo, sostituendo semplicemente la vecchia versione con la nuova, sia per dare la possibilità a chiunque voglia in un futuro utilizzare queste UI di poterlo fare liberamente. Questo è possibile proprio grazie all'uso di una Interface, che mostra, a chiunque voglia costruire un applicativo compatibile con esse, quali sono le funzioni implementate e messe a disposizione.

Le BI necessitano di una classe che implementi le funzioni dichiarate in essa e nel nostro applicativo tutte le funzioni sono dichiarate in una BI chiamata **UI_Interface** e successivamente implementate attraverso una classe HUD chiamata **HUD_UI_Logic**.

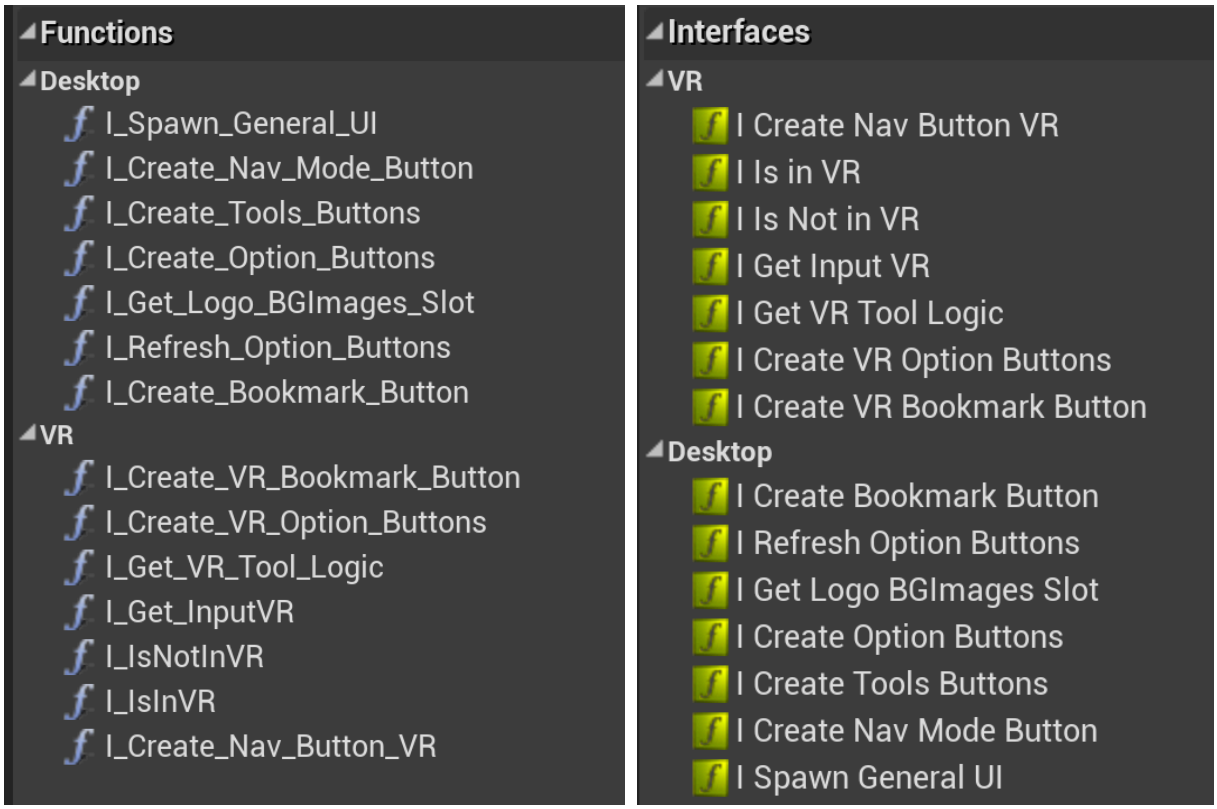


FIGURA 2.38: FUNZIONI DICHIARATE NELLA *UI_INTERFACE* A SINISTRA E IMPLEMENTATE NELLA *HUD_UI_LOGIC* A DESTRA

Le classi HUD[38] sono un retaggio che precede l'adozione dell'UMG in Unreal per la costruzione delle UI. Senza entrare nello specifico, le UI potevano essere costruite attraverso Canvas e Widget aggiunti alla HUD che aveva il compito di gestire la logica di visualizzazione e disegno degli stessi, a seconda di come venivano modificati i parametri nel gioco. Seppur come approccio di costruzione della UI, quello con l'HUD è superato, essa non è ancora stata completamente sostituita nella logica di gestione dell'interfaccia. Si possono costruire applicativi senza l'utilizzo di questa classe, essa però mette ancora a disposizione alcune utili proprietà: ad esempio nei multiplayer games gli oggetti HUD esistono sono in client. Questi oggetti possono essere visibili anche su server ma rimangono legati al Player Controller e quindi la classe HUD può essere utilizzata per memorizzare lo stato e il comportamento che ha un singolo utente in client, come può essere la personale UI. La classe HUD è dunque unica per ogni player controller e viene definita nel GameMode, insieme alla classe Pawn predefinita e la classe PlayerController, oltre che a qualsiasi comportamento o impostazione specifiche del gioco.

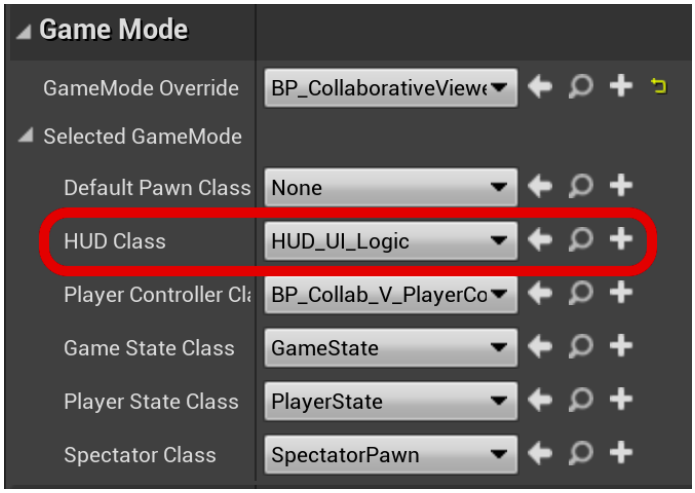


FIGURA 2.39: IMPOSTAZIONE PARAMETRI DEL GAMEMODE, DOVE VIENE INSERITA LA HUD_UI_LOGIC COME HUD CLASS DI RIFERIMENTO

Ad oggi dunque l'HUD resta parte integrante del PlayerController e rappresenta il Monitor di ogni utente, responsabile di riscrivere la Viewport ad ogni frame, dunque, un Widget generalmente aggiunto alla Viewport, diventa parte del PlayerController. Per cui un cambiamento che avviene in un Widget è immediatamente visibile sullo schermo poichè l'HUD ha il compito di aggiornare ad ogni frame la Viewport con le informazioni in suo possesso. Dunque diventa una buona soluzione quella di usare l'HUD come contenitore delle reference dei Widget utilizzati e di tutto ciò che compone le UI. Oltre a questo, nel nostro caso, aggiungendo alla nostra classe *HUD_UI_Logic*, la classe *UI_Interface*, essa diventa il centro di gestione dell'intera logica delle interfacce utente, poiché è ad essa che deve fare riferimento qualunque applicativo voglia utilizzare le UI da noi create, richiamando le apposite funzioni dichiarate nella *UI_Interface* e implementate nell'*HUD_UI_Logic*.

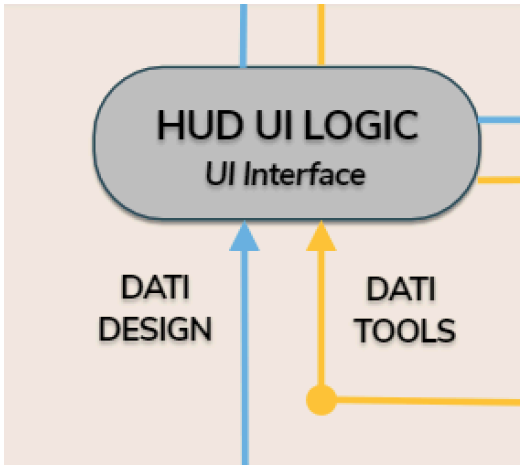


FIGURA 2.40: SEZIONE HUD_UI_LOGIC CON DATI IN INGRESSO

Se da un lato l'HUD_UI_Logic gestisce tutta la logica ed espone le funzioni necessarie al funzionamento delle UI, è chiaro che dal lato dell'applicativo queste funzioni devono essere correttamente richiamate. Essendo nostro obiettivo l'implementazione e studio delle User Interface, si è reso necessario creare una classe da aggiungere all'applicativo che si occupi di richiamare correttamente tutti gli oggetti e gli eventi presenti nel Collab Viewer per metterli in comunicazione con l'HUD_UI_Logic da noi creata.

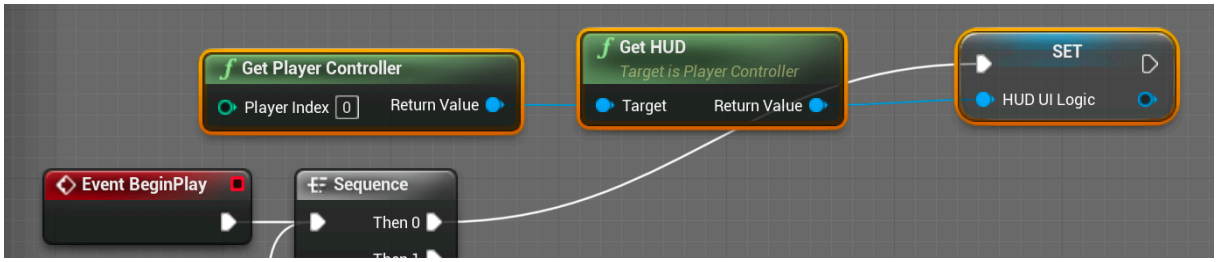


FIGURA 2.41: *EVENT BEGINPLAY* DELLA *DP_INTERFACE* DOVE IMMEDIATAMENTE VIENE CREATA UNA REFERENZA ALLA *HUD_UI_LOGIC* RECUPERANDOLA COME *HUD* DEL *PLAYERCONTROLLER*

Ci preme sottolineare che il Collab Viewer non è stato costruito per questo tipo di utilizzo, poiché nella classe *DP_Interface* per ottenere quanto necessario sono stati utilizzati approcci che, seppur corretti e funzionali, potrebbero risultare non ottimizzati, come invece sarebbe stato se si fosse andati ad agire direttamente sul codice del Template. Per questo si è scelto l'approccio di tenere il tutto più separato possibile, in vista di futuri aggiornamenti da parte di Unreal Engine, come è effettivamente già successo durante la programmazione di questo progetto, quando è avvenuto il passaggio alla versione 4.21.

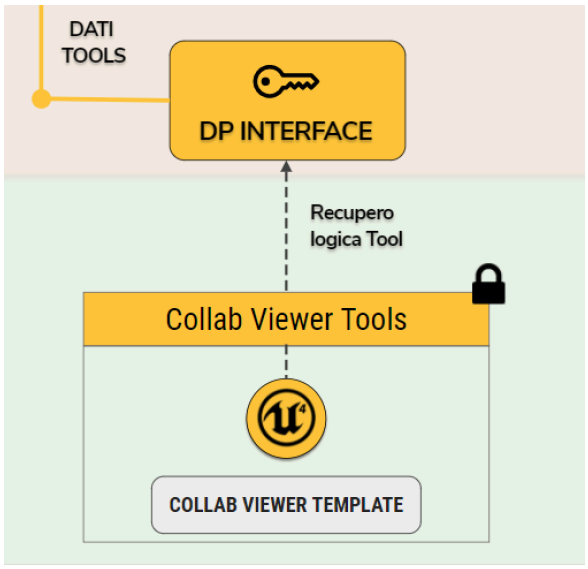


FIGURA 2.42: *BLOCCO RELATIVO AL COLLAB VIEWER TEMPLATE*

3 - INTERFACCIA UTENTE DESKTOP

3.1 - FASE DI IDEAZIONE

Come di consueto all'inizio di un nuovo progetto, prima di procedere con la fase di implementazione, è importante definire chiaramente quale sia il risultato finale che si vuole ottenere. Il nostro approccio in merito alla costruzione dell'interfaccia utente Desktop non si è dunque discostato da questo modus operandi. Innanzitutto abbiamo dedicato qualche settimana alla ricerca e studio di reference su quanto fosse già stato sviluppato in merito, per capire quali fossero gli approcci comunemente utilizzati e quali, tra questi, si rivelassero più funzionali alla costruzione di un'interfaccia utente. In seguito abbiamo proceduto a costruire dei mockup che ci aiutassero a visualizzare graficamente quanto da noi ideato, cercando di definire fin dal principio l'impatto visivo di quello che avremmo implementato.

3.1.1 - REFERENCE E DIFFERENTI APPROCCI

Oggigiorno non è raro trovare siti web e applicazioni che permettono di configurare un autoveicolo, per questo motivo non è stato difficile trovarne e provarne qualcuno in prima persona per valutarne i pregi e i difetti.
Precisiamo che le immagini e i video che seguono non intendono esaltare o denigrare nessuno dei marchi e modelli di autoveicoli presenti, ma esclusivamente fare uno studio oggettivo dei configuratori desktop attualmente presenti sul mercato, considerando anche che alcuni di questi sono progetti privati e non direttamente collegati ai marchi mostrati.

Tra le UI analizzate abbiamo notato che spesso veniva utilizzata una larga porzione dello schermo per mostrare le varianti di configurazione. La scelta è piuttosto ovvia se si considera il fatto che l'utente debba avere sempre la possibilità di visionare senza alcuna difficoltà queste varianti, ma ci siamo resi conto che è altrettanto importante lasciare spazio all'utente di controllare personalmente la resa grafica di quanto aggiunto al veicolo. Per cui non sempre è da considerarsi efficiente occupare buona parte della visuale con informazioni sul prodotto, poiché è proprio questo che deve essere al centro dell'attenzione.

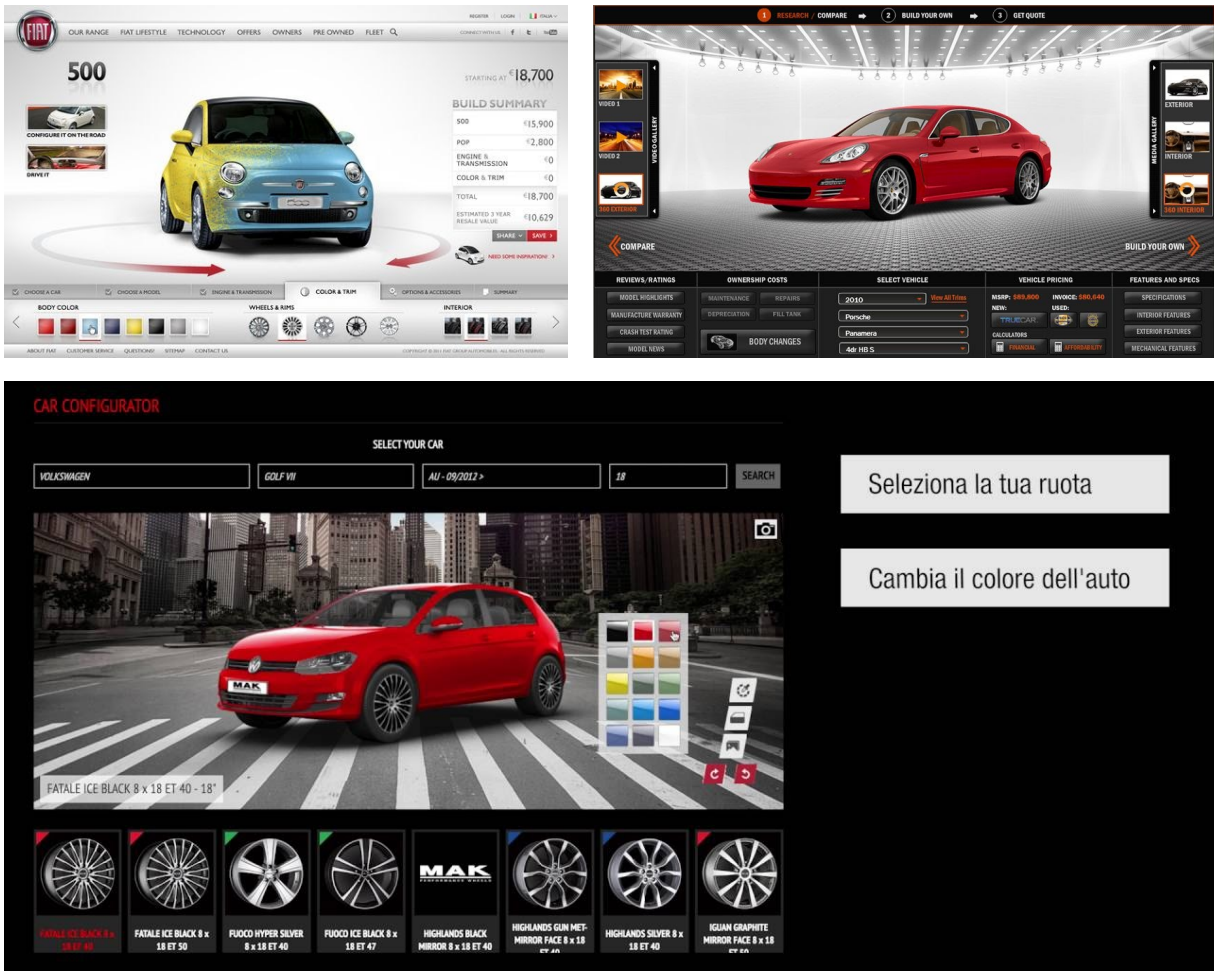


FIGURA 3.1: *ESEMPI DI CONFIGURATORI CHE SACRIFICANO MOLTO SPAZIO DELL'INTERFACCIA PER LE INFORMAZIONI, TOGLIENDONE AL VEICOLO*

Molti dei configuratori analizzati risolvono questo tipo di problema adattando lo spazio a disposizione a seconda della necessità del momento, dedicando alle varianti la giusta quantità di spazio e togliendo dalla visuale tutto ciò che risulta d'ingombro o non necessario al momento. Ad esempio se l'utente, una volta finito di configurare i cerchioni, decide di cambiare la colorazione della carrozzeria, i primi verranno tolti dalla schermata per lasciare spazio alle diverse colorazioni della carrozzeria offerte per quel modello di autoveicolo. Si verranno quindi a creare delle fasi di configurazione, l'una indipendente dall'altra, che verranno mostrate singolarmente su schermo.

Sulla base di questo approccio, indubbiamente il più funzionale data l'area ristretta e limitata offerta dallo schermo, abbiamo cominciato a progettare una nostro prototipo di interfaccia utente, disegnandone il relativo mockup.
In base alle nostre necessità abbiamo definito tre aree logicamente distinte, che di conseguenza andavano separate anche all'interno della UI, per trasmettere questa logica e quest'ordine

anche all'utente. Queste aree riguardano: gli **Asset modificabili** del veicolo, dunque le fasi di configurazione con le relative categorie e varianti; i **Tool** messi a disposizione per poter interagire con il veicolo configurato e con l'ambiente; i **Settings** dell'ambiente stesso e dell'applicazione.

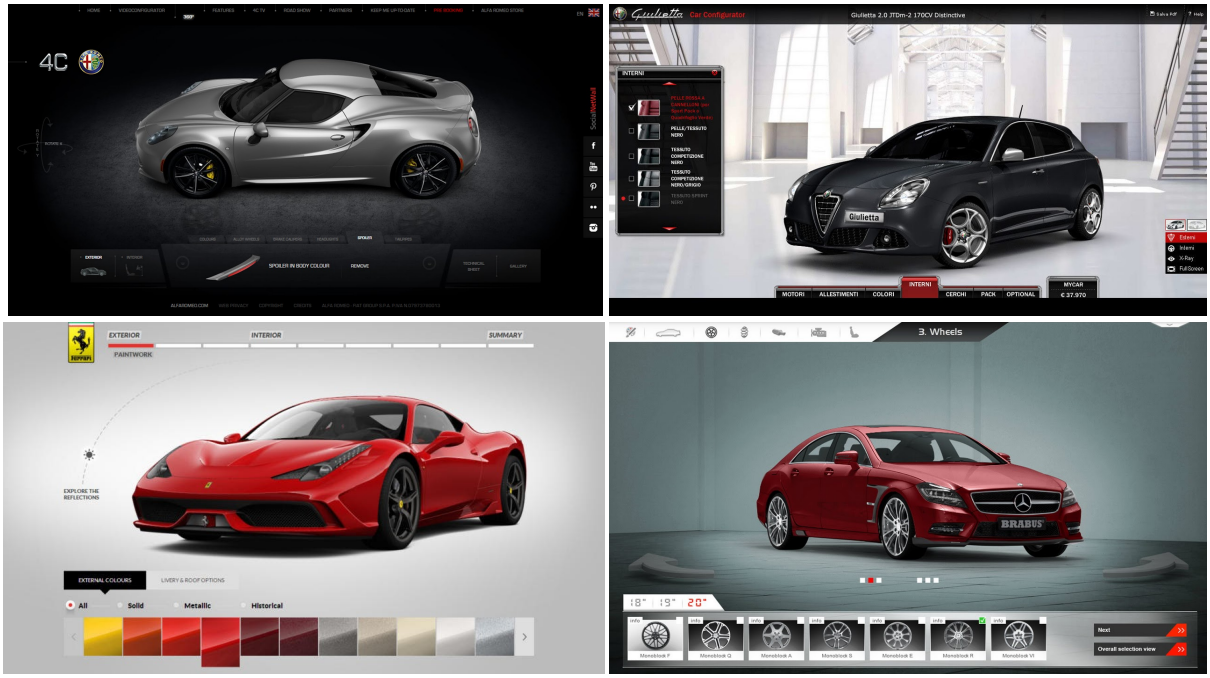


FIGURA 3.2: *ESEMPI DI CONFIGURATORI CHE DEFINISCONO DELLE FASI DI CONFIGURAZIONE, IN MODO DA AGGIORNARE LO SPAZIO A DISPOSIZIONE SECONDO NECESSITÀ*

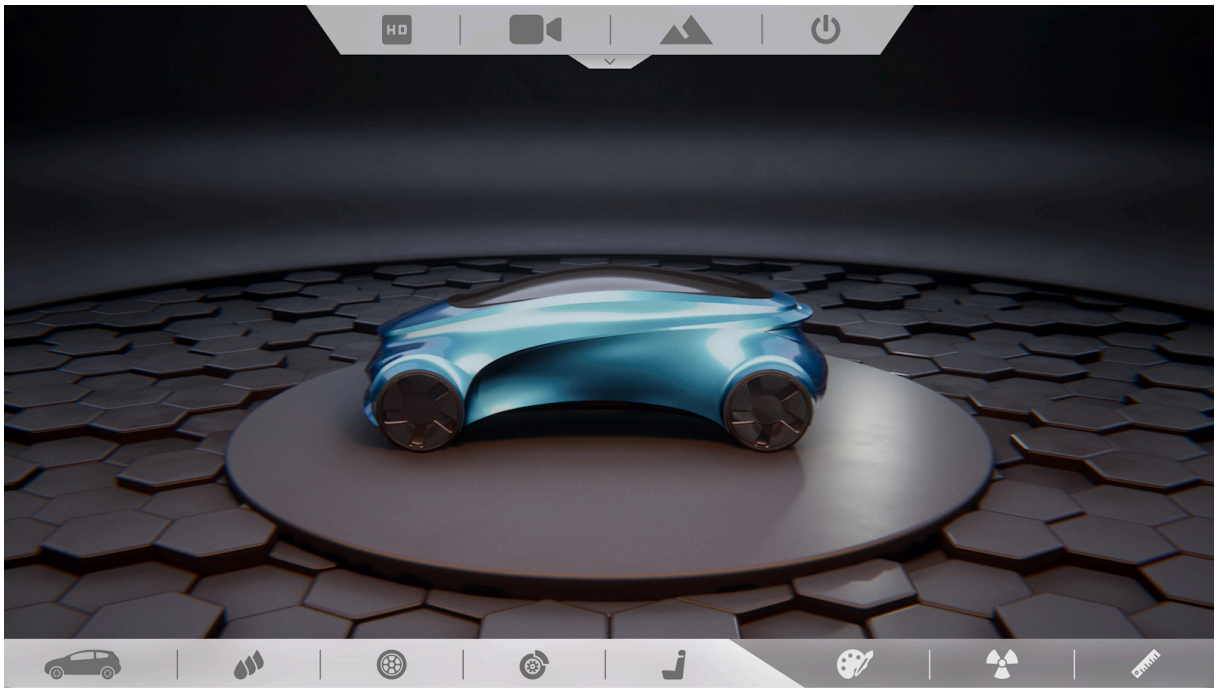


FIGURA 3.3: PRIMA VERSIONE DI INTERFACCIA UTENTE DESKTOP

Come si può vedere in figura 3.3, il primo prototipo di UI prevedeva una barra dei menu posta alla base dello schermo, divisa tra *Tool* e *Asset modificabili*, che descrivevano le fasi del Configuratore, e in alto i *Settings* posti in una sorta di tendina richiudibile a piacimento per ampliare la visuale.
Seppur fosse già questo un risultato convincente, nel quale era possibile gestire la barra dei menu a in base alle proprie necessità, aggiornandola con le categorie e le varianti, aumentandone la dimensione solo in quel contesto, non eravamo ancora pienamente convinti del nostro operato.

Non avevamo tenuto in considerazione il fatto che stavamo costruendo una interfaccia utente per un plug-in modulare e completamente customizzabile e che dunque doveva essere ancora più versatile di quanto non lo fosse questa prima versione.

La soluzione che sembrava risolvere queste problematiche, prestandosi ulteriormente alla nostra situazione, l'abbiamo trovata non tanto nelle interfacce dei configuratori illustrati sopra, più che altro riconducibili a siti web, ma in alcuni approcci sperimentali di alcuni sviluppatori che hanno utilizzato Unreal Engine.

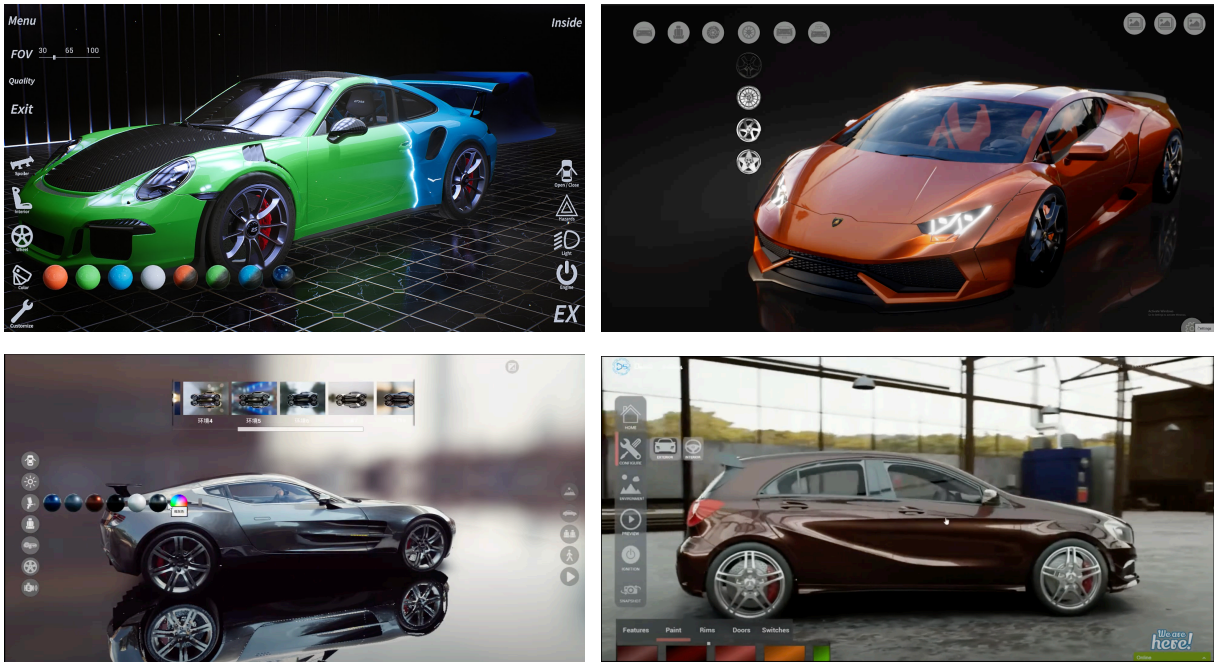


FIGURA 3.4: SCREENSHOT DI VIDEO CHE PRESENTANO ALCUNI CAR CONFIGURATOR COSTRUITI CON UNREAL ENGINE 4

Questo tipo di approccio richiama indubbiamente la regola, sempre più efficace e diffusa, del "*less is more*", in cui ogni menu viene ridotto alle singole icone che lo compongono, permettendo di fatto una maggiore visibilità dell''autoveicolo nella finestra desktop. Inoltre, in questo modo, è molto più semplice la costruzione di un menu modulare, dovendo occuparci esclusivamente tanti slot customizzabili quante sono le fasi o le categorie presenti nel configuratore, così come per le opzioni dei Tool.

Nella maggior parte dei casi in figura 3.4[39] gli insiemi di icone sono sempre visibili, continuando ad occupare una parte di schermo a prescindere che vengano utilizzati o meno. Fa eccezione il configuratore in alto a sinistra della figura 3.4, nel quale tutti i menu si possono chiudere, riducendoli ad una singola icona all'angolo dello schermo quando non vengono utilizzati, come illustrato in figura 3.5.



FIGURA 3.5: MENU CHE, RIDOTTO AL MINIMO, MOSTRA SOLO 4 ICONE AGLI ANGOLI DELLO SCHERMO

Ci è parso chiaro che fosse questa la logica da seguire per una interfaccia non invasiva, seppur ci fossero ancora alcuni dettagli che andavano modificati. Come si può notare dall'immagine 3.4, oltre a non essere considerate le sottocategorie che una fase di configurazione potrebbe avere, ad esempio i colori della carrozzeria suddivisi per tipo, oppure i cerchioni suddivisi per marchio o per grandezza, vediamo che le varianti vengono rappresentate nella direzione opposta a quella di apertura del menu, spesso andando ad invadere la parte di

schermo occupata dal veicolo. Seppur l'attenzione dell'utente venga indirizzata verso il menu aperto, la modalità di presentazione delle varianti ci ha comunque dato la sensazione di essere piuttosto invasiva. Dunque per la creazione della nostra interfaccia abbiamo voluto adottare un approccio differente, riapplicando, ancora una volta, la logica di riutilizzo degli spazi a disposizione.

3.1.2 - MOCKUP E RIUTILIZZO DEGLI SPAZI

Come è possibile vedere dalla figura 3.6, la nostra UI Desktop è composta da 4 menu, posti agli angoli dello schermo: in basso **Configurator** e **Tools**, in alto **Settings** e **Logout**. La loro apertura avviene orizzontalmente e comporta automaticamente la chiusura degli altri menu eventualmente aperti. Questa scelta è data dal fatto che non vi sono ragioni per cui l'utente dovrebbe usare in contemporanea, ad esempio, il menu *Configurator* con il menu *Tools*. Da questo segue che ogni menu, potenzialmente, possa occupare l'intero spazio orizzontale disponibile di fianco ad esso. Ciò è vantaggioso sia per i *Tools*, nel caso un domani se ne volessero aggiungere altri, ma soprattutto per gli Asset modificabili che compongono il menu *Configurator*, non potendo sapere a priori quante fasi di configurazione, categorie, sottocategorie e varianti dovranno essere gestite.

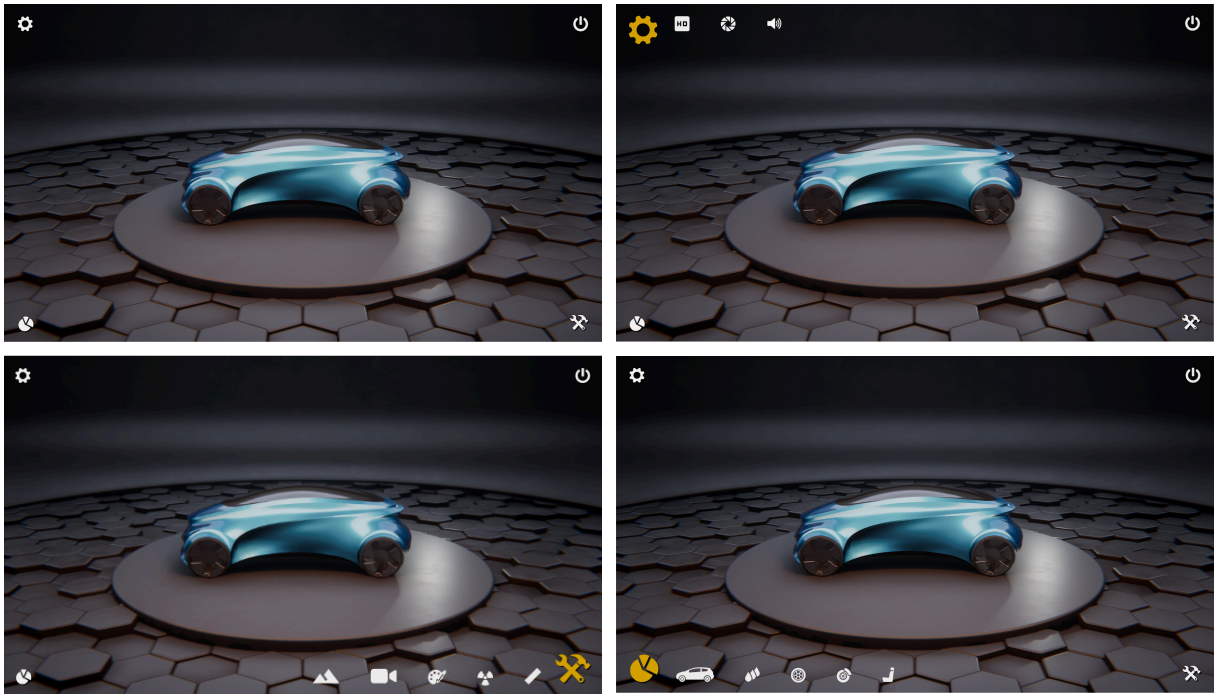


FIGURA 3.6: SECONDA VERSIONE DELLA UI DESKTOP, CON I VARI MENU APERTI

L'imprevedibilità del contenuto del menu *Configurator* (inizialmente chiamato *Asset*, per cui nei mockup spesso apparirà con questo nome) non si limita alle sole fasi di configurazione, ma anche al numero di categorie e varianti, risultando quindi il menu di più difficile gestione all'interno dell'interfaccia. Per cui si è definita una logica che si adattasse alle esigenze di modularità imposte da questo, per poi applicarla anche ai menu *Tools* e *Settings*, consci della loro struttura pressoché fissa e quindi meno problematica.

Così facendo ogni menu occupa sempre lo stesso spazio e non va mai ad invadere l'area centrale dello schermo, dove presumibilmente si trova il veicolo o il prodotto da configurare.
Oltre ai i vantaggi sopracitati, si sono presentati tuttavia alcuni problemi, tra questi vi era sicuramente quello di come fornire un costante feedback all'utente in merito a in che fase di configurazione si trovasse o di come tornare alle categorie precedenti della fase scelta.
Pertanto inizialmente si è scelto di sostituire l'icona del menu *Configurator* con quella della fase scelta, come si può vedere nell'esempio in figura 3.7. In questo modo viene costantemente dato all'utente un feedback riguardo a quale fase stia configurando, permettendo inoltre, dopo aver cliccato sull'icona di questa, di tornare alla sezione precedente. Ci siamo accorti però che questo approccio poteva non essere del tutto intuitivo, oltre al fatto che obbligava l'utente a cliccare più volte sulla medesima icona per tornare al menu principale. Questa versione è stata quindi migliorata nell'applicativo, lasciando comunque il feedback con il cambio di icona, ma facendo in modo che un click su di essa bastasse per tornare direttamente al menu principale, a prescindere dal numero di sottocategorie presenti. Per tornare invece indietro di una singola categoria, abbiamo aggiunto un bottone *Back* in testa alla sezione delle categorie attualmente visualizzate.



FIGURA 3.7: AGGIORNAMENTO DEL MENU *CONFIGURATOR* SE SCELTA, IPOTETICAMENTE, LA FASE *RIMS*, CON LE RELATIVE CATEGORIE DATE DAI DIFFERENTI MARCHI A DISPOSIZIONE E CON SOSTITUZIONE DELL"ICONA DEL MENU CON QUELLA DELLA FASE SCELTA. NON C'È TASTO BACK POICHÉ NON PREVISTO NEI MOCKUP, MA AGGIUNTO QUANTO GIÀ SI LAVORAVA ALL'APPLICATIVO.

Se da un lato ogni categoria e sottocategoria viene gestita come appena illustrato, d'altro canto la logica di visualizzazione delle varianti cambia. Queste, a differenza delle singole categorie, hanno l'esigenza di mostrare chiaramente all'utente come verrà modificata una specifica parte del prodotto, a seguito delle scelte fatte in precedenza, per cui sopraggiunge la necessità di rappresentarle su schermo con icone leggermente più grandi rispetto a quelle delle categorie.

Come si può vedere in figura 3.8, le varianti sono visivamente più grandi e allineate al centro del menu. Se poi ne viene selezionata una, questa si ingrandisce ancor più mostrando anche la rispettiva Label e un infobox a sinistra con tutte le informazioni recuperate dal *Payload* (Per *Payload* vedi capitolo 2.4.3). Questo infobox viene reso visibile solo dopo aver selezionato una variante, ma, sempre in ottica di una interfaccia utente non invasiva, può essere ridotto a icona e nascosto dalla visuale a discrezione dell'utente, nel caso volesse osservare più liberamente il modello del prodotto senza ingombri derivanti dalla UI.



FIGURA 3.8: VARIANTI DELLA A FASE *RIMS*, CON RELATIVO PANNELLO DELLE INFORMAZIONI

È stato scelto di mantenere la stessa struttura anche per i menu *Tools* e *Settings* per due ragioni: l'utente non si vede costretto ad apprendere diversi tipi di interazioni a seconda del menu, limitandosi quindi a comprendere la logica sopradescritta valida in tutta la UI; il tipo di approccio scelto resta comunque il più funzionale ed efficiente anche per gli altri menu.



FIGURA 3.9: COME VIENE AGGIORNATO IL MENU *TOOLS* QUANDO VIENE SCELTO, IPOTETICAMENTE, LO STRUMENTO *TELEPORT*, MOSTRANDONE LE RELATIVE *OPTIONS* E SOSTITUENDO L'ICONA DEL MENU CON QUELLA DEL *TOOL* SCELTO

Essendo ovvio che i mockup sono un mero esempio grafico di ciò che si vuole ottenere, una volta deciso quanto si voleva implementare siamo passati all'effettiva progettazione dell'interfaccia su Unreal Engine.

L'intera interfaccia utente è strutturata all'interno di un unico Widget, chiamato ***General_UI***, che si compone di parti statiche, già pre-costruite all'interno di esso, e parti dinamiche che verranno costruite in relazione a determinati eventi che spiegheremo nei sottocapitoli successivi.

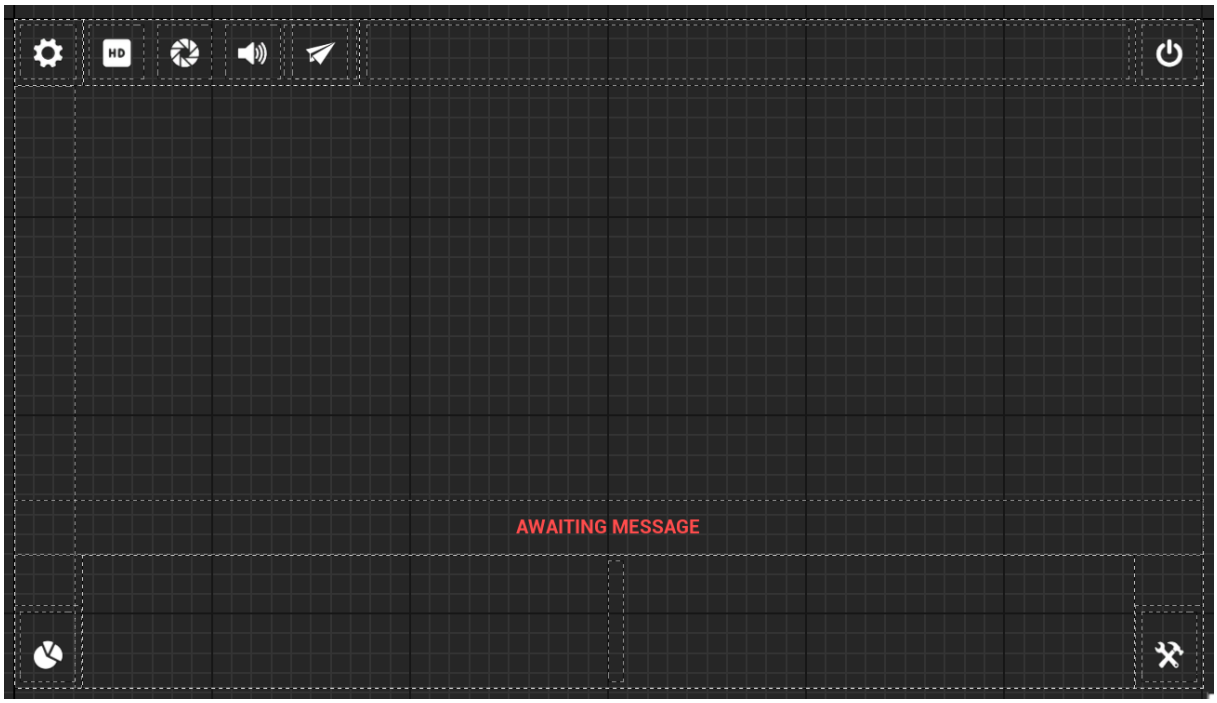


FIGURA 3.10: GENERAL UI NEL WIDGET EDITOR DI UNREAL ENGINE

Come si evince dall'immagine 3.10, le parti statiche, ossia quelle rappresentanti i quattro Menu illustrati nel Mockup che l'utente visualizzerà inizialmente all'avvio dell'applicativo, sono di fatto i quattro *Button* posti agli angoli della schermata, chiamati ***Asset Button*** per il *Menu Configurator*, ***Tools Button*** per il *Menu Tools*, ***Settings Button*** per il *Menu Settings* ed ***Exit Button*** per chiudere l'applicazione.

3.2 - MENU CONFIGURATOR

La sezione ***Configurator*** della *General_UI* è riservata alla visualizzazione delle varie categorie e varianti del configuratore. Interagendo con essa, l'utente potrà customizzare a proprio piacimento il veicolo scegliendo le varianti per ogni fase disponibile, come definito dallo sviluppatore nel LVS e quindi nella DT associata. Tutte le fasi, categorie e sottocategorie vengono graficamente rappresentate nella UI come mostrato in figura 3.11, ovvero con la struttura del widget **BP Asset Button.**

3.2.1 - LAYOUT GRAFICO DEL WIDGET ASSET BUTTON

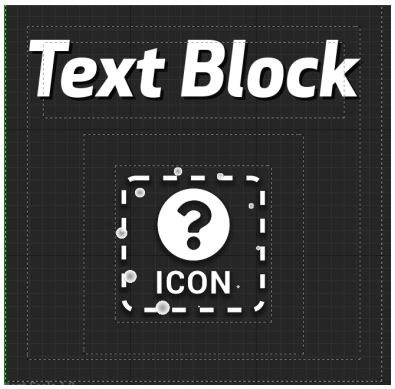


FIGURA 3.11: STRUTTURA BP ASSET BUTTON

Il widget BP Asset Button presenta un campo **Label** di tipo *Text*, in cui verrà scritta il nome della categoria o fase, ed un bottone con all'interno l'icona della suddetta categoria, definita dal designer tramite inserimento diretto della stessa nella DT oppure mediante l'interfaccia dell'Editor *W_Payload-Filler_TreeView.* In aggiunta a ciò, è presente anche un'immagine di caricamento per ogni *BP Asset Button* che viene visualizzata durante quegli istanti necessari al programma per recuperare da DT l'icona corretta per quel widget.

Per perseguire sempre uno stile minimale, si è scelto di rendere visibile la Label solo nel caso in cui l'utente passi il cursore del mouse sopra l'icona, così da invadere il meno possibile la viewport con informazioni testuali.
Questi *BP Asset Button* vengono poi aggiunti uno ad uno al ***BP Category Row***, widget adibito alla disposizione di tali bottoni nella sezione *Configurator* della *General_UI*. Inizialmente verranno visualizzate le fasi del configuratore, quindi all'interno del *BP Category Row* troveremo tanti *BP Asset Button* quante sono le fasi dichiarate nel LVS e di conseguenza nella DT.

3.2.2 - RECUPERO DEI DATI E CREAZIONE DEI RELATIVI WIDGET

Tutta la logica di recupero dei dati relativi alle diverse categorie e varianti da DT viene implementata via Blueprint all'interno del *BP Category Row*. Infatti, avendo al suo interno un'istanza della classe *VT_Manager*, è possibile richiamare la funzione ***Get Child Categories and Variants,*** la quale, ricevendo in input il Row Name di una categoria, restituisce: un Array

Categories di *Variant Category* con tutte le *Structure* delle sottocategorie di quella passata in input e un Array **Variants** di *Variant Entries* con tutte le *Structure* delle varianti di tale categoria. Il campo *Category* in input alla funzione di default possiede il valore *None*, caso in cui tale funzione debba restituire tutte le fasi che, per come sono costruite, non presentano nessuna categoria padre. Quindi ogni volta che vi è la necessità di visualizzare le fasi, basterà eseguire un reset del campo Category alla sua variabile di default.

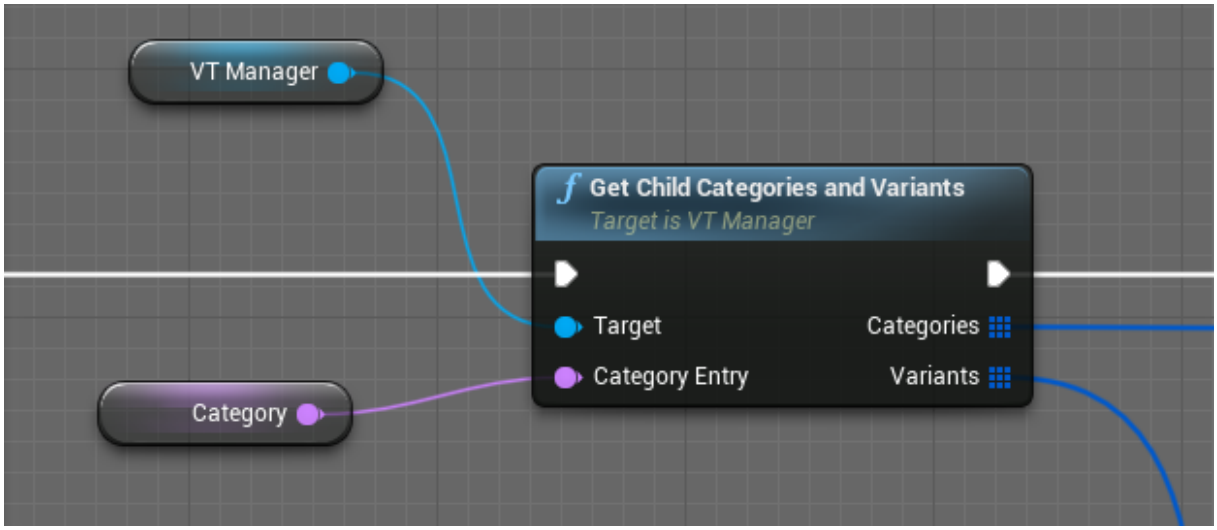


FIGURA 3.12: FUNZIONE GET CHILD CATEGORIES AND VARIANTS DELLA CLASSE VT MANAGER

L'Array di *Variant Entries* è mutualmente esclusivo all'Array di *Variant Category*, in quanto la presenza di sottocategorie esclude quella di varianti e viceversa. Se, per esempio, la lunghezza dell'Array di *Variant Category* è pari a zero significa che in input è stato passato il Row Name di una categoria che nella DT, alla voce *Child Category* della sezione *Phase-Category Data*, non presenta nessuna sottocategoria annidata, ma che possiede invece delle varianti ad essa associate. Viceversa se la lunghezza dell'Array di *Variant Category* è maggiore di zero, significa che la categoria passata in input presenta altre sottocategorie annidate e quindi non vi saranno varianti in output alla funzione. Questo ci permette di sapere ogni volta che viene invocata la funzione *Get Child Categories and Variants* se è il caso di aggiornare il menu *Configurator*, creando un nuovo *BP Category Row* con le nuove sottocategorie.

A seconda di quale Array in output alla funzione possieda una dimensione maggiore di zero, che sia *Categories* o *Variants,* viene chiamata rispettivamente la funzione ***Create Button Category*** o la funzione ***Create Button Variant.*** Entrambe eseguono all'incirca lo stesso compito: recuperare i dati di ogni categoria o variante contenuti nelle Structure associate (*Variant*

Category o Variant Entries); creare per ognuna di esse un widget *BP_Asset Button* che, come oramai è chiaro, viene utilizzato per rappresentare graficamente sia le varianti che le categorie; associare infine un indice a tale widget per identificarlo univocamente. Nel caso si tratti di una categoria o fase, questo indice rappresenta la posizione occupata da questa all'interno dell'Array di *Variant Category,* mentre per le varianti corrisponde all'indice delle stesse nell'Array di *Variant Entries.* L'unica differenza tra queste due funzioni è che per *Create Button Variant* ogni variabile booleana ***isLeaf*** di ogni *BP Asset Button* viene settata a *TRUE* per differenziarli da quelli omonimi che rappresentano le categorie in cui invece tale variabile è settata di default a *FALSE*.

Nonostante fasi/categorie e varianti condividano lo stesso widget, si è deciso di distinguerle graficamente le une dalle altre utilizzando un **WidgetSwitcher** all'interno del *BP_Asset Button* stesso, che permette di cambiare il layout del bottone a seconda che debba rappresentare una categoria o una variante.

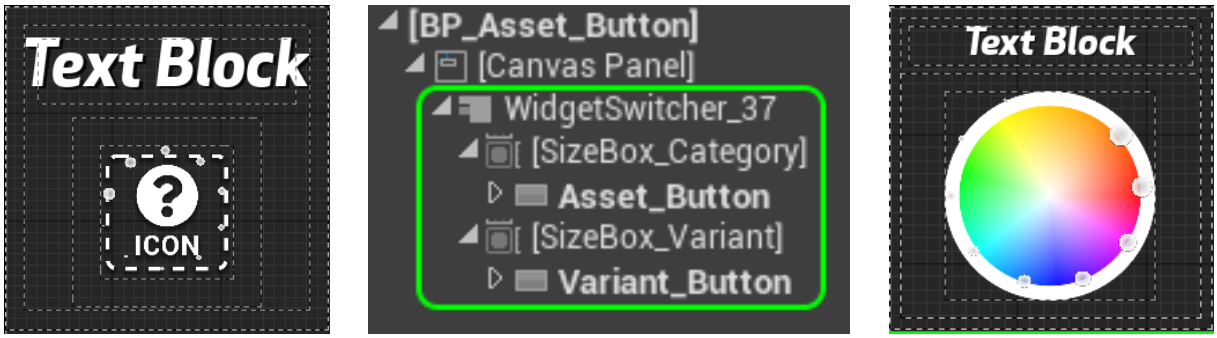


FIGURA 3.13: WIDGETSWITCHER DEL BP ASSET BUTTON CON ALL'INTERNO UN ASSET_BUTTON (SX) E UN VARIANT_BUTTON (DX)

Prendendo in esame *Create Button Category (*il discorso è pressoché identico anche per *Create Button Variant),* analizziamo quali campi di una singola *Variant Category,* medesimi a quelli presenti nel Row Editor della DT, sono fondamentali al *BP Asset Button* per essere configurato correttamente. In figura 3.14 vediamo come per ogni *Variant Category* dell'Array *Categories* viene recuperata sia l'icona, o più precisamente la **Soft Object Reference** alla Texture 2D dell'icona, che il nome della categoria all'interno del campo Label. Queste poi vengono passate alla funzione ***Add Button***, incaricata di creare un nuovo *BP Asset Button* associando alla variabile Label di quest'ultimo il nome della categoria, passato alla funzione come Name Category, e la reference dell'icona alla variabile Icon Soft.

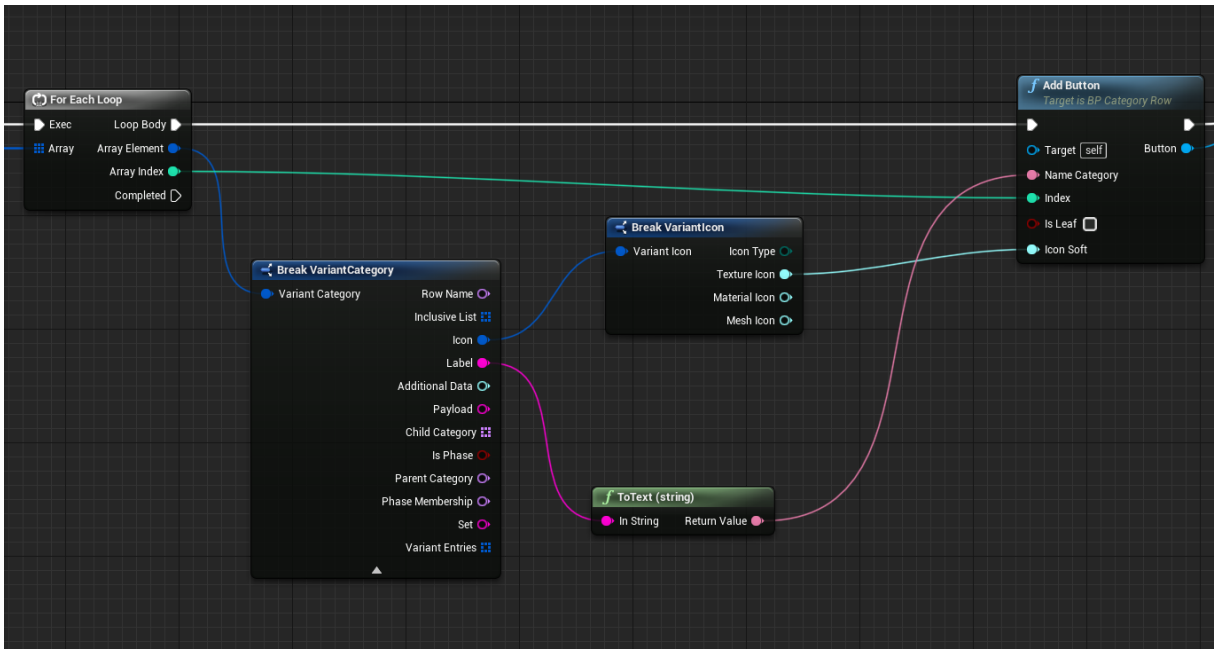


FIGURA 3.14: SEQUENZA DI NODI PER LA CREAZIONE DI UN NUOVO BP ASSET BUTTON CON I DATI PRELEVATI DAL VARIANT CATEGORY

3.2.3 - CARICAMENTO ASINCRONO DELLE ICONE

La scelta di utilizzare una *Soft Object Reference* e non una classica *Hard Reference* per la *Texture2D* dell'icona è dovuta alla volontà di ottimizzare le prestazioni dell'applicativo, evitando quindi un sovraccarico dello stesso durante il caricamento delle icone, oltre che ad una condizione impostaci dalla struttura stessa della DT scritta via codice. Infatti le Hard Reference definite in un widget vengono caricate immediatamente dal programma alla creazione del widget stesso, a prescindere dal tipo di asset a cui esse fanno riferimento (immagine, texture, mesh, etc..). Può verificarsi quindi il malsano evento in cui il software debba caricare molto velocemente e nello stesso momento molteplici variabili di diverso tipo per poter inizializzare correttamente il widget interessato, anche se alcune di queste, per esempio quelle di tipo *Texture2D*, risultano essere molto pesanti. Per questo motivo si è soliti utilizzare le Soft Object Reference per le Texture, in modo che possano essere caricate in modalità asincrona durante l'esecuzione del restante flusso di dati, non dovendo creare stati di sospensione d'esecuzione dell'applicativo o addirittura eccessivo sovraccarico che porti ad un crash. Questo caricamento asincrono su Unreal Engine viene eseguito dal nodo ***Async Load Asset***, mostrato in figura 3.15, che in input prende la Soft Object Reference d'interesse, in questo caso di tipo *Texture2D*, e di fatto avverte il programma che il caricamento di tale risorsa non debba essere immediato, ma parallelizzato all'esecuzione delle istruzioni successive, non impedendo così il prosieguo del restante flusso di codice. Non appena viene caricata correttamente l'icona, il

nodo Async Load Asset esegue i nodi collegati al pin *Completed* che, nel nostro caso, dopo aver recuperato l'icona caricata, restituita dal nodo come generico Object, ne esegue un *Cast* a *Texture2D* per poi finalmente caricarla come icona del *BP Asset Button*.

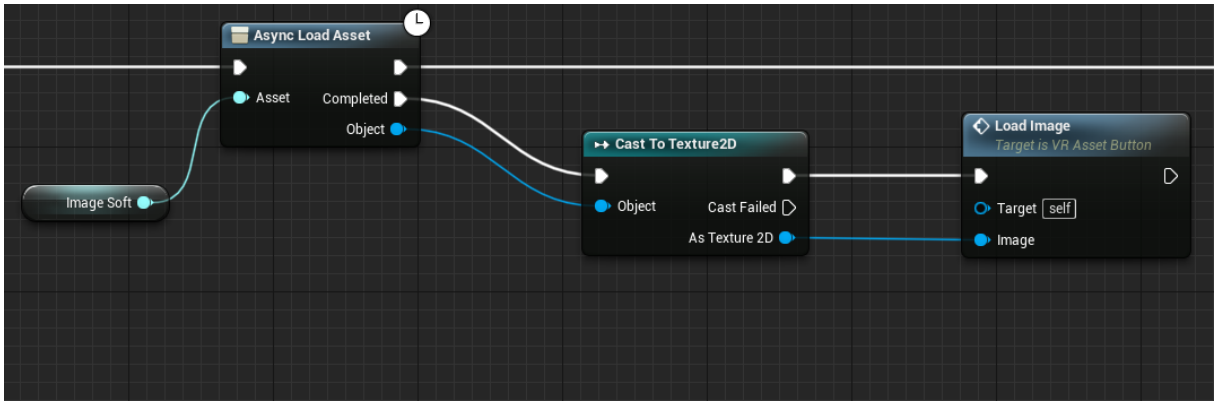


FIGURA 3.15: CARICAMENTO ASINCRONO DELLA TEXTURE2D DELL'ICONA MEDIANTE ASYNC LOAD ASSET

3.2.4 - LEAFS BOX E AGGIORNAMENTO DELLE CATEGORIE

Per le varianti il discorso è pressoché il medesimo se non per il fatto che in questo caso è fondamentale prelevare dalla *Structure* d'interesse, quindi *Variant Entries*, anche il *Payload* associato, in modo da poter mostrare correttamente all'utente tutte le informazioni che caratterizzano quella determinata variante. Dunque, alla creazione di un widget ***BP_Infobox,*** gli viene passato il *Payload* come stringa in formato JSON, oltre che l'indice che identifica univocamente la variante a cui questo widget fa riferimento. In questo modo allo stesso indice viene associato sia un *BP Asset Button* con una variante che il relativo *BP_Infobox* con tutti i campi del *Payload* ad essa riferiti. Una volta inizializzati correttamente l'icona, la label e il *Payload* nei rispettivi widget, il *BP Asset Button* viene aggiunto ad un Horizontal Box, denominato **LEAFS BOX**, di default non visibile e posizionato nella parte inferiore della *General_UI*. Allo stesso tempo, ogni *BP_Infobox* creato viene inserito in un apposito Array in modo da poterli poi recuperare con facilità specificando solamente l'indice distintivo.

Riepilogando, non appena l'utente clicca sull'icona del *Menu Configurator,* la *General_UI* provvede alla creazione del primo *Category Row*, delegando a quest'ultimo il compito di recuperare i dati delle fasi dalla DT e creando di conseguenza un *BP Asset Button* per ognuna di queste (figura 3.16).



FIGURA 3.16: ESEMPIO DI BP ASSET BUTTON PER DELLE FASI

Ogni volta che viene cliccato un qualunque *BP_Asset Button* viene chiamato un *Event Dispatcher* ***On Click Asset*** che notifica alla *General_UI* se il bottone cliccato corrisponde ad una categoria/fase (variabile interna *isLeaf* settata a FALSE) oppure ad una variante. A seconda del valore della variabile *isLeaf* , la *General_UI* decide se richiamare la funzione *Get Child Categories and Variants,* in caso siano presenti delle sottocategorie o varianti da visualizzare, oppure se invece è opportuno recuperare l'infobox relativo al *BP Asset Button* cliccato tramite indice e quindi attivare nel VM la variante scelta.

Nel primo caso, se vi sono delle sottocategorie annidate a quella cliccata, come input alla funzione *Get Child Categories and Variants,* attraverso la variabile *Category,* viene passato il Row Name associato alla categoria cliccata dall'utente e viene creato di fatto un nuovo *Category Row* con le nuove sottocategorie (vedi esempio in figura 3.17), se invece sono presenti delle varianti, viene nascosto il Category Row e reso visibile il LEAFS BOX con tutti i *BP Asset Button* delle varianti nel formato del secondo layout definito (figura 3.18).



FIGURA 3.17: BP ASSET BUTTON DELLE SOTTOCATEGORIE DELLA FASE CAR PAINT CON CURSORE SULLA CATEGORIA PASTELLO



FIGURA 3.18: BP ASSET BUTTON DELLE SOTTOCATEGORIE DELLA FASE CAR PAINT CON CURSORE SULLA CATEGORIA PASTELLO

Se invece l'*Event Dispatcher* rileva il click su una variante, viene chiamata la funzione ***Change By Category And Index*** del VT_Manager che, prendendo in input l'indice della variante e il nome dell'ultima sottocategoria cliccata, attiva da Blueprints la variante scelta. Parallelamente a questa funzione viene anche recuperato e visualizzato sulla viewport il rispettivo infobox contenente il *Payload* della variante in questione (figura 3.20).

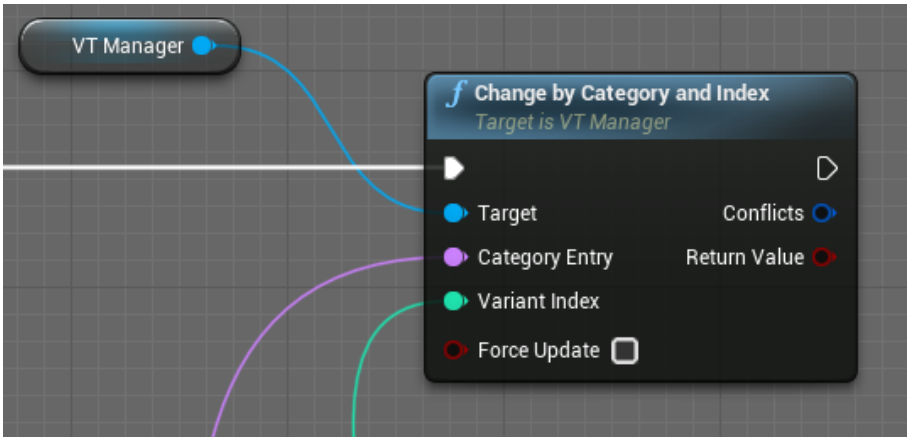


FIGURA 3.19: FUNZIONE CHANGE BY CATEGORY AND INDEX



FIGURA 3.20: INFOBOX CON LE CARATTERISTICHE DELLA VARIANTE BLU MONTECARLO

Per facilitare la navigazione delle varie categorie e varianti, in ogni *Category Row* è presente un ***BP Back Button*** che permette di tornare all'ultima sottocategoria cliccata e visualizzare quindi l'apposito *Category Row*, evitando di dover per forza ri-cliccare sull'icona *Configurator,* riprendendo quindi la navigazione dalle fasi per recuperare la categoria d'interesse. Per implementare ciò, si è reso necessario memorizzare tutti i *Category Row* creati in un Array, così che fossero velocemente reperibili dall'applicativo in caso di aggiornamento della UI su azione dell'utente. Ogni volta che viene cliccato il *BP Back Button*, la General_UI provvede a rendere invisibile il *Category Row* corrente e visualizzare invece quello precedente memorizzato nell'Array.

3.3 - MENU TOOLS

La sezione ***Tools*** della ***General_UI*** è adibita invece alla visualizzazione dei vari strumenti messi a disposizione dal Collab Viewer e del tool *Environment* creato interamente da noi, per agire sul prodotto configurabile e sull'ambiente circostante. Come anticipato nel capitolo **2.4**, abbiamo scelto di non modificare la struttura del template, ma solo di sfruttare a nostro vantaggio quanto messo a disposizione. Per fare ciò abbiamo dovuto costruire la logica di funzionamento di questo menu in modo che recuperasse correttamente i tool di nostro interesse, le relative opzioni e che, alla scelta di una di queste, attivasse correttamente lo strumento scelto.

Il Collab Viewer è un Template che di base non è programmato per interfacciarsi con Blueprint esterne ad esso, infatti possiede una struttura "chiusa" non predisposta ad essere modificata dallo sviluppatore se non agendo direttamente su di esse. Si è resa quindi necessaria la creazione di una classe adibita a mettere in comunicazione la struttura del Collab Viewer con la nostra UI.

La classe *Actor* ***DP_Interface*** viene infatti impiegata per recuperare tutte le informazioni del Collab Viewer necessarie da passare all'***HUD_UI_Logic***, classe HUD adibita alla creazione dei Widget, mettendole a disposizione della *General_UI*. Per fare ciò ogni Widget creato contiene al suo interno un Button, al cui evento *OnClicked* è legato un ***Event Dispatcher***. In seguito alla creazione di un qualunque Widget nella *HUD_UI_Logic* vengono impiegati una serie di *Event Dispatcher,* oltre a quello interno al Widget creato, che permettono di mettere in comunicazione questa classe HUD con la *DP_Interface,* legando agli eventi descritti nella *HUD_UI_Logic* la logica di funzionamento del Collab Viewer recuperata dalla *DP_Interface.* Questi eventi richiamati implementano le funzioni dichiarate nella *UI_Interface* spiegata nel capitolo **2.4**. Consci che sia un procedimento complesso ed intricato, ma necessario per mantenere la nostra interfaccia utente separata dal *Collab Viewer Template*, in figura 3.21 è stato inserito uno schema riassuntivo di tutta la logica di funzionamento, in modo da far comprendere meglio al lettore la successione degli eventi e di tutti i passaggi che verranno spiegati in seguito.

Sempre per comodità al lettore verranno in seguito riportati anche frammenti dello schema completo, utili a seguire il filo del discorso, ma si è scelto di mostrare prima lo schema nella sua interezza, in modo che vi si possa fare riferimento qualora fosse necessaria una visione più ampia per comprendere l'insieme dei passaggi che descriveremo.

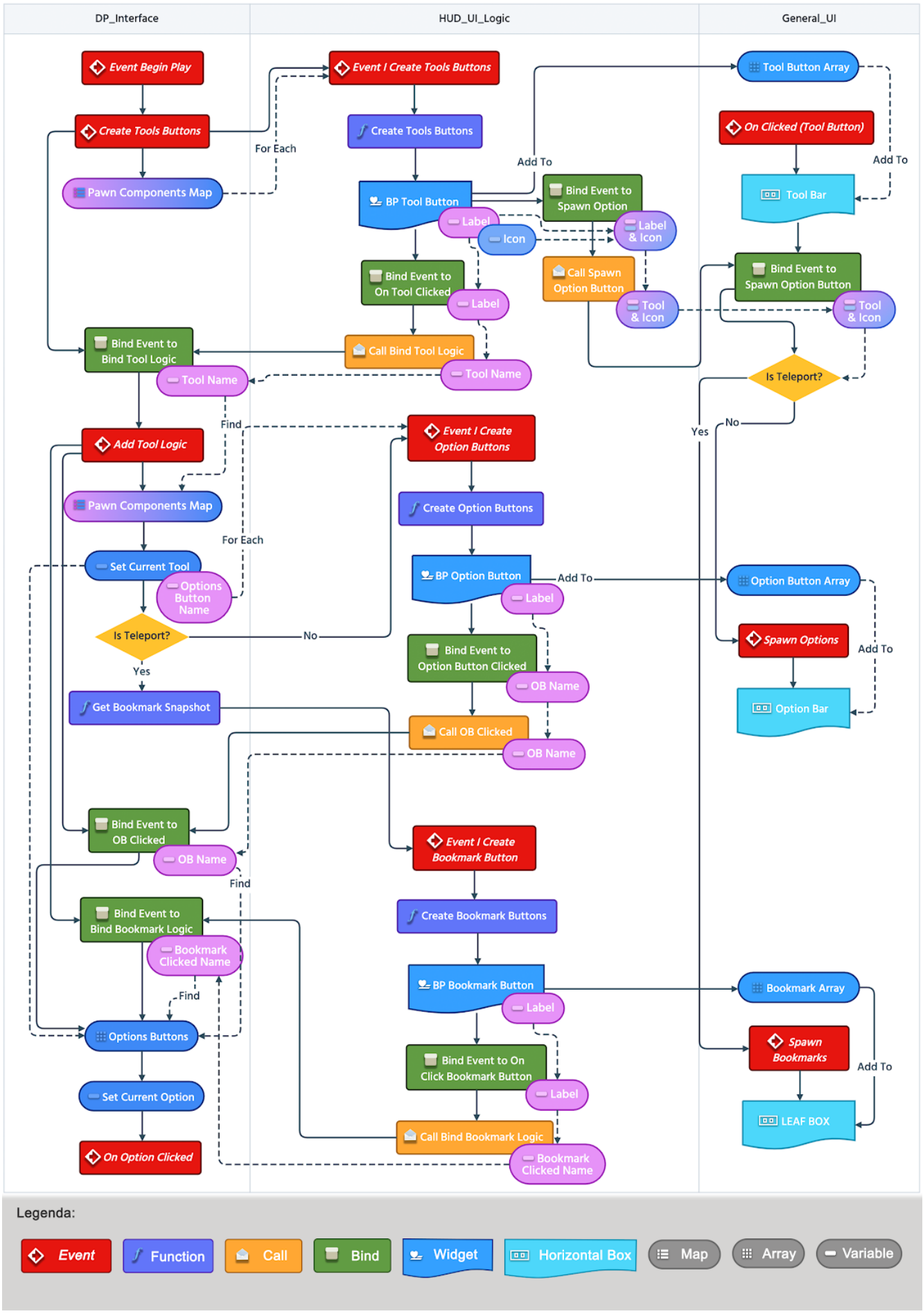


FIGURA 3.21: SCHEMA FUNZIONAMENTO TOOL

3.3.1 - WORKFLOW DI RECUPERO DELLE INFORMAZIONI

All'avvio dell'applicazione vengono chiamati tutti gli *Event Begin Play* di tutte le classi dell'applicativo. Al suddetto evento appartenente alla *DP_Interface* sono stati collegati due eventi che, chiamati in successione, sono alla base di tutta la creazione e il funzionamento dei Tool della nostra UI: **Add Tool Component** e il conseguente **Create Tools Buttons**.

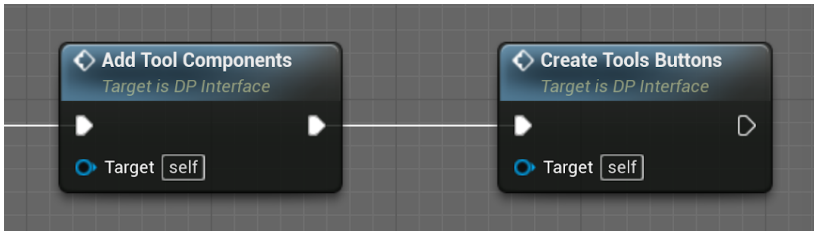


FIGURA 3.22: FUNZIONI ADD TOOL COMPONENTS E CREATE TOOLS BUTTONS

Nel *Collab Viewer Template* i Tool sono degli Actor Component del Player Pawn, classe che definisce la logica di navigazione e interazione con l'ambiente. Per poterli quindi utilizzare a nostro piacimento abbiamo prima recuperato dal Pawn corrente tutti i suoi Component di tipo ***BP_BaseCommand***, la classe base da cui tutti i Tool ereditano, e poi li abbiamo messi in una apposita *Map*, ***PawnComponentsMap***, composta da delle chiavi di tipo *Name* e dalle relative istanze *BP_BaseCommand*, per e poterli recuperare in qualsiasi momento specificandone semplicemente il nome.

Il successivo evento *Create Tools Buttons* utilizza questa *Map* per associare ad ogni Component il relativo *Widget* ***BP Tool Button***. Per ognuno di questi viene chiamato l'evento **I_CreateToolsButtons**, definito nella *UI_Interface* e implementato nell'*HUD_UI_Logic*, che riceve in input la variabile *Name* con il nome del Tool per cui si vuole creare il widget.

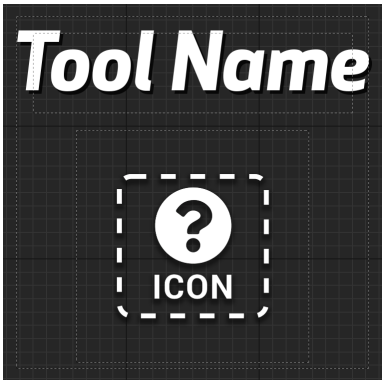


FIGURA 3.23: BP_TOOL_BUTTON

Nell'*HUD_UI_Logic* l'evento *I_CreateToolsButtons* richiama l'omonima funzione che riceve in input la variabile *Name* passata all'evento stesso e restituisce in output il widget *BP Tool Button*.

Come si evince dal nome, il nodo principale della funzione *Create Tools Buttons*, è ***Create BP Tool Button Widget*** che riceve in input tutte le informazioni necessarie alla creazione dello stesso, tra cui Label, Icon, Font e tutti i colori definiti nell'EUW spiegato nel capitolo **2.2.1**, e

restituisce il Widget correttamente creato che poi viene aggiunto al ***Tool Button Array*** della *General_UI*.

Il suddetto Widget viene restituito come output della funzione, in modo da costruire il Bind all'*Event Dispatcher* ***On Tool Clicked***, chiamato al suo interno dall'evento *OnClicked*. In questo modo si da il via alla catena di *Event Dispatcher* che permettono di collegare l'*OnClicked* di un *BP Tool Button* al rispettivo evento della *DP_Interface* che ne regola il funzionamento.

Nello specifico il Bind a *On Tool Clicked* chiama a sua volta l'*Event Dispatcher* ***Bind Tool Logic*** dell'*HUD_UI_Logic*, il cui Bind nella *DP_Interface* viene posto immediatamente dopo la chiamata all'evento *I_CreateToolsButtons*. In questo modo si "chiude il cerchio" di creazione dei Widget dei Tool, collegandoli alla *DP_Interface* per potervi associare la corretta reazione all'evento *OnClicked* su di essi.

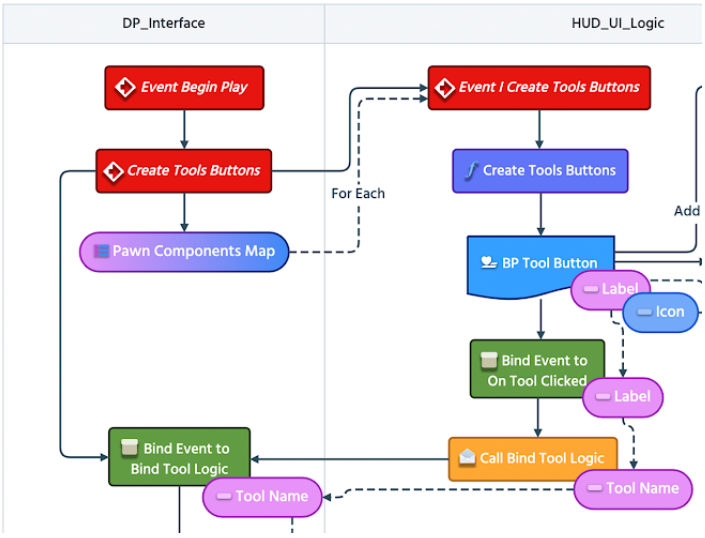


FIGURA 3.24: SCHEMA DI CREAZIONE DEI TOOL BUTTONS

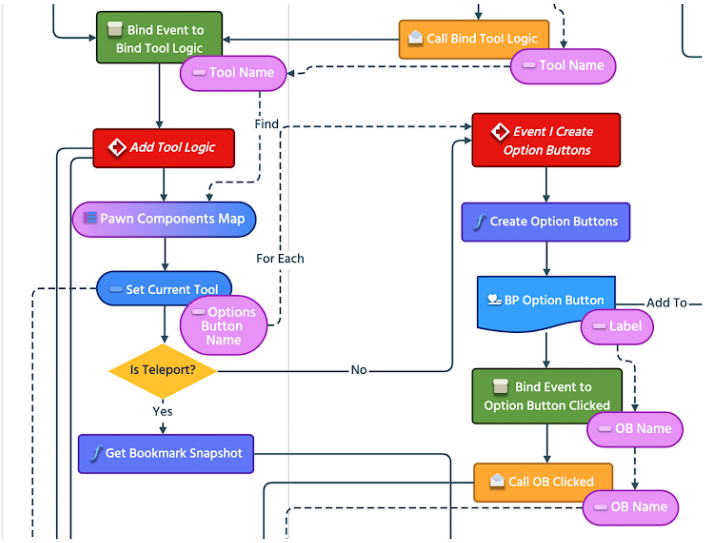


FIGURA 3.25: SCHEMA DI CREAZIONE DELLE OPTIONS

Una volta che l'utente clicca su un Tool è necessario recuperare e mostrare le corrette ***Options*** appartenenti ad esso. Per fare questo, *Bind Tool Logic*, nella *DP_Interface*, chiama un evento denominato ***Add Tool Logic*** a cui viene passata una variabile con il nome del Tool cliccato che viene utilizzata per recuperare dalla *PawnComponentsMap* il *BP_BaseCommand* corretto.

Il contenuto di questa variabile poi viene scritto in ***Current Tool***, da cui viene recuperato un Array di variabili *Text*, chiamato ***Options Button Name***, con l'elenco di tutte le Options associate al Tool scelto.

A questo punto si è inserito un nodo Branch per verificare se il Tool in questione fosse di tipo *Teleport*, poiché quest'ultimo è stato trattato in maniera leggermente diversa dagli

altri e per questo necessita di una specifica trattazione che rimandiamo a fine capitolo.
Se quindi tale nodo in output restituisce *False*, per ogni voce dell'*Options Button Name* viene chiamato un evento della *HUD_UI_Logic* denominato ***I_CreateOptionButtons***.
Quest'ultimo viene quindi incaricato di invocare ***Create Option Buttons***, l'omonima funzione adibita alla creazione di tutti i Widget delle Options coerentemente al Tool scelto.
Similmente a quanto descritto per la funzione *Create Tools Buttons*, il nodo principale della funzione *Create Option Buttons* è ***Create BP Option Button Widget*** che si occupa della creazione del corretto Widget per ogni Options con le informazioni passate in input, provenienti sia dall'EUW che dalla *DP_Interface*. In output a questo nodo viene restituito il Widget creato che viene poi aggiunto in un apposito Array della *General_UI*, denominato ***Option Button Array.***

Al widget restituito dalla funzione *Create Option Buttons* viene collegato il Bind ad un *Event Dispatcher* presente dentro il *BP Option Button*, chiamato ***Option Button Clicked,*** che registra il click dell'utente su tale widget e restituisce il nome della Options scelta. Questo notifica a sua volta l'evento di click anche alla *DP_Interface* chiamando un ulteriore *Event Dispatcher* ***OB_Clicked***.

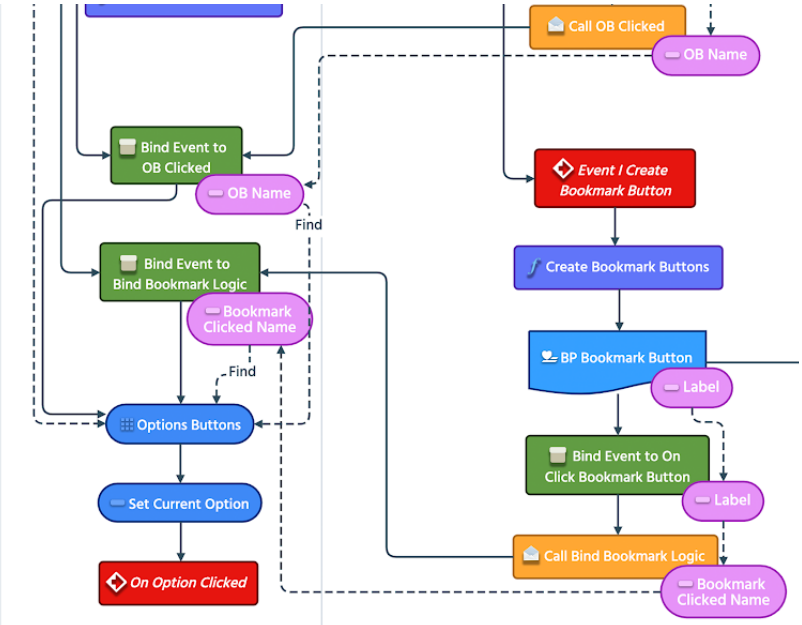


FIGURA 3.26: ATTIVAZIONE EVENTO ON OPTION CLICKED

Il Bind correlato nella *DP_Interface* con il nome dell'*Option* scelta recupera il corretto ***Option Button*** da un *Array* ***Options Buttons*** presente nel *Current Tool* settato in precedenza. Fatto ciò, tale *Option Button* viene settato come ***Current Option*** nella apposita variabile.

Quanto appena descritto, a partire dalla chiamata all'evento *Create Tools Buttons*, serve a definire la logica di costruzione dei widget dei Tool e delle relative Option. A questo punto, a seconda delle scelte compiute dall'utente, va attivato il Tool con l'Option corretta in modo che l'utente possa usarlo liberamente all'interno dell'applicazione.

Per fare ciò, in coda a quanto precedentemente descritto, viene chiamato l'evento ***On Option Clicked*** all'interno della *DP_Interface*, che si occupa, in base al *Current Tool* e *Current Option* settati in precedenza, di richiamare il corretto funzionamento.
Data la struttura chiusa del Collab Viewer, l'evento *On Option Clicked* si occupa di richiamare correttamente tutti gli eventi descritti in esso, ricostruendo di fatto tutto l'iter descritto nel template in seguito al click su un Tool, come se l'utente avesse interagito effettivamente con le UI proprie del Collab Viewer, e completando quindi la comunicazione tra questo e la *General_UI*.

Riprendendo il discorso interrotto in precedenza sul diverso trattamento delle Option del Tool ***Teleport,*** andremo ora ad esporre quanto implementato per tale Tool**.**
La scelta che ha portato ad utilizzare un diverso widget per le Option di questo Tool è data dal differente funzionamento di questo strumento. Dal momento che esso permette all'utente di "tele-trasportarsi" in punti definiti dell'ambiente è importante dare da subito un feedback chiaro sul luogo in cui si sta per spostarsi, informazione che con solamente il nome dell'Option potrebbe non essere sufficiente chiara.
Sostanzialmente la logica di creazione delle Option di tale Tool è identica a quanto descritto fino ad ora, ma impone l'utilizzo di una funzione differente a *Create Option Buttons* per il fatto che in questo caso viene impiegato un diverso widget.
Per evitare incomprensioni, ci preme sottolineare che tale Tool è chiamato *Bookmark* nel Collab Viewer, dunque anche noi in alcuni passaggi abbiamo mantenuto questo nome, seppur all'utente venga mostrato come *Teleport.*

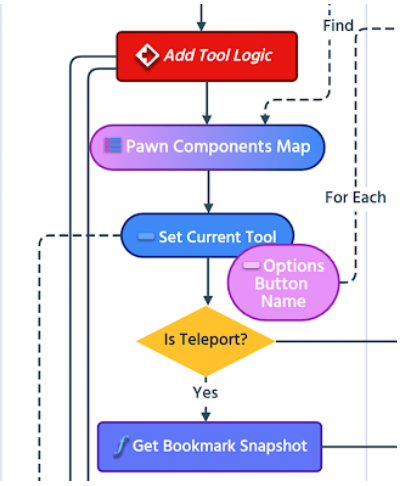


FIGURA 3.27: SCELTA DEL TOOL TELEPORT

Nel caso venga selezionato *Teleport* come Tool, viene comunque chiamato l'evento *Add Tool Logic* nella *DP_Interface*, che recupera tutte le Option dal *Current Tool,* che in questo caso risulta essere *Teleport, ma,* a differenza di quanto descritto in precedenza, per ognuna delle Option viene chiamata la funzione ***Get Bookmark Snapshot*** che permette di recuperare uno ***Snapshot*** che rifletta la posizione di questi punti di teletrasporto nell'ambiente. Per fare ciò, tale funzione recupera dal Pawn l'Array contenente i ***BP_-Bookmark,*** anch'essi Actor Component, ognuno dei quali contiene una Camera. Viene dunque fatto lo Spawn di un

Scene Capture nella posizione di ognuna di queste camere, il quale cattura un Render di quella inquadratura in quell'istante. Questo Render viene poi restituito come output della funzione *Get Bookmark Snapshot* e passato, insieme al nome della corrente Option, all'evento ***I_CreateBookmarkButton*** della *HUD_UI_Logic*.

Identicamente a quanto già descritto per gli altri Tool, l'evento chiama una omonima funzione che è adibita a creare correttamente i widget necessari, aggiungerli ad un Array della *General_UI* chiamato ***Bookmark Array*** e restituirli in modo che nell'*HUD_UI_Logic*, si possa fare il Bind dell'*Event Dispatcher* ***On Click Bookmark Button*** presente all'interno del widget. Similmente a quanto accade per gli altri Tool, verrà chiamato a sua volta un altro *Event Dispatcher*, in questo caso ***Bind Bookmark Logic***, il cui Bind verrà eseguito nella *DP_Interface* appena dopo la chiamata di *I_CreateBookmarkButton*.



FIGURA 3.28: BP BOOKMARK BUTTON

Dopo tutti questi passaggi, ormai noti al lettore, viene settata la *Current Option* e chiamato l'evento *On Option Clicked* descritto in precedenza.

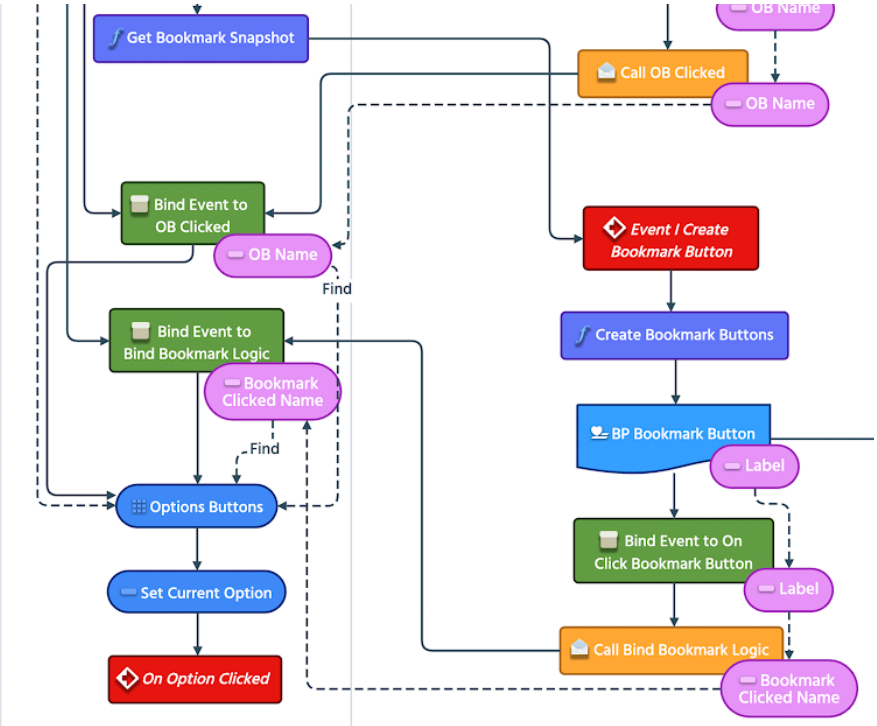


FIGURA 3.29: CREAZIONE DELLE OPTION DEL TOOL TELEPORT

Identicamente a quanto già descritto per gli altri Tool, l'evento chiama una omonima funzione che è adibita a creare correttamente i widget necessari, aggiungerli ad un Array della *General_UI* chiamato ***Bookmark Array*** e restituirli in modo che nell'*HUD_UI_Logic*, si possa fare il Bind dell'*Event Dispatcher* ***On Click Bookmark Button*** presente all'interno del widget. Similmente a quanto accade per gli altri Tool, verrà chiamato a sua volta un altro *Event Dispatcher,* in questo caso ***Bind Bookmark Logic***, il cui Bind verrà eseguito nella *DP_Interface* appena dopo la chiamata di *I_CreateBookmarkButton.*
Dopo tutti questi passaggi, ormai noti al lettore, viene settata la *Current Option* e chiamato l'evento *On Option Clicked* descritto in precedenza.

Una volta che l'utente termina l'utilizzo di un Tool e decide quindi di sceglierne giustamente un'altro o di utilizzare liberamente il Configuratore, è di fondamentale importanza che l'applicativo disattivi correttamente il Tool attivo. Per fare ciò, abbiamo dichiarato nella *UI_Interface* la funzione ***I_RefreshOptionButtons***, la cui implementazione nella *HUD_UI_Logic* si compone della sola chiamata all'*Event Dispatcher* ***Refresh Option Buttons.***
La *DP_Interface,* alla chiamata dell'*Event Begin Play*, fa un Bind a questo *Event Dispatcher* legando ad esso un evento ***Refresh*** che si occupa di richiamare tutti gli eventi descritti nel Collab Viewer necessari ad ottenere l'effetto desiderato.

3.3.2 - POPOLAMENTO DEL MENU

Nel sottocapitolo precedente si è visto come tre funzioni della *HUD_UI_Logic*, ovvero *Create Tools Buttons, Create Option Buttons* e *Create Bookmark Button,* dopo aver creato i corrispettivi widget d'interesse, provvedessero anche ad inserirli in tre Array della *General_UI*, rispettivamente: ***Tool Button Array, Option Button Array*** e ***Bookmark Array***.
Questi Array fungono da punto di partenza per la creazione del *Tools Menu*, in modo che la *General_UI* possa completamente ignorare la logica di creazione dei vari Widget e occuparsi semplicemente di renderli visibili, così come sono stati creati, al momento opportuno. Abbiamo visto come l'evento *Create Tools Buttons* sia legato all'*Event Begin Play* della *DP_Interface*, dunque *Tool Button Array* avrà al suo interno gli opportuni Widget già all'aggiunta della *General_UI* alla Viewport.

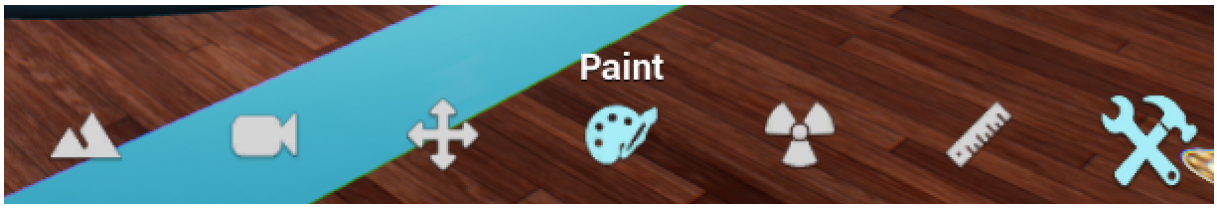


FIGURA 3.30: MENU TOOL APERTO CON LA TOOLS BAR VISIBILE E CON CURSORE SU PAINT

La corretta visualizzazione del *Tools Menu* nella *General_UI* è gestita da due *Horizontal Box*: ***Tools Bar*** che dovrà contenere i *BP_ToolButton* creati e la ***Options Bar*** che invece conterrà i *BP_OptionButton* del Tool scelto. Non potendo essere contemporaneamente visibili su schermo, è stato utilizzato un *WidgetSwitcher* che alterna la visibilità dei due *Horizontal Box* a seconda del valore posseduto dal suo indice, che viene opportunamente aggiornato nei vari eventi che regolano questo menu.

L'apertura del *Tools Menu* avviene tramite click sul ***Tool_Button*** presente nella struttura di base della *General_UI*. Viene attivato così l'evento *On Clicked (Tool Button)*, che recupera tutti i Widget presenti nel *Tool Button Array* e li aggiunge alla *Tools Bar* per poi renderla visibile nella Viewport.

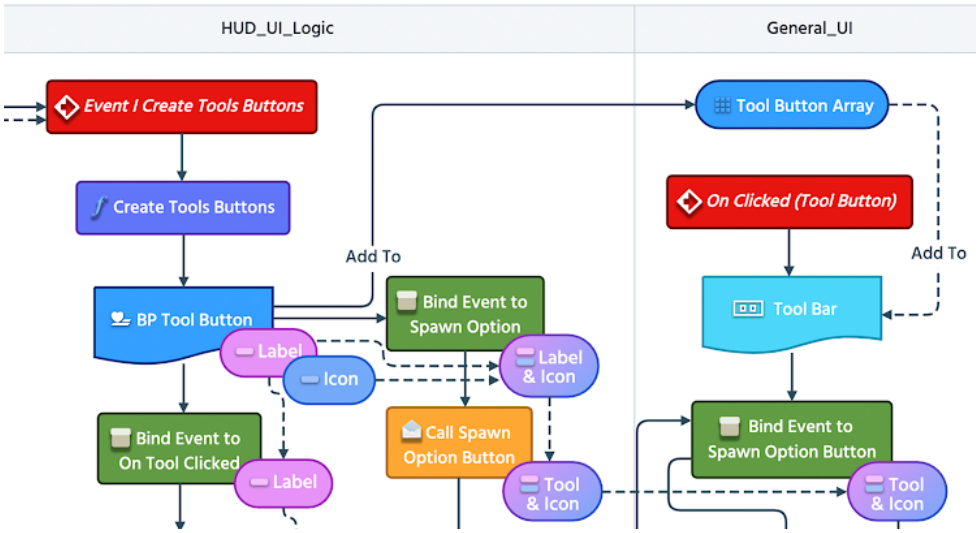


FIGURA 3.31: SCHEMA CREAZIONE TOOL BUTTON ARRAY

A questo punto l'utente avrà piena libertà di scegliere il Tool che intende usare cliccando sul Widget che lo rappresenta. Il click su un *BP_ToolButton* attiva la catena di *Event Dispatcher* descritta nel sottocapitolo precedente, che porta alla creazione dei corretti *BP_OptionButton* all'interno del *Option Button Array*. Nella *HUD_UI_Logic* in questo caso viene fatto un Bind ad un altro *Event Dispatcher* presente all'interno del *BP_OptionButton*, denominato ***Spawn Option.*** A sua volta verrà chiamato da quest'ultimo un secondo *Event Dispatcher* della *HU-*

D_UI_Logic, ***Spawn Option Button***, posto in coda all'evento *On Clicked (Tool Button)* della *General_UI* appena descritto.
Vengono così inviate alla *General_UI* sia la Label che l'Icon del Tool scelto, quest'ultima utile per modificare l'icona del Menu Tool, per dare all'utente un feedback, sempre visibile su schermo, in merito a quale Tool ha scelto.

Legato al Bind di ***Spawn Option Button*** è presente un evento che recupera tutti i widget presenti nel *Option Button Array* e aggiunti all'*Options Bar.* Prima di rendere visibile quest'ultima, nascondendo di conseguenza la *Tools Bar,* viene aggiunto in coda un Widget ***Back Button.*** Questo Widget, oltre a dare la possibilità di tornare a visualizzare l'elenco dei Tool a disposizione, richiama l'evento ***I_RefreshOptionButtons*** della *HUD_UI_Logic*, permettendo all'utente di tornare alla condizione iniziale del menu, senza alcun Tool attivo.

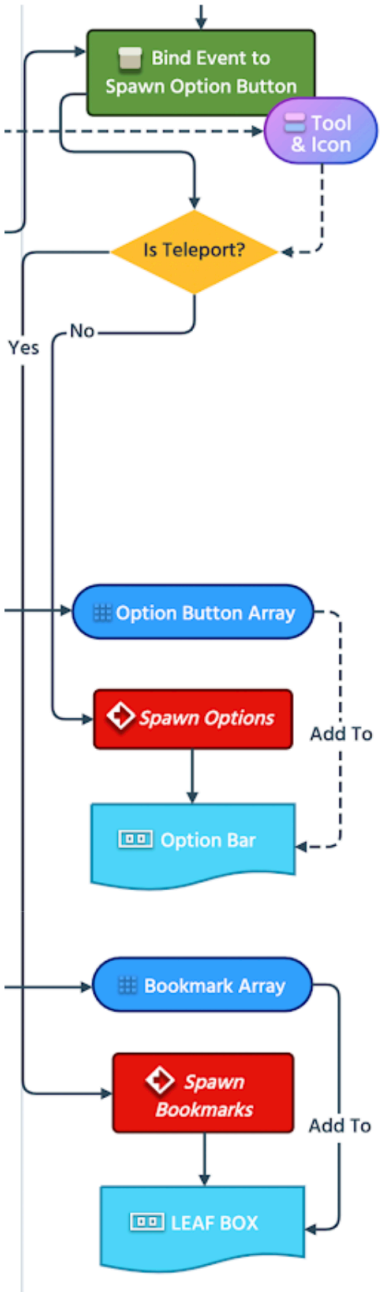


FIGURA 3.32: SCHEMA CREAZIONE OPTION BUTTON ARRAY E BOOKMARK ARRAY

Se il Tool scelto è di tipo *Teleport* il discorso cambia leggermente. In questo caso, arrivati al Bind di *Spawn Option Button*, non vengono presi i Widget del *Option Button Array*, che risulterebbe ovviamente vuoto, ma quelli del *Bookmark Array***.** Sempre per dare una migliore visibilità a queste differenti Option, questa volta non viene riempito l'*Options Bar*, ma viene invece utilizzato lo stesso *Horizontal Box* utilizzato per le varianti del Menu Configurator, il **LEAFS BOX.**
Poiché non esiste condizione per cui debbano essere visualizzate in contemporanea le varianti e le opzioni del *Teleport*, abbiamo fatto questa scelta per perseguire una coerenza di struttura e di estetica, permettendo all'utente di ritrovare con facilità, negli spazi a lui oramai familiari, tutti gli elementi messi a disposizione dalla UI.
Anche in questo caso, per ritornare alla visualizzazione di tutti i Tool disponibili, in coda a tutte le Option, viene aggiunto un Widget ***Back Button***.



FIGURA 3.33: OPTIONS DEL TOOL PAINT POSTE NELLA OPTIONS BAR RESA VISIBILE A SEGUITO DELLA SCELTA DI QUEST'ULTIMO TOOLS

3.3.3 - TOOL ENVIRONMENT

Il Tool ***Environment***, a differenza di tutti gli altri, non viene ripreso dal Collab Viewer, ma è interamente gestito all'interno della nostra UI, quindi dal widget *General_UI*. Ciò comporta che per questo Tool non vi sia alcuna necessità di utilizzo della *DP_Interface* o della *HUD_UI_Logic*.
Si è deciso di realizzare questo Tool, prendendo ispirazione da altri progetti, poiché può essere interessante, per un utente che vuole configurare il suo autoveicolo, poterlo osservare in ambienti e soprattutto sotto luci differenti.

Per realizzarlo abbiamo sfruttato lo strumento di ***Level Streaming*** messo a disposizione da Unreal Engine. In generale in un qualunque videogame, il mondo e ogni oggetto interagibile risiede in ciò che è comunemente noto come Livello. In Unreal Engine 4 un livello è generalmente composto da una parte chiamata ***Persistent Level,*** sempre esistente, ed eventualmente da ulteriori sottolivelli che vengono chiamati mappe. Il *Level Streaming* permette, definito un livello di base per il mondo di gioco, di gestire differentemente varie parti dello stesso, caricando o scaricando dalla memoria le mappe che lo compongono e di attivarne la visibilità durante il gioco.
Noi abbiamo usato questo sistema per dare la possibilità all'utente di cambiare momento del giorno, in modo che possa decidere di osservare il veicolo che sta configurando sotto le luci dell'alba, al tramonto, in pieno giorno o in piena notte.

Il funzionamento del Tool *Environment* è basato su un Array ***Levels*** di *World* posto all'interno della *General_UI*. Alla creazione dell'interfaccia viene creato un widget con Label e Icon del Tool in questione e viene aggiunto, insieme agli altri Tool, alla *Tool Bar* quando viene cliccata l'icona del Tools Menu.

Le Options di questo Tool vengono create a seguito del click sul widget che lo rappresenta, prendendo i livelli presenti nell'Array *Levels* e recuperando nome e icona da una Map appositamente creata, denominata ***Env Image Map***, che servono per creare un Widget ***BP_EnvOptionButton*** per ogni Option presente.

Trovandoci già nella *General_UI*, in questo caso non risulta necessaria la costruzione di un Array di Option, come per gli altri Tool, ma i Widget creati vengono direttamente aggiunti al *LEAFS BOX*, come accaduto per le Option del *Teleport*.

In coda a tutte le Option viene caricato un ***Back Button***, avente la stessa funzione già spiegata per gli altri Tool.

Al click su una delle Option, viene attivato, sempre tramite *Event Dispatcher* e relativo Bind, l'evento ***Load Env On Clicked***, che si occupa di scaricare il livello visibile in quel momento e caricare quello appena scelto dall'utente. Il nome del livello attivo è salvato in una variabile Name ***Last Level Loaded***, per cui in primis viene scaricato questo livello, tramite il nodo ***Unload Stream Level (by Name)***, passando in input la suddetta variabile, per poi caricare il nuovo livello con ***Load Stream Level (by Name)***, passandogli il nome della Option scelta, che a sua volta verrà settato anche nella variabile *Last Level Loaded*.

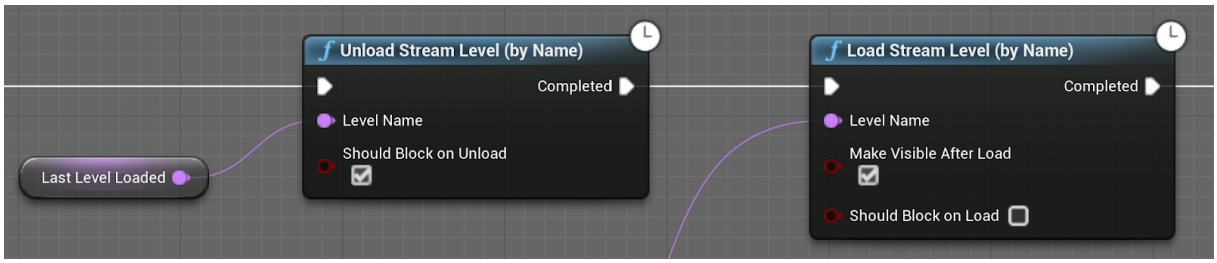


FIGURA 3.34: NODI ADIBITI AL CARICAMENTO E SCARICAMENTO DEI LIVELLI

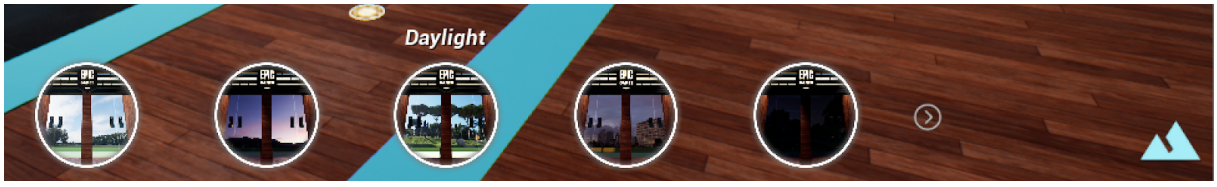


FIGURA 3.35: OPTION DEL TOOL ENVIRONMENT

3.4 - MENU SETTINGS

Il Menu **Settings** con la sua logica di creazione e funzionamento non si discosta da quanto già spiegato per i Tool.
Al momento, tra le funzionalità attualmente presenti, solo una è stata implementata, poiché indispensabile per il passaggio da UI Desktop a UI VR, ed è lo strumento che permette il cambio di modalità di navigazione. Essendo tale funzionalità già implementata nel Collab Viewer, anche in questo caso abbiamo utilizzato la *DP_Interface* e l'*HUD_UI_Logic* per collegare la *General_UI* con il rispettivo funzionamento descritto nel Template.

Nella UI del Collab Viewer sono presenti dei Widget Button in cui è stata costruita la logica del cambio di navigazione, che avviene attraverso un cambio del *Player Pawn* attivo con un differente Pawn avente una differente logica di interazione e movimento.
Per riportare questo funzionamento nella nostra UI, abbiamo collegato all'*Event Begin Play* della *DP_Interface*, insieme agli eventi relativi alla creazione dei Tool, l'evento ***Create Nav Button*** che si occupa di gestire tutta la logica di costruzione dei Widget per il cambio di navigazione.

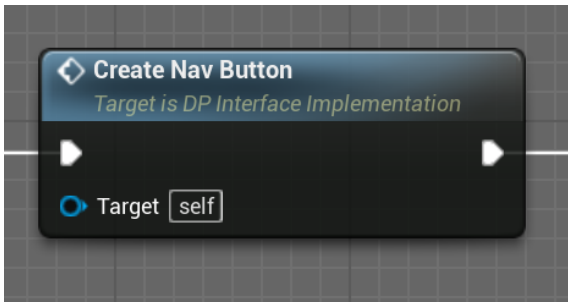


FIGURA 3.36: FUNZIONE CREATE NAV BUTTON

Questo evento si occupa di richiamare dalla *Desktop_UI*, interfaccia utente creata dal Collab Viewer, tutti i ***Navigation Button*** presenti in essa e di aggiungerli in una *Map*, chiamata ***Nav Map,*** catalogandoli per nome in modo da poterli recuperare facilmente in seguito.
Creata la *Map*, per ogni sua occorrenza viene chiamato l'evento ***I_CreateNavModeButton*** dell'*HUD_UI_Logic*. Questo evento ha il compito di richiamare la funzione ***Create Nav Button,*** sempre all'interno dell'*HUD_UI_Logic*, che è adibita alla creazione di tutti i Widget per le opzioni di cambio di navigazione, da aggiungere alla nostra interfaccia utente. Essa è composta principalmente dal nodo ***Create BP Nav Details Widget,*** che crea il Widget a partire sia dalle informazioni provenienti sia dalla *DP_Interface* che dal EUW. Una volta creato, il *BP Nav Details* viene aggiunto ad un apposito Array della *General_UI*, ***Nav Detail Array***, e poi restituito come output della funzione *Create Nav Button*.

Esattamente come per la logica di funzionamento dei Tool, è necessario che l'evento *OnClicked* del *BP Nav Details* richiami il suo corretto funzionamento descritto nella *DP_Interface*, e questo è possibile attraverso l'ormai nota catena di *Event Dispatcher.*
A seguito della funzione *Create Nav Button* viene fatto un Bind all'*Event Dispatcher* ***Call Nav Mode*** presente all'interno di *BP Nav Details* e chiamato dal suo evento *OnClicked.* Il Bind a *Call Nav Mode* a sua volta invoca un nuovo *Event Dispatcher* dell'*HUD_UI_Logic*, ***Bind Nav Logic***, con relativo Bind nella *DP_Interface*, restituendo la Label con il nome del tipo di navigazione scelto.
Grazie a questa Label è dunque possibile recuperare il corretto *Navigation Button* dalla *Nav Map* precedentemente creata e dunque chiamare l'evento ***Change Pawn*** ad esso associato.

Quanto descritto è sufficiente a cambiare *Player Pawn*, ma poiché sia i tipi di navigazione che i Tool sono strettamente collegati ad esso, dopo la chiamata dell'evento *Change Pawn* viene fatta una verifica sul tipo di Pawn attivo. Se questo non è *BP_VRPawn*, caso particolare che verrà opportunamente trattato nel successivo capitolo, vengono semplicemente richiamati in successione tutti gli eventi di creazione dei Widget di Tool e navigazione, preceduti dall'evento *Is Not In VR*, utile nel caso venga fatto un passaggio da navigazione VR a Desktop, per ristabilire alcune condizioni essenziali alla corretta transizione tra interfacce differenti.

A questo punto nella *General_UI*, all'interno di *Nav Detail Array*, sono presenti tutti i Widget *BP Nav Details.* Essi vengono recuperati a seguito dell'evento *OnClicked* su ***Nav Button,*** elemento già creato nella struttura della *General_UI* all'interno del *Menu Settings*, e vengono aggiunti a ***Settings Details,*** un Horizontal Box posto anch'esso nel *Menu Settings.* Viene inoltre posto in esso anche un ***BackButton***, utile per tornare al *Menu Settings.*
Una volta che il *Settings Details* è stato correttamente popolato, viene nascosto l'Horizontal Box contenente il *Menu Settings* al contrario di quest'ultimo che invece viene reso visibile per dare modo all'utente di potervi interagire e scegliere quindi la modalità di navigazione che più lo aggrada.



FIGURA 3.37: MENU SETTINGS E VARI DETAILS DEI NAVIGATION

Oltre all'appena citata sezione Navigation, è in corso di sviluppo una sezione **LOD** (Level of Detail) per cambiare la risoluzione delle textures in modo da dare la possibilità all'utente sia di abbassarne il livello di dettaglio e ottimizzare quindi le performance dell'applicativo, che invece di aumentarne la qualità restituendo un'esperienza più fotorealistica a discapito delle prestazioni. Vi saranno poi una sezione **Volume** dedicata alle impostazioni audio dell'applicazione e una **FOV** (Field of View) per cambiare la distanza focale della camera associata al Pawn corrente, consentendo di zoomare su alcune parti del prodotto normalmente molto piccole e quindi difficilmente ispezionabili.

3.5 - UTILIZZO DELLE RISORSE SALVATE SU DATA ASSET

Nel sottocapitolo **2.3** abbiamo illustrato come, attraverso l'*Editor UI,* si possano scrivere in un apposito Data Asset, tutte le informazioni utili a configurare a piacimento l'interfaccia utente.

Dunque questi dati, se correttamente inseriti, sono a disposizione durante l'esecuzione runtime dell'applicazione. Poiché l'*HUD_UI_Logic* gestisce tutta la logica di creazione delle UI, dei Menu e dei relativi Widget, fatta eccezione solo per il Menu Configurator, ci è sembrato opportuno assegnare a questa classe il compito di recuperare tutti i dati presenti nel Data Asset e assegnarli ad apposite variabili stanziate dentro di essa, esattamente come fatto per l'*Editor UI* e come mostrato in figura 2.18 del sottocapitolo **2.3.2**.

Tutti i Widget dei Menu Tools e Settings, come ampiamente illustrato in questo capitolo, vengono creati all'interno dell'*HUD_UI_Logic,* dunque a questi vengono semplicemente passate le corrette variabili al momento opportuno, come mostrato in figura 3.38.

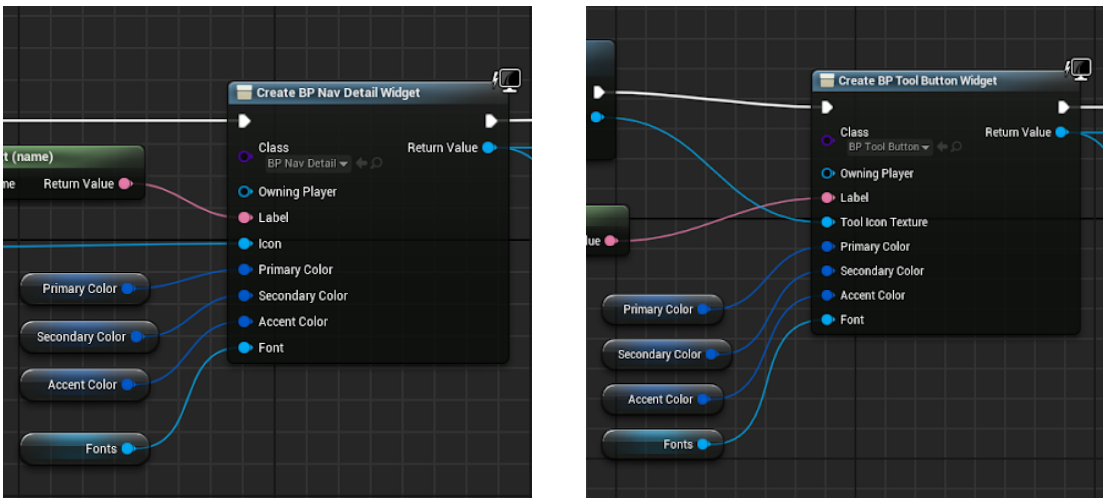


FIGURA 3.38: NODI DI CREAZIONE DEI NAV DETAIL E TOOL BUTTON

Un'altra ragione per cui il recupero di tutti i dati da Data Asset nell'*HUD_UI_Logic* ci è sembrata la scelta più logica ed efficiente è data dalla semplicità con cui si può creare una reference a tale classe ovunque sia necessario, essendo stata impostata come HUD del *Player Controller,* come spiegato nel capitolo **2.5** e mostrato in figura 2.46 di tale capitolo.

Basta solamente aggiungere una referenza alla *HUD_UI_Logic* e utilizzarla per recuperare le opportune variabili che contengono i dati necessari.

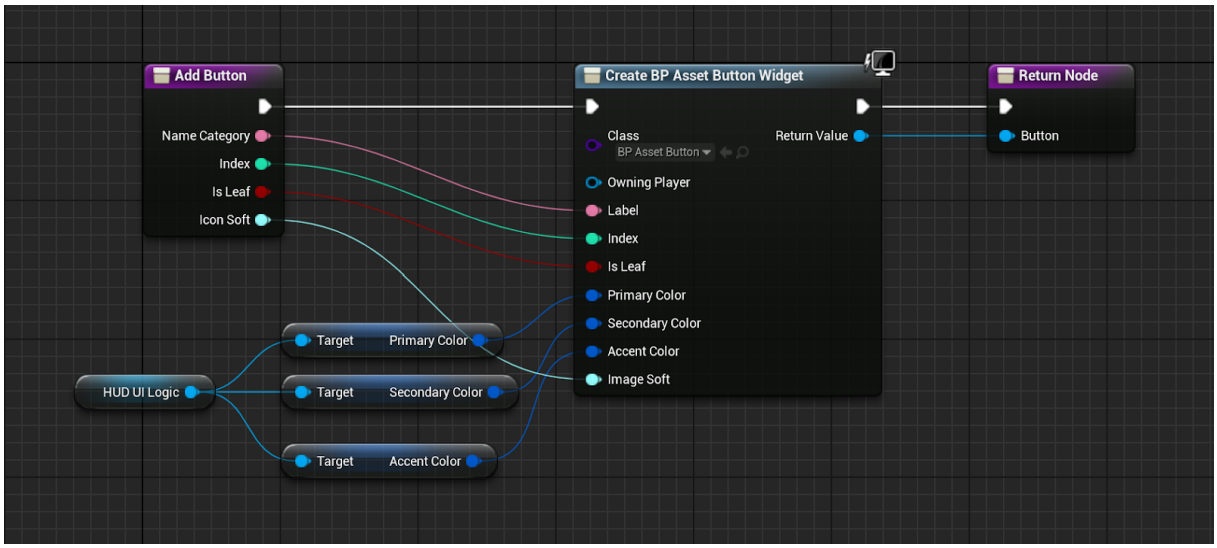


FIGURA 3.39: NODI DI CREAZIONE DEGLI ASSET BUTTON

Vorremmo dunque ora soffermarci brevemente su come vengono utilizzati i dati stanziati nel Data Asset, non tanto lato software trattandosi principalmente di variabili opportunamente assegnate all'interno dei vari Widget, ma più che altro sul loro utilizzo nel dare corretti feedback di funzionamento all'utente.
Come si nota dalle figure precedenti, alla creazione di un Widget vengono richieste sostanzialmente sempre le stesse informazioni: *Label, Icon, Primary Color, Secondary Color, Accent Color* e *Font*.

La ***Label***, come più volte ripetuto nei capitoli precedenti, è indispensabile come identificativo di un determinato Widget per poterne recuperare il corretto funzionamento, mandando questo dato nella catena di *Event Dispatcher.* Risulta però indispensabile anche a livello grafico, poiché ogni Widget, sull'evento *OnHovered* ad esso associato, mostra una etichetta con il contenuto della variabile in questione. Ciò permette all'utente, con un semplice passaggio del cursore del mouse, di comprendere meglio la funzione di un determinato Widget ove l'icona non fosse per lui sufficientemente esplicativa.
Per quanto riguarda ***Icon e Font*** non vi è necessità di aggiungere ulteriori spiegazioni al fatto che rispettivamente rappresentano uno la figura che viene costantemente mostrata per una corretta distinzione e identificazione di una voce nei vari Menu, mentre l'altro il carattere assegnato al testo che viene mostrato nell'etichetta di Widget.



FIGURA 3.40: TOOLS MENU APERTO, CON RELATIVA ICONA EVIDENZIATA IN ACCENT COLOR

Vorremmo invece porre una particolare attenzione alle tre variabili di tipo *Linear Color* e al loro utilizzo, fondamentali in diverse occasioni per dare corretti e costanti feedback all'utente in merito allo stato in cui si trova l'applicazione e i relativi Menu.
Il ***Primary Color*** è il colore di base di tutte le icone, quello relativo allo "stato di quiete" di ogni Widget nel momento in cui non è soggetto a nessuna interazione. Il ***Secondary Color*** invece è il colore mostrato sull'effettiva interazione, dunque all'evento *OnClicked* quando un Widget è stato cliccato ma il pulsante del mouse non è ancora stato rilasciato.
A dare maggiori informazioni è sicuramente l'***Accent Color***, che ha il compito di attirare l'attenzione o evidenziare le informazioni più rilevanti in quel momento. Viene dunque usato in diverse occasioni, prima fra queste sull'evento *OnHovered* di ogni Widget, dove l'icona viene colorata appunto con il colore in questione. Inoltre viene utilizzato ovunque vi sia necessità di sottolineare lo stato in cui si trova l'applicazione in un determinato istante. Ad esempio assume il colore d'accento l'icona di un Menu quando questo viene aperto, oppure quando viene scelto un Tool e l'icona del Menu Tools viene sostituita con quella del Tool attivo che rimane del colore d'accento, ma anche in seguito alla scelta di una Option di un Tool, in cui la corrispondente icona rimane del colore d'accento fino a che l'Option scelta risulta essere attiva e utilizzabile.



FIGURA 3.41: OPTION DEL TELEPORT, CON L'OPZIONE GROUND FLOOR CHE REAGISCE AL CURSORE, MOSTRANDO LABEL E ACCENT COLOR. L'ICONA DEL TOOLS MENU È STATA SOSTITUITA DA QUELLA DEL TOOL TELEPORT.

4 - INTERFACCIA UTENTE PER REALTÀ VIRTUALE

4.1 - FASE DI IDEAZIONE

Terminata l'implementazione dell'Interfaccia Utente per Desktop abbiamo intrapreso, come fatto in precedenza, una fase preliminare di ricerca di quanto ad oggi venisse utilizzato, e fosse effettivamente funzionante, in merito di UI per Realtà Virtuale.
Se da un lato per il design di applicazioni web/pc e mobile gli standard di progettazione risultano ben consolidati, a seguito di miliardi di test svolti, per quanto riguarda la VR, come più volte ripetuto nel capitolo **1**, vi è una quasi totale assenza di standard predefiniti e modelli di creazione, senza contare che quelli ora presenti sono ancora in costante cambiamento. Abbiamo infatti immediatamente riscontrato, già dai primi utilizzi di applicazioni in VR, che vi fossero delle costanti in tutte le interfacce, ma anche come venissero utilizzati approcci a volte diametralmente opposti per adempiere agli stessi obiettivi di utilizzo.
In questo caso non ci siamo limitati ad analizzare solamente interfacce utente di configuratori come era stato fatto per la versione Desktop, non esistendone molti ad oggi, ma abbiamo deciso di studiare in generale tutte quelle applicazioni che mettevano a disposizione dell'utente degli strumenti da utilizzare liberamente nell'ambiente tridimensionale e che possedevano dei menu che fossero in grado di gestire una moltitudine non indifferente di contenuti. Questo per capire come implementare al meglio sia il nostro Menu Tools, in modo che potesse fornire degli strumenti d'interazione adeguati, che il Menu Configurator, e il pressoché identico Menu Settings, adibito principalmente alla visualizzazioni di dati e informazioni riguardo il configuratore in sé.

4.1.1 - REFERENCE E DIFFERENTI APPROCCI

Abbiamo avuto modo di analizzare, sia personalmente con l'Oculus Rift che attraverso articoli[40], i menu di queste applicazioni: **Mindshow, Tvori, EXA: The Infinite Instrument, Big Screen, Tilt Brush, Gravity Sketch, Microsoft Maquette, Google Blocks, Sketchbox, Oculus Quill, Noda** e **Oculus Menu.**

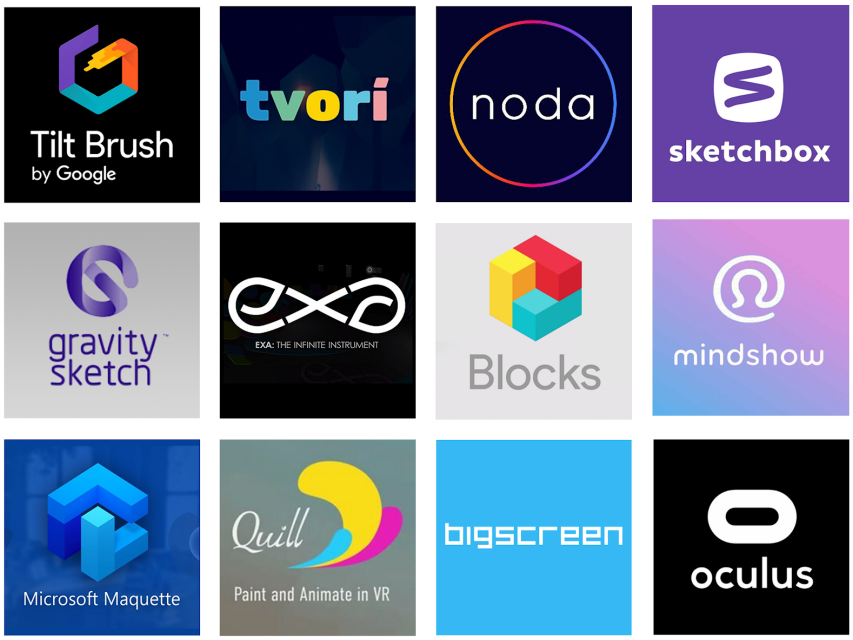


FIGURA 4.1: APP DI CUI ABBIAMO ANALIZZATO LE INTERFACCE UTENTE.

Una costante che sembra essere trasversale a tutte le applicazioni analizzate è la tendenza a riproporre menu con logica bidimensionale adattati però all'ambiente tridimensionale. Viene quasi sempre proposto o un pannello con varie schermate oppure una modalità di scorrimento delle pagine disponibili spesso utilizzando effetti 3D per evidenziare l'elemento selezionato. Questa soluzione ha il vantaggio di essere già familiare all'utente, poiché non richiede interazioni o conoscenze sulla VR che non siano già abbastanza note con la UI Desktop, ma proprio per questo pecca di originalità, nel senso che non sfrutta a pieno ciò che uno spazio tridimensionale mette a disposizione.



FIGURA 4.2: OCULUS QUILL CON UI INTERAMENTE BIDIMENSIONALE



FIGURA 4.3: TILT BRUSH E I SUOI SHORTCUT DA MENU RADIALE

Un'altra tendenza comune a molti applicativi è quella di presentare le opzioni su un menu radiale che, pur essendo ancora legato ad una logica bidimensionale, risulta più elegante, comprensibile e rapido nel suo utilizzo se collegato alla thumbstick del controller.
Un altro approccio è quello di utilizzare un menu ad albero, che vada a ramificarsi ad ogni scelta, dando il vantaggio di aver sempre ben chiaro il percorso fatto, dove ci si trova e come tornare indietro.



FIGURA 4.4: UI DI EXA CON MENU AD ALBERO

Infine abbiamo analizzato menu che si discostano quasi totalmente dalla logica Desktop, sfruttando maggiormente le potenzialità della realtà virtuale. Questi menu tendono a rappresentare qualunque strumento o set di tool come fossero oggetti reali, sia nell'aspetto che nel loro funzionamento. Ad esempio uno strumento Brush viene rappresentato come un pennello, oppure un tool di registrazione come un microfono, con il tasto di accensione posto dove si troverebbe anche nel suo corrispettivo reale. In questo modo l'interazione con i tool risulta molto più intuitiva per l'utente, poiché non viene costretto ad imparare delle nuove logiche e, allo stesso tempo, viene invogliato ad interagire con essi come avrebbe fatto al di fuori della VR.



FIGURA 4.5: MENU SCOMPONIBILI DI TVORI CON UNA SERIE DI STRUMENTI VEROSIMILI PER FORMA, FUNZIONAMENTO E INTERAZIONE

Anche la struttura stessa dei menu viene gestita in due modi diametralmente opposti. Alcune applicazioni utilizzano un unico menu il cui spazio viene aggiornato a seconda del sottomenu scelto, organizzando dunque diverse schede, magari anche composte da diverse pagine, in un unico spazio ristretto. Il secondo metodo, quello che al momento sembra essere quello maggiormente diffuso, è quello di scomporre l'intera UI in tanti moduli separabili, spesso posizionabili a piacere nello spazio circostante l'utente, che lo seguono in caso di spostamenti all'interno dello spazio virtuale, come se fossero ancorati ad esso. Se nel primo caso si ha il vantaggio di avere a disposizione un menu che utilizza delle logiche desktop familiari e che si presenta fisicamente sempre nello stesso punto, nel secondo caso invece si evitano menu ingombranti che possano diventare molto complessi all'aumentare delle funzionalità disponibili, risolvendo il problema dividendo quest'ultime in appositi piccoli moduli, dando anche la possibilità all'utente di personalizzare il proprio laboratorio virtuale.

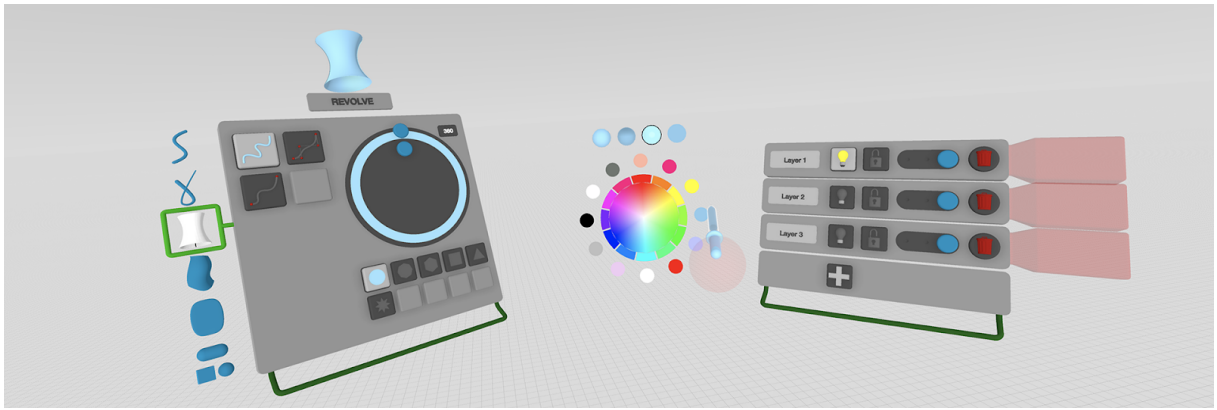


FIGURA 4.6: MENU SEPARABILI DI GRAVITY SKETCH

Anche nelle modalità d'utilizzo dei controller abbiamo riscontrato la presenza di due criteri antitetici fra loro. Se in alcuni casi si è scelto di assegnare ai due controller funzioni differenti, in modo tale che un pulsante o un gesto fatto con un controller non avesse lo stesso e identico significato se fatto con altro, in altri casi si è optato invece per rendere l'interazione completamente bilaterale. Se utilizzando controller non speculari si ha il vantaggio di avere più pulsanti e opzioni a disposizione, nell'altro caso invece si ha maggiore versatilità per l'utente che può scegliere che controller usare secondo personale comodità.
Alcune applicazioni raggiungono un buon compromesso dando la possibilità all'utente, attraverso il menu delle impostazioni, di definirsi destrorso o mancino.

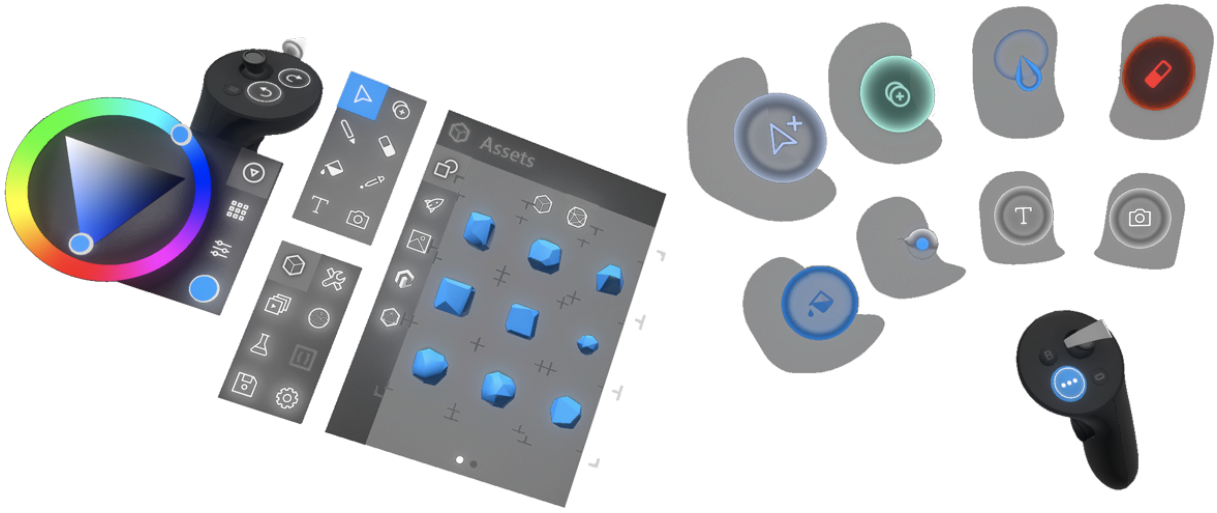


FIGURA 4.7: IN MICROSOFT MAQUETTE CONTROLLER DESTRO E SINISTRO HANNO FUNZIONI DIFFERENTI, INVERTIBILI DAL MENU SETTINGS

Anche per quanto riguarda la modalità di puntamento e selezione non sono presenti degli standard comuni a tutti gli applicativi: in alcuni casi si utilizza un laser, che parte dal controller e si estende nello spazio, in altri si utilizza una sorta di "punto attivo" posto sul controller, il quale riesce ad interagire con gli oggetti circostanti tramite contatto. Il vantaggio principale nell'utilizzare un laser è quello di poter sfruttare la possibilità di puntamento e selezione a distanza, molto comoda in determinati casi; mentre se il controller viene rappresentato attraverso delle mani virtuali il risultato è senza dubbio più realistico potendo cliccare direttamente con le dita, o qualora al posto delle mani si avesse un controller virtuale, cliccando mediante cursori e bacchette sui tasti dei vari menu. Alcune app analizzate usano sia il laser che il "punto attivo", differenziando l'utilizzo di una o dell'altra scelta a seconda dell'obiettivo, oppure dando una doppia possibilità di interazione pur avendo lo stesso fine.

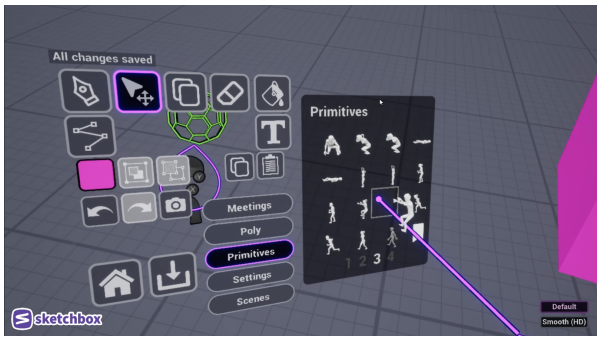


FIGURA 4.8: PUNTATORE LASER DI SKETCHBOX

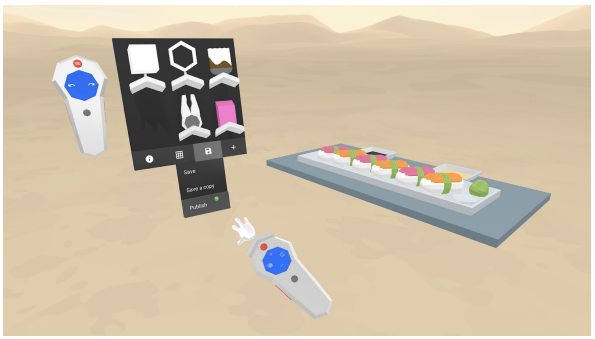


FIGURA 4.9: "PUNTO ATTIVO" A FORMA DI MINI MANO DI BLOCKS

L'ultima differenza principale risiede, come già accennato, nella scelta di rappresentazione della posizione dei controller nello spazio virtuale. C'è dunque chi predilige l'utilizzo di una mesh a forma di mano, simulandone al meglio i movimenti sfruttando i controller touch, e chi ricade su un modello di controller, avendo cura di riproporre quello effettivamente utilizzato dall'utente. La prima scelta è efficace in termini di immersione e resa realistica, ma non sempre si riesce ad ottenere questo effetto per la difficoltà di riprodurre efficacemente tutti i micromovimenti di una mano reale e in tal caso la loro presenza può risultare eccessivamente statica o addirittura fastidiosa; la seconda opzione invece è maggiormente predisposta per dare feedback sull'utilizzo dei comandi, a patto che sia correttamente aggiornata in ogni situazione, ma di rimando l'interazione perde di realisticità..

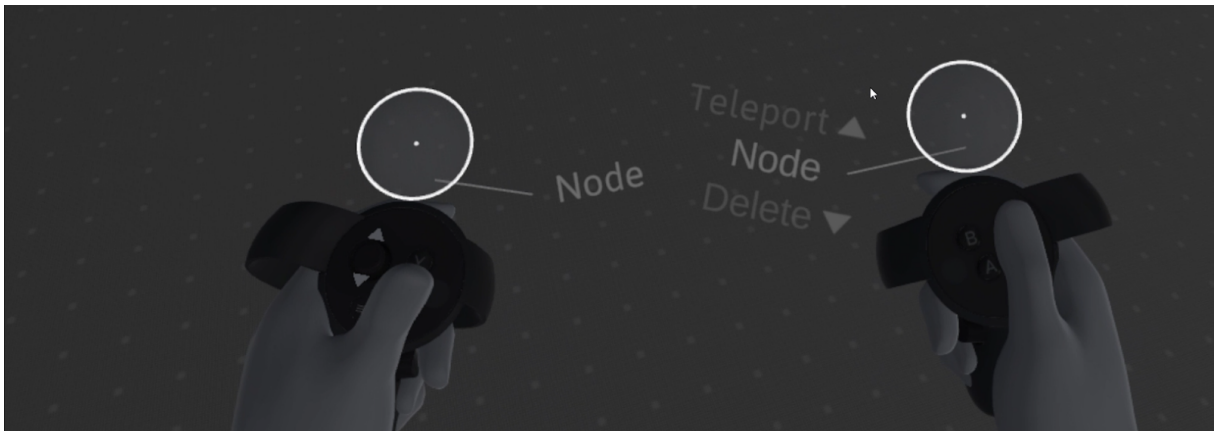


FIGURA 4.10: NODE SFRUTTA SIA LA RESA REALISTICA DELLE MANI CHE LE RAPPRESENTAZIONI DEI CONTROLLER, CON COSTANTI FEEDBACK DI UTILIZZO DEI PULSANTI

4.1.2 - BEST & WORST

Non essendo ancora in grado di definire a priori quali criteri seguire, avendo a disposizione per la VR molti meno test e casi studio che non per le applicazioni PC/Mobile, l'unico modo di procedere è stato quello di basarsi sui dati raccolti e di provare in prima persona quante più applicazioni possibili, per poi avanzare delle ipotesi in merito a ciò che abbiamo ritenuto maggiormente efficace per raggiungere gli obiettivi prepostoci.

Innanzitutto abbiamo escluso ciò che, delle UI analizzate, risultava non essere efficace nel nostro caso. Per quanto la logica bidimensionale non possa essere completamente eliminata, non è nemmeno possibile riportarla in VR senza apportarvi alcuna modifica. In quel caso, oltre a non sfruttare i nuovi spazi a disposizione, si aggiungono complessità di utilizzo, dovendo interagire con schede e finestre nate per mouse e tastiera e non per controller VR, oltre ad una mancanza di chiarezza dovuta a scritte eccessivamente piccole o icone non adeguate che in VR diventano illeggibili (vedi figura 4.2).

Nella quasi totalità delle applicazioni analizzate, ove è possibile muoversi nello spazio, i menu sono ancorati ai controller. Ciò risulta comodo, poiché sono sempre a disposizione, ma potrebbero diventare snervanti se questi diventassero eccessivamente ingombranti e non si avesse modo di ridurli di dimensioni o chiuderli. Quindi riteniamo si debba sempre dare all'utente la possibilità di rimpicciolire o addirittura nascondere tutti gli oggetti invasivi, senza rinunciare all'immediatezza di accessibilità ai menu (esempio: menu in figura 4.8 esaustivo ma ingombrante e non richiudibile).

Molte applicazioni inoltre tendono a rendere l'intero stile dei menu molto low poly, colorato e giocoso. L'approccio low poly è indubbiamente efficace ove non vi è necessità di fotorealismo assoluto, poiché migliora le prestazioni generali dell'applicativo, ma con questo "effetto cartoon" a volte ci si allontana da una apparenza di serietà e professionalità. Se in alcuni casi l'aspetto vivace funziona, in altri il target stesso non ammette uno stile eccessivamente giocoso.

D'altro canto abbiamo anche definito quali aspetti avrebbero potuto funzionare per il nostro progetto e ci potevano aiutare a progettare le diverse interfacce utente che intendevamo implementare per la VR.

Dove le interazioni non richiedono di usare i controller in modo complesso, la rappresentazione di quest'ultimi attraverso delle mani virtuali risulta avere un effetto più immersivo per l'utente. Quanto descritto per le mani virtuali, vale anche quando si vuole riprodurre in modo realistico gli strumenti che si possono utilizzare all'interno dell'applicazione, infatti per renderli efficaci non basta progettarli in modo che il loro comportamento sembri il più realistico possibile ma bisogna corredare a ciò anche un'opportuna rappresentazione fotorealistica dello strumento utilizzato. Questo ovviamente non è sempre possibile, per cui in alcuni casi è necessario far coesistere elementi in 3D con altri che invece si appoggiano ancora ad una logica di interazione bidimensionale. In quest'ottica risultano essere molto intuitivi i menu radiali, sia per la loro semplicità che per la velocità di utilizzo.

Eccessiva complessità dei menu o di utilizzo degli stessi può diventare disorientante, soprattutto per gli utenti meno esperti, che ad oggi sono la maggioranza per quanto riguarda la VR. Dunque perseguire un approccio di semplicità e di coerenza, sia di stile che di logiche di utilizzo, aiuta l'utente a non avere la sensazione di non essere in grado di capire o utilizzare l'applicazione. Risulta poi efficace suddividere i diversi menu in slot o strutture differenti, ancora meglio se separabili fra loro e posizionabili a piacere dell'utente nello spazio circostante. Ultimo fattore importante da tenere in considerazione è che questi menu, una volta posizionati, dovrebbero seguire l'utente nei suoi spostamenti all'interno dello spazio, rimanendo così sempre a disposizione.

Va sempre tenuto in considerazione che con questa tecnologia difficilmente si ha a che fare con utenti esperti, infatti, pur rispettando tutti i punti sopra descritti, potrebbe non essere comunque chiaro per l'utente medio come si debba interagire con menu e strumenti dell'applicazione. Per cui è necessario dotare ogni strumento o pulsante di una apposita Label, non invasiva ma allo stesso tempo esaustiva del tipo di funzionalità associata ad esso. Altro metodo usato soprattutto da Oculus nel suo menu principale, è quello di fornire all'utente molteplici vie per raggiungere il medesimo obiettivo, offrendogli differenti mezzi per facilitargli il compito.

4.1.3 - IDEE, MOCKUP E PRIME IMPLEMENTAZIONI

A seguito di questa analisi abbiamo pensato a tre possibili soluzioni per lo sviluppo dell'interfaccia utente.

La prima è quella di creare un menu radiale ancorato al controller che gestisca l'intera logica dei menu, aggiornandosi a seconda delle scelte dell'utente. Scelta una variante del configuratore questa verrà caricata sul modello e una volta attivato un tool con l'apposita opzione questo diventerà attivo e utilizzabile. Per quanto possa essere sicuramente efficace, tale scelta non è altro che la traslazione di quanto fatto per la UI Desktop sulla UI VR, trasformando i menu lineari in menu radiali.
Spingendoci un po' oltre ad una soluzione di mera transizione e provando a sfruttare meglio lo spazio tridimensionale, la seconda idea prevede l'utilizzo di sole componenti 3D, a partire dal menu iniziale fino ad arrivare alle singole mesh degli strumenti. Sicuramente una scelta più affine alla nuova tecnologia a disposizione, ma che risulta di complessa realizzazione per quanto riguarda la rappresentazione del configuratore. Se le varianti sono facilmente rappresentabili in 3D, ciò non vale per fasi e categorie, che anche nella realtà sarebbero rappresentate in un catalogo o sullo schermo di un computer.
Da qui si arriva alla terza idea, ovvero un ibrido delle due precedenti , in cui viene utilizzato sempre un menu radiale per la gestione di tutte le sezioni dei vari sottomenu difficilmente rappresentabili con oggetti reali, per poi passare ad una visualizzazione in 3D ove più realistico ed efficace.

Scelta quest'ultima opzione, siamo passati alla realizzazione dei mockup per definire al meglio ogni singola interazione. Se per la versione Desktop i mockup rispecchiavano fedelmente ciò che sarebbe stato il risultato finale, per la VR non poteva essere così. Abbiamo cercato strumenti per la realizzazione di mockup per la VR, ma ne abbiamo trovati solo di molto limitati, dove la complessità di utilizzo non valeva la resa e soprattutto nessuno dove fosse possibile lavorare in più di una persona o condividere il modello creato, dunque siamo rimasti su strumenti per mockup desktop, consci che la resa in tre dimensioni sarebbe stata molto diversa.

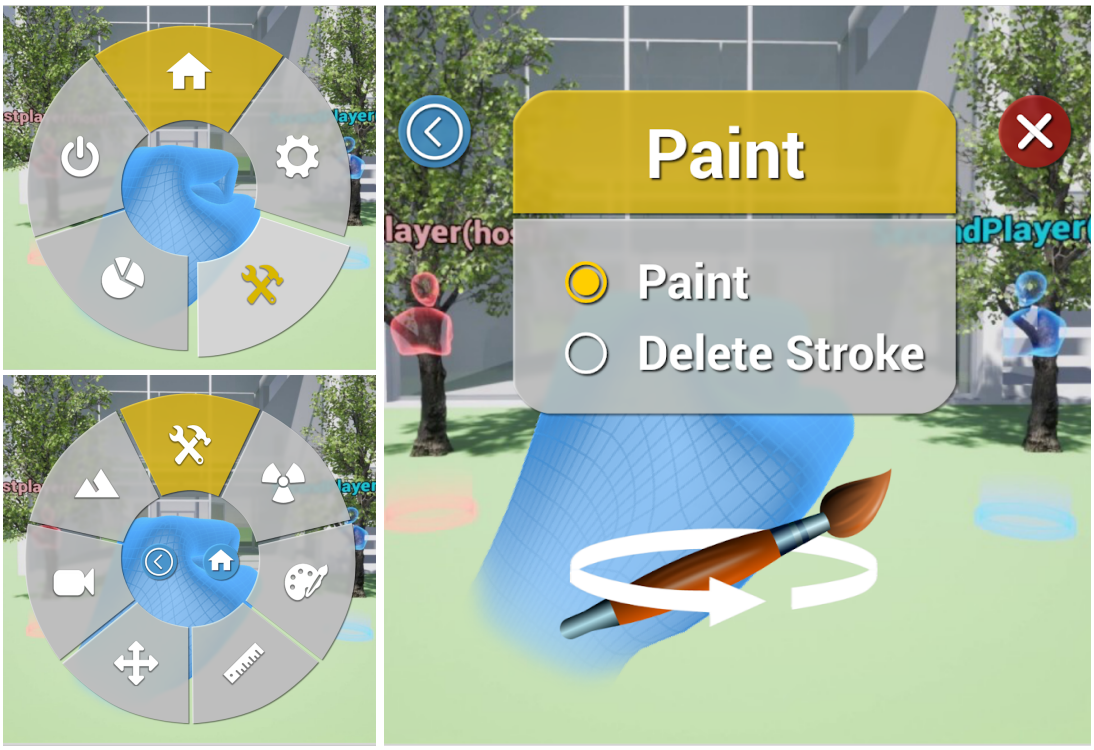


FIGURA 4.11: MENU RADIALE CHE SI AGGIORNA AD OGNI SCELTA E UN PANEL CON LE OPTION E LA MESH DEL TOOL PAINT

In figura 4.11 è mostrato il primo mockup realizzato. Come si può notare il menu radiale, alla scelta ad esempio del *Menu Tool*, viene suddiviso nuovamente in più spicchi, mostrando in alto la sezione scelta e attorno i Tool a disposizione. Scelto poi uno di essi viene mostrato al posto del radiale un pannello con le relative Option del Tool e sotto di esso un oggetto 3D rappresentante il Tool stesso, che per essere utilizzato deve essere preso dall'utente.

Per evitare il passaggio ad un ulteriore menu, si è modificata l'idea iniziale scegliendo di disporre le Option direttamente sopra il Tool, la cui Mesh sostituisce la mano virtuale dell'utente non appena viene scelto dal menu radiale. Selezionata un'opzione questa viene iconizzata e lasciata visibile sopra il Tool, in modo che funga da costante feedback per l'utente, come mostrato in figura 4.12.

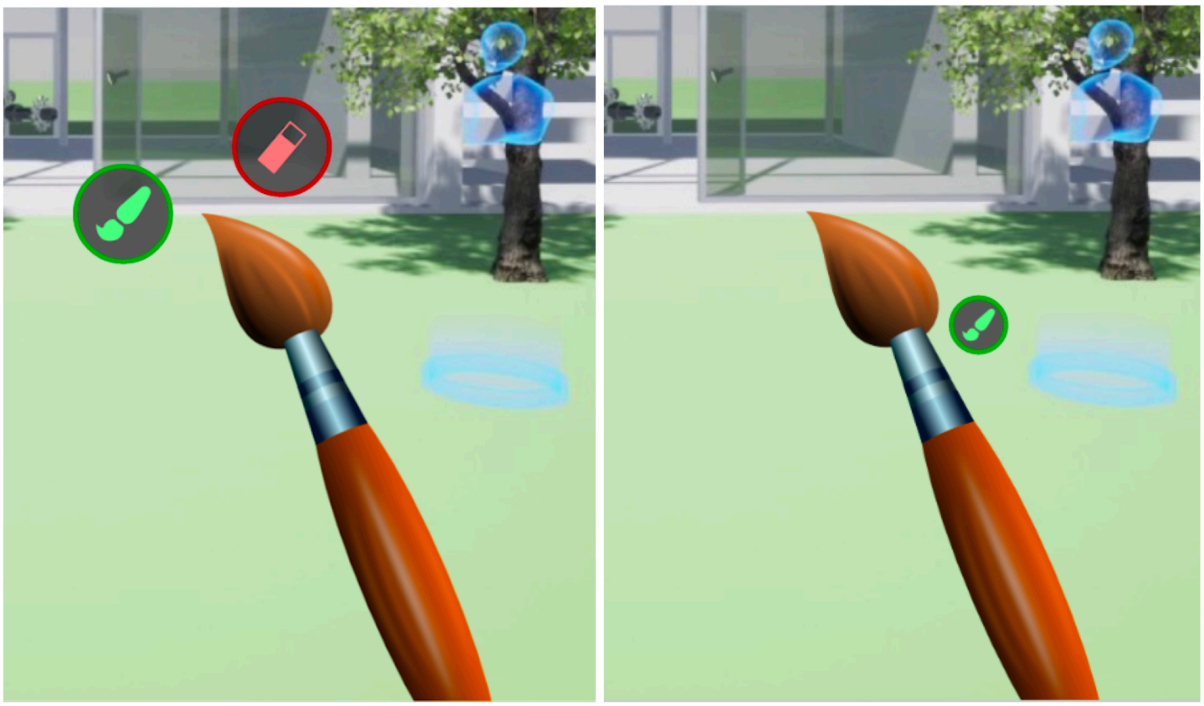


FIGURA 4.12: TOOL PAINT CON LE PROPRIE OPTION MOSTRATE SOTTO FORMA DI ICONA SOPRA LA MESH E L'OPZIONE SCELTA VIENE POI ICONIZZATA VICINO ALL'OGGETTO 3D

Così facendo si evita l'introduzione di un pannello solo per le Option, ma viene a mancare anche il gesto di presa del Tool, che a nostro parere avrebbe reso l'interazione maggiormente realistica. Abbiamo quindi pensato di utilizzare il menu radiale solo per il configuratore e di posizionare i Tool attorno alla vita dell'utente, in una sorta di *Toolbelt* sempre disponibile. L'idea è stata abbandonata pressoché subito poiché si andava a perdere una coerenza di stile, se non consideriamo che staccare completamente quella parte di menu poteva essere disorientante per l'utente oltre che scomodo, dal momento che i Tool sarebbero rimasti ancorati attorno all'utente anche quando non erano necessari.

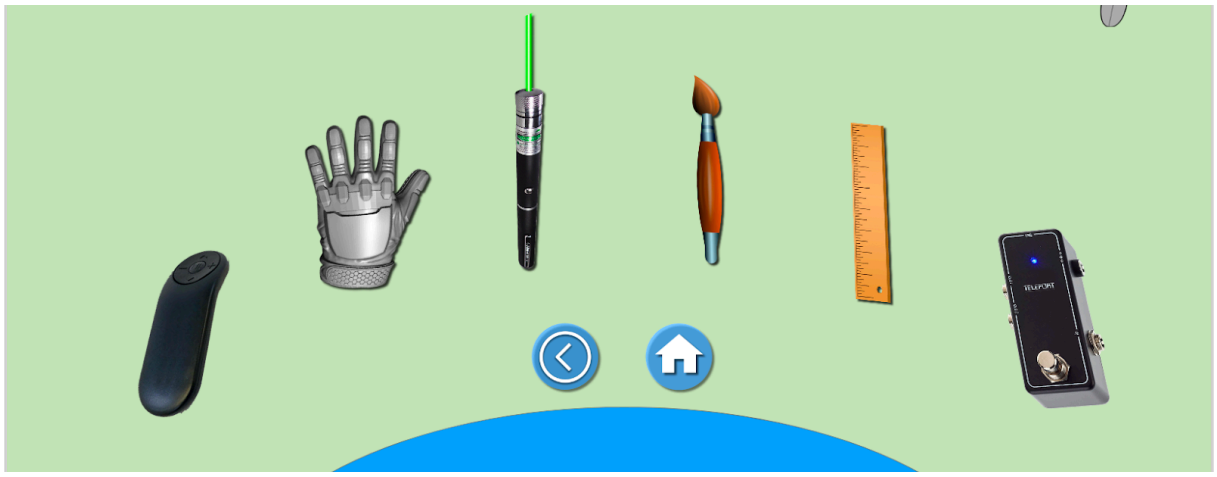


FIGURA 4.13: MOCKUP DELLA TOOLBELT ATTORNO ALLA VITA DELL'UTENTE, COMPOSTA DAI MODELLI 3D DEI TOOL A DISPOSIZIONE

Tralasciando l'ipotesi di una toolbelt, andava comunque definito come gestire le fasi e categorie del configuratore all'interno del menu radiale. Tenendo conto dell'imprevedibilità del numero di fasi o categorie e dovendo creare un menu che fosse gestibile in ogni circostanza, ci siamo resi conto che il menu radiale mostrato in figura 4.11 poteva non essere molto versatile nel caso le categorie da mostrare fossero di numero elevato. Allo stesso modo per le varianti, rappresentarle sul menu radiale poteva creare problemi sia per quanto riguarda il loro numero, sia in termini di dimensioni delle immagini che su uno spicchio del menu radiale potevano essere eccessivamente piccole. Per di più, nel caso di diverse sottocategorie innestate, diventa complesso notificare correttamente il percorso fatto o come tornare indietro.

Da qui siamo arrivati all'idea di tenere fisse le 5 parti che compongono il menu radiale, sfruttando lo spazio esterno per mostrare fasi, categorie e varianti. Disporre una serie di pulsanti attorno al menu radiale è una soluzione più scalabile: offre molto più spazio per collocare i vari pulsanti relativi ad una categoria o variante, che potenzialmente possono arrivare a ricoprire l'intera circonferenza, ed è una soluzione tranquillamente applicabile anche per Tool e Settings, come mostrato in figura 4.14.

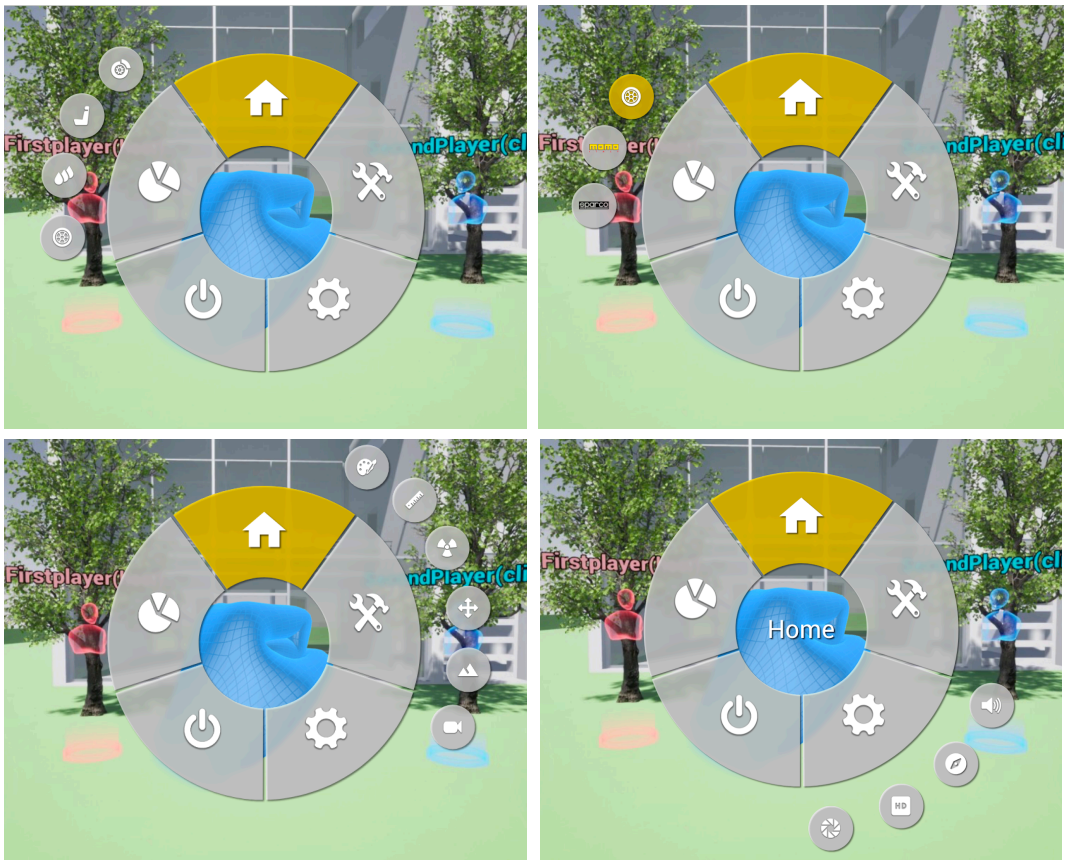


FIGURA 4.14: IN ALTO A SINISTRA UN ESEMPIO DEL MENU CONFIGURATOR CHE MOSTRA LE FASI DI CONFIGURAZIONE, IN ALTO A DESTRA LO STESSO CON LE CATEGORIE DELLA FASE RIMS, SOTTO IL MENU TOOLS E IL MENU SETTINGS CON LE RELATIVE OPZIONI

Se da un lato questo tipo di soluzione risolve il problema del fornire maggiore scalabilità alla rappresentazione di fasi e categorie, dall'altro però rimane inefficace per quanto riguarda la rappresentazione delle varianti. Queste, anche aumentando la dimensione dei pulsanti che circoscrivono il menu radiale, richiedono comunque uno spazio maggiore di quello attualmente disponibile, anche in ottica di dover visualizzare le informazioni aggiuntive del Payload.
Abbiamo dovuto dunque nuovamente re-inventare il menu radiale, questa volta aggiungendo anche migliorie grafiche che andassero ad aggiungere feedback a disposizione dell'utente e sfruttando meglio la parte centrale del menu, fino ad ora quasi inutilizzata.
In questa versione ogni spicchio rappresentante un menu apre un'area laterale dedicata alla corretta visualizzazione di tutto il contenuto del menu in questione, come mostrato in figura 4.15.

In questo modo è possibile sfruttare l'intera area laterale per rappresentare le varianti, aggiungendo un ulteriore area sottostante, apribile a piacimento, per contenere le informazioni del Payload ad esse associato. .
A questo punto, considerato che quest'ultima soluzione ci sembrava essere la più adatta e completa, è sorta la necessità di passare alla fase di implementazione, anche se solamente a livello strutturale. Si è reso necessario questo passaggio poiché non era scontato che ciò che sembrava funzionale sui mockup lo fosse effettivamente anche in realtà virtuale, essendo il mockup una versione semplificata e in due dimensioni di ciò che invece avrebbe dovuto funzionare adeguatamente in uno spazio tridimensionale..
Dopo aver ricreato, in un apposito widget, la struttura base del menu radiale, fedele a quanto raffigurato in figura 4.15, esso è stato aggiunto, attraverso l'utilizzo di un *Widget Component*, al Motion Controller del Player Pawn utilizzato dal Collab Viewer per la navigazione in realtà virtuale. In questo modo ci è stato possibile posizionarlo sul dorso della mano sinistra, rendendolo costantemente a nostra disposizione per poterne valutare la resa grafica e la semplicità di utilizzo in VR.
Ci è immediatamente parso chiaro quanto fosse enorme la differenza tra la rappresentazione grafica del suddetto menu su mockup bidimensionale e la relativa visualizzazione in ambiente 3D. Se dunque per la versione Desktop i mockup davano una rappresentazione estremamente fedele di ciò che sarebbe stato il risultato finale, ciò non era assolutamente possibile per quanto riguarda la realtà virtuale.

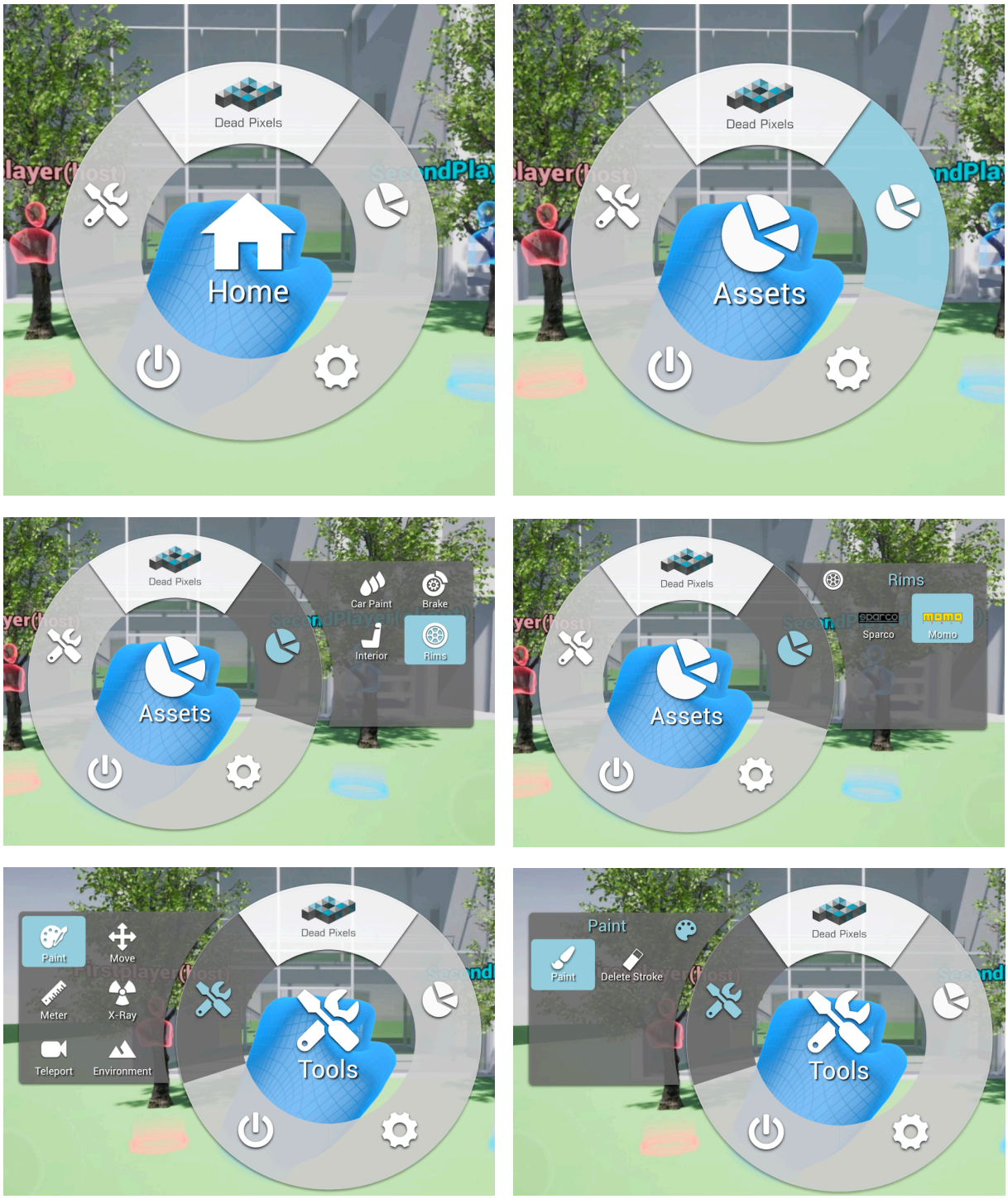


FIGURA 4.15: MENU RADIALE CHE NOTIFICA AL CENTRO IL MENU SELEZIONATO O APERTO IN QUELL'ISTANTE. A DESTRA E SINISTRA SI APRONO DEI PANNELLI LATERALI CHE SI AGGIORNANO AD OGNI SCELTA FATTA E RIPORTANO L'ICONA DELL'OPZIONE SELEZIONATA IN ALTO SUL PANNELLO INSIEME AL NOME DELLA FASE O TOOL SCELTO

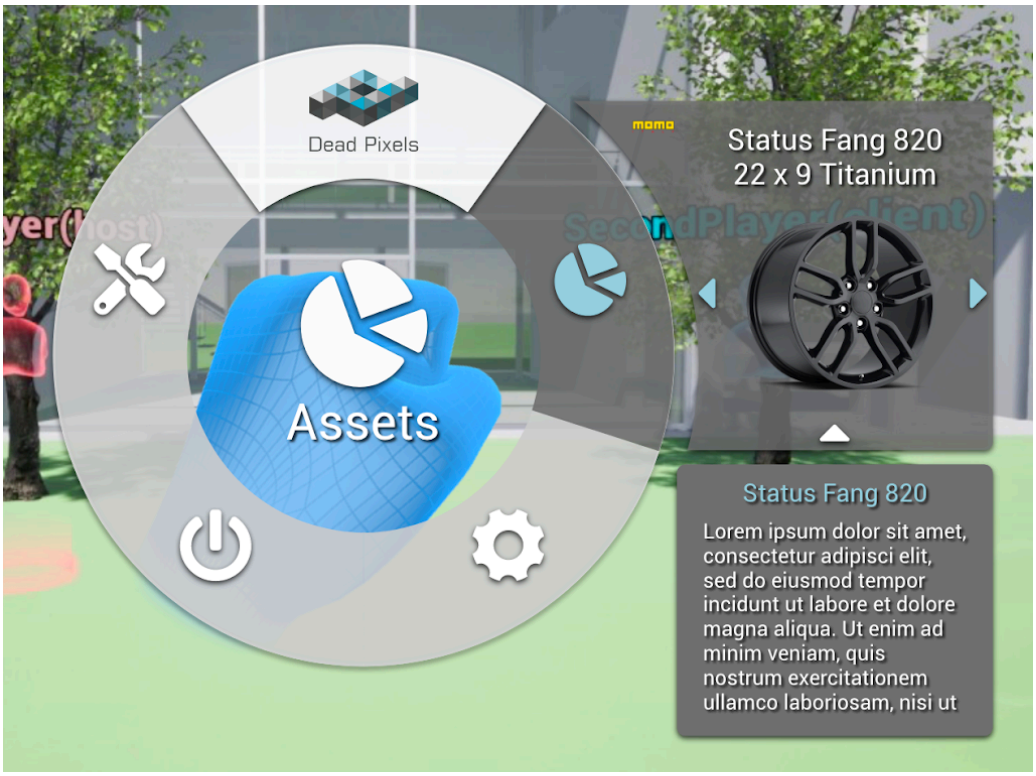


FIGURA 4.16: MENU RADIALE E PANNELLO LATERALE CONTENENTE LE VARIANTI E SOTTOSTANTE AD ESSO UN ULTERIORE PANNELLO RICHIUDIBILE CONTENENTE IL PAYLOAD DELLA VARIANTE IN QUESTIONE

4.1.4 - MAIN MENU E UI MODULARE

A questo punto, prima di passare all'effettiva implementazione, collegando dunque tutte le informazioni da Variant Manager e Data Table e tutto il funzionamento dei Tool rappresentati da oggetti 3D, vi è stato un approfondito confronto con il Team di Dead Pixels che ci ha seguito in questo progetto.
Seppur avessimo piena libertà di implementazione, abbiamo ritenuto fondamentale avere un confronto con chi, lavorando nel settore, avesse sicuramente più esperienza di noi.
Da parte loro ci è stato suggerito di sfruttare meglio lo spazio messo a disposizione dall'ambiente, poiché l'interfaccia progettata sino a quel momento, seppur funzionale, era ancora molto legata ad una logica 2D. Di fatto stavamo limitando l'intero menu ad uno spazio ristretto sul dorso della mano virtuale, quando potenzialmente potevamo sfruttare tutto quello disponibile attorno all'utente.
Quanto progettato fino a quel punto non era propriamente sbagliato, considerato anche che quasi tutte le applicazioni analizzate gestivano il menu fissandolo ad un controller, ma semplicemente limitato e vi era sicuramente possibilità di fare meglio.

Una soluzione alternativa era quella di dividere i vari menu in moduli differenti, apribili singolarmente e posizionabili a piacere dell'utente nello spazio a lui circostante (ad esempio vedi figura 4.4 e 4.5).

Prendendo ispirazione dall'interfaccia utente di Oculus, abbiamo pensato che potesse essere funzionale disporre le 4 icone del Main Menu su una sorta di piccolo smartband, ancorato all'altezza del polso nel Motion Controller, dando modo all'utente di aprire da lì tutti gli altri menu.
Abbiamo modellato dunque un piccolo smartband, implementando un widget con le 4 icone dei vari menu e posizionandolo al posto dello schermo del bracciale. Seppur l'idea sembrasse realistica ed efficiente, ci siamo accorti fin da subito che in realtà virtuale bisognava scendere a compromessi con le sue dimensioni eccessivamente ridotte. Pur provando a trovare un compromesso tra la dimensione del bracciale e una interazione efficace, ci siamo presto resi conto di dover abbandonare l'idea.

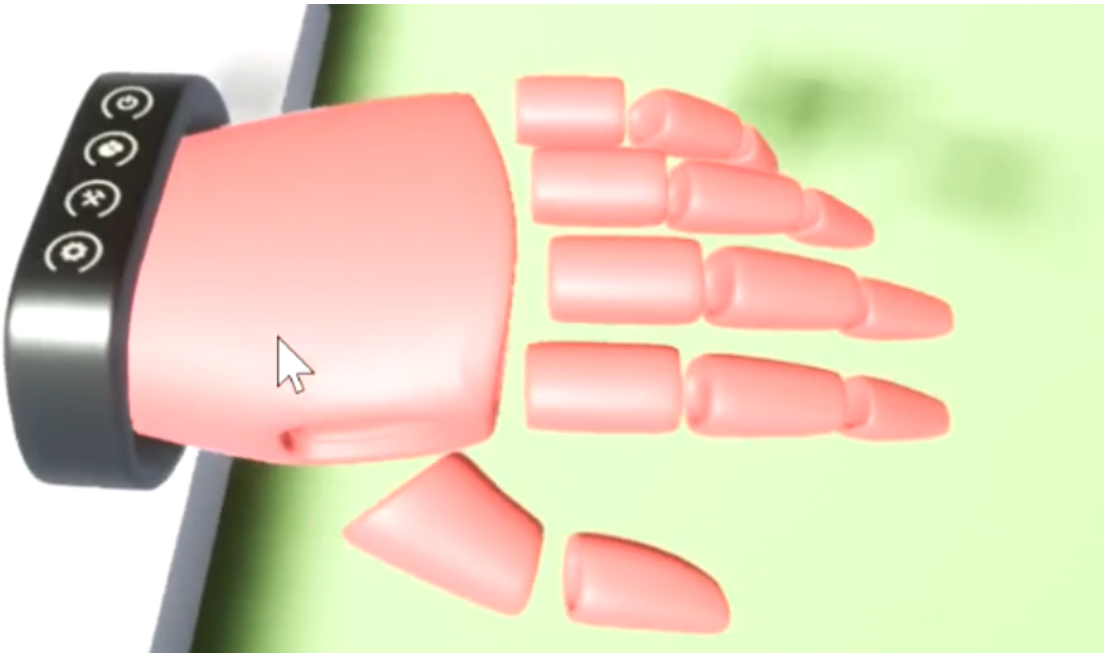


FIGURA 4.17: SMARTBAND SU MANO VIRTUALE DELL'AVATAR DELL'UTENTE

Nonostante non fosse una soluzione praticabile, resta comunque efficiente l'idea di avere un Main Menu di partenza, dal quale aprire tutti gli altri menu secondo necessità. Per cui abbiamo deciso di riprogettare l'interfaccia utente, non cancellando però quanto di buono era stato fatto fino a quel momento. Infatti il menu radiale, per quanto limitato nel gestire interamente i menu della UI, risulta essere comunque una struttura di semplice e rapido utilizzo, dunque ottimo come punto di partenza per aprire tutte le altre sezioni.

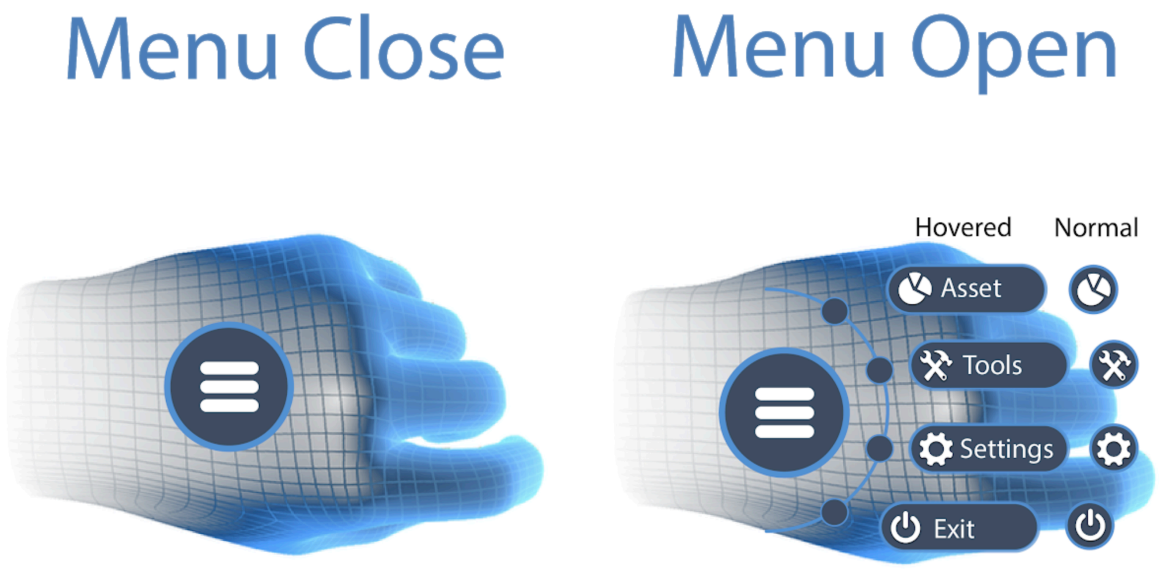


FIGURA 4.18: MOCKUP DEL MAIN MENU ADIBITO ALL'APERTURA DEI VARI MODULI

Andavano dunque ideati tutti gli altri moduli che nel loro insieme sarebbero andati a definire l'intera struttura di questa nuova UI.
Innanzitutto, per comprendere come rappresentare i vari menu dell'interfaccia utente, andava compreso che tipo di informazioni dovevamo far visualizzare. I menu da creare erano sostanzialmente tre, se consideriamo che quello di **Logout** è completamente definito all'interno del Main Menu stesso, e questi erano a loro volta raggruppabili in due categorie in base al tipo di informazioni visualizzate: il **Menu Tools** mette a disposizione dell'utente degli strumenti, rappresentati fisicamente come oggetti di comune utilizzo, per interagire col prodotto in scena e con l'ambiente circostante ad esso; per quanto riguarda **Menu Configurator** e Settings invece si hanno informazioni astratte, difficilmente rappresentabili fisicamente, fatta eccezione per le sole varianti, oltre al fatto di avere una notevole mole di informazioni rappresentabili solo attraverso icone, come fasi e categorie del prodotto da configurare o strumenti i vari di setting.

Per quanto riguarda il *Menu Tools* siamo ripartiti dall'idea di *toolbelt* avuta inizialmente. Non essendo pienamente convinti da una soluzione che prevedesse Tool sospesi nel vuoto all'altezza della vita dell'utente, abbiamo scelto di provare ad implementare qualcosa che più si avvicinasse alla realtà. Ci siamo inizialmente ispirati al menu iniziale di Oculus, nel quale è

disponibile una Dashboard all'altezza della vita dell'utente che rende facilmente accessibili tutti i menu più importanti attraverso pulsanti con effetti 3D.
La nostra necessità non si discostava molto da quella di Oculus, dovendo rendere facilmente accessibili degli oggetti 3D che rappresentassero e rendessero utilizzabili i Tool.
Nella realtà di ogni giorno, quando si deve lavorare con degli strumenti, si tende solitamente a disporli su un piano in modo che possano essere costantemente a portata di mano. Da qui l'idea di realizzare una Dashboard, apribile dal Main Menu e posta all'altezza della vita dell'utente, che lo segua ovunque esso si sposti nell'ambiente virtuale e che sia posizionabile a piacere attorno all'avatar dell'utente semplicemente prendendola e spostandola.

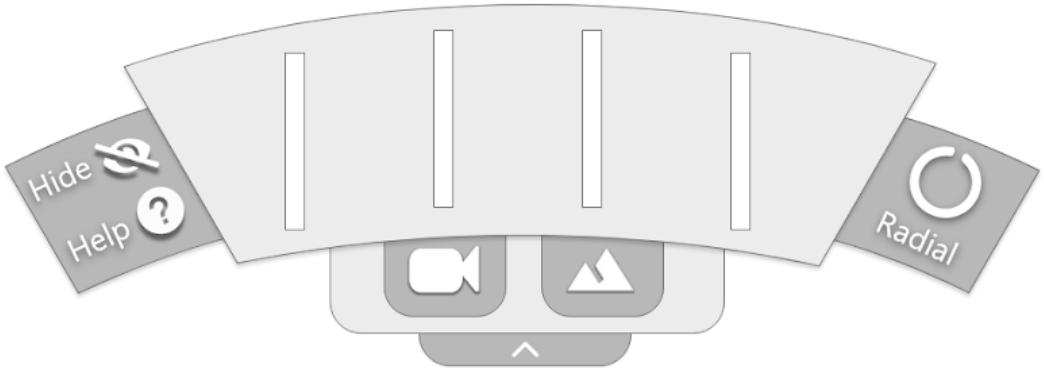


FIGURA 4.19: MOCKUP DELLA TOOLS DASHBOARD REALIZZATO IN FASE DI IDEAZIONE

Poiché non era possibile associare ai Tool Environment e Teleport degli strumenti di comune utilizzo che esplicassero visivamente la loro funzione, si è pensato di gestirli diversamente e rappresentarli quindi come pulsanti posti in un cassetto a loro dedicato. Alla scelta di uno di questi seguirà l'apertura di una serie di portali, posti a semicerchio attorno all'utente e rappresentanti le opzioni del Tool in questione.



FIGURA 4.20: MOCKUP DELLE OPZIONI DEL TOOL ENVIRONMENT DELLA UI

Le considerazioni appena fatte per il Menu Tools non valgono per il Menu Configurator e Settings, nei quali vengono visualizzate principalmente informazioni astratte. Per riuscire a trovare una soluzione appropriata al nostro caso, abbiamo deciso di prendere ispirazione da alcuni modelli di interfaccia utente presenti nell'applicazione ***VR Advanced Framework***, scaricabile dal *Marketplace* di *Unreal Engine* e consigliataci dal Team di Dead Pixels.
Normalmente nel mondo reale il processo di configurazione di un prodotto può avvenire o attraverso un catalogo o, nel caso si debbano visualizzare una grande mole di categorie, varianti e informazioni, attraverso un computer / tablet.

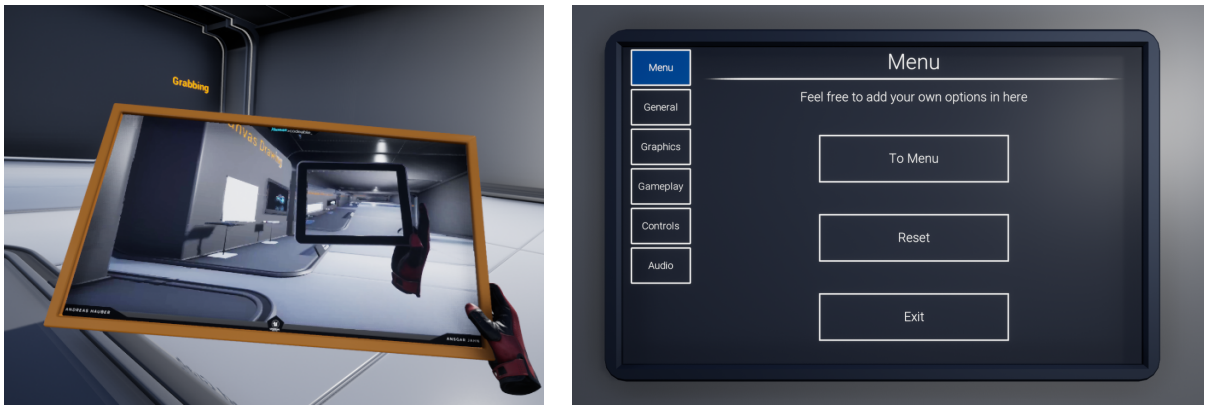


FIGURA 4.21: REFERENCE DI SCHERMI/TABLET DA VR ADVANCED FRAMEWORK

Dopo aver riscontrato che anche all'interno del Framework sopracitato veniva utilizzato un tablet per ospitare menu con grandi quantità di dati, come mostrato in figura 4.21, si è ritenuto di poter perseguire in questo caso una fusione tra una logica in due e una in tre dimensioni per riuscire a rappresentare correttamente ed efficacemente tutte le informazioni necessarie.
Abbiamo pertanto scelto che il *Menu Configurator* e il *Menu Settings* venissero rappresentati attraverso un Tablet virtuale, che fungesse da base per due differenti Widget adibiti alla corretta gestione di questi menu.
Tuttavia, visto che in questo modo si sarebbe sacrificata la rappresentazione 3D delle varianti, relegandole ad una mera immagine, abbiamo scelto di aggiungere una modalità "aumentata" di visualizzazione di queste, utilizzabile su richiesta dell'utente. Così facendo l'utente può scegliere di visualizzare le varianti sia con il Tablet, che aprendo una apposita Dashboard, mediante la quale esse potranno essere visionate come modelli in 3D.

4.2 - STRUTTURA GENERALE DELL'INTERFACCIA UTENTE VR

In questo sottocapitolo verrà illustrata la struttura generale che regola l'interazione dei vari moduli che compongono l'interfaccia utente. Vorremmo soffermarci brevemente anche sulle basi di interazione e movimento in VR, non essendo per nulla scontate. In tutta la spiegazione che segue verrà fatto riferimento ai controller di Oculus Rift, l'headset che avevamo a disposizione durante l'implementazione, nonostante i comandi nell'applicazione siano utilizzabili anche da HTC Vive.

4.2.1 I MENU E LE INTERAZIONI DI BASE

Di default l'applicazione, dopo aver creato una sessione con tutti i player, nelle modalità definite nel Collab Viewer, si avvia in modalità desktop. Per passare alla visualizzazione in realtà virtuale, e di conseguenza alla relativa interfaccia, è necessario cambiare modalità di navigazione dall'apposita voce nel Menu Settings.
In questo modo, una volta indossato il visore, l'utente sarà immerso nello spazio virtuale e troverà, in concomitanza della posizione occupata dai due controller, due mani virtuali dello stesso colore con cui è stato caratterizzato l'Avatar dell'utente.
La scelta di rappresentare i **controller come mani** è data sia dal fatto che queste venivano già rese disponibili dal Collab Viewer, sia perché riteniamo possano migliorare il senso d'immersione. I comandi associati ai pulsanti dei due controller sono stati progettati per essere sia speculari, lasciando quindi libertà all'utente di scegliere se utilizzare prevalentemente un controller o l'altro, che immutabili durante l'intera esperienza, così una data funzionalità rimarrà sempre associata al medesimo pulsante. Presumiamo dunque che l'utente, una volta comprese quelle 4 interazioni di base necessarie a poter utilizzare correttamente gli strumenti e le interfacce offerte dall'applicativo, non abbia bisogno di ricevere costanti feedback di utilizzo del controller.

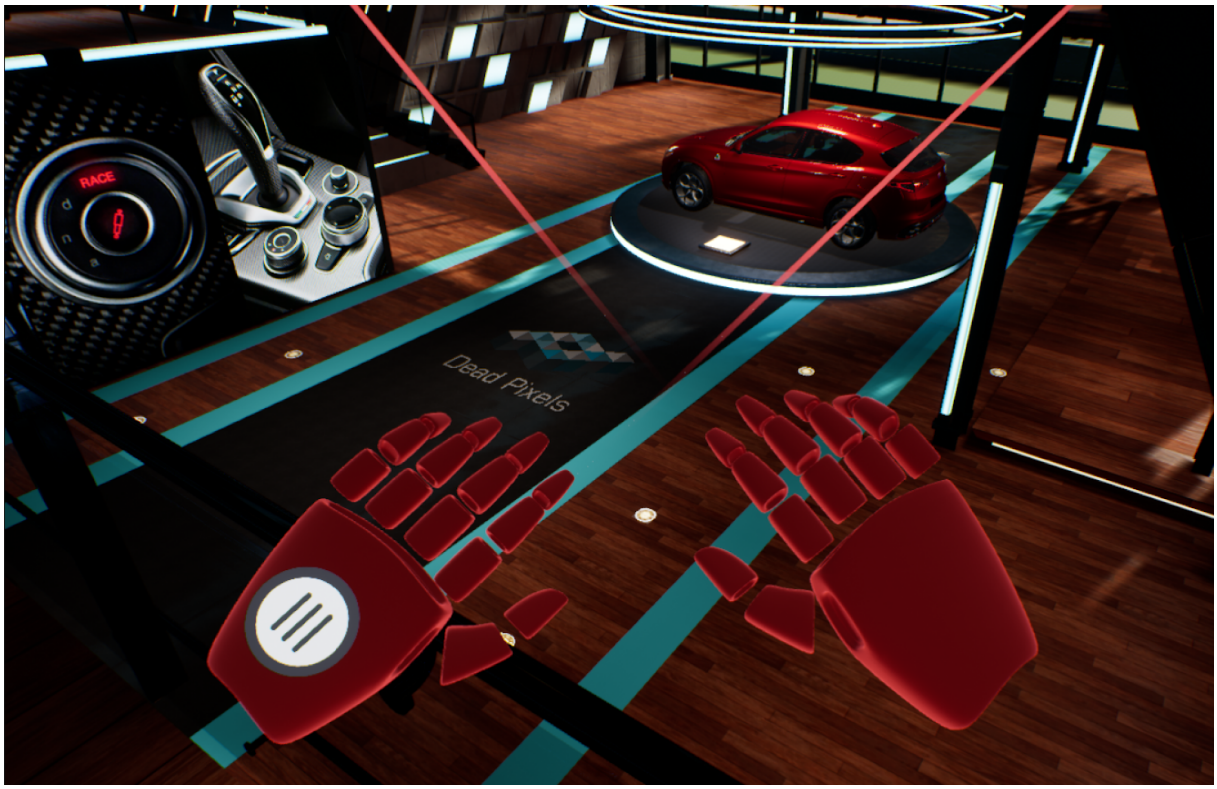


FIGURA 4.22: MANI VIRTUALI NELLA POSIZIONE OCCUPATA DEI CONTROLLER, DALLE QUALI PARTONO DUE LASER CHE FUNGONO DA PUNTATORI

L'utente potrà immediatamente notare come da entrambe le mani parta un laser dello stesso colore di queste. Il laser è lo strumento principale di puntamento verso qualunque oggetto dell'ambiente, comportandosi dunque come fosse il cursore del mouse, volendolo paragonare alla rispettiva funzione in due dimensioni. Con lo stesso parallelismo si può definire la funzione del **pulsante Trigger**, posto nella parte frontale del controller e utilizzabile con il dito indice della mano, che è adibito alla selezione di oggetti e all'interazione con i widget presenti, più o meno come il pulsante sinistro del mouse in un'applicazione bidimensionale..

Il movimento nello spazio tridimensionale avviene tramite "**teletrasporto a breve distanza**". Questo tipo di spostamento è molto comune nelle applicazioni in VR poiché limita, se non annulla, l'effetto di **Cybersickness** tipico quando l'utente si muove traslando nello spazio il proprio avatar come se stesse camminando o addirittura volando.
Per potersi muovere l'utente deve tenere premuto il **pulsante Y o B** del controller. Questa operazione modifica il puntatore laser in modo che esso assuma la forma di un arco che parte dalla mano e arriva fino al punto indicato, creando in corrispondenza di tale punto una circonferenza con una freccia indicante l'orientamento nel punto di spostamento. Muovendo il

controller l'utente può dunque scegliere dove spostarsi, mentre ruotandolo può invece scegliere la direzione della visuale che avrà a spostamento ultimato.

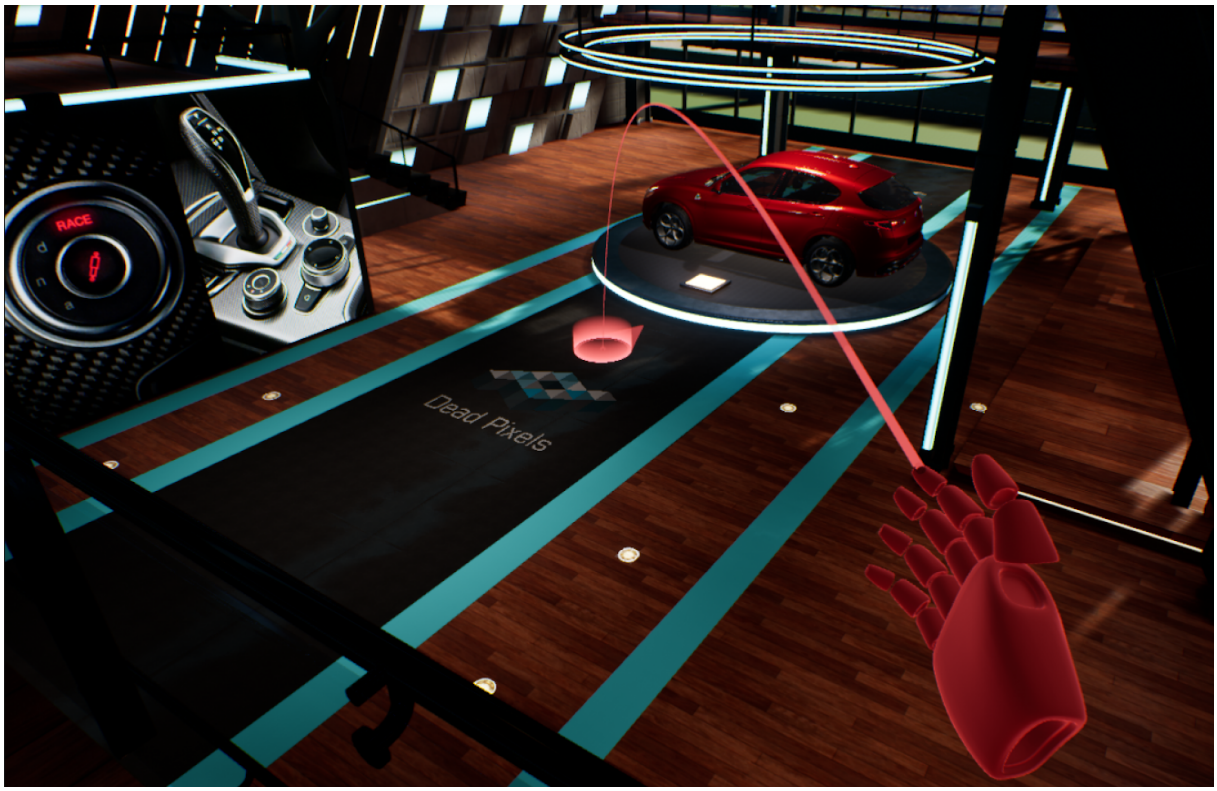


FIGURA 4.23: STRUMENTO DI SPOSTAMENTO NELL'AMBIENTE, CHE PERMETTE DI DEFINIRE IL PUNTO DOVE SPOSTARSI

L'interfaccia utente, come anticipato nel sottocapitolo precedente, è suddivisa in moduli, uno per ogni menu a disposizione, tutti richiamabili attraverso il ***Main Menu***.
Esso è posto sul dorso della mano sinistra dell'utente e inizialmente presenta solo l'icona rappresentante il menu, in modo da essere accessibile quando serve ma per nulla invasivo i quando non necessario. Esso è apribile sia puntando il laser sull'icona e cliccando il relativo tasto Trigger, che cliccando sul **pulsante X o A** dei controller, lasciando così libertà all'utente di scegliere l'interazione per lui più comoda.



FIGURA 4.24: MAIN MENU CHIUSO E APERTO CON I RELATIVI MENU DISPONIBILI

Una volta aperto, il Main Menu mostra a destra dell'icona principale 4 icone più piccole disposte a semicerchio. Dall'alto verso il basso queste icone rappresentano il Menu Configurator, il Menu Tools, il Menu Settings e il pulsante di chiusura dell'applicazione, rispecchiando esattamente le stesse dell'interfaccia Desktop.
Puntando il laser su ognuna di queste viene mostrato il rispettivo nome del menu, che potrà essere aperto cliccandoci sopra col pulsante Trigger. Se ai primi tre menu vengono riservati tre nuovi moduli distaccati dal Main Menu, per l'ultimo pulsante di chiusura invece viene aperta una piccola finestra lateralmente al menu principale, che chiede conferma di uscita dall'applicazione.
Tutti gli altri moduli, quando aperti, vengono posti di fronte all'utente e su differenti livelli, sia di altezza che di distanza, a seconda del tipo di utilizzo richiesto da questi, ma per dare anche all'utente la possibilità di poter tenere aperto più di un menu in contemporanea senza che essi si intralcino tra loro. La posizione di questi è relativa all'utente, dunque lo seguono in ogni suo spostamento all'interno dello spazio virtuale, ma non è fissa, poiché egli può scegliere di prenderli e spostarli a piacimento attorno a sé per personalizzare il proprio spazio di lavoro.



FIGURA 4.25: REAZIONE ALL'EVENTO ONHOVERED DEI 4 BUTTON RAPPRESENTANTI I MENU

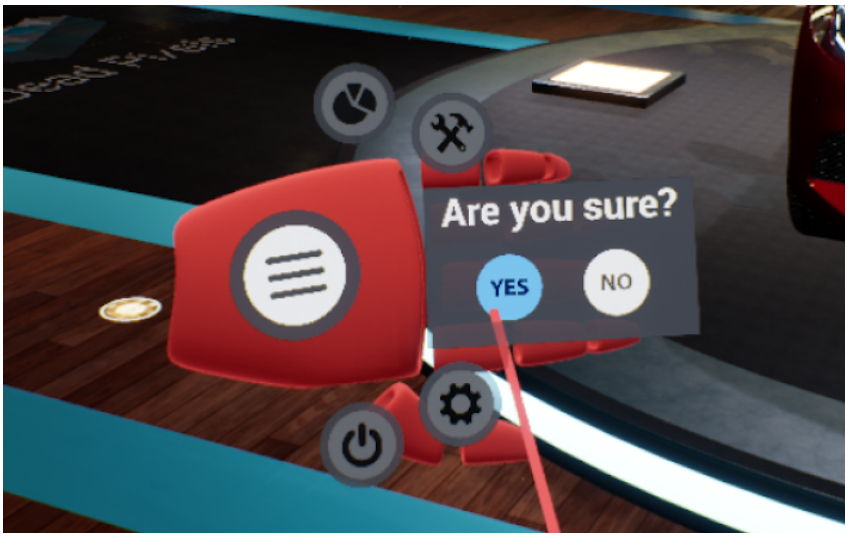


FIGURA 4.26: FINESTRA CHE SI APRE CLICCANDO SUL MENU QUIT

Se viene scelto il Menu Configurator, il Main Menu si riduce nuovamente a icona e viene aperto un Tablet, all'altezza dello sguardo dell'utente e leggermente distante da lui. In questo Tablet vengono mostrate le fasi di configurazione, con le quali l'utente può interagire per scegliere quali varianti visualizzare. A questo punto può decidere se caricarle sul prodotto da configurare oppure se accedere alla versione "aumentata" delle stesse, attraverso un apposito pulsante. Cliccandolo si chiuderà il Tablet e verrà aperta una Dashboard, adibita a mostrare le varianti con i rispettivi modelli 3D. Attraverso appositi pulsanti è poi possibile scorrere le pagine per vedere tutte la varianti disponibili, leggere le specifiche tecniche di ognuna di esse, chiudere il menu o tornare al Tablet di partenza per cambiare categoria o fase.
L'esposizione del funzionamento di questo menu è rimandata al sottocapitolo **4.3** in cui verrà spiegata nel dettaglio ogni singola parte di cui esso è costituito.

Per il Menu Settings viene usato lo stesso Tablet del Menu Configurator, ma ovviamente mostrando un widget diverso, destinato alla gestione di questo menu. Come nel caso della UI Desktop, questo menu presenta alcune opzioni non implementate poiché non fondamentali a questo progetto di tesi, ad eccezione di quella di navigazione, che permette di tornare alla versione Desktop dell'applicativo.



FIGURA 4.27: TABLET MENTRE MOSTRA LE VARIANTI DELLA FASE BRAKES, SOTTOCATEGORIA STANDARD



FIGURA 4.28: ASSET DASHBOARD CON I MODELLI DELLE VARIANTI DELLA FASE BRAKES, SOTTOCATEGORIA STANDARD, E RELATIVA FINESTRA DELLE INFORMAZIONI CON IL PAYLOAD DELLA VARIANTE SELEZIONATA

Infine è stata realizzata la Tools Dashboard seguendo il mockup mostrato in figura 4.19, apportando alcune migliorie di estetica e interazione. Sul ripiano della Dashboard sono presenti 4 oggetti, rappresentanti 4 dei 6 tool messi a disposizione. Esattamente come nella vita reale, per utilizzare questi strumenti l'utente dovrà prenderli. Questo tipo di interazione è possibile avvicinando la mano all'oggetto di interesse e premendo il **pulsante Grip**, posto lateralmente nel controller, simulando appunto il gesto di presa. Con lo stesso gesto possono essere presi e spostati a piacere i vari menu nello spazio circostante, così come descritto in precedenza.
I due Tool non rappresentati sono posti nel cassetto sottostante la Dashboard, apribile e chiudibile cliccando sull'apposita sfera. Sono stati inseriti poi lateralmente dei bottoni che permettono di chiudere la Dashboard e attivare un tutorial sul funzionamento di questa.

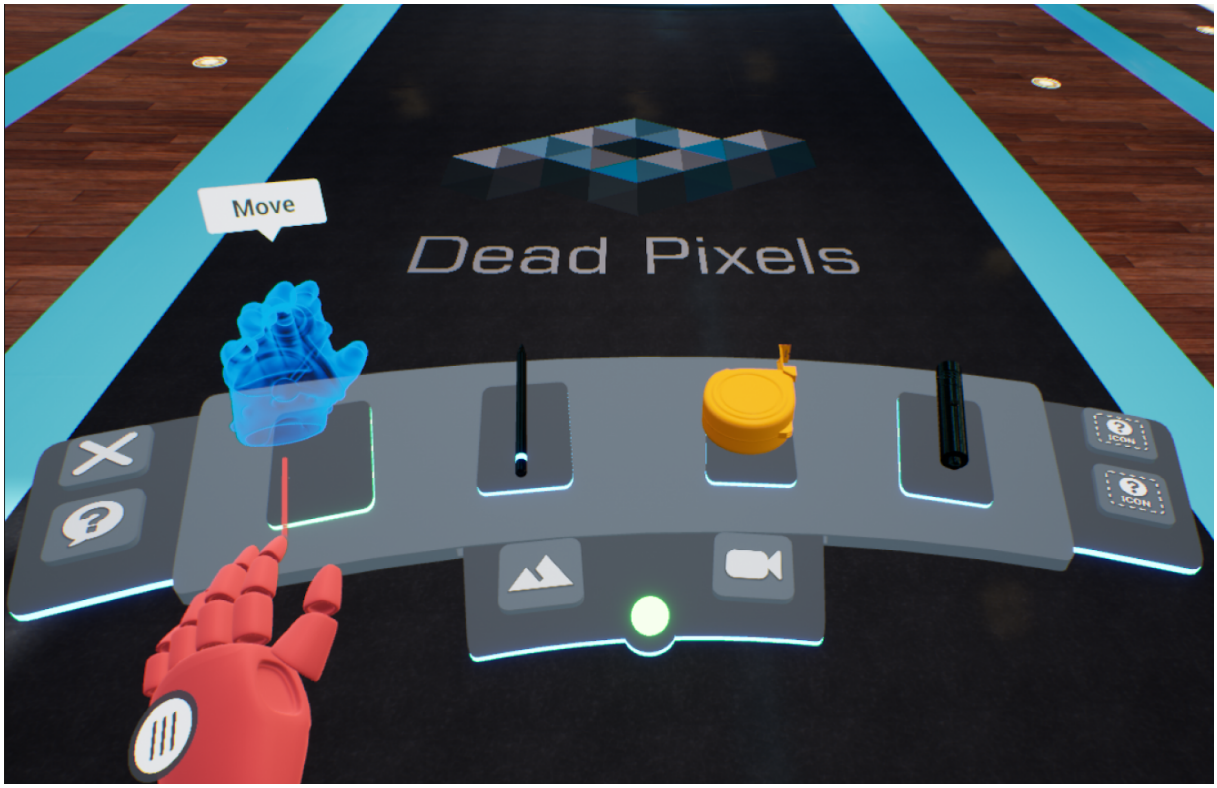


FIGURA 4.29: TOOLS DASHBOARD CON LO STRUMENTO MOVE SELEZIONATO

4.2.2 PASSAGGIO IN VR E CREAZIONE DEI MENU

L'interfaccia utente per la Realtà Virtuale va a sostituire quella Desktop non appena l'utente sceglie di passare alla modalità di navigazione VR.



FIGURA 4.30: SEZIONE NAVIGATION DEL MENU SETTINGS, DA DOVE È POSSIBILE PASSARE IN REALTÀ VIRTUALE

Come era stato anticipato nel capitolo **3.4**, nella *DP_Interface* dopo aver chiamato l'evento *Change Pawn* adibito al cambio di navigazione, viene nuovamente fatto un controllo sul Player Pawn corrente attraverso un nodo cast a ***BP_VRPawn***. Se questo non fallisce a significa che il Pawn attivo è quello legato alla navigazione in VR, e si può quindi procedere a richiamare tutti gli eventi necessari a togliere l'interfaccia utente Desktop e a costruire quella per la realtà virtuale.



FIGURA 4.31: CAST A BP_VRPAWN CON CONSEGUENTE CHIAMATA ALLA FUNZIONE IS IN VR IN CASO DI ESITO POSITIVO

Per prima cosa è necessario costruire il Main Menu e tutti gli altri moduli che compongono la UI per poi attaccarli nelle posizioni adibite ad essi. Dunque il primo evento ad essere richiamato, a seguito del Cast riuscito, è ***Is_in_VR*** della *DP_Interface.*

Esso ha il compito di recuperare tutte le informazioni necessarie a chiamare l'omonimo evento della *UI_Interface*, implementato nella *HUD_UI_Logic* e designato all'effettiva costruzione dell'intera Interfaccia Utente.

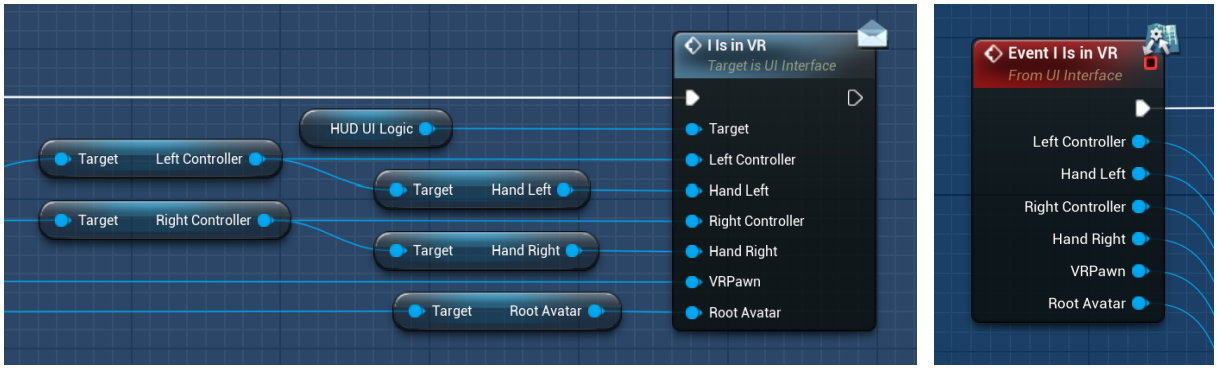


FIGURA 4.32: PARTE FINALE DELLA FUNZIONE *IS_IN_VR* DELLA *DP_INTERFACE*, CHE RICHIAMA LA FUNZIONE *I_IS_IN_VR* CON TARGET *HUD_UI_LOGIC*

Dopo aver rimosso la *General_UI* utilizzata nella UI Desktop, viene creato in primis il ***Main Menu*** e quindi l'apposito Widget che lo rappresenta, inserendolo successivamente in un *Widget Component.* Questo poi viene attaccato al *Left Controller,* istanza dell'*Actor BP_MotionController* del Collab Viewer che definisce la corretta rappresentazione dei controller destro e sinistro. In questo modo il *Main Menu* rimarrà ancorato alla mano sinistra, sempre a disposizione dell'utente quando dovesse servire.

Successivamente vengono creati due ***Capsule Collision Component*** i quali vengono attaccati al *Scene Root Component* del *Motion Controller.* Questi Component sono indispensabili per riconoscere l'oggetto più vicino alla mano, e dunque quello che l'utente ha intenzione di afferrare quando viene cliccato il tasto *Grip.*

Si prosegue poi con la creazione dei tre principali *Actor Component* che definiscono le varie componenti in cui la UI è divisa: ***Tools Dashboard***, ***Tablet*** e ***Asset Dashboard***.

A partire da questi tre *Actor* da noi definiti, vengono creati tre differenti ***Actor Component*** che vengono successivamente aggiunti e posizionati correttamente in tre rispettivi ***Scene Root Component,*** che verranno inseriti a loro volta nel *Scene Component* ***Root Avatar*** appartenente al *BP_VRPawn.* Ciò ci permette di definire a priori la posizione che avranno questi menu rispetto all'utente, in termini di altezza e distanza da esso, oltre che ogni menu richiamato venga aperto in posizione frontale rispetto all'avatar, ruotando opportunamente il *Scene Root Component* del menu.

La *DP_Interface*, dopo aver chiamato l'evento *I_Is_in_VR* della *HUD_UI_Logic*, ne chiama un secondo, denominato ***I_Get_Input_VR***, recuperando prima tutti gli input necessari attraverso *Event Dispatcher* definiti nel *BP_VRPawn*. Questa operazione, seppur non ortodossa, è necessaria per riportare tutti gli input utili, elencati in un apposito Enum ***Input VR***, nella *HUD_UI_Logic*, poiché gli eventi legati agli input da controller possono essere implementati solo in una classe Pawn, che in questo caso appartiene al Collab Viewer, mentre a noi interessa tenere separate le due parti dell'applicativo.

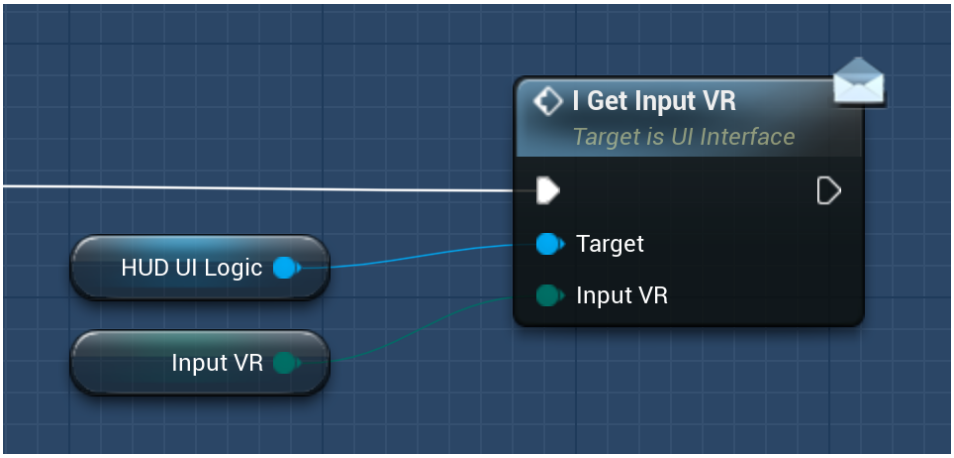


FIGURA 4.33: EVENTO *I_GET_INPUT_VR* DELLA *HUD_UI_LOGIC* CHE VIENE RICHIAMATO DALL'EVENTO *IS_IN_VR* DELLA *DP_INTERFACE*

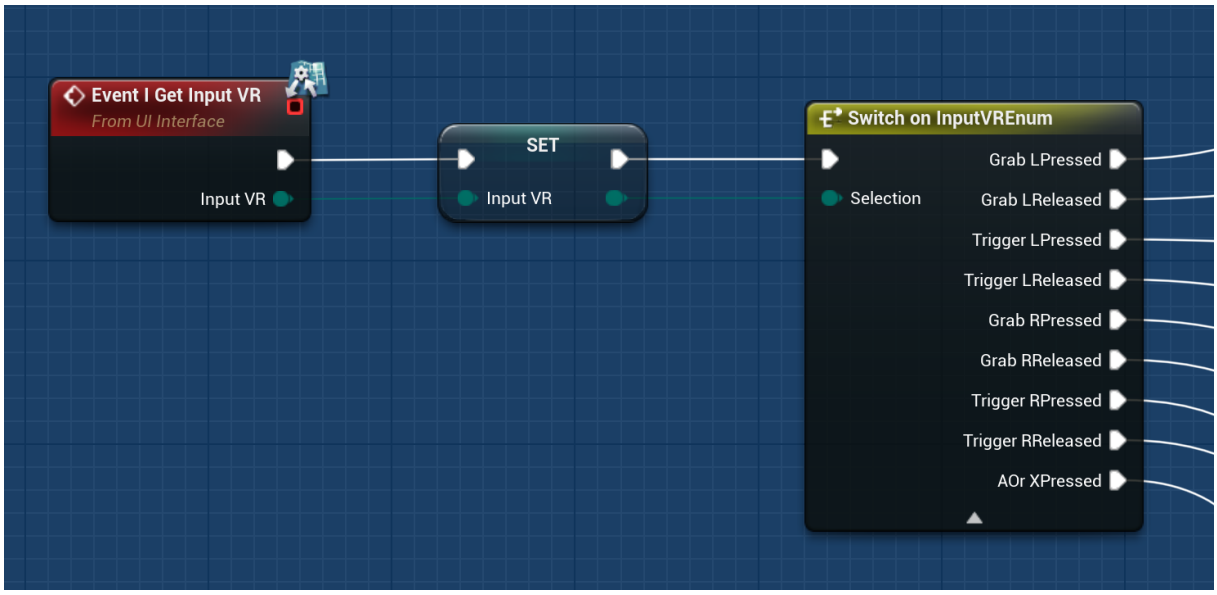


FIGURA 4.34: EVENTO *I_GET_INPUT_VR* DELLA *HUD_UI_LOGIC* CON NODO *SWITCH ON INPUTVRENUM* CHE ESEGUE IL CODICE OPPORTUNO A SECONDA DELL'EVENTO RICEVUTO IN INPUT

In questo modo la *HUD_UI_Logic* è costantemente a conoscenza di quale pulsante dei controller viene premuto e reagisce di conseguenza eseguendo il flusso di codice definito nell'opportuno pin di uscita del nodo ***Switch on InputVREnum*** mostrato in figura 4.34.

4.2.3 SELECT E GRAB INTERFACE

Nella UI Desktop ogni interazione, ad eccezione di quelle previste per l'utilizzo effettivo dei Tool, avviene tramite Widget. Per cui la maggioranza di queste quando eseguite attivano principalmente due tipi di eventi appartenenti ai Button presenti nei vari Widget che costituiscono l'interfaccia utente: *OnHovered* e *OnClicked*.
Il Collab Viewer, nella UI di default che noi andiamo a sovrascrivere, prevede lo stesso tipo di interazione anche per la UI in realtà virtuale, poiché l'interfaccia utente è quasi totalmente identica a quella per Desktop, definita dal solo *Contestual Widget* come menu. Infatti viene sfruttato un *Widget Component* aggiunto al *Motion Controller* e un *Widget Interaction Component* nell'Actor *Laser* che abilita l'interazione con questo unico widget di cui è composta l'interfaccia, per poter interagire con esso sfruttando gli stessi eventi progettati per Desktop. La UI VR da noi progettata invece si basa sia sull'interazione con dei Widget che sull'interazione con *Actor* o *Actor Component* aggiunti al *Motion Controller* e al *Player Pawn,* che sfruttano principalmente i tasti *Trigger* e *Grip* del controller.

Per l'interazione con i Widget non abbiamo dovuto aggiungere nulla a quanto fosse già stato definito dal Collab Viewer, poiché già nell'Actor **BP_MotionController** è presente un *Widget Interaction Component* che gestisce questo tipo di interazione, associando al tasto *Trigger* del controller le stesse funzioni del pulsante sinistro del mouse.
Invece l'interazione con *Actor* o *Actor Component* è stata completamente costruita da zero, insieme a tutte le varie parti della UI.
Per il corrispettivo dell'evento *OnHovered* sui Widget abbiamo usato per gli *Actor* dei *Box Collision* che mettono a disposizione gli eventi di *Begin*/*End Overlap*, che danno la possibilità non solo di controllare quando inizia o finisce una sovrapposizione tra diverse componenti, ma restituiscono anche l'*Actor* o il *Component* che ha scatenato l'evento. In questo modo risulta semplice controllare quando il laser si sovrappone al *Box Collision* e invocare di conseguenza tutti i successivi eventi.

Altre due interazioni molto frequenti nella nostra UI sono quella di **selezione** e di **presa** di un oggetto. Queste comportano che determinati Actor in alcuni momenti possano trovarsi in stati come **Selected** e **Grabbed**, situazioni abbastanza comuni in VR, e che questi debbano essere adeguatamente gestiti a seconda dell'Actor coinvolto.

Abbiamo dunque costruito due apposite Interface, ***VR Select Interface*** e ***VR Grab Interface***, che, come già visto nel capitolo **2.4**, permettono di assegnare a diverse classi le stessa funzionalità, potendola implementare in modo differente a seconda dell'*Actor* in questione.

VR Select Interface mette a disposizione solo una funzione ***Select*** e viene implementata da tutti gli Actor che, per ragioni date dal loro scopo nella UI, devono essere selezionabili e, dopo tale evento, compiere una serie di azioni, più o meno complesse, per definire il loro stato di selezionati. Questa spiegazione verrà approfondita nei capitoli successivi, non appena si analizzeranno gli Actor che implementano l'Interface in questione.
Dunque quando viene puntato col laser un *Actor* selezionabile e di seguito premuto il tasto Trigger del rispettivo controller, viene chiamato l'evento *Pressed* legato a tale pulsante che a sua volta, nella *HUD_UI_Logic* chiama l'evento ***Select,*** evidenziato in rosso in figura 4.35**.** Quest'ultimo come prima cosa, tramite la funzione ***Get Actor Overlapped,*** recupera tutti gli elementi che implementano questa interfaccia e tra quelli trovati sceglie il più vicino al laser. In questo modo si ottiene sempre l'*Actor* che l'utente intende selezionare e si richiama l'evento ***Select*** in esso contenuto che, a seconda dell'Actor selezionato, eseguirà precisi comandi.

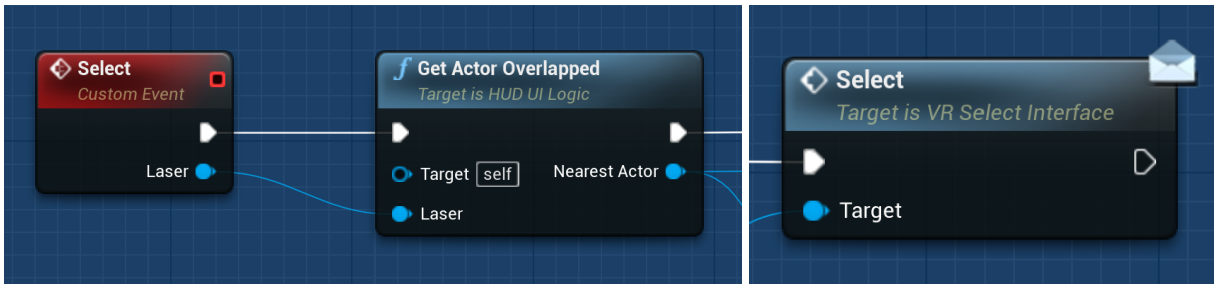


FIGURA 4.35: EVENTO *SELECT* DELLA *HUD_UI_LOGIC* CHE RECUPERA L'ACTOR IN OVERLAP E RICHIAMA IL SUO EVENTO *SELECT*

In ***VR Grab Interface*** sono invece dichiarate due funzioni: ***Pickup*** e ***Drop***, adibite rispettivamente alla presa e al rilascio di un oggetto interagibile con il tasto Grip.
Dunque, avvicinando una mano virtuale ad un *Actor* afferrabile e premendo il rispettivo tasto Grip, tenendolo premuto, viene chiamato l'evento ***Grab Actor*** nella *HUD_UI_Logic,* al quale vengono passati in input il *Capsule Collision Component* ***Grab Box*** e la ***Hand Mesh*** della mano in questione, insieme ad un valore booleano ***Is Right*** che comunica se si tratta della mano destra o sinistra.
L'evento *Grab Actor* in primis chiama la funzione ***Get Actor Near Hand*** alla quale viene passata la *Grab Box*. Tra tutti gli elementi che in quel momento fanno overlap su quest'ultima, la

funzione sceglie quelli che implementano l'interfaccia *VR Grab Interface* e poi tra questi sceglie il più vicino alla mano. Dopo aver individuato l'elemento che l'utente vuole afferrare, lo salva in una apposita variabile ***Attached Actor***, per poi richiamare il suo evento ***Pickup***, passandogli in input il *Default Scene Component* del Motion controller come oggetto a cui attaccarsi, la *Hand Mesh* e l'informazione di quale delle due mani ha eseguito l'azione.

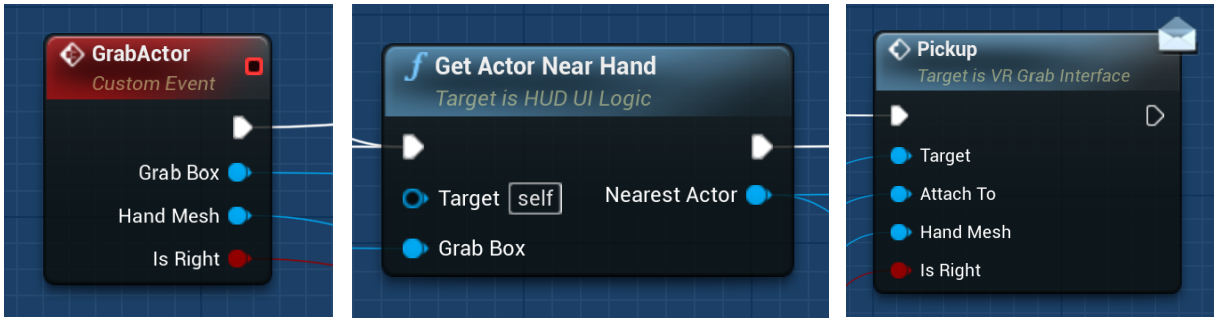


FIGURA 4.36: EVENTO *GRAB ACTOR* DELLA *HUD_UI_LOGIC* CHE RECUPERA L'ACTOR PIÙ VICINO ALLA MANO VIRTUALE E RICHIAMA IL SUO EVENTO *PICKUP*

Dopo aver preso un oggetto, al rilascio del tasto Grip, viene chiamato, sempre nell'*HUD_UI_Logic*, l'evento ***Release Actor*** adibito alla corretta gestione del rilascio dell'oggetto preso. Ad esso infatti viene passata in input ***Hand*** la Mesh della mano relativa al controller dove è stato rilasciato il tasto Grip, in questo modo l'evento può eseguire il controllo che la mano che sta rilasciando l'oggetto sia effettivamente quella che lo ha preso. Se la condizione risulta vera allora viene richiamato, dall'*Actor* preso in precedenza e salvato in *Attached Actor*, il suo evento ***Drop*** per eseguire correttamente il rilascio.

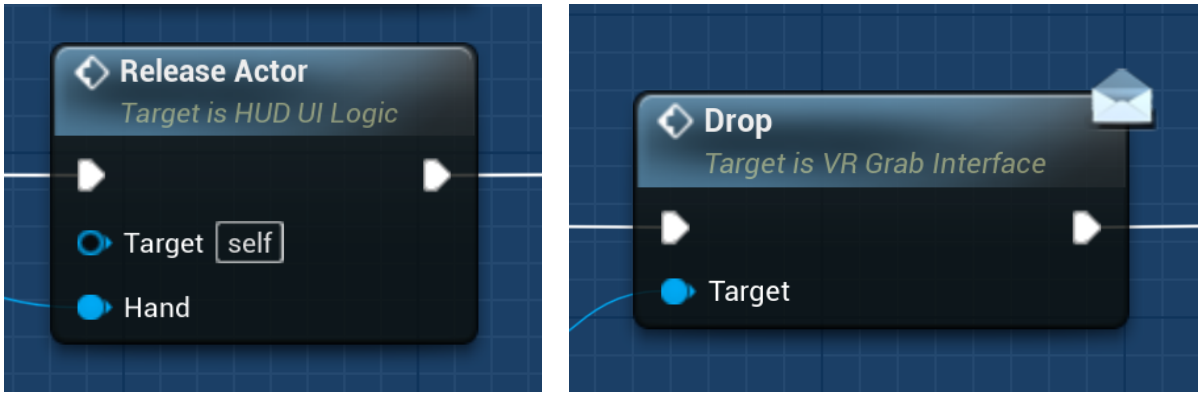


FIGURA 4.37: EVENTO *RELEASE ACTOR* DELLA *HUD_UI_LOGIC* CHE RECUPERA L'ACTOR CHE È STATO AFFERRATO E RICHIAMA IL SUO EVENTO *DROP*

Tutti i menu che compongono la UI, per poter essere presi e spostati a piacimento, implementano *VR Grab Interface;* dunque negli *Actor Tablet, Asset Dashboard e Tablet Dashboard* sono dichiarati e implementati gli eventi *Pickup* e *Drop*.

L'evento *Pickup* restituisce da *Attach To* il *Component* a cui l'*Actor Component* in questione deve essere attaccato e poi esegue il nodo ***Attach Component To Component***. In questo modo l'utente vedrà che il menu da lui preso seguirà i movimenti della mano, fino al rilascio del tasto Grip.

L'evento *Drop* si occupa di eseguire l'operazione inversa: chiamare prima il nodo ***Detach From Actor*** e di seguito il nodo *Attach Component To Component* che si occupano rispettivamente di staccare l'*Actor Componen*t del menu dalla mano e riattaccarlo al *Root Avatar* del *Pawn*, in modo che continui comunque a seguire correttamente gli spostamenti dell'utente.

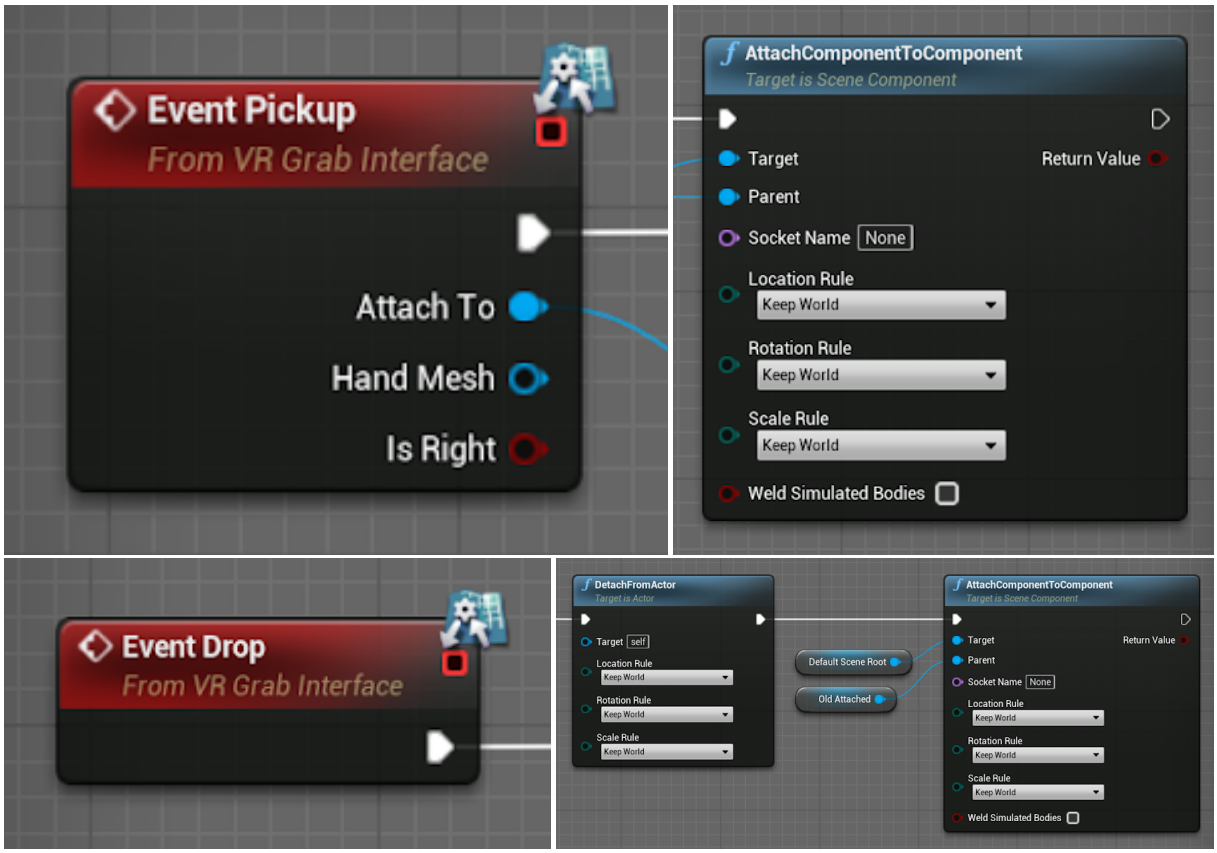


FIGURA 4.38: EVENTO *PICKUP* E *DROP* CON RELATIVI NODI CHIAVE

Nei sottocapitoli che seguono intendiamo spiegare più nel dettaglio la struttura e il funzionamento di ogni menu con le relative interazioni. Vorremmo sottolineare però che non verranno spiegati i singoli dettagli di implementazione, ma verrà eseguita piuttosto una panoramica generale del funzionamento di ogni menu, dando al lettore le informazioni necessarie alla comprensione delle scelte fatte per ognuno di essi e successivamente dei test svolti.

4.3 - MENU CONFIGURATOR

Dal Main Menu posto sul dorso della mano sinistra virtuale, una volta cliccato col tasto Trigger del controller sull'icona Configurator, è possibile aprire un tablet che permette di cambiare la configurazione del prodotto in scena.



FIGURA 4.39: SEZIONE CONFIGURATOR DEL MAIN MENU



FIGURA 4.40: TABLET CON MENU CONFIGURATOR

Per fare ciò, viene sfruttata l'*HUD_UI_Logic* per creare dinamicamente un *Actor* ***Tablet*** e attaccarlo come *Component* al *BP_VRPawn* controllato dall'utente in modalità VR. Ciò permette di rendere visibile questo menu ogni qual volta l'utente lo richiami.

4.3.1 - STRUTTURA E FUNZIONAMENTO DEL TABLET

Osservando nel *Components Tab* la struttura dell'*Actor Tablet*, ritroviamo sia degli *Static Mesh Component* utili al caricamento delle varie sezioni del modello 3D utilizzato per rappresentare graficamente il Tablet, ma anche due *Widget Component*, **Asset Tablet** e **Settings Tablet**, a cui sono associati rispettivamente un widget ***Tablet_Widget*** per la visualizzazione del *Menu Configurator* e uno ***Tablet_Widget_Settings*** per il *Menu Settings*, su cui ci soffermeremo nel sottocapitolo **4.4**.

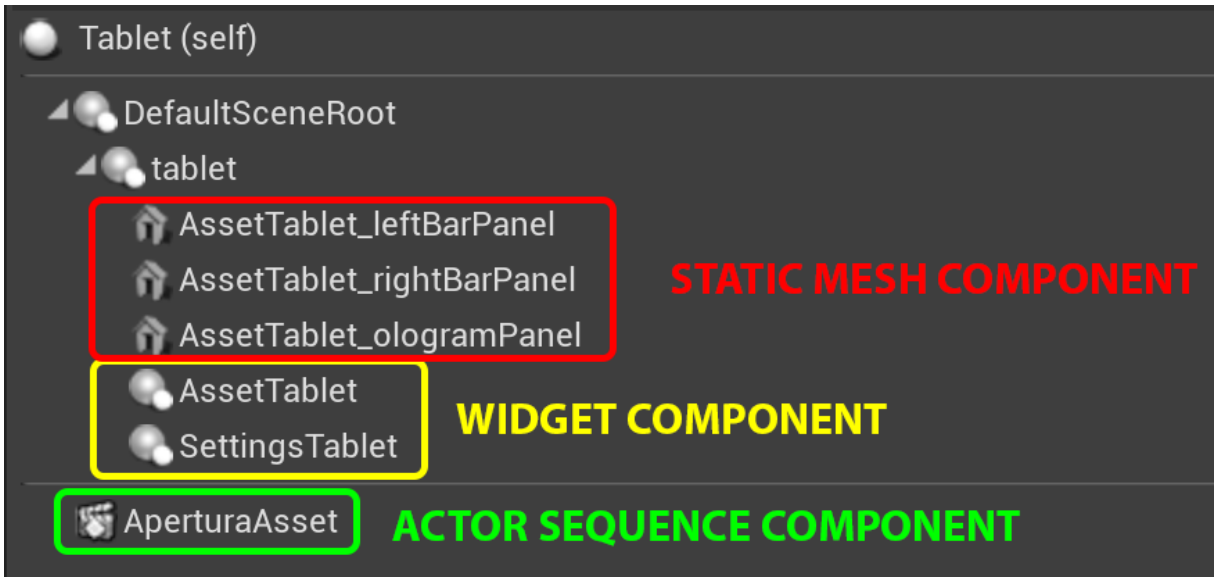


FIGURA 4.41: COMPONENTS TAB DELL'ACTOR TABLET

La presenza di due *Widget Component* per lo stesso Actor, come già detto in precedenza, è una conseguenza della nostra scelta di voler riciclare il Tablet per i due sopracitati menu, così da proporre all'utente modalità di interazione a lui già familiari. Pertanto, data l'impossibilità di coesistenza dei due diversi menu nello stesso istante per ovvie ragioni, la visibilità di uno preclude quella dell'altro. Ciò è stato implementato mediante due eventi speculari **Show Asset Tablet** e **Show Settings Tablet,** che rendono visibile il *Widget Component* d'interesse nascondendo l'altro a seconda del Menu richiesto dall'utente.
Come illustrato in figura 4.41, è stato inserito inoltre un *Actor Sequence Component*, denominato ***Apertura Asset,*** che gestisce l'animazione di apertura del Tablet sia che esso venga chiamato per visualizzare le impostazioni dell'applicativo sia che debba mostrare il menu di configurazione. Nel caso il Tablet venga aperto con un preciso menu, ma l'utente decida poi di aprire da Main Menu l'altro menu associato, l'Actor Tablet provvederà prima a far partire

l'animazione *Apertura Asset* in modalità *Reverse,* chiudendo il Tablet con il vecchio widget, per poi riaprirlo normalmente con la stessa animazione con l'altro widget visibile.

4.3.2 - TABLET WIDGET

Tutto il sistema di visualizzazione delle varie categorie e varianti è gestito dal widget ***Tablet_Widget,*** la cui rappresentazione nell'ambiente 3D è garantita dal *Widget Component* Asset Tablet che permette all'utente di interagire con il widget ad esso associato sfruttando la presenza di un *Widget Interaction Component* nel **BP_MotionController,** Actor utilizzato dal *BP_VRPawn* del Collab Viewer per rappresentare graficamente le mani virtuali dell'avatar dell'utente.

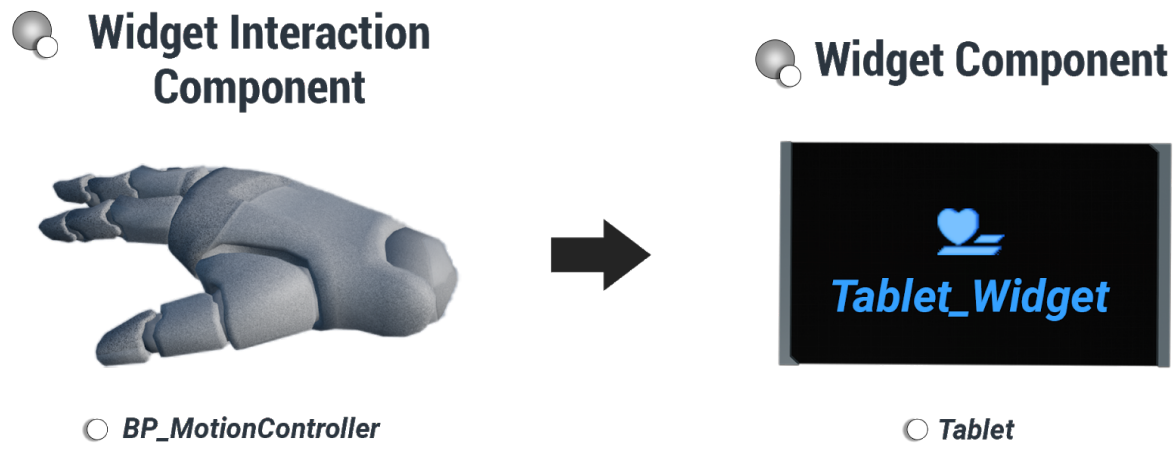


FIGURA 4.42: COMPONENT NECESSARI ALL'INTERAZIONE TRA TABLET_WIDGET E BP_MOTIONCONTROLLER

Attraverso tale widget l'utente può esplorare tutte le fasi, categorie e varianti disponibili per il prodotto in questione e scegliere quindi di conseguenza come configurarlo. Al suo interno vi sono due macro sezioni, ***Path e Categories & Variants,*** che gestiscono tutto il Menu Configurator e che vengono opportunamente aggiornate ad ogni scelta compiuta dall'utente.

La sezione *Categories & Variants* è contenuta in un *Vertical Box* ed è adibita a visualizzare tutte le fasi, categorie e varianti di tale prodotto sfruttando un *Widget Switcher* che alterna, a seconda che si debbano visualizzare delle sottocategorie o delle varianti, il layout della stessa. Inizialmente, come avviene anche per la UI Desktop, vengono visualizzate le fasi di configurazione, anche se in questo caso con la struttura del ***VR Asset Button***. Tale bottone presenta un layout, nonché una logica di creazione e funzionamento, pressoché identico al *BP Asset*

Button della *General_UI* che abbiamo già abbondantemente trattato nel sottocapitolo **3.2**, per questo ci esentiamo dal fornire qualsiasi spiegazione in merito poiché risulterebbe rindondante. Ogni volta che viene cliccato un *VR Asset Button* la sezione *Categories & Variants* si aggiorna con le nuove sottocategorie finché non vi sia la necessità di visualizzare delle varianti, caso in cui invece viene reso visibile, tramite Widget Switcher, il secondo layout e viene utilizzato il widget ***VR Leaf Button*** per rappresentarle.

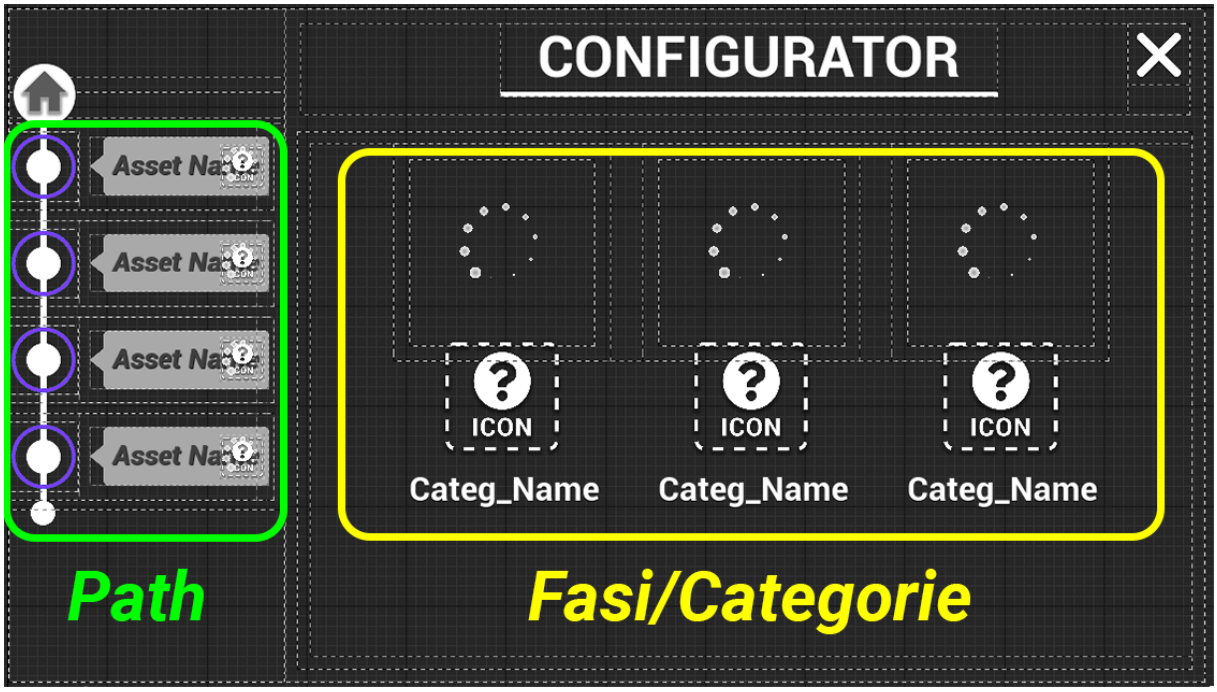


FIGURA 4.43: SEZIONI PATH E CATEGORIES & VARIANTS (PRIMO LAYOUT) DEL TABLET_WIDGET

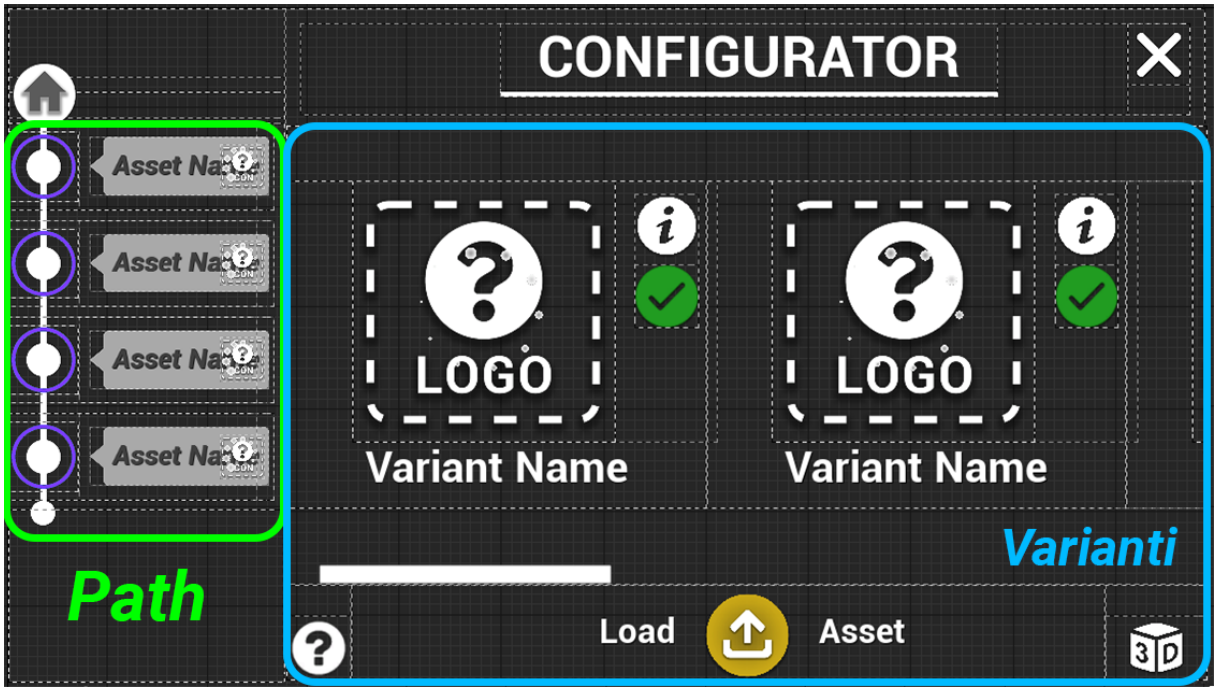


FIGURA 4.44: SEZIONI PATH E CATEGORIES & VARIANTS (SECONDO LAYOUT) DEL TABLET_WIDGET

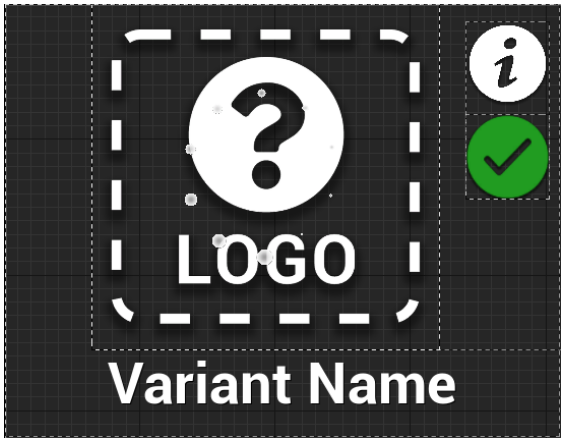


FIGURA 4.45: STRUTTURA *VR LEAF BUTTON*

Graficamente il *VR Leaf Button* presenta, come mostrato in figura 4.45, sia la solita icona e label distintive del tipo di variante, ma anche una nuova **icona di selezione verde**, inizialmente non visibile, che viene utilizzata per fornire un feedback visivo all'utente quando questo seleziona la variante, e un nuovo bottone **Info Button.** Quest'ultimo, se cliccato, permette di aprire il widget ***VR Leaf Detail*** contenente un infobox con il *Payload* associato, un bottone **Load Asset** per il caricamento della variante selezionata sul prodotto e un bottone **3D Mode** per cambiare la modalità di presentazione di tale variante, non più mostrata come *VR Leaf Button* ma rappresentata invece tridimensionalmente con la rispettiva Mesh, recuperata dal VM, e disposta su un'apposita **Dashboard**.

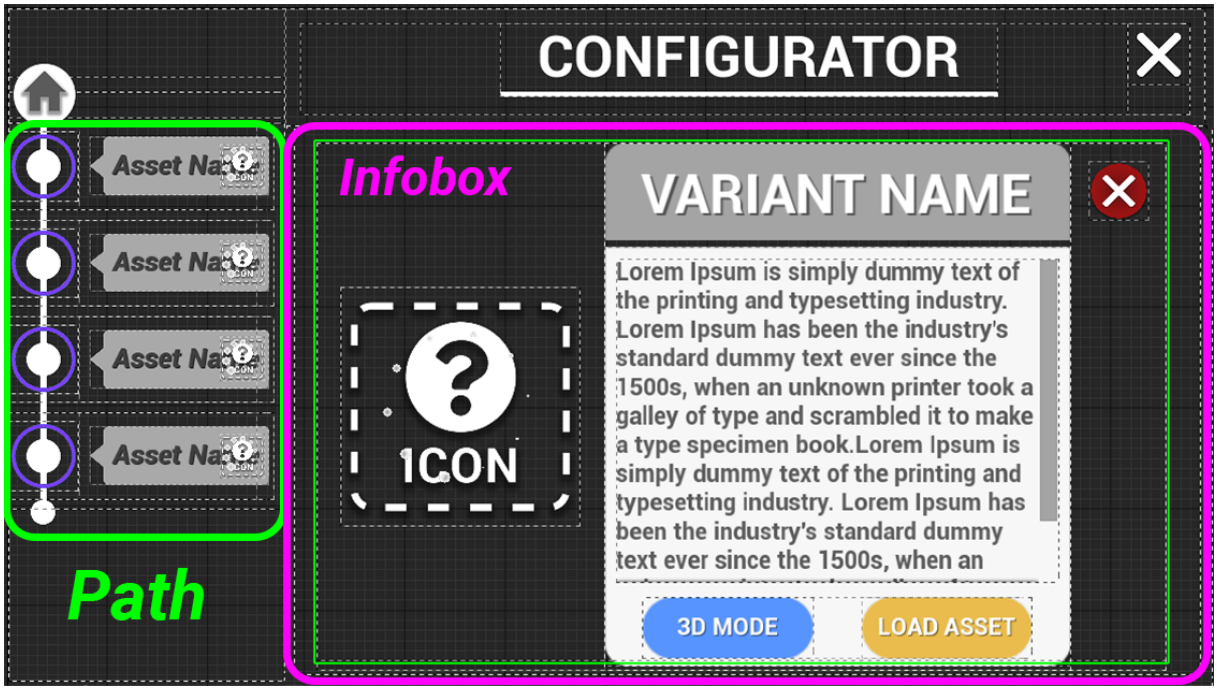


FIGURA 4.46: VR LEAF DETAIL PER LA VISUALIZZAZIONE DELL'INFOBOX DI UNA VARIANTE

Per quest'ultima funzione abbiamo riservato un opportuno sottocapitolo, precisamente il **4.3.3,** in cui illustriamo nel dettaglio il funzionamento di questa modalità "aumentata" di visualizzazione delle varianti.

Per gestire un numero imprevedibile di categorie e varianti, si è scelto di inserire in entrambi i layout della sezione *Categories & Variants* uno *Scroll Box* che distribuisse i vari *VR Asset Button* e *VR Leaf Button* in modo opportuno nello spazio disponibile, abilitando in caso di necessità una scrollbar per "scorrere" tutte le categorie o varianti in esse contenute. In figura 4.44 è possibile notare come sia presente per il secondo layout, sotto lo *Scroll Box*, una *footbar* con 3 diversi bottoni: sulla sinistra un **Tutorial Button** per la visualizzazione di un apposito tutorial sul funzionamento del tablet; al centro un **Load Asset Button** per il caricamento della variante scelta; sulla destra un **3D Button** per attivare la già citata modalità "aumentata".

L'altra macro sezione ***Path***, cerchiata in verde nelle immagini precedenti, è stata inserita per due principali motivi: aiutare l'utente a orientarsi all'interno delle varie categorie proprie di una fase, mostrando una sorta di riepilogo dei vari *VR Asset Button* cliccati durante la navigazione del menu per raggiungere tale categoria o sottocategoria; permettere all'utente di tornare alle categorie cliccate in precedenza, se non addirittura direttamente alla fase scelta, in caso volesse cambiare il tipo di varianti da visualizzare. Quindi ogni qualvolta viene cliccato un qualunque *VR Asset Button* nella sezione *Categories & Variants*, viene automaticamente aggiunto al *Vertical Box* della sezione *Path* un widget ***VR Cronology Button*** mostrante l'icona e la label della categoria o fase scelta. Tutti i widget del *Path* sono cliccabili dall'utente e permettono di aggiornare il contenuto di *Categories & Variants* richiamando la funzione *Get Child Categories and Variants* del VT_Manager con in input il Row Name della categoria specificata nella label del *VR Cronology Button*. Nel caso invece vi sia necessità di tornare direttamente alle fasi di configurazione, basterà cliccare sull' ***Home Button*** collocato sempre nel medesimo *Vertical Box* prima di tutti gli altri widget del *Path.*



FIGURA 4.47: CREAZIONE DEL VR CRONOLOGY BUTTON DELLA CATEGORIA TRI-STRATO NELLA SEZIONE PATH

4.3.3 - ASSET DASHBOARD

Come già anticipato nel precedente sottocapitolo, cliccando sul bottone *3D Mode* sia del *Tablet_Widget* che del singolo *VR Leaf Detail*, è possibile aprire una Dashboard che permette di visualizzare in 3D le varianti della fase scelta. In questo caso, viene coinvolto l'Actor ***Asset Dashboard*** che viene aggiunto come *Component* al *BP_VRPawn*, in modo che possa seguire i movimenti dell'utente e rimanere quindi sempre agganciato all'avatar ad esso associato, come spiegato nel sottocapitolo **4.2.2**. Nella figura 4.48 che segue abbiamo riportato graficamente la struttura del suddetto Actor, evidenziando in blu tutti gli *Static Mesh Component* impiegati e in rosso tutti i *Child Actor Component*, ad ognuno dei quali è stata associata una precisa funzionalità.

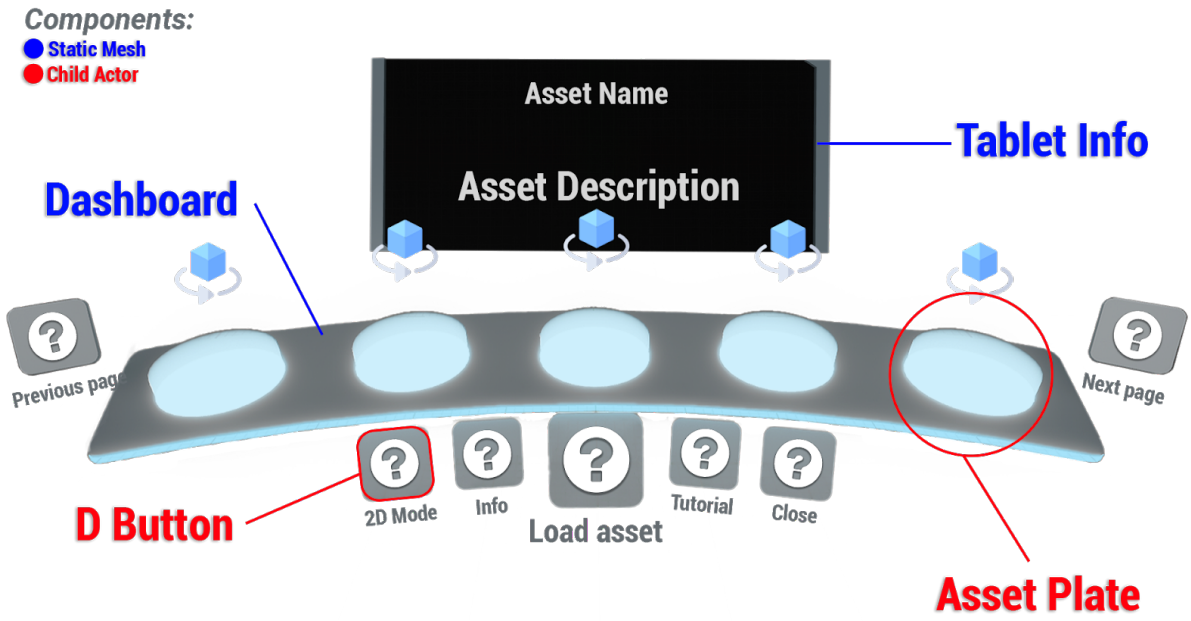


FIGURA 4.48: STATIC MESH E CHILD ACTOR COMPONENT UTILIZZATI NELL'ASSET DASHBOARD

La funzione degli Actor ***Asset Plate*** è quella di contenere i modelli 3D delle varianti della fase attiva. Tutte le varianti che devono essere visualizzate su Dashboard vengono memorizzate in un Array **Variants** di *Variant Entry* nel *Tablet_Widget* che poi viene passato in input ad un *Event Dispatcher*, denominato **Spawn Variants,** chiamato sempre dal *Tablet_Widget*. Nell'*Asset Dashboard* viene poi eseguito il rispettivo Bind in cui vengono chiamati una serie di eventi che, per ogni variante ricevuta, permettono di recuperare da VM la *Static Mesh* ad essa associata e le relative proprietà. Ad ogni variante viene quindi associato un diverso Asset Plate, univocamente identificato mediante un indice interno, che ne garantisce la corretta visibilità.

Tuttavia, dato il numero esiguo di Asset Plate disponibili in relazione a quello delle possibili varianti da visualizzare, tale sistema si è rivelato presto limitante. Per questo si è scelto di inserire nella Dashboard un sistema di suddivisione delle varianti per pagine, che permetta, a prescindere dal loro numero effettivo, di renderle visibili a gruppi di 5 per volta. Questa suddivisione per pagine nella pratica comporta l'inserimento di più varianti nello stesso Asset Plate, o più precisamente, all'aggiunta di più *Static Mesh* al *Scene Component* ***Variant_Component*** adibito al contenimento delle varianti.

L'evento **Show Variants** dell'*Asset Dashboard* possiede proprio questo compito: leggere le prime cinque varianti dall'apposito Array, recuperare da VM le Mesh associate ad ognuna di esse e aggiungere quest'ultime ai vari *Variant_Component* di ogni *Asset Plate*, infine ripetere, se necessario, il ciclo appena descritto fin tanto che non state distribuite tutte le varianti. A seconda del numero di volte con cui viene ripetuto tale ciclo e dalla lunghezza stessa dell'Array *Variants*, cambierà il numero delle Static Mesh inserite in ogni *Asset Plate*. Questo ci permette di avere sempre in ogni *Asset Plate* la situazione in cui: il primo *Static Mesh Component* inserito rappresenti una delle prime cinque varianti recuperate e quindi destinato ad essere visualizzato nella prima pagina; il secondo, aggiunto al secondo ciclo di ripartizione delle varianti, visualizzabile nella seconda pagina e così via. Inizialmente infatti, nella prima pagina, vengono rese visibili tutte le prime varianti inserite e nascoste le altre presenti nel *Variant_Component*. L'*Asset Dashboard* mette poi a disposizione un evento ***Next Page*** e uno ***Previous Page*** che, a seconda del numero della pagina attualmente aperta, provvedono a cambiare la visibilità di ogni variante in ogni *Asset Plate* in base al loro ordine d'inserimento e al nuovo numero di pagina. Per chiarire meglio il concetto, nell'immagine 4.49 abbiamo inserito uno schema riassuntivo della logica seguita prendendo come esempio il caso di 8 varianti da visualizzare.

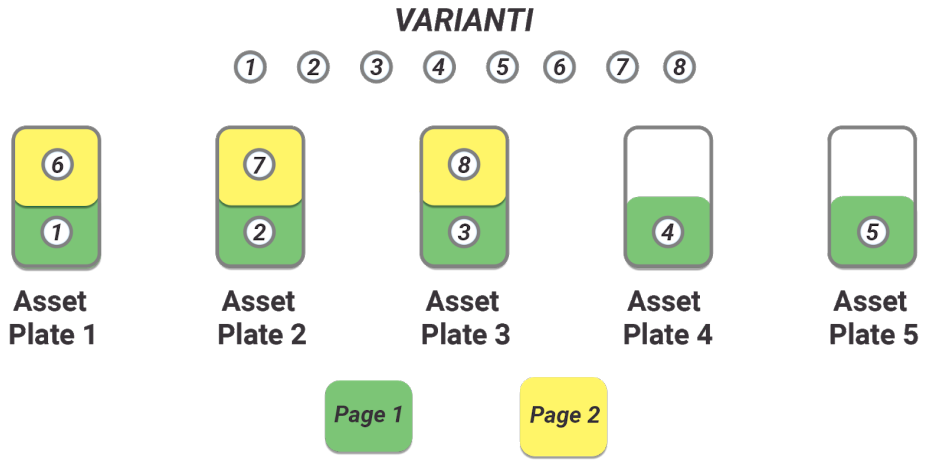


FIGURA 4.49: ILLUSTRAZIONE DEL FUNZIONAMENTO DEL SISTEMA DI VISUALIZZAZIONE DELLE VARIANTI PER PAGINA

Andando ad analizzare la struttura del singolo *Asset Plate* notiamo subito la presenza di: l'ormai noto *Variant Component*; un *Box Collision*, posto in concomitanza della posizione assegnata alla singola variante, utile alla rilevazione del passaggio del puntatore Laser sopra l'area circoscritta dal Box e di conseguenza sopra la variante; un *Widget Component* ***Variant Label,*** riservato alla visualizzazione di una opportuna label con il nome della variante attualmente visibile; due *Actor Sequence Component*, utilizzati per animare il *Variant Component* a seguito della selezione della variante associata da parte dell'utente.

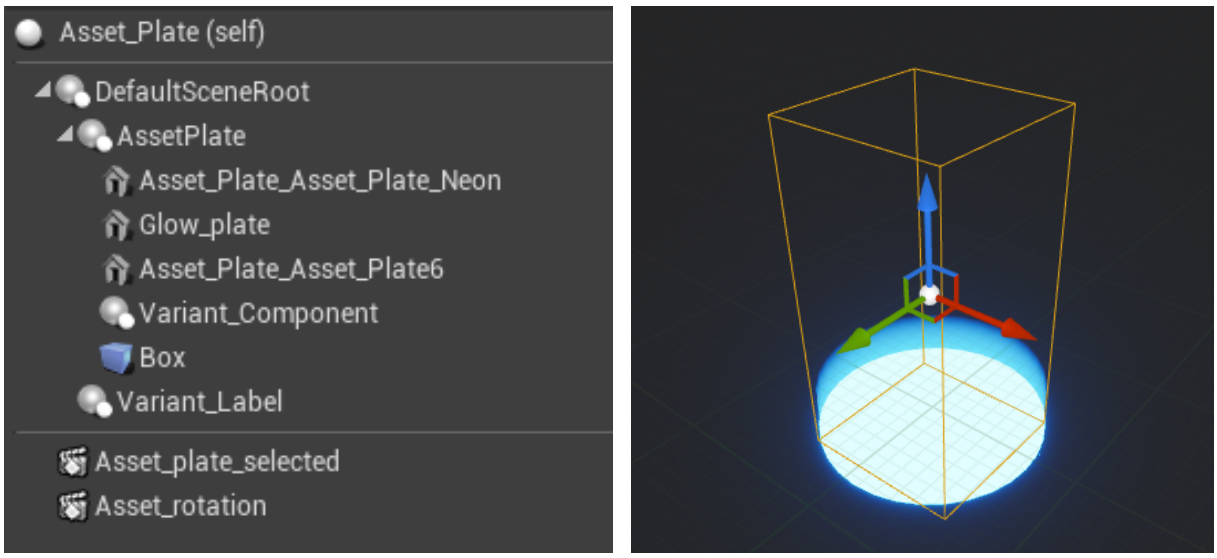


FIGURA 4.50: COMPONENTS TAB DELL'ASSET PLATE E RAPPRESENTAZIONE GRAFICA DEL RELATIVO COLLISION BOX

Per selezionare una variante sull'*Asset Dashboard* l'utente dovrà indirizzare il proprio laser verso la variante scelta e cliccare col tasto Trigger del controller. Fatto ciò vengono attivate le due animazioni **Asset plate selected** e **Asset rotation** che cambiano la colorazione del neon dell'*Asset Plate* cliccato e attivano un'animazione di rotazione della variante in esso contenuta, questo per mettere in evidenza quale variante è attualmente attiva e quindi pronta per essere caricata sul prodotto. Allo stesso tempo viene dato modo all'utente di ispezionare da tutte le angolazioni la Mesh della suddetta variante così da fornirgli ulteriori elementi per compiere la sua scelta.



FIGURA 4.51: SELEZIONE DI UNA VARIANTE SU ASSET DASHBOARD

Altro Actor fondamentale per il funzionamento dell'*Asset Dashboard* è il ***D Button*** che, come visibile in figura 4.52, viene utilizzato per attivare delle specifiche funzionalità. Lateralmente alla Dashboard vi sono due *D Button* che, se cliccati, svolgono la funzione di cambio pagina, richiamando i già citati eventi *Next Page* e *Previous Page*. Al centro della Dashboard troviamo invece una serie di *D Button* disposti a semicerchio che mettono a disposizione: una funzionalità **2D Mode,** che permette di ritornare alla visualizzazione delle varianti tramite Tablet, chiudendo di conseguenza la Dashboard; un'opzione **Info,** che apre un Tablet più piccolo riservato alla visualizzazione del *Payload* della variante attiva; la funzionalità **Load Asset,** disposta centralmente e identica a quella presente nel *Tablet_Widget;* un tutorial sul funzionamento della Dashboard attivabile dall'opzione **Tutorial,** nel caso l'utente non riuscisse a capire le corrette modalità d'interazione; l'opzione **Close** di chiusura del Menu Configurator e di conseguenza della relativa Dashboard.

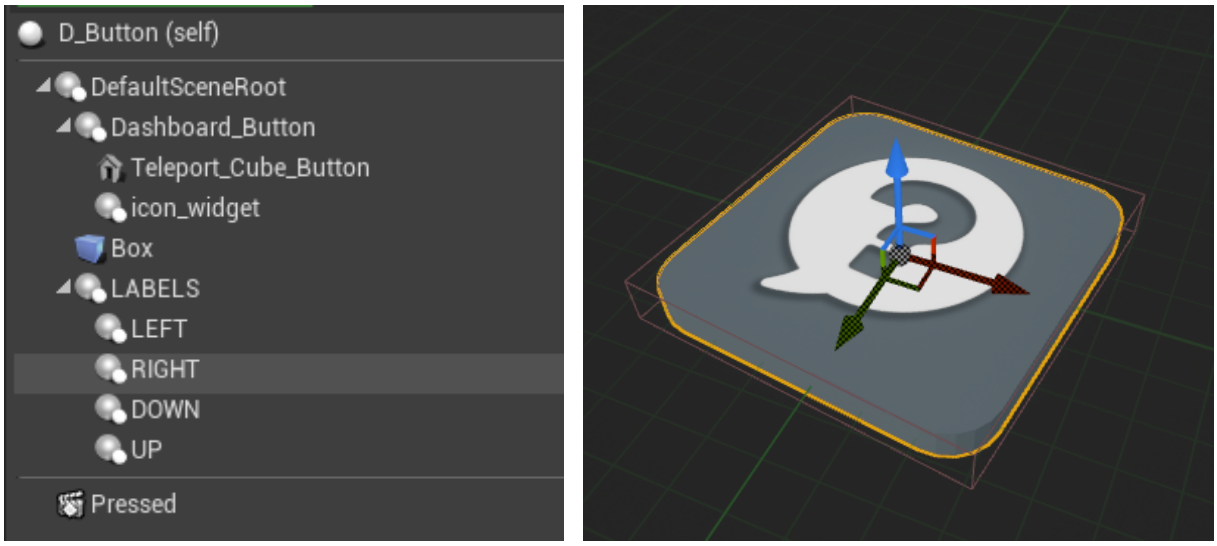


FIGURA 4.52: COMPONENTS TAB DEL D BUTTON CON ESEMPIO DI UN D BUTTON TUTORIAL

Anche per questo Actor è presente un *Box Collision* per controllare quando il *D Button* viene intercettato dal Laser dell'utente e far partire pertanto sia l'animazione **Pressed** associata**,** a seguito del click sul tasto Trigger, che tutti gli eventi necessari a richiamare la funzionalità da esso controllata.

Il *Scene Component* **LABELS** gestisce invece le diverse posizioni assumibili dalla label della funzionalità attorno al bottone: a sinistra, a destra, sopra o sotto di esso. Ciò si è reso necessario dal momento che lo stesso Actor viene utilizzato in parti diverse della Dashboard, infatti vi sono casi in cui la posizione assunta dalla label possa interferire con altri elementi della UI ed è quindi indispensabile spostarla in una delle altre tre posizioni prefissate in modo da renderla facilmente leggibile.

Al *Widget Component* ***Icon Widget*** di ogni D Button viene associato un diverso Widget contenente l'icona rappresentante la funzionalità assegnata, customizzabile dall'utente mediante l'Editor Customize UI. Sono stati costruiti quindi 7 Widget diversi, uno per ogni funzionalità, al cui interno sono presenti i necessari *Event Dispatcher* che notificano all'*Asset Dashboard* il click eseguito dall'utente su di essi e quindi la necessità di invocare tutti gli eventi propri del tipo di funzionalità chiamata, oltre che al *D Button* di far partire l'animazione *Pressed* in esso descritta come reazione al click.

Per inizializzare correttamente ogni *Icon Widget* con il giusto widget all'apertura della Dashboard, all'evento *Begin Play* dell'*Asset Dashboard* viene chiamata la funzione ***Initialize D Buttons*** in cui vengono fissate le posizioni di tutte le label di tutti i *D Button* e vengono assegnati i vari Widget ai rispettivi bottoni con le corrette icone.

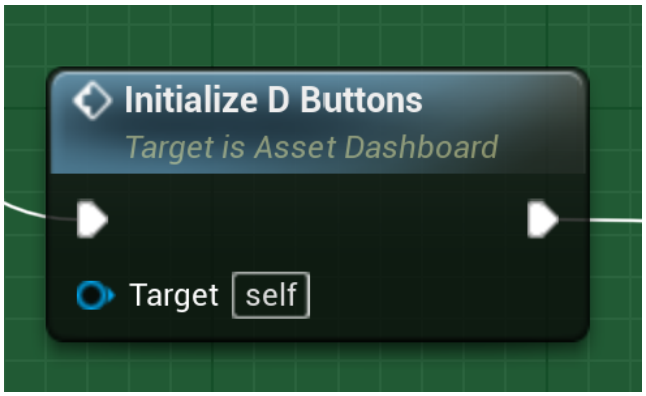


FIGURA 4.53: FUNZIONE *INITIALIZE D BUTTONS*

In generale al momento sono customizzabili tutte le icone dei vari widget associati ai D Button, la colorazione dei neon inseriti nella *Static Mesh* della Dashboard e nel singolo *Asset Plate* e infine anche il secondo colore assunto dai neon dell'*Asset Plate* quando viene selezionata una variante. Questo grazie all'inserimento in ogni Actor e Widget customizzabile di una istanza della *HUD_UI_Logic,* dalla quale è possibile recuperare tutte le informazioni necessarie salvate su *Data Asset*.

4.4 - MENU SETTINGS



FIGURA 4.54: TABLET CON MENU SETTINGS

Una volta selezionato il *Menu Settings* dal *Main Menu*, viene aperto lo stesso Actor *Tablet* utilizzato per il *Menu Configurator*, e in questo caso viene reso visibile il *Widget Component* ***Settings Tablet*** e quindi il relativo ***Tablet_Widget_Settings***.

Come già anticipato nel sottocapitolo **3.4** sul *Menu Settings* della versione Desktop, è stata implementata attualmente solo la sezione ***Navigation*** nella quale, identicamente a quanto fatto per la *General_UI*, vengono visualizzate le modalità di navigazione del Collab Viewer. Cliccando su una di queste l'utente potrà tornare velocemente alla versione Desktop con la modalità di navigazione scelta senza perdere però tutte le modifiche apportate al prodotto o all'ambiente fino a quel momento.

4.5 - MENU TOOLS

Similmente a quanto avviene per i menu Configurator e Settings, cliccando con il tasto Trigger sull'icona Tools del Main Menu è possibile aprire una Dashboard con tutti gli strumenti già visti nella versione Desktop, utili a poter agire sul prodotto e sull'ambiente.



FIGURA 4.55: SELEZIONE DEL MENU TOOLS DAL MAIN MENU E RELATIVA TOOLS DASHBOARD

La ***Dashboard*** si presenta come un ripiano semicircolare che, alla sua apertura, è posto di default di fronte all'utente all'altezza dei fianchi. Al centro sono presenti 4 piastre aventi sopra di esse 4 oggetti: un *guanto*, una *penna*, un *metro* e un *puntatore laser;* che rappresentano rispettivamente i Tool *Move*, *Paint*, *Meter* e *X-Ray*.
Nella parte inferiore del ripiano principale è presente una sfera che se cliccata permette di aprire un cassetto, dentro il quale sono riposti due pulsanti che attivano i due Tool rimanenti: *Environment e Teleport*.
Lateralmente sono stati aggiunti due piccoli sottoripiani anch'essi con dei pulsanti adibiti a funzioni di utility, come la chiusura della *Dashboard* o alla richiesta di un Tutorial sul funzionamento della stessa.
Tutti i pulsanti presenti nella *Dashboard* sono degli *Actor* di tipo ***D Button***, la cui struttura e funzionamento sono stati descritti nel sottocapitolo **4.3.3**.

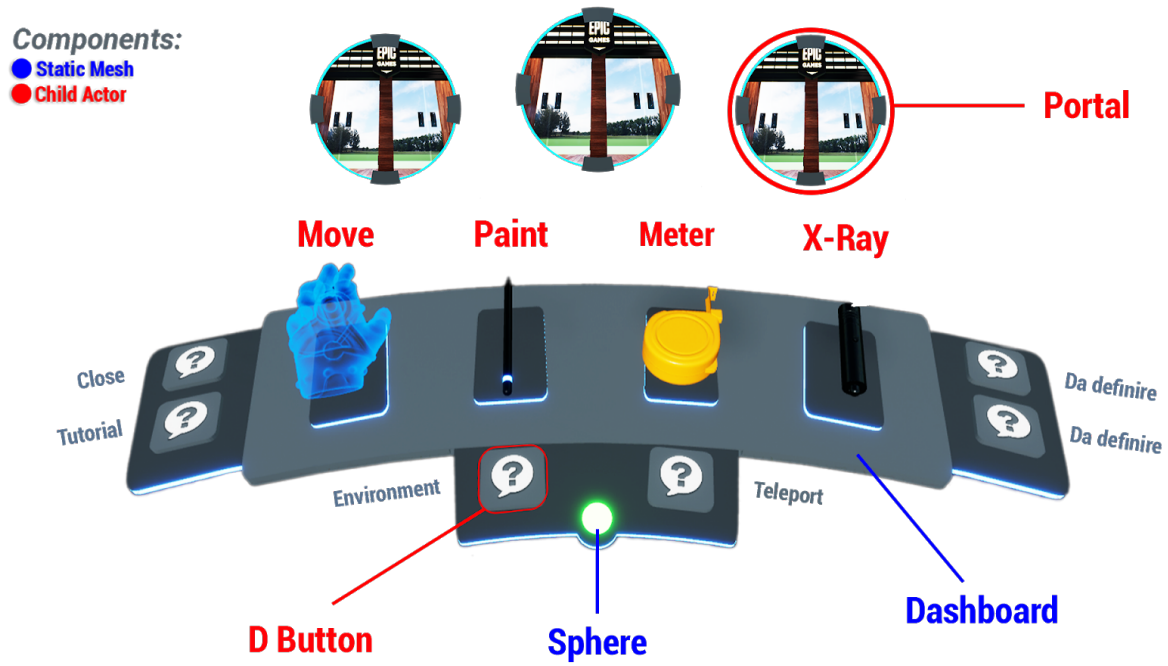


FIGURA 4.56: STATIC MESH E CHILD ACTOR COMPONENT UTILIZZATI NELLA TOOLS DASHBOARD

L'interazione con gli oggetti rappresentanti i Tool si compone di due fasi: la prima, in cui il Tool interessato deve essere selezionato puntando il laser verso di esso e cliccando poi il tasto Trigger; la seconda, in cui è necessario afferrare l'oggetto selezionato, avvicinando la mano a quest'ultimo e premendo il tasto Grip. Per tenere in mano un Tool è fondamentale che il pulsante Grip rimanga premuto per tutta la sua durata d'utilizzo, come nella realtà è necessario mantenere la presa su un oggetto affinché possa essere utilizzato. Per terminare l'utilizzo è sufficiente rilasciare la presa e il Tool tornerà nella sua postazione sulla Dashboard.

Per permettere la selezione dei Tool, è stato posizionato sulla *Dashboard*, per ognuno di essi, un *Collision Box*, in modo da poter rilevare sia l'Overlap del laser, rendendo visibile al suo passaggio la relativa label, che per selezionare effettivamente il Tool.
La *Dashboard* riesce a gestire correttamente la selezione poiché implementa l'interfaccia *VR Select Interface* e quindi anche la funzione ***Select*** in essa definita, che ha il compito di recuperare correttamente il Tool selezionato e impostare la variabile ***Is Selected*** dell'*Actor* del Tool su *True*.
Ogni Tool è descritto da un *Actor* differente ma, a parte alcune accortezze necessarie al corretto funzionamento di ognuno di loro, la logica generale di funzionamento è la stessa, per cui tutto ciò che viene spiegato ora per un singolo Tool vale anche per tutti gli altri.

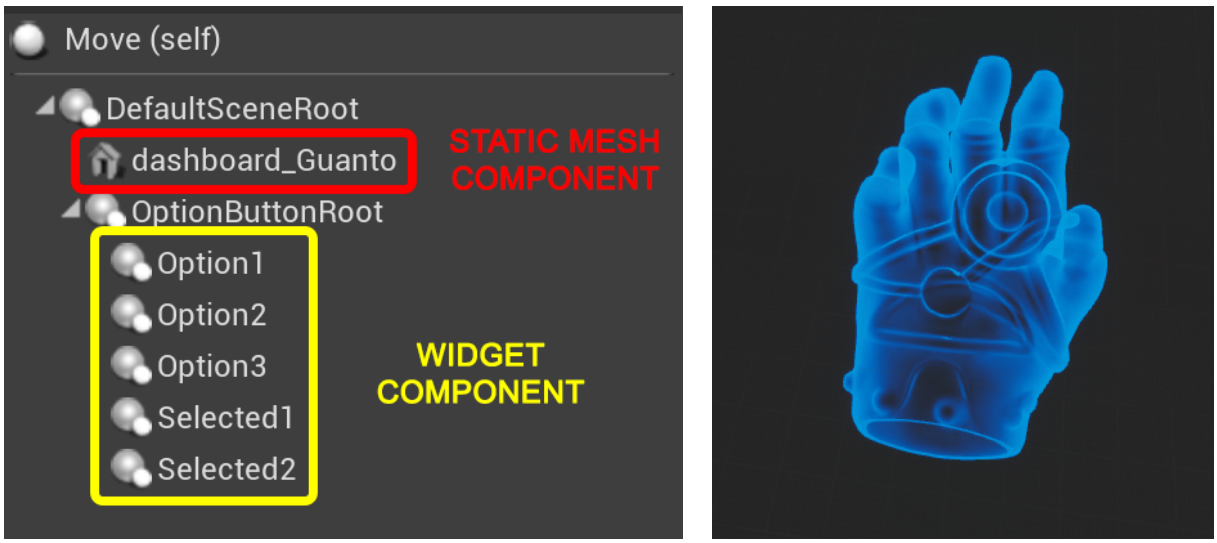


FIGURA 4.57: COMPONENT DEL TOOL MOVE E RELATIVA RAPPRESENTAZIONE

Ogni Tool, dopo essere stato selezionato, può essere afferrato, in quanto ogni *Actor* associato implementa la *VR Grab interface.*
L'evento *Pickup* in primis si occupa di attaccare la Mesh che rappresenta il Tool alla mano che lo ha afferrato. La Mesh della mano viene quindi resa invisibile, in modo da dare all'utente l'impressione che lo strumento afferrato abbia effettivamente sostituito la mesh di questa. Attraverso poi una catena di *Event Dispatcher* viene notificato quale Tool è stato appena afferrato dall'utente, in modo da recuperare correttamente le sue Option e crearne i relativi Widget.
Così come era stato progettato nei mockup, le Option di un Tool possono essere attivate cliccando con il tasto Trigger sui Widget assegnati a queste, che vengono in seguito inseriti in appositi *Widget Component* posizionati attorno alla Mesh che rappresenta il Tool. Una volta scelta una Option, tutti i Widget Component vengono nascosti e viene reso visibile solo quel-

lo adibito a notificare all'utente quale Option è stata scelta ed è dunque attiva in quel momento. Questo Widget, posto normalmente al centro dell'oggetto, serve anche per poter tornare indietro alle Option disponibili, disattivando quella scelta in precedenza. Le funzioni e le catene di *Event Dispatcher* che regolano il funzionamento dei Tool in VR sono del tutto simili a quelle descritte per la versione Desktop e dunque non verranno ri-spiegate.

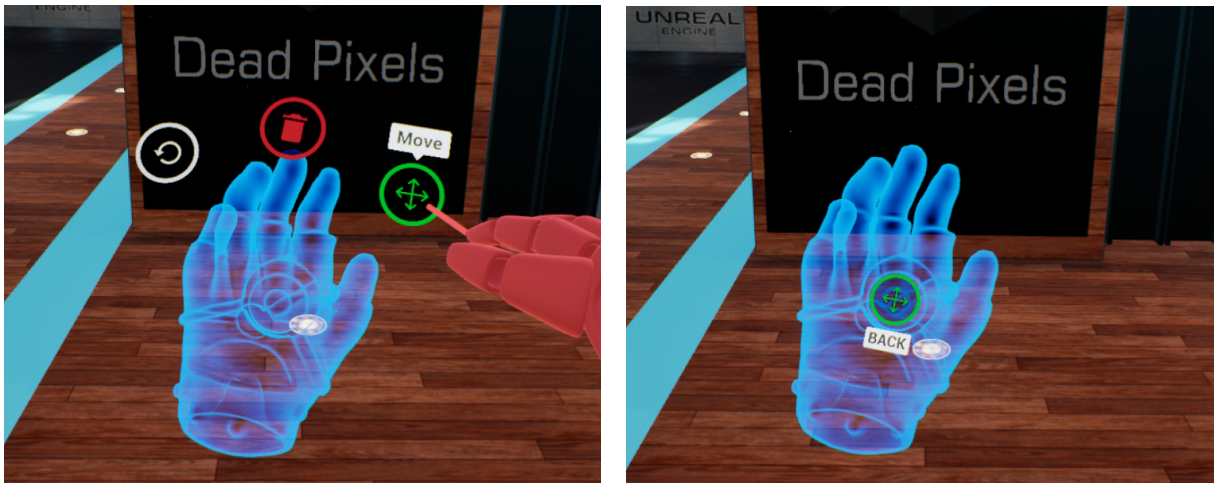


FIGURA 4.58: TOOL MOVE CON LE OPZIONI VISIBILI E CON WIDGET DI NOTIFICA DELL'OPZIONE ATTIVA

Attivata una Option l'utente sarà libero di usare il Tool scelto cliccando sul tasto Trigger, allo stesso modo di come avrebbe usato il tasto sinistro del mouse nell'interazione bidimensionale. Al rilascio del tasto Grip viene chiamato l'evento *Drop*, che si occupa di riposizionare la Mesh del Tool nella sua posizione sulla Dashboard, disattivare il funzionamento del Tool e rendere nuovamente visibile la Mesh della mano fino ad ora sostituita dallo strumento attivo.

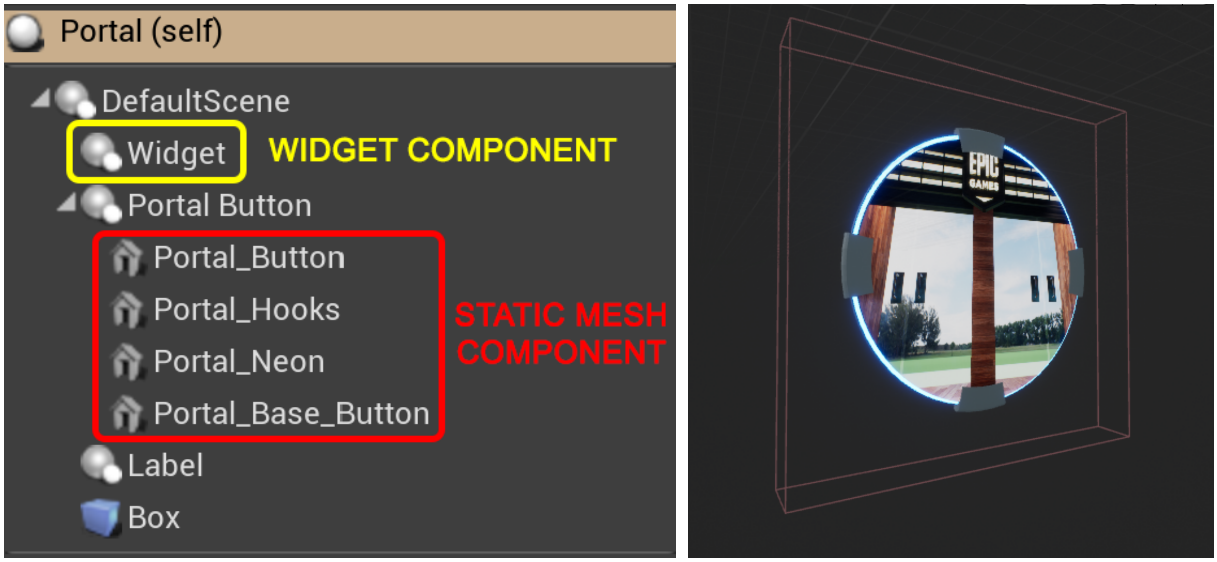


FIGURA 4.59: STRUTTURA DELL'ACTOR PORTAL E RELATIVA RAPPRESENTAZIONE

L'interazione per quanto riguarda i Tool *Teleport* ed *Environment* è stata gestita diversamente. Essi sono attivabili attraverso due appositi *D Button* posti nel cassetto della *Dashboard* ed entrambi mostrano le loro Option attraverso dei portali che si aprono semi circolarmente attorno all'utente. Gli *Actor* ***Teleport*** ed ***Env,*** incaricati di gestire i due Tool sopracitati, sono composti da diversi *Actor Component* a forma di portale. L'*Actor* ***Portal*** è la classe che descrive questi portali ed è composta dalla Mesh dell'oggetto, da una *Label* che si aggiorna con il nome della Option che deve rappresentare, da un *Widget Component*, dentro al quale viene inserito il Widget assegnato alla Option, e da un *Collision Box* per gestirne l'interazione (vedi figura 4.59). Pertanto alla selezione di uno di questi Tool vengono recuperate le relative Option, a partire dalle quali è possibile costruire i corrispettivi Widget che vengono poi inseriti negli *Actor Component* di tipo *Portal* presenti negli Actor che descrivono i Tool.
A questo punto l'utente dovrà solo puntare e selezionare l'Option desiderata per attivare una animazione che simulerà la sua entrata nel portale, eseguendo parallelamente una transizione di ambientazione o spostamento nel punto dell'ambiente scelto.

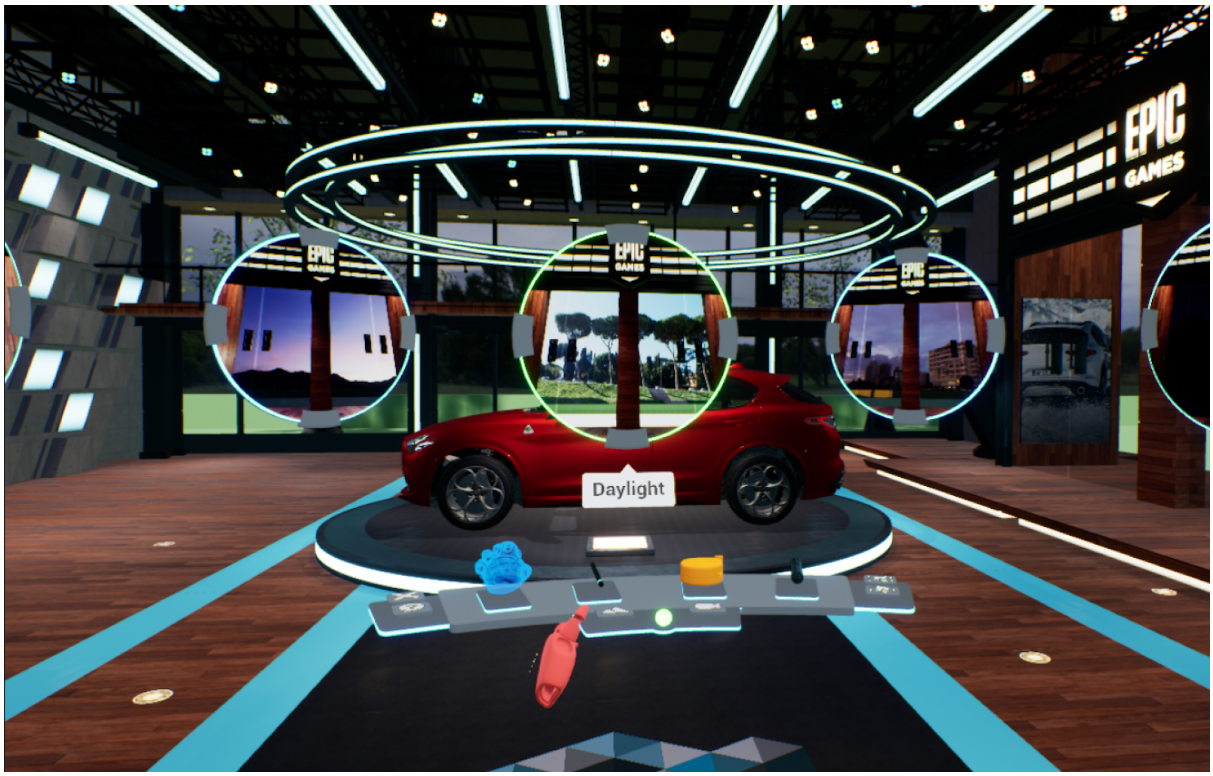


FIGURA 4.60: TOOL ENVIRONMENT CON I DIVERSI ACTOR COMPONENT PORTAL RELATIVI ALLE SUE OPTION

5 - TEST

5.1 - PERFORMANCE TEST CON DIVERSE TIPOLOGIE DI PRODOTTO

Dal momento che il plugin per cui stiamo implementando le varie interfacce è progettato per essere modulare, quindi in grado di ospitare qualsiasi genere di prodotto lo sviluppatore o il designer scelgano di inserire, ci siamo subito preposti di testare l'usabilità del plugin, e quindi anche delle nostre interfacce, inserendo Mesh diverse sia per tipologia che per complessità poligonale. Ovviamente, ogni qualvolta è stato inserito un diverso tipo di prodotto, si è proceduto a costruire il rispettivo *Variant Manager* con le opportune fasi di configurazione, categorie e varianti; nonché la relativa *Data Table* con tutte le informazioni necessarie al popolamento del Menu Configurator presente nelle due UI del nostro applicativo.
Inizialmente abbiamo scelto di creare un configuratore di **figure geometriche semplici**, con un numero di poligoni estremamente basso, per capire se le interfacce da noi progettate riuscissero, a prescindere da come era stato implementato tutto il sistema di catalogazione delle varianti, a recuperare correttamente tutti i dati della Data Table e tutte le varie Mesh dal VM. Come previsto, le UI vengono inizializzate correttamente dall'applicativo, il quale riesce anche a gestire fluentemente il configuratore in runtime, sia nella versione Desktop che in quella VR. In questo caso la facilità e la velocità con cui l'applicativo ha eseguito l'operazione di recupero e visualizzazione nell'ambiente 3D delle diverse varianti sono giustificate sia dal fatto che fosse stato progettato un VM molto semplice nella struttura, con poche fasi e categorie, sia dalla bassa complessità poligonale di ogni Mesh costituente il configuratore.

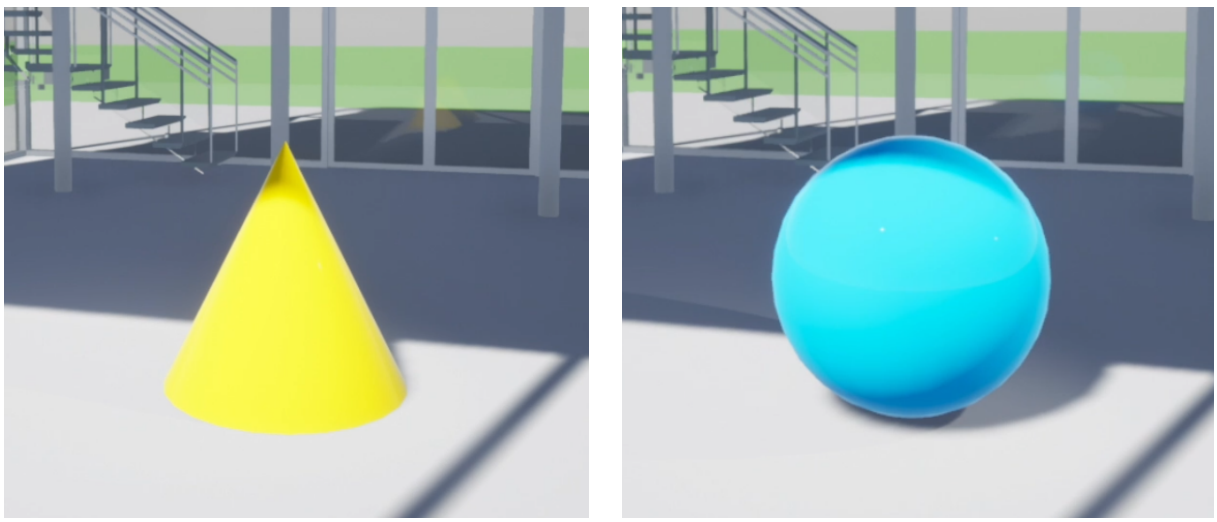


FIGURA 5.1: ESEMPIO DI FIGURE GEOMETRICHE USATE PER TESTARE IL CONFIGURATORE

Si è così deciso di inserire un modello semplificato di una **Lamborghini Huracàn**, decisamente più dettagliato delle precedenti figure geometriche, e di costruire un VM molto più articolato, con numerose fasi di configurazione e sottocategorie. In questo caso inizialmente l'applicativo ha faticato a caricare le varie Mesh delle varianti sull'Asset Dashboard, creando diversi crash di sistema e bloccando quindi l'esecuzione del programma. Il problema è stato risolto ottimizzando quella parte di codice governante l'operazione di recupero delle Mesh da *Variant Manager* e cambiando la logica di ripartizione delle varianti nei vari Asset Plate, nonché la medesima già trattata nel sottocapitolo **4.3.3**. Dopo aver apportato tali modifiche, si è osservato un deciso aumento delle performance dell'applicativo, il quale riusciva tranquillamente a gestire tutto il Menu Configurator e a caricare sull'Asset Dashboard le Mesh relative ad ogni variante della Lamborghini.



FIGURA 5.2: UI DESKTOP UTILIZZATA NEL USER TEST



FIGURA 5.3: ASSET DASHBOARD CON LE VARIANTI DELLA CARROZZERIA DELLA LAMBORGHINI

Alla luce di questi risultati, si è passati all'inserimento di un modello di autoveicolo ancora più complesso, con circa 6 milioni di vertici, per testare l'efficienza delle nostre interfacce in un contesto semi-professionale, vicino a quello che si avrebbe per un determinato cliente nel mondo Automotive. È stato utilizzato un modello di **Alfa Romeo Stelvio Quadrifoglio** estremamente dettagliato**,** al quale è stato parallelamente associato un configuratore fedele a quello messo a disposizione dalla stessa casa automobilistica sul proprio sito web[41]. Una volta completata la costruzione del configuratore, abbiamo notato un leggero e prevedibile calo di performance per la versione Desktop ed uno invece sostanziale per la versione VR. In questa modalità infatti l'applicativo faticava a processare la grande quantità di informazioni derivante dalle varie riflessioni della luce ambientale sul modello, portando ad una diminuzione della frequenza di aggiornamento del visore e causando quindi lag che rendevano difficile e frustrante, se non impossibile, l'interazione con le interfacce utente VR. Il motivo di ciò si deve molto probabilmente sia all'eccessiva complessità del modello adottato e del nuovo ambiente introdotto, che agli oggettivi limiti della GPU del PC utilizzato per testare l'applicativo. Esso infatti dispone di una scheda video *Nvidia GeForce GTX 1060* che, secondo quanto specificato nelle linee guida dell'Oculus Rift, è la minima richiesta per poter utilizzare tale visore. Non avendo quindi l'opportunità di testare l'applicativo su un dispositivo più performante, si è deciso di utilizzare nella fase di user testing il modello della Lamborghini e un ambiente più semplice, così da poter garantire ai tester un'interattività più fluida durante

l'esperienza e dare modo a noi di poterci concentrare esclusivamente sui test da svolgere. Nonostante tale deficit tecnico del nostro computer, che rendeva impossibile eseguire i test in VR sulla Stelvio, la versione Desktop con questo modello risultava comunque funzionante e comodamente utilizzabile anche dai tester.
In seguito abbiamo comunque provato a ridurre la complessità del modello a 3 milioni di vertici e a cambiare tutta l'illuminazione ambientale, passando da un sistema di shading di tipo **Deferred,** usato di default da Unreal Engine per renderizzare luci e materiali, ad uno **Forward** più performante per applicativi VR. In questo nuovo contesto si è notato immediatamente un netto miglioramento delle prestazioni dell'applicativo, che risulta ora utilizzabile anche in modalità VR, pur presentando ancora qualche lag di sistema dovuti esclusivamente alla nostra GPU.

5.2 - COSTRUZIONE DEI TEST

5.2.1 - TARGET E TEST QUALITATIVI

La maggior parte del parco consumatori delle nuove tecnologie, e quindi anche della VR, generalmente è composta da utenti appartenenti alla Generazione Y, nati dal 1980 al 1995, e alla Generazione Z, dal 1995 al 2010. Per questo abbiamo scelto di individuare come target per il nostro User Test tutti quegli utenti che appunto rientrano nella categoria dei Millenials e Post-Millennials.
Le ragioni che ci hanno spinto a scegliere questo tipo di Target sono principalmente due ed entrambe consequenziali: una nuova tecnologia in via di sviluppo, come appunto la VR, per ovvie ragioni, risulterà disponibile soprattutto al target scelto, essendo composto da persone che sono o saranno al centro della società nei prossimi anni; la VR richiede un utenza con una buona affinità tecnologica, che sia quindi abituata ad usare computer, smartphone o qualsiasi altra tecnologia, come normale attributo della vita quotidiana, e che non sia estranea all'utilizzo di controller da gaming.
È noto come gli User Test si possano dividere in due categorie, quali test qualitativi e test quantitativi[42]: i primi servono a riscontrare problemi di utilizzo, feedback, intuitività della UI e lacune di UX, dando modo di individuare i principali problemi di progettazione; i secondi invece producono dati statistici in merito alle differenti scelte compiute dagli utenti durante il test. Nel nostro contesto risultava più corretto compiere **test qualitativi**, poiché più opportuni a dare risultati inerenti ai nostri obiettivi riguardo una interfaccia funzionale, ma ancora non ottimizzata per un utilizzo su larga scala.
Complice di questa scelta però è stata anche la situazione di emergenza sanitaria globale, dovuta alla pandemia di Covid-19, che ci ha imposto di svolgere i test con un numero ridotto di persone, più che sufficiente per un test qualitativo ma ben lontano dal minimo necessario per test quantitativi. Abbiamo infatti sottoposto alla fase di User Testing 17 volontari rientranti nel target precedentemente descritto. È stato accertato, mediante opportuni moduli e liberatorie, che non fossero stati a contatto con soggetti positivi nei 14 giorni precedenti e che fossero consci dei potenziali rischi di contagio derivanti da una seduta di test. Oltre a queste precauzioni, abbiamo comunque distribuito equamente tutti i vari partecipanti in slot orari giornalieri, così da evitare assembramenti; sono state costantemente indossate mascherine, sia da noi che dai tester e sia controller che visore è stato accuratamente disinfettato dopo ogni utilizzo. Sarebbe stato interessante svolgere test anche con utenti fuori dal target consi-

derato, ma date le restrizioni abbiamo preferito concentrarci sul target strettamente necessario.
Seppur per un test qualitativo generalmente bastino 3 o 4 utenti, abbiamo scelto di ripetere il test su più persone, arrivando appunto a 17, poiché secondo noi vi erano ulteriori fattori che potevano condizionare i risultati dei test: primo fra questi l'esperienza pregressa con la VR, quanto i nostri utenti fossero "*technology-addicted*", quanto invece fossero riconducibili al prototipo di gamer abile con l'uso di controller, e così via...
Anche in questo caso ci siamo scontrati col fatto che per la VR non esistano standard predefiniti su come svolgere dei test qualitativi per le interfacce utente, al contrario invece delle UI Desktop. Pur essendo a conoscenza di alcuni standard consolidati, come VRUSE[43], che ormai risale a oltre 20 anni fa, abbiamo optato per costruire un nostro test creando: un **questionario preliminare** per profilare correttamente tutti gli utenti; delle **task** adeguate alle interazioni che offre il nostro applicativo; un **questionario post-esperienza**, sottoposto all'utente appena concluso lo svolgimento delle task, per raccogliere qualsiasi tipologia di dato relativo alla sua esperienza, sia fisiologico che psicologico.
Per la costruzione dello User Test ci siamo basati su diverse ricerche e articoli trattanti l'argomento, partendo da questionari per UI Desktop e adattandoli poi alla UI VR. Nonostante nei sottocapitoli che seguono vengano comunque indicate le fonti di riferimento, qui ci preme sottolineare che per quanto riguarda la struttura generale dei test e le accortezze da prendere in considerazione per la UX, in VR e non, ci siamo basati sulle ricerche e articoli di: Merche Gómez Sánchez (Senior UX Designer & Ricercatrice)[44]; Ana Pavuna (Antropologa Digitale)[45]; Tímea Falmann (Ricercatrice UX)[46]; Dionysios Georgios Papadimitriou (tesi 2019 in Human-Computer Interaction)[47]; Kit di Usabilità, fornito dal Gruppo di Lavoro e Usabilità (GLU-X) del Ministero della Pubblica Amministrazione[48].

5.2.2 - QUESTIONARIO PRELIMINARE

Nella prima fase di user testing, ci premeva capire se il target individuato possedesse o meno delle conoscenze riguardo le modalità d'interazione offerte dalla VR e, più in generale, se avesse mai utilizzato in precedenza un visore per la Realtà Virtuale. Era infatti necessario effettuare una profilazione dei vari tester in base alle loro conoscenze ed esperienze pregresse con la VR, per aiutare da un lato noi a strutturare meglio tutta la fase di test successiva, proponendo precise task ad ogni utente in linea con le competenze da lui possedute, dall'altro

gli utenti stessi a completare tranquillamente il test e non bloccarsi durante l'esperienza per lacune conoscitive sul funzionamento di questa nuova tecnologia.

Abbiamo quindi creato un **questionario preliminare** da sottoporre agli utenti prima di essere sottoposti alle Task, nel quale sono state inserite diverse domande mirate a comprendere il loro livello d'interesse e competenza nei confronti della Realtà Virtuale. Per avere una profilazione più accurata dei nostri tester, abbiamo deciso di inserire nel questionario alcune domande relative ad indagare quanto essi fossero effettivamente "*technology-addicted*", quindi conoscitori, in generale delle nuove tecnologie e delle loro logiche, ma anche quanti di questi fossero catalogabili in **Hardcore** o **Casual Gamers**. Infatti, a nostro parere, gli utenti che rientravano nella categoria di gamer esperti od erano semplicemente degli appassionati del mondo videoludico, a prescindere che avessero già provato o meno un HMD, possedevano una mentalità diversa dagli altri tester non videogiocatori, che li avrebbe aiutati ad apprendere con più rapidità le meccaniche e le interazioni offerte dal nostro applicativo. Questo nostro presupposto si basa sul fatto che molte delle logiche governanti il mondo videoludico sono presenti anche nella nuova tecnologia della Realtà Virtuale, basti pensare che in entrambi i casi l'interazione tra player e programma viene gestita da dei controller.
Tuttavia, indipendentemente che i nostri utenti fossero competenti o meno in materia, abbiamo provveduto a fornire ad ognuno di loro, al termine della compilazione del questionario preliminare, un breve tutorial sul funzionamento dei Touch Motion Controller dell'Oculus Rift utilizzato per questa fase di User Test e delle modalità d'interazione progettate sul nostro applicativo.

Ci esentiamo dal riportare in questo sottocapitolo tutte le domande presenti nel suddetto questionario, poiché visionabili personalmente dal lettore all'appendice **A** di questo documento, nella quale è stato inserito l'intero questionario preliminare utilizzato.

Siccome volevamo garantire l'anonimato dei nostri utenti, in modo che non ci fosse nessuna correlazione tra questi e i rispettivi dati rilevati, ad inizio esperienza è stato assegnato ad ogni partecipante al test un **ID** che lo identificasse univocamente e che ci permettesse di collegare i dati raccolti col questionario preliminare con quelli dello svolgimento delle Task e del questionario post-esperienza, mettendoci poi nelle condizioni di poter fare le opportune analisi e conclusioni.

A seconda delle competenze dichiarate da ogni singolo tester, si è provveduto poi a catalogarli, mediante ID assegnato, in 3 macro-categorie da noi ideate, chiamate rispettivamente **Base Level**, **Medium Level** ed **High Level,** che riflettessero il livello di esperienza posseduto dall'utente in merito di VR. Con le stesse categorie abbiamo poi suddiviso anche le task ideate, in modo da proporle agli utenti coerentemente al loro livello di complessità. Così facendo, al termine della compilazione di ogni questionario preliminare, eravamo a conoscenza del livello d'esperienza dichiarato e di conseguenza della serie di task che potevamo proporre all'utente.

5.2.3 - TASK

Il fulcro di un test qualitativo sono le **Task** che vengono sottoposte ai tester. Se scelte opportunamente, esse permettono di sondare ogni singolo aspetto implementato nelle UI e dunque di metterne in evidenza le problematiche.
Prima di pianificare le varie Task, abbiamo definito le **Personas** e gli **Scenari** in cui queste potevano trovarsi all'interno dell'applicazione. Chiariti dunque i tre livelli di suddivisione delle varie Personas, a seconda della loro esperienza pregressa con la VR, si è passati alla definizione dello Scenario attraverso alcune "macro-azioni" che potevano essere svolte utilizzando la nostra interfaccia utente, in modo da aver ben chiaro fin da subito ciò che sarebbe stato testato. Nello specifico queste sono: *spostarsi nell'ambiente, aprire e chiudere i menu, chiudere l'applicazione, spostare a piacimento i menu attorno a sé (solo VR), cambiare modalità di navigazione, utilizzare i Tool e configurare il prodotto*.
A questo punto, per ogni Scenario sono state definite le Task, sia per la UI Desktop che VR, dividendo quest'ultime in tre **livelli di difficoltà** (Base, Medium, High), potendo così sottoporre al tester delle task adatte al suo livello di competenza. Detto ciò, abbiamo deciso di rendere comunque flessibili i tre livelli e di **adattare l'andamento del test** a seconda di come reagisse l'utente. Ad esempio, se un utente si fosse dichiarato livello Base, ma avesse risolto le prime Task con facilità, si sarebbe provveduto a sottoporgli Task del livello superiore, stesso discorso anche nel caso opposto, indagando allo stesso tempo le ragioni per cui si fosse dimostrato più o meno abile nell'utilizzo della Realtà Virtuale a differenza di quanto dichiarato. Per ogni Task scelta abbiamo **descritto ogni singola azione** che ci aspettavamo l'utente compiesse per portarla a termine, dopo di che abbiamo provato personalmente ognuna di esse per definirne i **tempi di completamento**, aggiungendo al tempo da noi ottenuto circa 15 secondi di ambientamento e orientamento dell'utente con l'applicativo.

Pertanto, durante il test, ad ogni utente sono state sottoposte circa 10 Task: 2-3 per la versione Desktop, 1 per il movimento in VR, 3 in merito all'utilizzo dei Tool, 3 sull'utilizzo del configuratore e 1 sul cambio di navigazione e/o chiusura dell'applicazione.
Durante lo svolgimento delle task abbiamo opportunamente misurato e quantificato quanto è stato complesso svolgere una determinata attività, ovvero la *task level satisfaction*[49]. Esistono diverse tipologie di questionari proponibili all'utente alla fine dello svolgimento di una task, che sia stato o meno raggiunto l'obiettivo; noi abbiamo optato per la ***SEQ (Single Easy Question)***. Questo tipo di *Survey* prevede una sola ed unica domanda: *Quanto ritieni difficile questa task da 1 a 10?* (con 1 molto facile e 10 molto difficile). Con questa domanda è stato particolarmente facile comparare i dati provenienti dalle task, seppur la difficoltà percepita da ogni utente avesse sempre una componente soggettiva fortemente influenzata dal proprio background.
Rimandiamo il lettore all'appendice **B** del documento nel caso volesse visionare la struttura della scheda utilizzata durante i test, nonché tutte le task e i relativi tempi di completamento.

5.2.4 - QUESTIONARIO POST-ESPERIENZA

Il questionario post-esperienza viene somministrato ai tester immediatamente dopo la fase di completamento delle Task, in modo che l'esperienza sia ancora ben impressa nella memoria del soggetto e che dunque le risposte al questionario siano quanto più veritiere possibili. È stato anche qui chiesto all'utente di inserire l'ID assegnatogli a inizio esperienza, per poter ricondurre i risultati alla sua profilazione e alle task da lui svolte, mantenendo l'anonimato.
La prima sezione di questo questionario è il ***SSQ (Simulator Sickness Questionnaire)***[50], questionario standard per indagare tutti i sintomi di *Cybersickness* che il soggetto deve quantificare su una scala da 0 a 3. Questa sezione è fondamentale per comprendere se la UI VR da noi progettata introduce sintomi di malessere. È stata inserita come prima sezione del questionario post-esperienza poiché, se l'utente avesse manifestato alcuni dei sintomi specificati nel SSQ, sarebbe stato opportuno, per ottenere dati veritieri, riportarli fintanto che fossero ancora da egli ben percepiti.
Di seguito sono stati indagati tutti quei fattori coinvolti nel creare ***senso di immersione e presenza***. Per la stesura di questa sezione è stato preso come riferimento il questionario sulla immersione e presenza della Tesi di Laurea Magistrale "*User Experience Evaluation in Virtual Reality*" di Dionysios Georgios Papadimitriou, della Tampere University[51], documento nel

quale abbiamo trovato conferme anche in merito alla corretta stesura del *SSQ*, oltre che informazioni sulla struttura generale della fase di User Testing.

La sezione successiva è riservata allo ***UEQ (User Experience Questionnaire)***, strumento standard per la misurazione in generale della UX, in grado di sondare sia aspetti di usabilità (efficienza, chiarezza, affidabilità) che di user experience (originalità, stimolazione)[52]. Si tratta di una serie di coppie di aggettivi, diametralmente opposti per significato, mediante i quali l'utente può scegliere, su una scala da 1 a 5, quale aggettivo rispecchia meglio l'applicazione e l'esperienza fatta.

Successivamente si è condotta una **valutazione di usabilità**. Per questa sezione abbiamo fatto riferimento a "*Usability Evaluation 2.0*" di Francesco di Nocera, Ph.D del Dipartimento di Psicologia de "La Sapienza" di Roma[53]. La Us.E 2.0 (Usability Evaluation) è un questionario multi-dimensionale, basato su 19 domande volte a valutare le tre principali dimensioni dell'usabilità: Maneggevolezza, Soddisfazione e Attrattiva. Nel nostro caso le domande scendono a 16, avendo eliminato quelle relative agli acquisti, non previsti per ora dal nostro applicativo. Le domande sono state ripetute due volte, dovendo valutare in modo differenziato sia l'usabilità della UI Desktop che quella VR. Nella terza parte, riguardante l'attrattiva, è stata inserita una domanda per la misurazione del ***Net Promoter Score (NPS)***, necessaria a comprendere se l'utente, al termine dell'esperienza, si possa classificare come promotore, neutro o detrattore dell'applicazione.

Il questionario si conclude con un'**intervista** al tester, che si compone principalmente di domande aperte. Lo scopo dell'intervista è raccogliere quante più possibili informazioni e suggerimenti l'utente voglia fornirci, non essendo più vincolato dalle domande a risposta multipla. Questa sezione non è stata utilizzata per raccogliere dati, ma più che altro commenti e sensazioni legate ad ogni fase dell'esperienza, soprattutto quelle derivanti dall'interazione in Realtà Virtuale, rendendo più facile per noi sondare quali parti dell'applicativo fossero risultate critiche o problematiche.

Il questionario post-esperienza è visionabile per intero all'appendice **C** di questo documento.

5.3 - ANALISI DEI DATI

5.3.1 - RISULTATI DEL QUESTIONARIO PRELIMINARE

Dalle risposte al questionario preliminare si evince che gli utenti sottoposti a test erano al **52,9**% donne e al **47,1**% donne, la cui età era compresa tra i 22 e i 30 anni, con una media ponderata pari a **25 anni**, dunque perfettamente rientrante nel target stabilito.

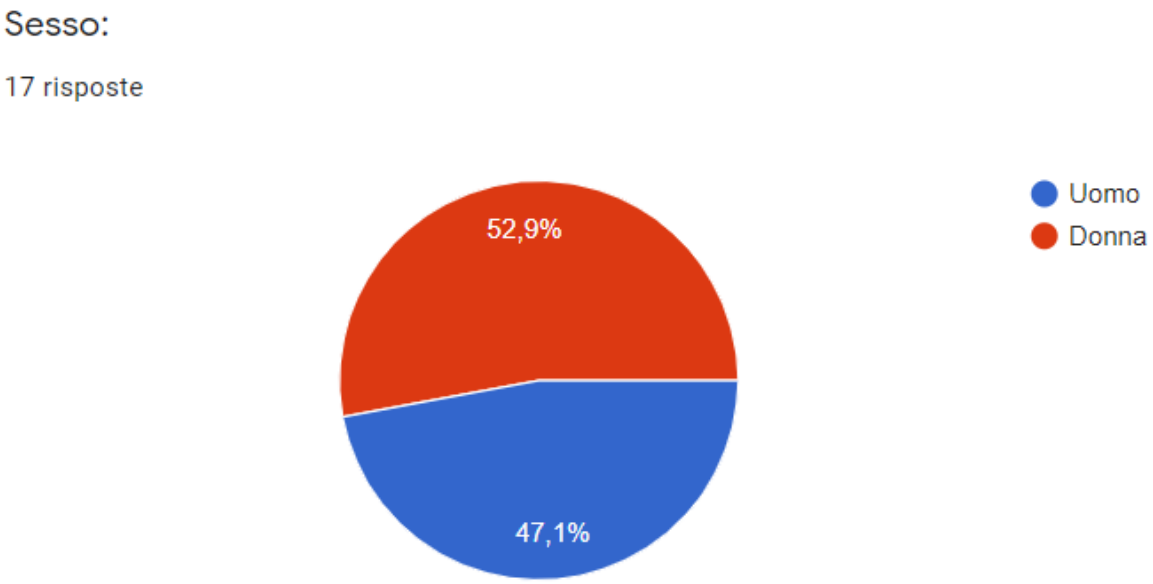


FIGURA 5.4: PERCENTUALE DI UOMINI E DONNE APPARTENENTI ALL'UTENZA CHE È STATA SOTTOPOSTA AL TEST

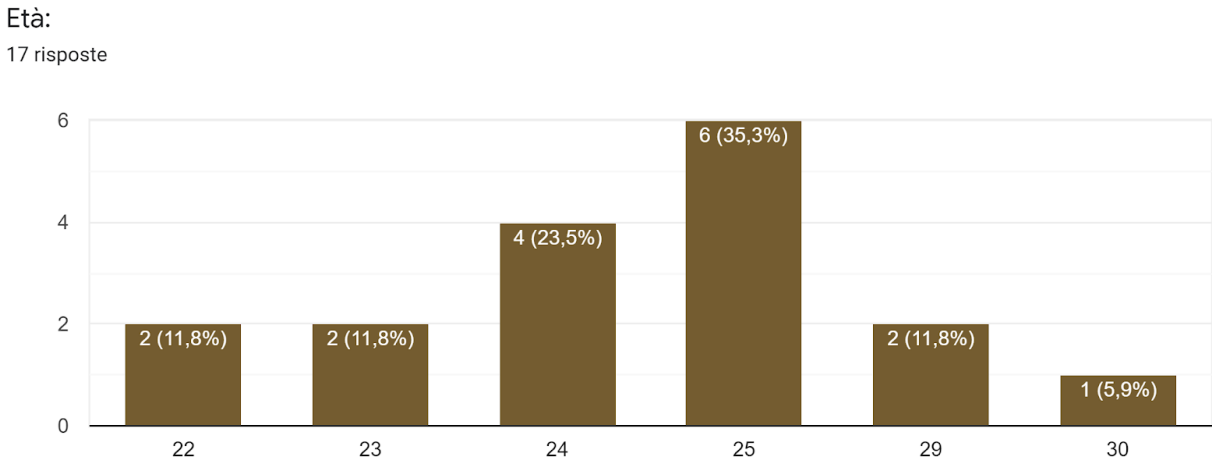


FIGURA 5.5: DISTRIBUZIONE DELL'ETÀ DEGLI UTENTI

La quasi totalità dei 17 tester frequenta l'università, ad eccezione di un solo utente che lavora attualmente nel settore Automotive. Purtroppo non siamo riusciti ad invitare alla seduta di test altri lavoratori del settore, che ci avrebbero potuto dare, con la loro esperienza, molti

più feedback sui pregi e difetti dell'interfaccia utente realizzata, e consigli su come ottimizzare o modificare quanto finora implementato. Gli studenti partecipanti sono iscritti a corsi di laurea inerenti al mondo dell'informatica, della telecomunicazione, della meccatronica, della sociologia, del cinema, della psicologia e in generale dell'umanistica. Tra questi ci interessava soprattutto il parere di coloro che facevano parte del nostro stesso dipartimento universitario poiché, avendo in comune con noi diverse competenze e conoscenze, ci hanno potuto dare importanti consigli di miglioramento dell'interfaccia sia a livello informatico che di design.
Per quanto riguarda l'utilizzo e la conoscenza delle varie tecnologie, un solo utente ha affermato di passare meno di 2 ore al giorno davanti ad computer, telefono o tablet, **5** utenti invece tra le 3 e le 4 ore, **8** tra le 5 e le 8 ore ed infine **3** oltre le 8 ore al giorno. La maggior parte di loro quindi utilizza tali device in modo continuativo durante la giornata e di conseguenza conosce le logiche d'interazione e di utilizzo offerte da questi, presentando quelle competenze tecnologiche comuni a tutti i soggetti rientranti nel target.

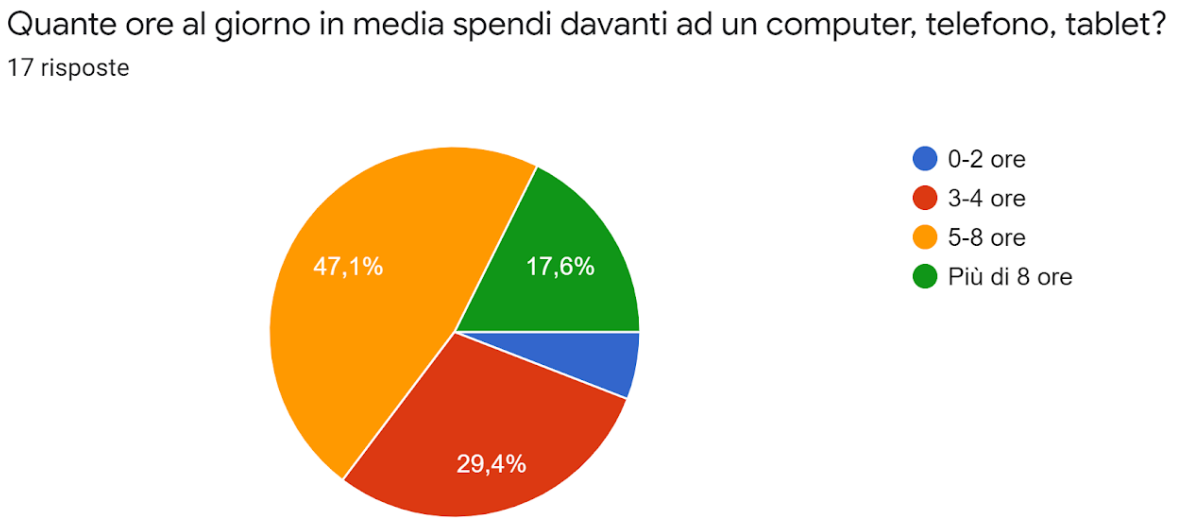


FIGURA 5.6: ORE IN MEDIA PASSATE DAGLI UTENTI DAVANTI AD UN COMPUTER, TELEFONO O TABLET

Si è poi verificato quanti di loro rientrassero nella categoria dei *Gamer*, pertanto se fossero soliti giocare ai videogiochi, per PC o altre piattaforme, e con quale frequenza. Il **29,4**% degli utenti ha dichiarato di non essere interessato al mondo videoludico e quindi di non giocare di norma ad alcun videogioco, il **17,6**% di non giocare al PC ma piuttosto su altri dispositivi come la console o il mobile, il **23,5**% di giocare al computer qualche volta al mese, uno solo, pari al **5,9**%, una volta a settimana, l'**11,8**% 4 o 5 volte a settimana e infine un altro **11,8**% tutti i giorni o quasi. Vi è quindi un buon **70**% di questi che, almeno una volta al mese, è solito videogiocare al PC ed è a conoscenza perciò delle dinamiche tipiche del mondo videoludico.

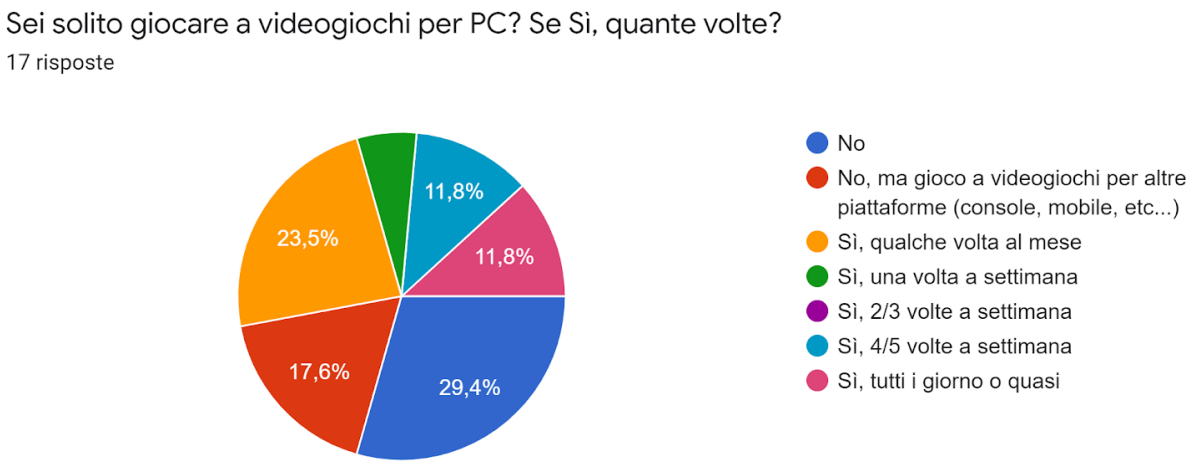


FIGURA 5.7: TENDENZA DEGLI UTENTI A GIOCARE AL PC E CON QUALE FREQUENZA

Passando invece alla VR, abbiamo chiesto quale fosse il loro livello d'interesse nei confronti di questa nuova tecnologia. Come avevamo previsto, quasi la totalità degli utenti ha manifestato un notevole interesse verso questa, complice la curiosità per una nuova esperienza, anche se oltre la metà di loro ha affermato di non conoscerne bene il funzionamento. Solo 2 di loro si sono dimostrati indifferenti alla Realtà Virtuale, mentre il **52,9**% si è definito incuriosito da questa e il restante **35,3**% molto interessato. Questo entusiasmo in parte non riflette quello effettivo mostrato ad oggi dal pubblico di massa che, come visto nel sottocapitolo **1.1.1** sull'Hype Cycle della VR, ha ancora forti dubbi sulle sue concrete potenzialità. Probabilmente ciò è dovuto alla scelta di un campione di utenti principalmente giovane ed inserito nel mondo scientifico universitario, costantemente aperto alle nuove tecnologie e quindi promotore di queste.

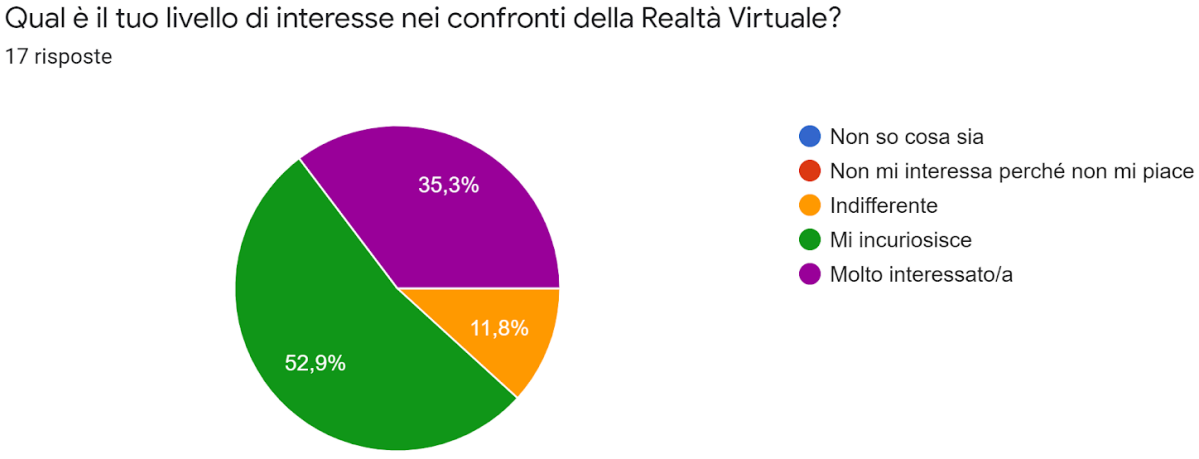


FIGURA 5.8: LIVELLO D'INTERESSE ESPRESSO VERSO LA REALTÀ VIRTUALE

Dalle successive domande è emerso che solo 4 utenti su 17 presenti non aveva mai utilizzato un HMD, mentre per i restanti 13 il **38,5**% lo aveva provato solo una volta, il **30,8**% meno di 5 volte, il **15,4**% dalle 5 alle 20 volte e il **15,4**% oltre le 20.



FIGURA 5.9: OCCASIONI DI UTILIZZO DI UN VISORE TRA I TESTER

Nonostante quasi tutti gli utenti avessero provato almeno una volta un visore, molti di loro non reputavano ancora facile interagire con un ambiente virtuale. Nello specifico, il **30,8**% dei 13 utenti aventi già provato un'esperienza in VR ha trovato complicata l'interazione con l'ambiente, indicando, in una scala di complessità inversa che andava da 1 (Molto complicato) a 5 (Molto semplice), un livello di semplicità pari a 2 su 5; il **53,8**% ha valutato non particolarmente semplici le proprie esperienze pregresse in VR, scegliendo invece il valore 3; solo 2 candidati su 13, pari al **15,4**%, hanno trovato molto semplice interagire con un ambiente VR, poiché, come è stato verificato in seguito con altre domande, risultano essere possessori o frequenti utilizzatori di un visore e quindi abituati ad interagire con dei controller per la Realtà Virtuale. Se si vanno ad analizzare le risposte degli utenti che hanno già sperimentato la VR, alla domanda relativa al loro livello di confidenza nell'utilizzo dei controller, si nota che 11 di loro, pari all'**84,7**% di questo sottocampione, hanno espresso un valore di confidenza minore o uguale a 2, mentre i restanti 2, nonché i medesimi citati sopra, un valore superiore o uguale a 4. Le singole risposte sono comunque visionabili nell'istogramma in figura 5.11.

Alla luce di questi risultati, da noi in gran parte previsti prima della seduta di test, è chiara la motivazione per cui si è deciso di effettuare, prima dell'esecuzione delle varie task, una velo-

ce dimostrazione del funzionamento dei Touch Motion Controller e delle principali interazioni con l'ambiente virtuale e la UI da noi progettata.

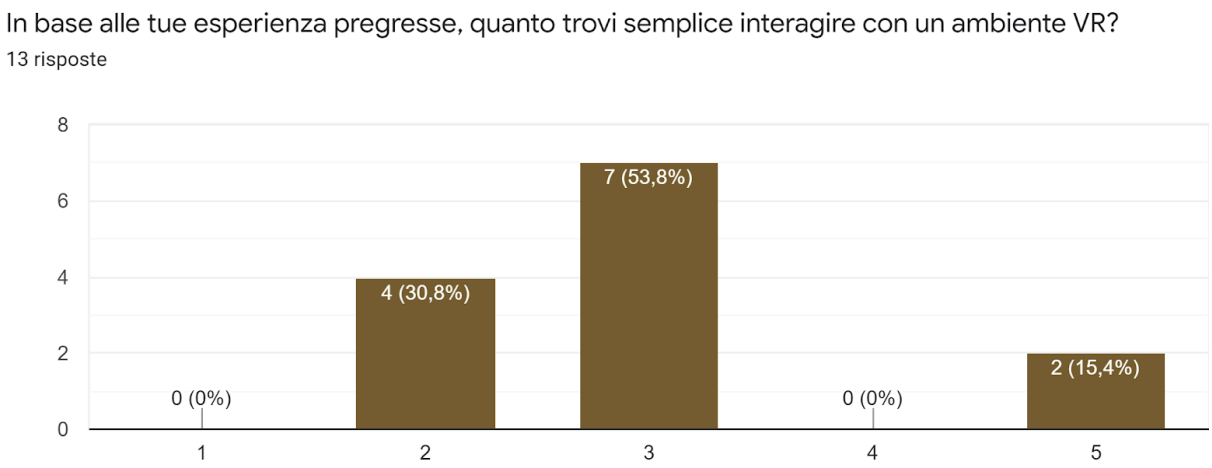


FIGURA 5.10: SEMPLICITÀ D'INTERAZIONE CON UN AMBIENTE VIRTUALE RIPORTATA DAGLI UTENTI

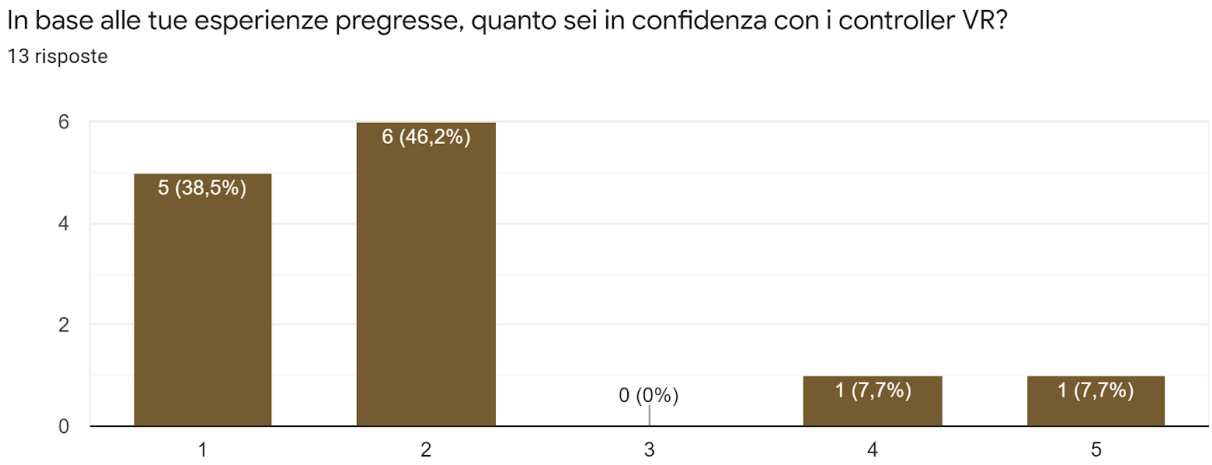


FIGURA 5.11: LIVELLI DI CONFIDENZA ESPOSTI NEI CONFRONTI DEI CONTROLLER VR

Per concludere la sezione del questionario riguardante le esperienze pregresse in VR, si è chiesto agli utenti di autovalutare, mediante una scala a 5 valori, il proprio grado di esperienza verso di questa. Di loro, 3 si sono dichiarati incapaci di utilizzare questa tecnologia, 2 invece non molto abili optando per il valore 2, 5 tester si sono definiti discretamente abili, 2 di loro hanno ritenuto di essere a proprio agio con la Realtà Virtuale e hanno scelto pertanto il valore 4, infine solo uno si è sbilanciato dichiarandosi esperto di tale tecnologia.

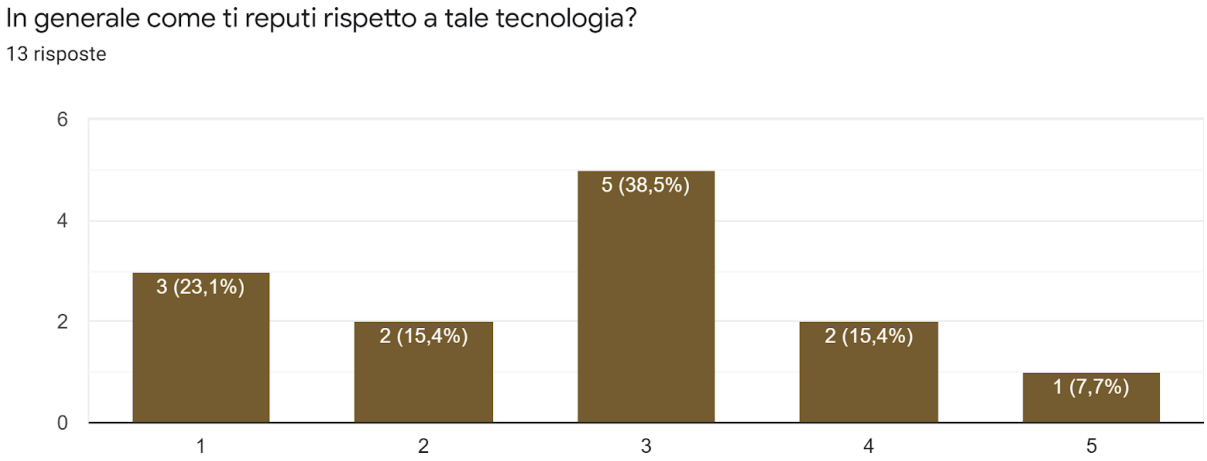


FIGURA 5.12: LIVELLO DI ESPERIENZA DICHIARATO DAGLI UTENTI CON LA VR

Gli utenti che hanno scelto il valore 1 o 2 sono stati automaticamente inseriti nella categoria Base Level citata ad inizio capitolo, quelli dichiaranti un livello di esperienza pari a 3 o 4 nella Medium Level e infine nella categoria High Level coloro che hanno specificato un valore pari a 5.

Benché la maggior parte degli utenti avesse dichiarato in principio di aver già provato un HMD, anche più di una volta, si è poi visto che solo una minima parte di questi ritiene di saper interagire correttamente all'interno di un ambiente virtuale e quindi di avere una buona confidenza coi controller. Questo, presumibilmente, deriva dal fatto che quasi tutti gli utenti che hanno avuto l'opportunità di testare il visore lo abbiano fatto solamente a delle fiere, conferenze relative al mondo della tecnologia, mostre con particolari installazioni artistiche e non perchè ne fossero effettivamente possessori. Questo rimarca ancora una volta la **difficile accessibilità** di un visore per il pubblico di massa.

Abbiamo così deciso di indagare quali fossero le motivazioni che spingessero gli utenti a non acquistare un visore e a capire se, in qualche modo, ci fosse una correlazione con le problematiche già riscontrate nei vari casi studio e articoli di ricerca. Alla domanda a risposta multipla nella quale veniva chiesto se ci fosse un particolare motivo che impedisse loro l'acquisto di un visore, gli utenti per ben **7 volte** hanno scelto la risposta "*Nessuno, semplicemente non mi interessa così tanto*", **4 volte** invece *"Nessuno, ma ho intenzione di farlo prima o poi"*, **3 volte** *"È troppo costoso"*, **2 volte** *"Sto aspettando che tecnologicamente si evolva perché ora non mi convince pienamente"*, mentre si sono scelte **una volta** sola le seguenti risposte: *"Non ci sono abbastanza contenuti disponibili per la VR (giochi, applicazioni)"*, *"Mi provoca nausea, vertigini o mal di testa"*, *"L'esperienza di gioco non è migliore di un qualunque altro tipo di videogioco"*, "*il mio pc non lo supporta*", "*non abbastanza maturo nella grafica*".

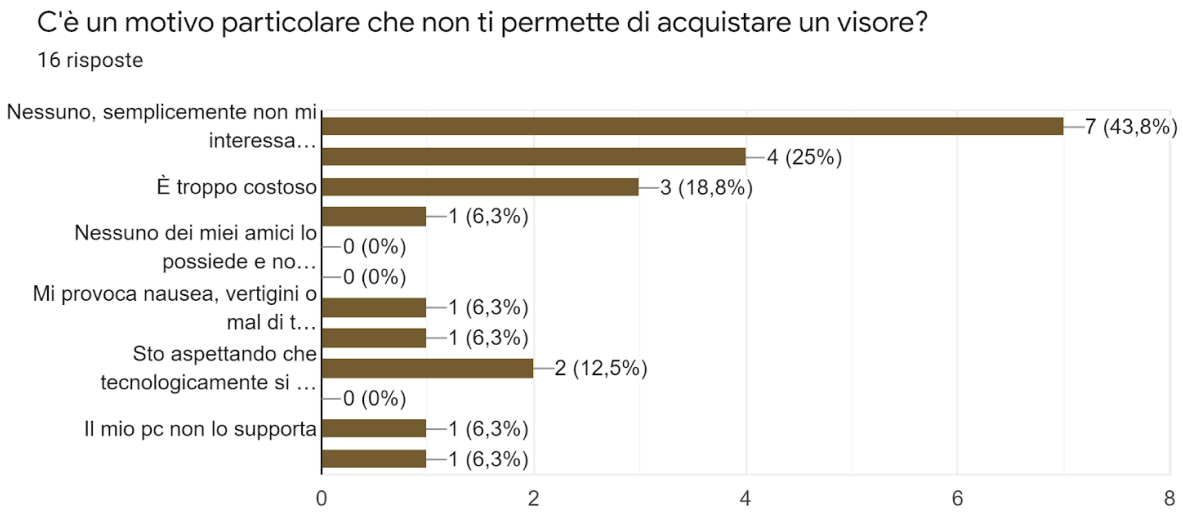


FIGURA 5.13: PROBLEMATICHE RELATIVE ALL'ACQUISTO DI UN VISORE

Dalle risposte ricevute si è visto che esiste ancora un certo grado di disinteresse nei confronti di questa nuova tecnologia, dovuto probabilmente al fatto che il pubblico non percepisce ancora le potenzialità di questo nuovo strumento e perché oggettivamente non si è ancora arrivati a quella fase di maturità che garantisce una distribuzione su larga scala. Si è poi appurato, mediante un'apposita domanda, che **solo uno** dei 17 utenti possedesse un visore, l'HTC VIVE, provando di fatto come ancora non si possa definire una tecnologia "*di massa*". Inoltre quest'ultimo candidato ha dichiarato che le funzionalità offerte al giorno d'oggi dal suo visore, che precisiamo è uno dei migliori attualmente sul mercato, non ne giustificano appieno il prezzo e quindi anche l'acquisto, poiché ritenuto ancora troppo costoso in relazione all'esperienza di gioco offerta e ai pochi contenuti disponibili sullo store. Le altre principali problematiche esposte rispecchiano appieno quanto già detto nel sottocapitolo **1.1.2**, ossia che non risulta ancora supportabile dalla maggioranza dei PC mediamente posseduti dal target, non riesce a garantire una risoluzione grafica tale da fornire un'esperienza completamente immersiva e infine fa insorgere ancora fastidiosi fenomeni di cybersickness.

5.3.2 - ANALISI DELLE TASK

La fase di esecuzione delle Task, dove i tester utilizzano l'applicazione, è quella in grado di mettere in risalto sia i fattori positivi che le criticità delle UI testate e dell'applicazione in genere. Come anticipato, per ogni Task è stato chiesto all'utente un punteggio SEQ, per definire la difficoltà percepita nel svolgerla. Le Task sono state state suddivise sia per livello di com-

petenza dell'utente sulla VR che per argomento, e quest'ultimi nello specifico riguardavano:: *UI Desktop, interazioni generali nella UI VR, Tool in VR* e *Configuratore in VR*.
I punteggi forniti dai tester alle SEQ risultano tutti inferiori a 5 (su una scala da 1 a 10), con una media totale pari a **2,5**, mentre il valor medio più alto registrato, appartenente al tester UX1905, è stato **4,88**.
Questi risultati evidenziano come i tester in generale non abbiano percepito quasi nessuna difficoltà; tuttavia, per analizzare meglio e più in dettaglio quanto sperimentato dagli utenti, è necessario esaminare i risultati filtrandoli per genere di Task proposta. Così facendo infatti, come si può vedere dal grafico in figura 5.14, è stato più evidente notare come vi sia un deciso aumento della difficoltà percepite per quelle Task inerenti all'utilizzo dei Tool in VR.

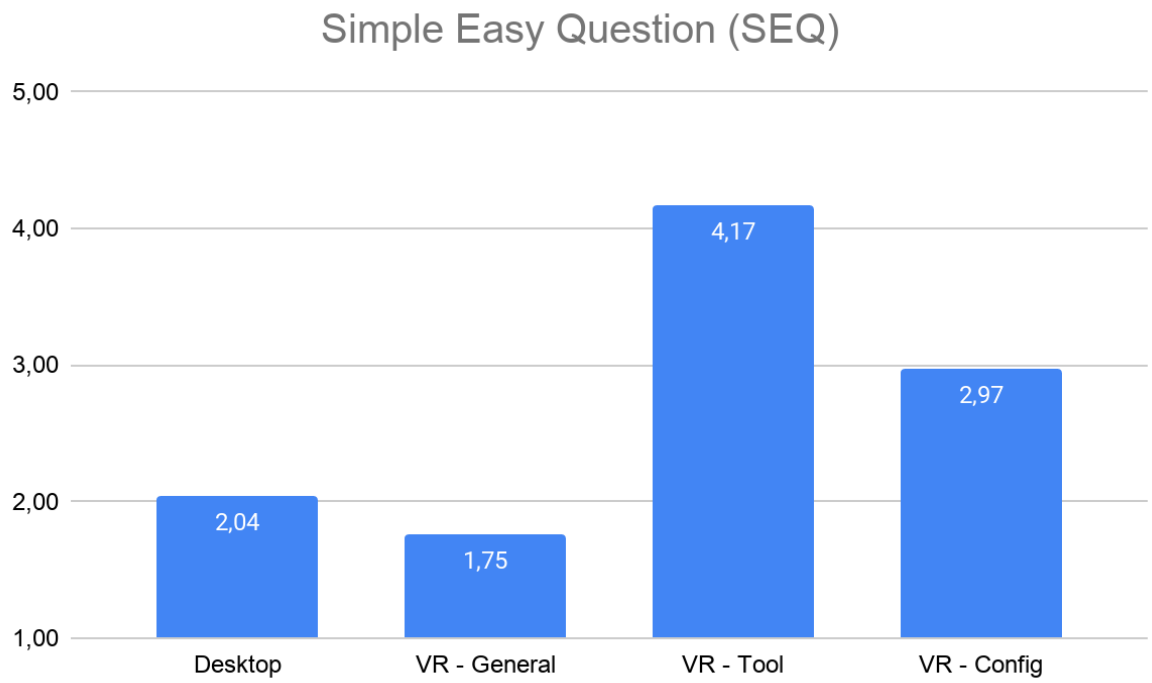


FIGURA 5.14: VALORI DELLE SEQ SUDDIVISI PER GENERE DI TASK SOTTOPOSTA AI TESTER

Questi risultati si riflettono specularmente a quelli riguardanti le percentuali di successo delle Task, che non risultano mai inferiori al 50% per ogni utente, presentando una media complessiva di **94,23**% ed il punteggio medio più basso pari a **62,5**% riportato dal tester UX1905.
Anche in questo caso i dati di maggior interesse si riscontrano dividendo le Task per genere, come riportato nel grafico in figura 5.15. Seppur in generale le percentuali di successo siano molto alte, si riscontrano anche questa volta palesi criticità per quanto riguarda le Task inerenti all'interazione con i Tool in VR.

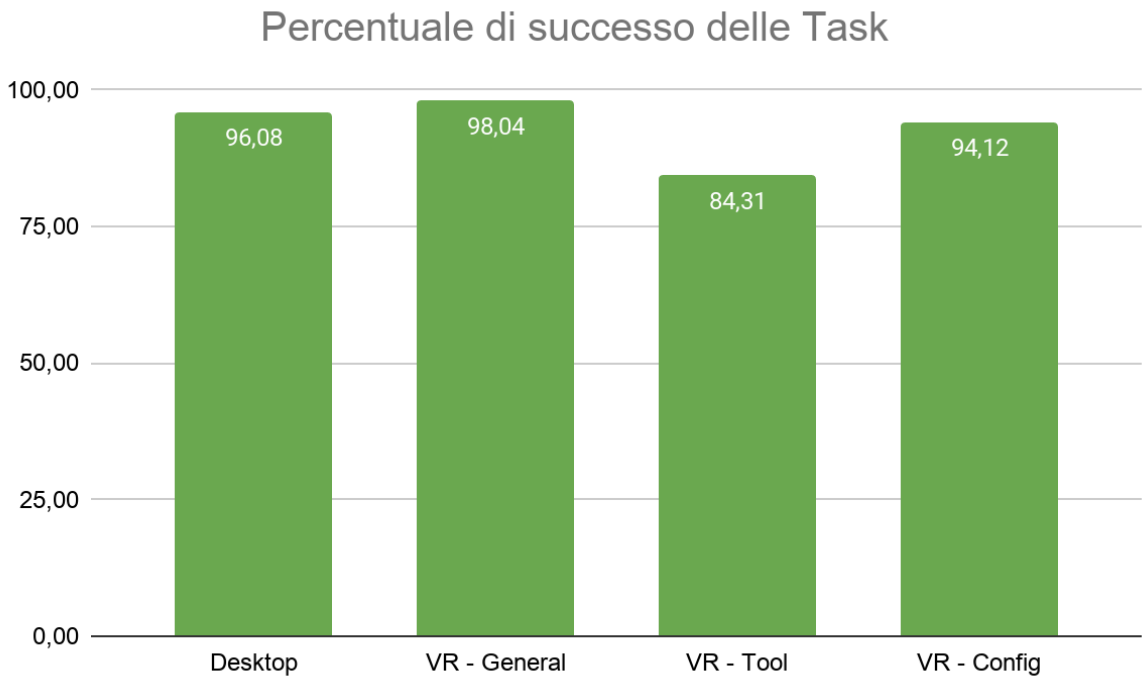


FIGURA 5.15: VALORI DELLE SEQ SUDDIVISI PER GENERE DI TASK SOTTOPOSTA AI TESTER

Abbiamo notato come alcuni tester, sia nella UI Desktop che in quella VR, inizialmente non avessero ben chiara la distinzione tra Menu Tools e Menu Configurator. La Task che ha più messo in evidenza ciò è stata quella che richiedeva di cambiare colore alla carrozzeria dell'autoveicolo, infatti alcuni tester hanno prima cercato il Tool Paint, per poi comprendere che in realtà questa azione era inerente al configuratore. Questo genere di problema potrebbe essere semplicemente risolto cambiando nome al tool in questione, sostituendolo ipoteticamente con "Draw" o un nome che possa confondere meno.

Altro problema riguardante l'interazione in generale, che diversi utenti hanno lamentato anche nel questionario post esperienza, riguarda la loro difficoltà a ricordare i comandi, evidenziata più volte da oltre 8 tester su 17. Le ragioni di ciò si devono probabilmente alla generale poca esperienza degli utenti con i controller, ma plausibilmente anche dall'impossibilità di avere un feedback visivo della posizione di questi nell'ambiente, che li avrebbe sicuramente aiutati a ricordare anche le posizione di tutti i relativi pulsanti disposti sopra. È interessante però notare che oltre 8 tester su 17, seppur consapevoli di dover eseguire precise task, spesso prendessero iniziativa, completando task differenti senza che venisse espressamente richiesto da noi. Queste iniziative riguardavano, ad esempio, cancellare o resettare quanto appena fatto con il Tool nella task richiesta, guardare le info delle varianti di loro iniziativa o utilizzare la versione "aumentata" di queste.

Non sono state evidenziate particolari criticità in merito all'utilizzo dei Tablet, sia Configurator che Settings, con cui i tester si sono immediatamente trovati a proprio agio. In un solo caso l'utente non ha avuto inizialmente chiaro l'utilizzo della sezione Path o su come tornare all'elenco delle fasi cliccando sul bottone Home. Diversamente da quanto era stato previsto, nel caso della Dashboard delle varianti, 5 utenti su 17, per caricare le varianti sull'autoveicolo hanno provato a prenderle e caricarle avvicinandole alla vettura o lanciandole su di essa. Tale comportamento si giustifica in quanto alcuni tester risultano essere gamer più o meno costanti e questa logica di configurazione è molto comune nel mondo dei videogame. Quanto visto è senza dubbio interessante, poiché questo tipo di interazione sfrutta maggiormente lo spazio tridimensionale rispetto al caricamento tramite pulsante "Load Asset", più affine ad una logica desktop.
Come evidenziano i dati relativi a SEQ e percentuale di successo, le maggiori criticità riguardano l'interazione con i Tool. Oltre 10 utenti su 17 dichiarano di trovare difficile il loro utilizzo e di ritenere troppo macchinosa la sequenza di azioni necessaria per arrivare all'effettiva applicazione della option scelta. In merito all'utilizzo dei Tool rappresentati come oggetti nella parte centrale della Dashboard, sono principalmente due le osservazioni fatte: *ridondanza della selezione iniziale* e *difficoltà nel mantenere la presa sull'oggetto*.
Nel primo caso si riteneva fosse necessaria la selezione per evitare che l'utente, per sbaglio, prendesse un altro Tool, essendo essi molto vicini fra loro, ma ci è stato fatto notare che nella vita quotidiana il gesto si limita solamente alla presa dello strumento d'interesse. Riteniamo dunque di dover togliere questo primo passaggio ritenuto ridondante dalla maggior parte dei tester, rendendo tutti i Tool attivi già all'apertura della Dashboard.
La scelta di dover tenere premuto il tasto Grip per utilizzare il Tool era data dal voler simulare una azione reale. Questa azione viene usata, nel medesimo modo, anche per quanto riguarda la possibilità di spostare dei menu attorno all'utente ed è interessante notare come in questo caso non sia mai risultato complesso tenere premuto il tasto Grip, a differenza dell'interazione con i Tool. Abbiamo dunque analizzato le differenze tra queste due interazioni che portano uno stesso gesto ad essere apprezzato in un caso e fastidioso nell'altro. È parso subito evidente che l'azione di spostare un menu risulta sempre di breve durata e non sommata ad altre azioni, mentre nel caso dei Tool viene richiesto all'utente di mantenere la presa per un lungo periodo, durante il quale deve compiere sia l'azione di scelta dell'option desiderata, che quelle necessarie per utilizzarla, arrivando a dover controllare più tasti del controller in contemporanea e di conseguenza allentare la presa sul tasto Grip, che porta a disattivare il

Tool e dover ripetere l'intero iter di azioni. Bisogna dunque modificare anche questa parte di interazione, eliminando la necessità di tener premuto il tasto Grip per semplificare la presa di un oggetto, oppure gestendo in modo differente la scelta delle Option, rendendola più immediata, in modo che l'utente non perda inevitabilmente la concentrazione sulla presa del Tool.
Un'altra sezione della Dashboard dei Tool che ha riscontrato molti problemi è stato il cassetto contenente i pulsanti per attivare *Teleport* ed *Environment*. Tutti i tester a cui è stata sottoposta una task inerente ad uno di questi due Tool non sono stati in grado di portarla a termine, se non con molta fatica o a seguito di nostri suggerimenti. Il problema qui, fortunatamente facilmente risolvibile, è la mancanza di feedback in merito alla possibilità di interagire con la sfera che apre il cassetto. Ci è comunque stato lamentato il fatto che questi due Tool non siano ben visibili all'interno della Dashboard, perché appunto inizialmente celati alla vista. Al momento questo menu presenta, a destra, due pulsanti inutilizzati, che eventualmente potrebbero ospitare questi due Tool, eliminando di conseguenza il cassetto oppure adibendolo a funzioni più marginali, dando invece maggiore risalto ai Tool che al momento ospita.
Infine vorremmo soffermarci sull'analisi di 6 tester in particolare. Come anticipato, i livelli in cui i tester erano divisi facevano riferimento alla loro esperienza con la realtà virtuale e non erano livelli "rigidi", in quanto l'esperienza è stata adattata a seconda del comportamento dell'utente. Gli utenti UX1903, UX1904, UX1907, UX1911 e UX1915 nel test preliminare hanno dichiarato di avere poca esperienza con la VR, rientrano nel Base Level o al massimo a metà tra Base e Medium Level, ma durante lo svolgimento del test, vista la facilità con cui portavano a compimento i compiti assegnati, gli sono state sottoposte task più complesse, arrivando fino a chiedere quelle di livello High. Di questi utenti sono stati ri-analizzati i dati relativi al background personale e si è constatato come tre di loro giocassero spesso a videogiochi, abilità che li mette più a proprio agio nell'utilizzo di comandi, anche complessi, per completare dei compiti attraverso l'utilizzo di controller. Gli altri due, dichiarati non gamer, hanno affermato comunque di passare tante ore con un pc, tablet o smartphone, durante il giorno.
A sottolineare ulteriormente quanto il background dei tester influenzi fortemente i loro risultati nel test, la difficoltà percepita e il successo nelle task, prendiamo come esempio i punteggi ottenuti dall'utente UX1905. È il tester che ha dichiarato maggiori difficoltà durante l'esecuzione delle task, con conseguente percentuale di successo minore in assoluto. Nel test preliminare il soggetto in questione dichiara di non aver mai giocato al computer o con altre

piattaforme che prevedano l'uso di controller, di non aver mai provato prima la VR, di non esserne assolutamente interessato all'acquisto futuro e di passare mediamente al giorno meno di 3 ore utilizzando un pc, smartphone o tablet. Dunque le maggiori difficoltà sono state riscontrate dal tester che meno è legato all'utilizzo della tecnologia nella sua vita quotidiana e per le quali dimostra poco interesse.
Consapevoli di non poter dichiarare che quanto osservato possa valere come regola assoluta, a causa del numero di test fatti, molto lontano da quello ideale per test quantitativi e per ottenere dati statistici, vogliamo però sottolineare come nel nostro caso sia stato riscontrato questo parallelismo tra background degli utenti e relativo successo delle task o difficoltà percepita nello svolgerle.

5.3.3 - RISULTATI DEL QUESTIONARIO POST ESPERIENZA

La prima sezione è quella riguardante il ***SSQ (Simulator Sickness Questionnaire)***[54], in cui, per ogni sintomo percepito, veniva assegnato dagli utenti un valore di intensità, da Nessuno (0) ad Acuto (3). Basandoci sul calcolo introdotto da Kennedy et al (1993), i sintomi sono stati suddivisi in tre macrogruppi, non mutualmente esclusivi: **Nausea (N)**, sistema **Olocomotore (O)**, di **Disorientamento (D**). Questi, attraverso opportuni calcoli, restituiscono sia i rispettivi indici di gravità di ogni gruppo di sintomi che il **Total Score (TS) rappresentante** la gravità complessiva della cyber-sickness sperimentata dagli utenti del nostro applicativo. Definite rispettivamente come [1], [2] e [3] le medie algebriche dei valori di ogni gruppo (N, O e D), si i calcolano i rispettivi indici come segue:

$$[N] = [1] * 9{,}54 \qquad [O] = [2] * 7{,}58 \qquad [D] = [3] * 13{,}92$$
$$[TS] = ([1] + [2] + [3]) * 3{,}74$$

La moltiplicazione per i fattori indicati ha solamente lo scopo di riportarli su una stessa scala di valori per poterli mettere a confronto. I punteggi totali definiscono il livello dei sintomi come: ***trascurabile*** se indice <5; ***minimo*** se tra 5 e 10; ***significativo*** se tra 10 e 15; ***rilevante*** se tra 15 e 20.

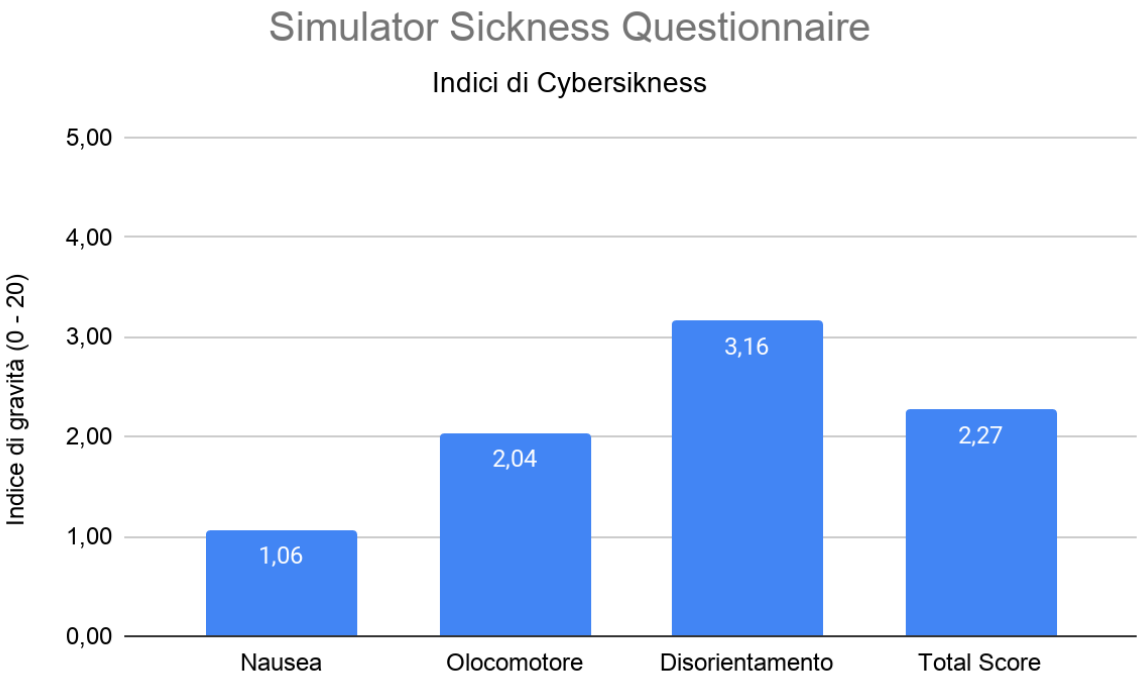


FIGURA 5.16: GRAFICO CON VALORI DI N, O, D E TS

Il grafico mostrato in figura 5.16 rappresenta il valore degli indici su una scala da 0 a 5 e non su scala completa da 0 a 20, per una più semplice lettura dei dati. Come è possibile vedere dal grafico, ogni gruppo di sintomi e il livello totale di cyber-sickness risultano avere un effetto trascurabile sugli utenti della nostra applicazione. Ci aspettavamo questo tipo di risultato poiché il movimento all'interno dello spazio virtuale avviene esclusivamente tramite teletrasporto "a breve distanza" e questo riduce significativamente qualunque disturbo fisico.

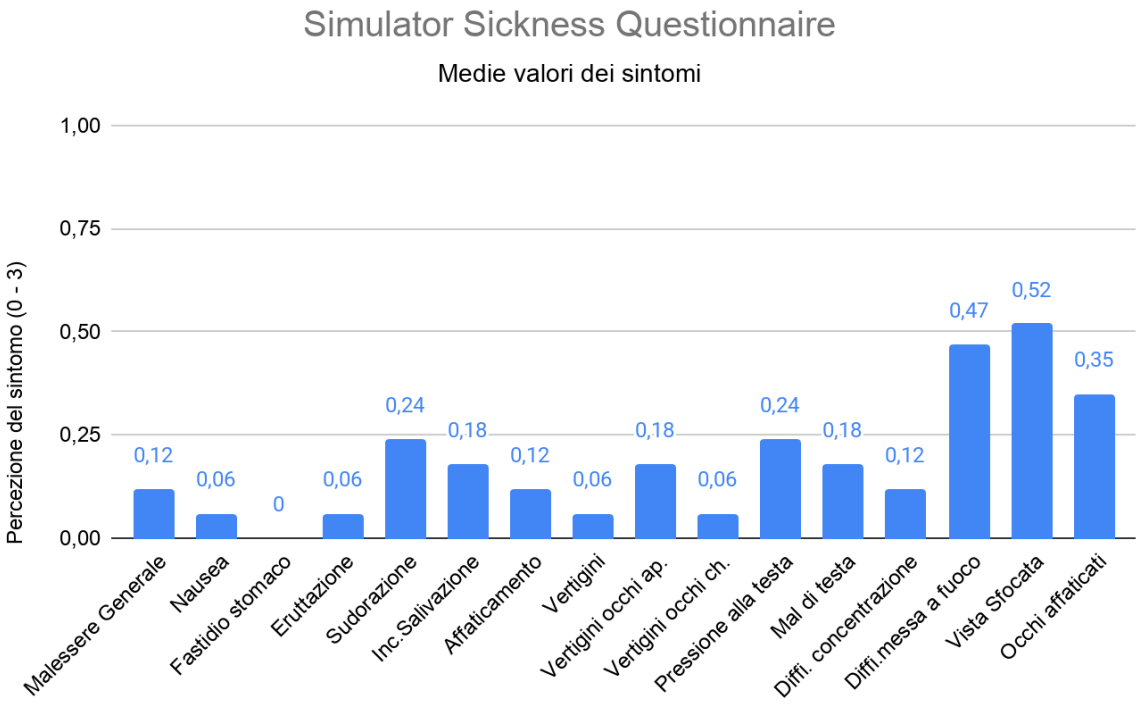


FIGURA 5.17: GRAFICO CON I VALORI MEDI DI OGNI SINTOMO

In figura 5.17 è invece riportato il grafico con i valori medi di ogni sintomo, anche in questo caso riportati su scala 0 - 1 e non da 0 - 3, pari al punteggio massimo che potevano dare i tester, per facilitarne la lettura. Si può notare che i valori medi più alti riguardano i sintomi di *Difficoltà di messa a fuoco*, *Vista sfocata* e *Occhi affaticati*. Possiamo attribuire la causa di questi sintomi a due fattori: risoluzione del visore ancora bassa, che unita ad un mal posizionamento dello stesso sul volto, anche di pochi millimetri, portano ad una scarsa messa a fuoco e conseguentemente ad un affaticamento della vista; unitamente a ciò vi è la necessità di leggere e comprendere illustrazioni e label, fattore che unito al precedente acuisce lo sforzo visivo necessario e quindi i suddetti sintomi.

Nella sezione riguardante lo stato di Immersione e Presenza, i soggetti sono stati sottoposti a 15 domande, relative a diversi aspetti che contribuiscono a restituire tale stato, nelle quali dovevano fornire un punteggio che andava da 1 a 5. Le domande sono riportati nell'appendice **C** e i risultati vengono qui riassunti in un grafico per poterli valutare nel loro insieme.

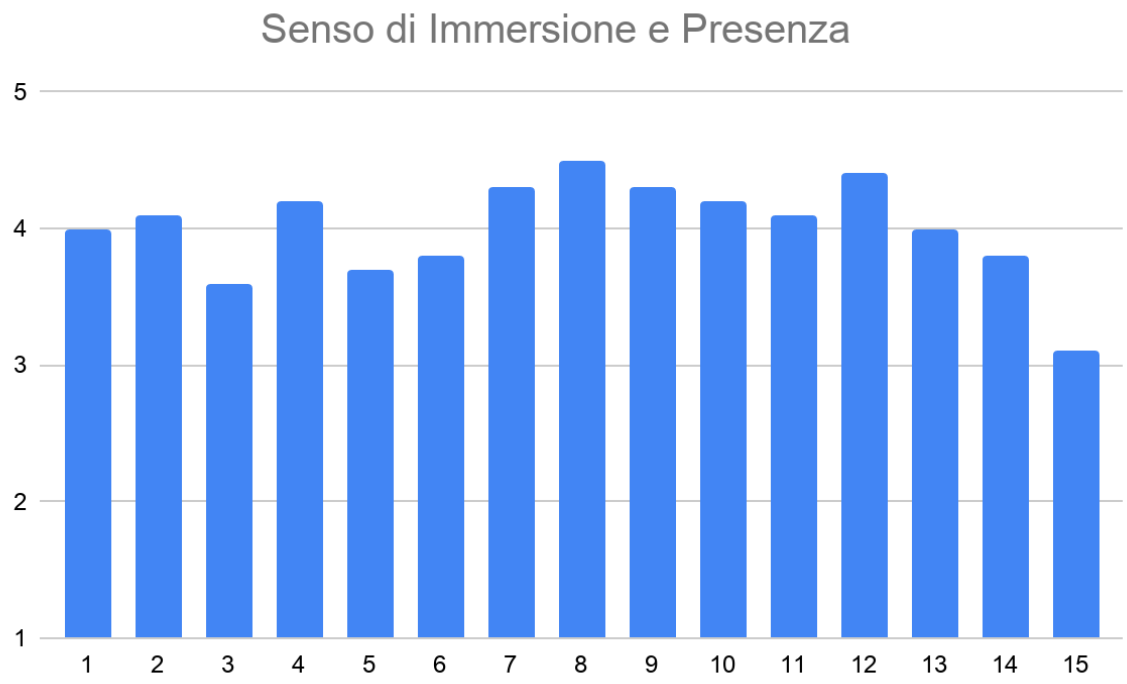


FIGURA 5.18: GRAFICO CON NUMERO DOMANDA SU ASCISSE E RELATIVO PUNTEGGIO SULLE ORDINATE

Come si può facilmente notare, ogni domanda ha ricevuto un punteggio superiore a **3**, portando la media totale di tutti i punteggi qui mostrati a **4**. È dunque evidente come il senso di Immersione e Presenza della nostra applicazione sia piuttosto elevato.

Segue la sezione dedicata all'analisi della ***User Experience***, i cui risultati sono riportati nel seguente grafico. I due aggettivi posti lateralmente ad ogni riga del grafico rappresentano rispettivamente il valore minimo assegnabile dall'utente alla suddetta domanda, ovvero 1, e quello massimo, pari a 5.

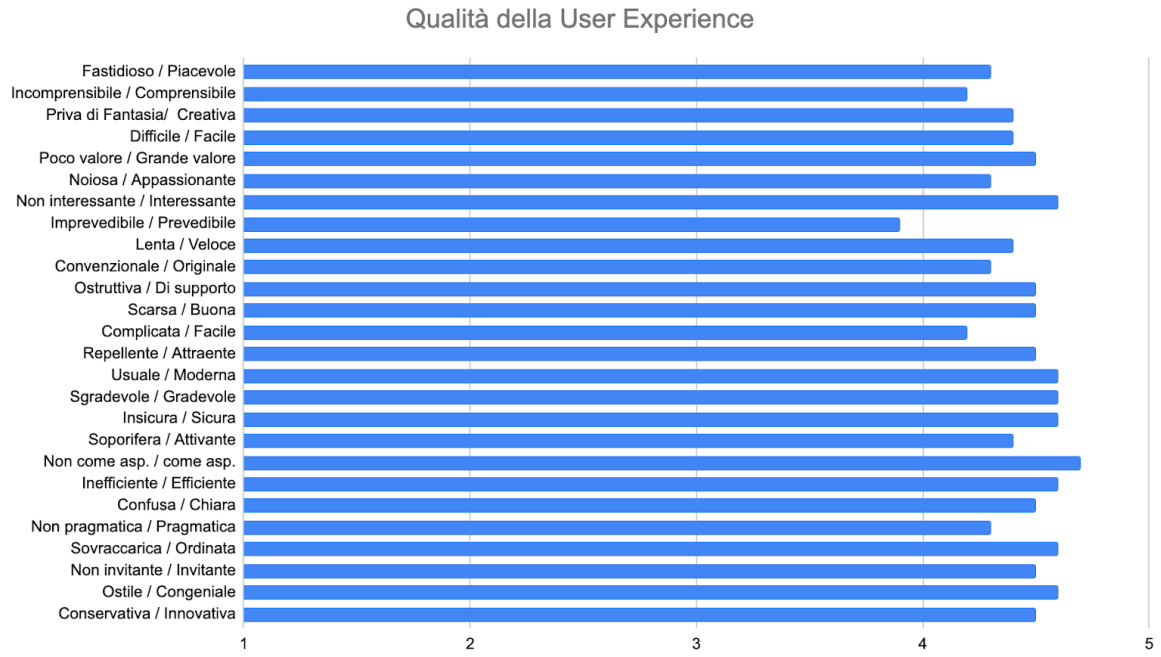


FIGURA 5.19: GRAFICO CON RISULTATI DELLA SEZIONE RELATIVA ALLA USER EXPERIENCE

Quindi ovviamente più il valore riportato si avvicina al secondo aggettivo della coppia, e quindi al punteggio 5, più aumenta il tasso di positività della *User Experience;* dal grafico pertanto si può riscontrare come l'applicazione nel complesso abbia fornito una esperienza più che positiva, con una media dei valori pari a **4,4**. Vogliamo però contestualizzare questo risultato, in quanto la maggior parte dei tester era alle sue prime esperienze in VR, se non alla prima in assoluto, per cui l'esito positivo è stato sicuramente influenzato dal loro entusiasmo verso questa nuova tecnologia, anche se nel complesso si è raggiunto un buon risultato di UX.

Per quanto riguarda l'***Usabilità*** dell'applicazione, abbiamo riportato qui sotto i relativi risultati suddividendoli in tre grafici, inerenti ai tre aspetti indagati: *Maneggevolezza, Soddisfazione* e *Attrattiva.* Per ogni gruppo sono stati ulteriormente divisi i risultati riguardanti la UI Desktop e quella VR, in modo da poterle mettere facilmente a confronto. Ogni domanda posta è stata riassunta nel grafico con delle parole chiave, per una migliore comprensione, e sono stati riportati, su una scala da 1 a 5, i valori medi dei punteggi dati dai tester ad ogni fattore di usabilità preso in esame. Si nota immediatamente come i risultati in merito alla UI VR siano

quasi sempre inferiori a quelli della versione Desktop. È un risultato che comunque ci aspettavamo per due ragioni: i tester sono quasi tutti neofiti della VR, per cui è naturale che pecchino in abilità di utilizzo di questo tipo di piattaforma; alcune interazioni, seppur considerate sia da noi che da loro intuitive e realistiche, non sempre sono risultate di semplice applicazione. Risulta positivo però che, per quanto i valori in VR siano inferiori, in media non scendono oltre un fattore di **0,2** rispetto alla versione Desktop.

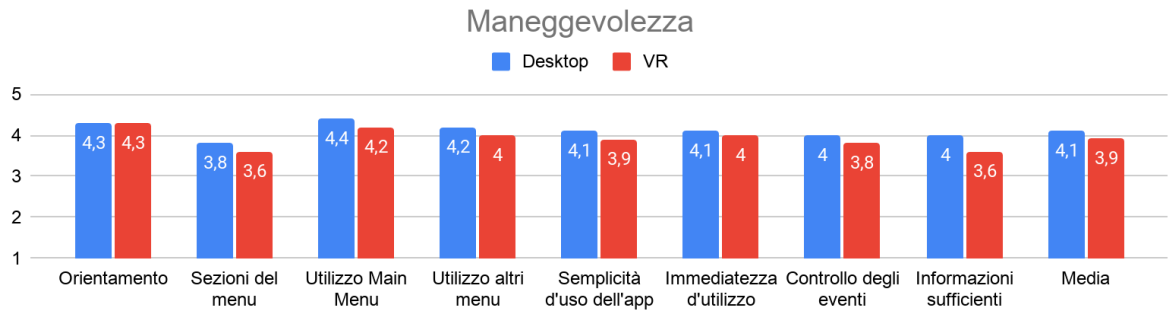


FIGURA 5.20: GRAFICO CON MEDIA RISPOSTE MERITO ALLA MANEGGEVOLEZZA

Nello specifico, in merito alla ***Maneggevolezza***, i fattori più critici risultano essere: *comprensione della suddivisione del Main Menu*, infatti è stato ritenuto da alcuni utenti non immediato e si è notato che, nei primi momenti di utilizzo, saltuariamente veniva confuso ciò che erano gli strumenti per agire sul veicolo e sull'ambiente, e ciò che invece faceva parte del configuratore del veicolo; *controllo degli eventi*, problema legato quasi esclusivamente alla UI VR e dovuto, come detto in precedenza, sia all'inesperienza dei tester che oggettivamente ad alcune interazioni più complesse; *informazioni sufficienti*, che pur essendo state inserite in gran misura in VR per fornire maggiori feedback sull'operato dell'utente, risultano ancora poche ed è evidentemente necessario aggiungerne altre per guidarlo al meglio.

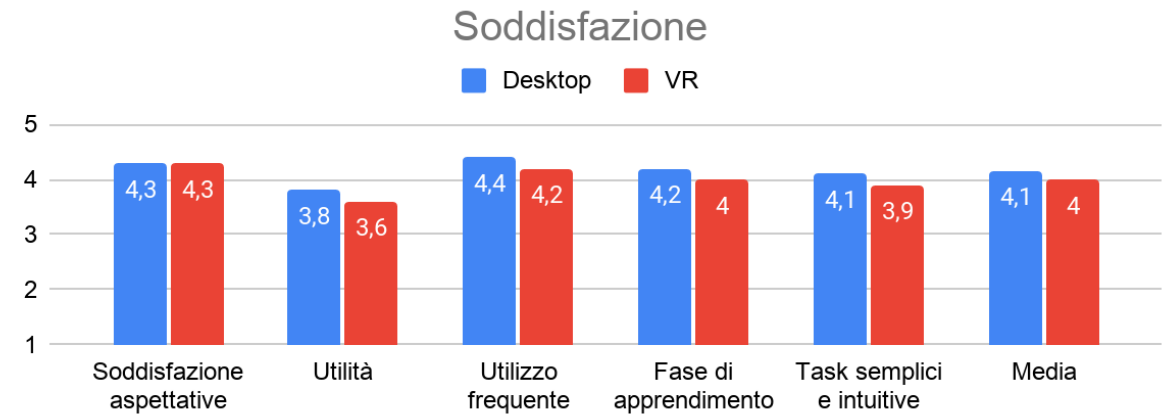


FIGURA 5.21: GRAFICO CON MEDIA RISPOSTE MERITO ALLA SODDISFAZIONE

Nel complesso i fattori inerenti alla ***Soddisfazione*** risultano essere positivi, con una media uguale o superiore a **4**. Il fattore con punteggio inferiore è stato quello legato all'*utilità dell'applicazione*. Questo dato rispecchia la visione attuale della Realtà Virtuale, ovvero una nuova tecnologia che risulta essere affascinante, ma che ancora fatica ad essere considerata utile, ponendo noi e tutti gli sviluppatori nella condizione di dover lavorare nell'ottica di costruire applicazioni che riescano a dare un valore aggiunto a questa tecnologia affinché venga reputata dall'utenza utile anziché solamente divertente.

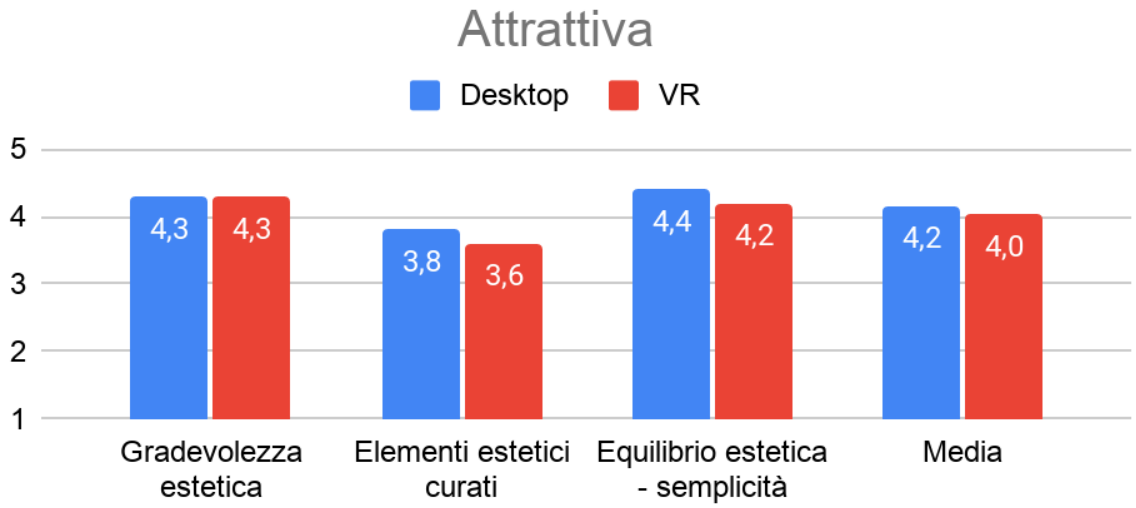


FIGURA 5.22: GRAFICO CON MEDIA RISPOSTE MERITO ALL'ATTRATTIVA

In merito ai fattori di ***Attrattiva***, anch'essi con una media uguale o superiore a **4**, si può notare come il fattore critico sia stato la *cura degli elementi estetici*. Come anticipato nel sottocapitolo **5.1**, per i test è stata utilizzata una versione semplificata, e quindi più fluida, dell'applicativo, avente un ambiente e un autoveicolo più semplici, oltre che una illuminazione piuttosto basilare. Una applicazione di questo tipo è chiaro che pecchi di estetica e fotorealismo, dettagli già migliorati nella versione dell'applicativo con la *Stelvio* all'interno di un nuovo *showroom* più curato, ma che comunque necessitano di ulteriori aggiornamenti. Va sottolineato che la complessità di ottenere una applicazione molto fotorealistica per VR non è assolutamente paragonabile a quella per Desktop, nettamente inferiore, poiché nel primo caso cresce drasticamente la richiesta di capacità computazionale della scheda video e del computer in genere. L'hardware non è ancora in grado di sostenere una VR totalmente fotorealistica ed è infatti uno dei problemi che ancora affligge questa nuova tecnologia, come è stato descritto nel sottocapitolo **1.1.2**.

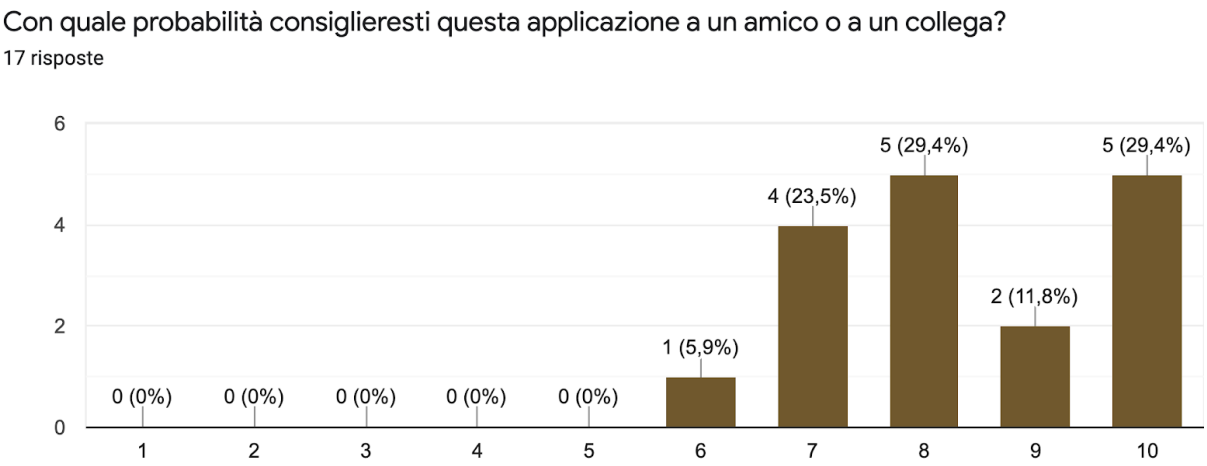


FIGURA 5.23: RISPOSTE ALLA DOMANDA DI NET PROMOTER SCORE

In coda alle domande relative all'attrattiva è stata inserita la domanda di ***Net Promoter Score***, dalla quale risultano **7** utenti come ***promotori***, vale a dire il **41,2**% dei tester, seguono **9** utenti ***neutri***, che coprono il **52,9**% delle persone sottoposte al test e infine **un solo** utente classificabile come ***detrattore***, seppur il suo voto sia stato pari a **6**, dunque non gravemente negativo.
Segue infine la sezione dedicata all'***intervista***, per la quale non ci siamo limitati a riportare esclusivamente dati numerici, trattandosi perlopiù di domande aperte, ma abbiamo cercato di riassumere al meglio la grande quantità di feedback datici dai tester.
Le prime domande riguardavano l'esperienza in generale con un HMD e la Realtà Virtuale, volte a confermare (o smentire) lo stato dell'arte, consapevoli, già da prima del test preliminare, che pochi dei tester erano soliti all'utilizzo di applicazioni in VR. In primis è stato riscontrato che gli utenti hanno valutato in modo molto positivo l'esperienza con il visore e tutti hanno dichiarato di volerlo utilizzare di nuovo in futuro.

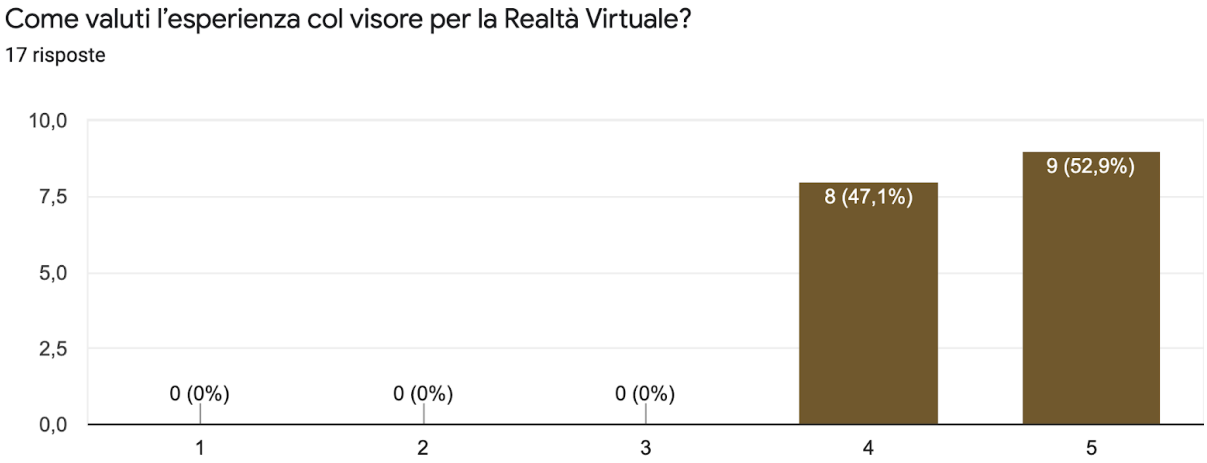


FIGURA 5.24: RISPOSTE ALLA DOMANDA DI VALUTAZIONE DELL'ESPERIENZA

Alla richiesta di descrivere l'esperienza e sottolineare cosa avesse attirato la loro attenzione, risalta la sorpresa per il forte senso di immersione provato, per alcuni inizialmente quasi disorientante, dovuto ad una completa libertà di movimento che li ha indotti a percepire di essere fisicamente da un'altra parte. Molti hanno sottolineato come, dopo una breve fase di apprendimento iniziale, siano stati sorpresi di quanto fosse immediata l'interazione, poiché molto simile a quella reale. Inoltre ha attratto molto la verosimiglianza dell'ambiente e dell'autoveicolo, la velocità di risposta dell'applicazione e alcune precise funzionalità come disegnare in 3D o la possibilità di avere il Main Menu sempre letteralmente a portata di mano. Questo per noi è stato estremamente positivo, poiché nonostante avessimo utilizzato una versione semplificata dell'applicativo e fossimo già a conoscenza di avere ancora molto margine di miglioramento, gli utenti hanno apprezzato comunque quanto fatto finora.
Per quanto riguarda gli aspetti negativi, alcuni lamentano scomodità dell'HMD e un senso di leggera nausea e vertigini quando venivano eseguiti dei movimenti molto rapidi. Con ciò dunque si conferma quanto analizzato nello stato dell'arte della VR: che risulta essere una nuova tecnologia affascinante, fortemente immersiva, in grado di costruire realtà verosimili nelle quali si può anche sfuggire alle leggi che governano la vita quotidiana, rendendola ancora più interessante, ma ancora soggetta a forti limitazioni, come la scomodità e i sintomi di Cybersickness.
Successivamente abbiamo posto domande più inerenti alla nostra applicazione, per sondare al meglio gli aspetti di forza e debolezza, chiedendo cosa fosse piaciuto di più o di meno, cosa si volesse cambiare e cosa non fosse stato per niente chiaro.
Tra gli aspetti fortemente positivi riportati dai tester troviamo: il già citato senso di immersione, l'intuitività e l'interattività dell'applicazione e dei relativi menu, la fedeltà a gesti quotidiani per l'utilizzo dei Tool e il realismo del veicolo e delle modifiche su di esso. Sono stati apprezzati molto, oltre ai feedback aptici e la grafica in generale, sia il Main Menu sul dorso della mano che i vari menu posizionabili a piacere nell'ambiente.
Tra ciò che è risultato sgradito, non chiaro o che gli utenti cambierebbero, risaltano principalmente tre aspetti: *l'utilizzo dei controller*, *l'interazione con i Tool* e *la mancanza di feedback*. Per quanto riguarda l'interazione con i Tool è interessante notare come questa sia stata citata dai tester sia in positivo che in negativo, ovvero che è stata apprezzata l'interazione con degli strumenti in 3D, anche se questa nel suo insieme, per come è stata progettata, è risultata di non facile comprensione e utilizzo, come riscontrato anche dall'analisi delle Task. La percepita mancanza di feedback è probabilmente correlata al complesso uso dei controller, poiché

seppur i comandi disponibili fossero pochi e semplici, il fatto che l'utente non potesse guardarsi le mani per capire dove fossero collocati i pulsanti sul controller è stato più rilevante e problematico di quanto pensassimo, specialmente nella fase di apprendimento iniziale con soggetti neofiti della VR e probabilmente senza un background da gamer.
In questa versione dell'applicativo non vi era un tutorial funzionante, per cui abbiamo chiesto direttamente ai tester se ritenessero importante inserirlo e in che formato potesse essere più facilmente fruibile.

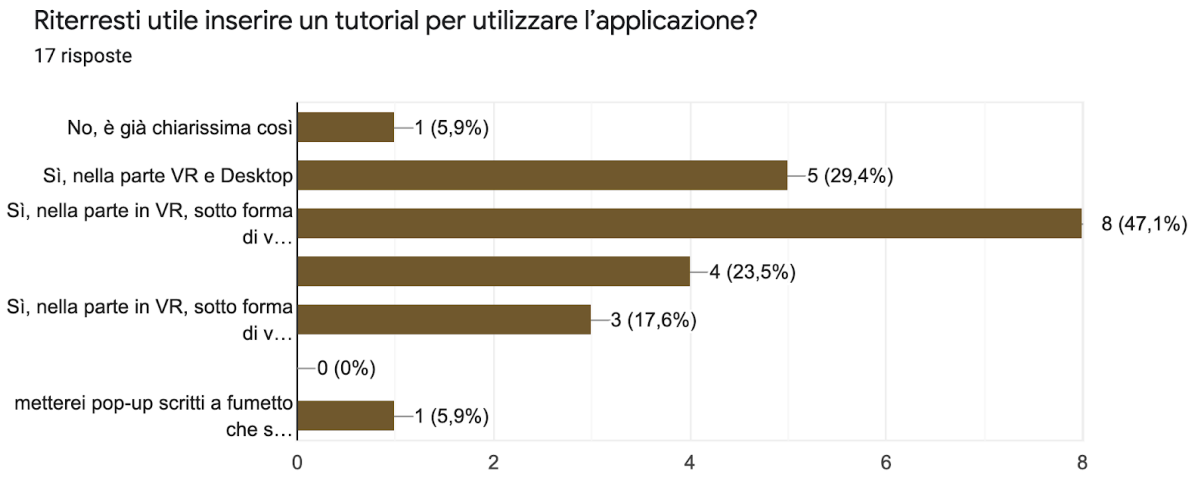


FIGURA 5.25: RISPOSTE ALLA DOMANDA IN MERITO ALL'INSERIMENTO DI UN TUTORIAL

La domanda era a risposta multipla, con possibilità per ogni tester di dare più di una risposta. La quasi totalità degli utenti ha suggerito di inserire un tutorial non obbligatorio, richiamabile secondo necessità, preferibilmente sotto forma di breve video piuttosto che testuale. Un utente ci ha consigliato di fare un compromesso tra un tutorial facoltativo e uno obbligatorio, suggerendoci di inserire dei pop-up che appaiono al primo utilizzo di una qualunque funzionalità.
L'ultima parte dell'intervista si compone di domande mirate a sondare ogni singola parte dei menu in VR, lasciando i tester liberi di sottolineare tutti gli aspetti da loro percepiti come critici che magari non si erano manifestati durante lo svolgimento delle Task.
Per la quasi totalità degli utenti il Main Menu si è rivelato semplice e intuitivo, anche se, quando veniva iconizzato, non sempre era facilmente individuabile. Quasi tutti i tester concordavano anche sull'intuitività di Tablet e Dashboard delle varianti, seppur riguardo quest'ultima vi fossero pareri discordanti su chi la riteneva ridondante e chi invece migliore del Tablet per intuitività di utilizzo.
Contrariamente a quanto riscontrato sia con le Task che con le precedenti domande meno specifiche, anche per quanto riguarda la Dashboard dei Tool e l'utilizzo di quest'ultimi la

quasi totalità degli utenti ne definisce l'utilizzo come intuitivo e non complesso. Però in questo caso sono stati lasciati un maggior numero di consigli in merito, riguardanti soprattutto il semplificare la serie di azioni necessarie ad attivare un Tool, l'esigenza di tenere premuto il tasto Grip per mantenere la presa sull'oggetto e alcune critiche sui due Tool posti nel cassetto, definiti difficilmente raggiungibili e utilizzabili, come è stato evidenziato anche dall'analisi delle Task. Tre utenti, in merito all'utilizzo in generale di questa Dashboard, hanno lamentano il fatto di saper cosa dovessero fare ma non aver ben chiaro come farla.
L'ultima domanda dell'intervista chiedeva ai tester di esprimere liberamente qualunque altro commento o consiglio in merito all'applicazione. Oltre ai diversi aspetti già analizzati in precedenza, vi sono stati ulteriori consigli in merito a: miglioramento di estetica e dell'audio ambientale; spostamento nello spazio sotto forma di camminata attivabile da thumbstick che, pur essendo consci che indurrebbe maggiore cyber-sickness, potrebbe comunque permettere una doppia modalità di movimento; maggiore interazione con l'autoveicolo, come l'opportunità di esplorarne gli interni e di aprirne e chiuderne le portiere; possibilità di annullare l'ultima azione fatta; disponibilità di cambiare lingua all'interfaccia.

CONCLUSIONI

RISULTATI OTTENUTI

Gli obiettivi prepostoci con questo progetto di tesi erano volti alla creazione di un applicativo che fosse: **modulare, multipiattaforma** ed incentrato sulla **user experience.**
In merito alla **modularità** sono stati fatti diversi test con modelli differenti, dai quali è emerso che il nostro applicativo, basato sul plugin di Dead Pixels, è riuscito a supportare tranquillamente qualsiasi modello inserito, a patto che questo venisse prima semplificato in termini di complessità poligonale. A livello prestazionale possiamo concludere che ci siano ancora buoni margini di miglioramento, infatti si potrebbe ottimizzare il programma in modo che restituisca un'esperienza più fotorealistica. Abbiamo constatato però che la struttura costruita per recuperare le informazioni da *Variant Manager* e *Data Table* risulta comunque stabile e funzionante con diversi tipi di modelli.
Così com'è **personalizzabile** il modello di autoveicolo e le relative varianti di configurazione, lo è anche la grafica di entrambe le UI che è stata resa facilmente customizzabile per qualunque potenziale cliente. Non sono stati fatti test di usabilità dell'Editor costruito, sia perché si tratta ancora di un prototipo, che per l'impossibilità di reperire degli sviluppatori o designer, vale a dire il target di tale strumento. Basandoci esclusivamente sulle osservazioni fatteci dall'azienda e sul nostro personale utilizzo dell'Editor possiamo concludere che ci siano anche qui possibilità di miglioramento. Di fatti il sistema ora adottato, con le anteprime interattive e con una logica di memorizzazione dati che si appoggia su *Data Asset*, risulta efficace e immediato, riuscendo a recuperare in poco tempo tutte le informazioni necessarie.
Per la fase di User Test è stato utilizzato come visore un Oculus Rift, collegato a un computer con sistema operativo Windows, benché il programma sia in grado di funzionare anche con HTC VIVE e MacOS, potendolo definire a tutti gli effetti **multipiattaforma**. È d'obbligo però una precisazione sul sistema operativo di Apple, in quanto, allo stato attuale, Unreal Engine risulta sì compatibile ma per nulla ottimizzato per tale OS. Inoltre per quest'ultimo è stato possibile creare degli eseguibili, ma funzionanti esclusivamente in modalità Desktop, poiché Oculus Rift non risulta compatibile con il sistema operativo in questione. Abbiamo riscontrato anche seri problemi con l'ultima versione del nostro applicativo, tali per cui ci impediscono di aprire il file di progetto, nel quale viene utilizzato un sistema di Shading di tipo Deferred non supportato da MacOS.

A livello di **compatibilità**, durante il passaggio da UI Desktop a VR, o viceversa, l'applicativo mantiene intatte le modifiche apportate dall'utente, sia al configuratore che all'ambiente. In questo modo viene data piena libertà di utilizzare l'applicazione con l'interfaccia utente o la piattaforma che si ritiene più consona ai propri scopi.
Essendoci concentrati principalmente sulle Interfacce Utente Desktop e VR dell'applicazione, gli aspetti più curati e testati risultano inevitabilmente essere l'**Usabilità** e la **User Experience**.
Alla luce dei risultati ottenuti è risultata ottimale la scelta di costruire una **UI Desktop** semplice e meno invasiva possibile. I tester, anche coloro che hanno dichiarato di avere poca dimestichezza nella navigazione di ambienti con mouse e tastiera, hanno trovato molto intuitivo l'utilizzo di questa interfaccia utente. Dopo una breve fase di orientamento iniziale, tutti hanno compreso più o meno facilmente la distinzione tra il *Tools Menu* e il *Configurator Menu*. Concludiamo dunque che, in merito a questa UI, non siano stati riscontrati errori critici, che possa essere migliorata in alcuni dettagli, ma che nel complesso risulta efficace ed efficiente secondo gli obiettivi definiti.
Per quanto riguarda la **UI VR** invece trarre conclusioni è risultato più complesso a seguito della grande quantità di informazioni ottenuta dai test. Non vogliamo peccare di supponenza definendo i risultati ottenuti come universalmente validi, poiché abbiamo riscontrato quanto vi siano innumerevoli fattori in grado di condizionare la percezione e prestazione dell'utente quando si tratta di interagire con questa nuova tecnologia. Tenendo conto di questi fattori soggettivi, abbiamo riscontrato sia alcune costanti di comportamento tra i tester, che errori più o meno critici di progettazione dell'applicazione.
Durante i test, spesso gli utenti tendevano a non esprimere o sminuire alcune difficoltà riscontrate attribuendole esclusivamente alla loro poca esperienza con l'interazione in Realtà Virtuale, per cui in alcuni casi è risultato complesso scindere ciò che era soggettivo rispetto al tester e il suo background e ciò che invece era una problematica oggettiva dell'applicativo. L'aver sottoposto i tester ad un questionario preliminare ci ha permesso di avere una profilazione più approfondita del loro background, risultato fondamentale nel comprendere alcune evidenti differenze riguardo le percentuali di successo e le difficoltà percepite durante lo svolgimento delle Task. Risulta logico che chi ha dichiarato numerose esperienze pregresse con la VR si sia trovato maggiormente a suo agio con il nostro applicativo. Abbiamo però riscontrato che utenti con poca esperienza in Realtà Virtuale, ma con un background da fruitore quotidiano di tecnologia o da gamer hanno affrontato con più facilità i compiti richiesti.

Da gran parte degli utenti è ritenuta essenziale la presenza di un **Tutorial in VR**, seppur non invasivo o obbligatorio, poiché in molti, senza di esso, faticavano a ricordare i comandi di interazione. Riteniamo che ciò sia dato, oltre che dalla poca esperienza o confidenza con questa tecnologia, anche dall'impossibilità di vedere i controller quando si indossa l'HMD. Siamo giunti alla conclusione che, seppur in VR vi siano molti più feedback che non in Desktop, sia comunque necessario fornire molte più informazioni di quante non ve ne siano già, preferendo piuttosto essere ridondanti, facendo però attenzione a non risultare eccessivamente invasivi. Per quanto riguarda l'interazione con i Menu abbiamo già predisposto un pulsante Tutorial su ognuno di essi, che sarà adibito ad aprire un apposito breve video che ricordi le interazioni base di quel preciso Menu. Per aiutare l'utente a ricordare dove siano posizionati i vari pulsanti e in generale facilitare l'utilizzo dei controller, intendiamo apportare opportune migliorie alle mani virtuali in modo che diano ulteriori feedback e supporto all'interazione.

Si è rivelata corretta la nostra supposizione che per la gestione di grandi quantità di dati, come nel caso del configuratore o delle impostazioni dell'applicazione, fosse efficace la scelta di riproporre una logica bidimensionale, rappresentando tali menu attraverso un **Tablet** in VR, infatti la totalità dei tester non ha avuto problemi ad utilizzarlo. In merito a ciò concludiamo che, seppur sia generalmente corretto sfruttare al meglio lo spazio tridimensionale, vi sono casi in cui non è sbagliato ri-proporre logiche di interazione appartenenti al mondo Desktop, specie se si ha a che fare con dati astratti e non facilmente rappresentabili in 3D.

Anche la scelta di non sacrificare le varianti ad una mera rappresentazione bidimensionale è stata corretta, seppur in questo caso gli utenti abbiano espresso pareri discordanti. Alcuni di loro hanno ritenuto la presenza della **Configurator Dashboard** ridondante, mentre altri estremamente utile e anche più intuitiva del Tablet. È risultato interessante come ad alcuni tester, durante l'utilizzo di questa versione "aumentata" delle varianti, venisse spontaneo prenderle e appoggiarle, o addirittura lanciarle, sulla vettura. È un tipo di interazione molto comune nei videogiochi, che effettivamente sfrutta al meglio l'ambiente tridimensionale a disposizione rispetto a quella del pulsante "Load Asset". Quanto detto, insieme ai diversi consigli raccolti con il questionario post-esperienza, ha sottolineato la volontà di alcuni tester di interagire maggiormente con il veicolo, il quale al momento risulta ancora relativamente statico. Questa condizione di staticità si deve probabilmente alla difficoltà di progettare una interazione che possa essere valida per qualunque tipologia di prodotto inserito e mantenere, allo stesso tempo, la modularità dell'applicativo.

Come ampiamente descritto nell'analisi dei test, gli aspetti più critici riscontrati riguardavano il **Tools Menu**. L'interazione con gli strumenti messi a disposizione è stata strutturata ispirandosi a gesti della realtà quotidiana, ri-adattandoli per essere svolti nell'ambiente virtuale. Dai risultati ottenuti possiamo concludere che questa interazione risulta efficace fintanto che il gesto da compiere non vada a sommarsi ad altri fino a risultare eccessivamente complesso. Nello specifico, per quanto riguarda il gesto di Grab, si è riscontrato che non creasse problemi fintanto che si limitava al semplice spostamento di Tablet o Dashboard, mentre invece ne generasse diversi in caso venisse applicato ai Tool, i quali impongono uno sforzo cognitivo maggiore, inducendo l'utente a dimenticarsi di tenere premuto il tasto Grip e perdere di conseguenza la presa.
Oltre a questo è stata richiesta una semplificazione dell'iter di attivazione di un Tool, eliminando la necessità di selezionarlo prima di poterlo prendere, poiché l'interazione, così strutturata risulta ridondante e complessa. Non è sbagliata pertanto la scelta di rimanere fedeli a gesti quotidiani, a patto di scendere a compromessi quando l'interazione diventa eccessivamente macchinosa, in modo da non farla risultare snervante pur di mantenerla "realistica".

SVILUPPI FUTURI

L'applicativo da noi progettato e implementato si basa sul Collab Viewer Template, ciò comporta che anche nel nostro eseguibile vi sia la possibilità di creare una sessione di gioco alla quale possono accedere fino a 10 utenti. Partire da questo Template, che già gestisce le logiche di sessione e di replicazione dei Tool, permetterà in futuro di sviluppare più agevolmente anche la corretta replicazione degli eventi legati alle nostre UI e al plugin di Dead Pixels..
Da un test in multipresenza con il Team di Dead Pixels abbiamo infatti verificato la possibilità di entrare con più utenti nella stessa sessione di gioco, utilizzando modalità di navigazione e UI diverse, potendo interagire fra noi. Allo stesso modo però sono emersi problemi di costruzione delle UI per gli utenti che accedevano alla sessione creata dell'utente Master, per il quale invece era tutto perfettamente funzionante. Inoltre tutti gli utenti potevano vedere gli effetti dei Tool utilizzati dal Master, ma non le modifiche apportate da esso alla configurazione del veicolo. Pertanto un possibile sviluppo futuro consiste nell'implementare da un lato in entrambe le UI la corretta replicazione degli eventi, dall'altro progettare un sistema di feedback tra utenti ed eventualmente la possibilità di collaborazione e condivisione dei menu in VR.

Quest'ultimo ultimo aspetto è dato dal fatto che l'interfaccia utente in realtà virtuale sia costruita col supporto di *Actor* fisicamente rappresentati nell'ambiente e potenzialmente condivisibili tra più utenti. Per esempio l'utente Master potrebbe aprire il proprio Tablet per la configurazione e in un secondo momento passarlo ad un secondo utente, che prosegue il lavoro da lui iniziato.
Oltre alla multiutenza, vi possono essere tutta una serie di miglioramenti più legati a quanto già implementato. Per esempio si potrebbe ampliare l'Editor UI, al punto da rendere le interfacce utente quasi totalmente personalizzabili e permettere di cambiare la lingua utilizzata nell'applicativo. Una personalizzazione più ampia darebbe la possibilità di decidere quali Tool mettere a disposizione e con quali mesh rappresentarli, di ampliare il numero di ambienti o di punti di Teleport e via dicendo. Inoltre, rispetto allo stato attuale, si possono migliorare le interazioni già presenti, specialmente quelle riguardanti il Tools Menu.
Infine un altro possibile sviluppo riguarda l'ampliamento dell'applicativo con una interfaccia utente per Mobile. Poiché un'applicazione come quella da noi implementata può risultare molto pesante, in questo caso è auspicabile sfruttare un nuovo strumento messo a disposizione da Unreal Engine: **Pixel Streaming**. Esso permette di eseguire il programma su un PC desktop o un server nel cloud, demandando ad esso l'onere di computazione, per poi trasmettere in streaming i frame e l'audio renderizzati a browser e dispositivi mobile tramite WebRTC, evitando agli utenti di scaricare o installare l'applicazione[55].

BIBLIOGRAFIA E SITOGRAFIA

[1] Hisour, *Immersione nella Realtà Virtuale*, https://www.hisour.com/it/immersion-virtual-reality-21313/.

[2] Stage, *VR has disappeared from the Gartner hype cycle 2018*, https://blog.vr-on.com/en/vr-has-disappeared-from-the-gartner-hype-cycle-2018.
Sandrine Lasserre, *VR Gartner hype cycle: Is virtual reality hype or hope?*, 10 Novembre 2020, https://blog.techviz.net/gartner-hype-cycle-virtual-reality-hype-or-hope.

[3] HT TECH, *Mark Zuckerberg says smart glasses will allow people to teleport*, 9 Marzo 2021, https://tech.hindustantimes.com/tech/news/mark-zuckerberg-says-smart-glasses-will-allow-people-to-teleport-71615268609099.html.
Account Instagram worldy.it: https://www.instagram.com/p/CNAzzK9sFGQ/.

[4] Oculus, *Facebook Horizon invite-only Beta is ready for virtual Explorers*, 27 Agosto 2020, https://www.oculus.com/blog/facebook-horizon-invite-only-beta-is-ready-for-virtual-explorers/.

[5] AcademyXi, *Why Virtual Reality is the next big thing*, https://discover.academyxi.com/blog/virtual-reality-next-big-thing/.

[6] Tom Warren, *Apple's first VR headset reportedly includes a fabric design, a fan, and expensive price tag*, 21 Gennaio 2021, https://www.theverge.com/2021/1/21/22242131/apple-vr-headset-2022-fabric-design-fan-price-details-rumors.
Andrea Guerriero, *Visore VR e AR da Apple: anche la mela sperimenta nella realtà* virtuale, 25 Settembre 2020, https://www.optimagazine.com/2020/09/25/visore-vr-e-ar-da-apple-anche-la-mela-sperimenta-nella-realta-virtuale/1936516.

[7] Tim Merel, *The reality of augmented and virtual reality venture capital*, 25 Maggio 2016, https://techcrunch.com/2016/05/24/the-reality-of-augmented-and-virtual-reality-venture-capital/.

[8] Pat O'Connor, *Is 2020 the point at which VR finally delivers on years of promise?*, 10 Febbraio 2020, https://eandt.theiet.org/content/articles/2020/02/is-2020-the-year-vr-finally-delivers-on-years-of-promise/.

[9] Juniper Research: https://www.juniperresearch.com/home.

[10] Beecham Research: www.beechamresearch.com/.

[11] Gartner: https://www.gartner.com/en.

[12] Francesco la Trofa, *Realtà Virtuale: oltre le barriere della trasformazione digitale*, 1 Aprile 2019, https://3dstories.protocube.it/tecnologie-innovative-barriere-trasformazione-digitale/.

[13] OpenXR by khronos: https://www.khronos.org/openxr/.
Massimo Morselli, *Rilasciato OpenXR 1.0, l'unificazione dell'ecosistema VR è più vicina*, 30 Luglio 2019, https://www.vr-italia.org/rilasciato-openxr-1-0-lunificazione-dellecosistema-vr-e-piu-vicina/.

[14] smartworld, *Miglior Visore VR – Aprile 2021*, Aprile 2021, https://www.smartworld.it/miglior-visore-vr.

[15] Manuel Micaletto, *HTC non molla: è in arrivo un nuovo visore VR stand alone per sfidare Oculus Quest 2*, 23 Ottobre 2020, https://www.hdblog.it/indossabili/articoli/n528492/htc-vive-visore-vr-proton-stand-alone-oculus-quest/.

[16] Teo Petruz, *Cybersickness, o nausea da realtà virtuale*, 17 Giugno 2018, https://www.horizonpsytech.com/2018/06/17/cybersickness-o-nausea-da-realta-virtuale/.

[17] Maria Gallagher, *Cybersickness: A Multisensory Integration Perspective*, Febbraio 2018, https://www.researchgate.net/publication/322400282_Cybersickness_A_Multisensory_Integration_Perspective.

[18] Wonjun Park, Hayoung Heo, Seongjun Park, Jinmo Kim, *A Study on the Presence of Immersive User Interface in Collaborative Virtual Environments Application*, 5 Marzo 2019, https://www.mdpi.com/2073-8994/11/4/476.

[19] Sol Rogers, *Why Is Presence Important For Virtual Reality?*, 4 Novembre 2017, https://www.vrfocus.com/2017/11/why-is-presence-important-for-virtual-reality/.

[20] Stereoscopia - Wikipedia: https://it.wikipedia.org/wiki/Stereoscopia#Fotocamera_stereoscopica.

[21] VRTL Academy, *Stereoscopic Vision and Virtual reality*, https://courses.vrtl.academy/lessons/whats-up-with-stereoscopic-and-virtual-reality/.

[22] Hansung Kim, Luca Remaggi, Philip J.B. Jackson, Adrian Hilton, *Immersive Spatial Audio Reproduction for VR/AR Using Room Acoustic Modelling from 360◦ Images*, https://core.ac.uk/download/pdf/187773242.pdf.

[23] bHaptics: https://www.bhaptics.com/tactsuit

[24] Teslasuit: https://teslasuit.io

[25] ivi 18

[26] Stephan Streuber, Astros Chatziastros, *Human Interaction in Multi-User Virtual Reality*, Gennaio 2007, https://www.researchgate.net/publication/41781665_Human_Interaction_in_-Multi-User_Virtual_Reality.

[27] The Wild: https://thewild.com.

[28] Resolve, *What is InsiteVR?*, 15 Settembre 2016, https://www.youtube.com/watch?v=DDf5SxpJcAQ

[29] Holodeck - NVIDIA Project:
- *Get a Tour of NVIDIA Holodeck*, 9 Agosto 2018, https://youtu.be/KcWZFpjjD34
- *NVIDIA Announces Project Holodeck*, 12 Maggio 2017, https://youtu.be/hUsP7fsjrdg
- *NVIDIA Holodeck: Photorealistic Collaborative Design in VR*, 14 Novembre 2017, https://youtu.be/goFZOTjCVFg.

[30] Unreal Engine, *BMW Brings Mixed Reality to Automotive Design | Project Spotlight | Unreal Engine*, 25 Aprile 2018, https://youtu.be/_A90IdpFe3o.

[31] All thing Media, *Multi-user Virtual Reality Experience*, 17 Gennaio 2019, https://youtu.be/orqKz7RcsBk.

[32] Scott W. Greenwald, Wiley Corning, Pattie Maes, *Multi-User Framework for Collaboration and Co-Creation in Virtual Reality, 2017*, https://dspace.mit.edu/handle/1721.1/108440
CocoVerse - MIT Media Lab: Scott Greenwald, *CocoVerse: A playground for co-creation and communication in virtual reality*, https://www.media.mit.edu/projects/cocoverse/overview/.

[33] Gravity Sketch, *Tool Belt - Tools Tutorial Gravity Sketch VR*, 22 Gennaio 2020, https://youtu.be/gF0VdSAIyzY.

[34] Epic Games: https://www.epicgames.com/site/it/home.

[35] Collab Viewer Template Documentation: https://docs.unrealengine.com/en-US/Resources/Templates/CollabViewer/index.html
徐紹容, *Unreal Engine - Collab Viewer Explode Bike*, 17 Dicembre 2020, https://www.youtube.com/watch?v=Hy0Gghh8AvM.

[36] ben ui, Data-driven Design in Unreal: https://benui.ca/unreal/data-driven-design/

[37] Blueprint Interface Documentation: https://docs.unrealengine.com/en-US/ProgrammingAndScripting/Blueprints/UserGuide/Types/Interface/index.html
Implementing Blueprint Interfaces Documentation: https://docs.unrealengine.com/en-US/ProgrammingAndScripting/Blueprints/UserGuide/Types/Interface/UsingInterfaces/index.html.

[38] Unreal Engine Forum: https://forums.unrealengine.com/development-discussion/c-gameplay-programming/72917-why-have-a-hud-class
https://forums.unrealengine.com/development-discussion/blueprint-visual-scripting/30025-what-is-the-appropiate-relationship-between-hud-and-widgets?59153-What-is-the-appropiate-relationship-between-HUD-and-Widgets=.

[39] Chris Burras, *Vehicle Configurator Darul Solutions*, 21 Luglio 2018,
https://www.youtube.com/watch?v=3rXzXo9ZaI4
YuanWu, *UE4 Aston Martin one77 Car configurator Demo*, 15 Novembre 2017,
https://www.youtube.com/watch?v=MJSdoIaOzQo
Ryan Duffield, *Lamborghini Huracan Configurator - UE4*, 8 Giugno 2019,
https://www.youtube.com/watch?v=1EbaBVugt3s
Syuya Mukai, *UE4 Porsche Car Configurator*, 1 Novembre 2018,
https://www.youtube.com/watch?v=VJIwZ8lsclw.

[40] Joe Connolly, *VR Design Review #2: Menus*, 27 Luglio 2018, https://blog.sketchbox3d.com/vr-design-review-2-menus-b0d7ddc3078.

[41] Configuratore Stelvio Quadrifoglio: https://www.alfaromeo.it/configuratore/stelvio-quadrifoglio/#/version/83630QAR2000/exteriors.

[42] Neuro Webdesign, *Usability Test: meglio un'analisi qualitativa o quantitativa?*, 31 Agosto 2020, https://www.neurowebdesign.it/it/usability-test-analisi-quantitativa-qualitativa/
Raluca Budiu, *Quantitative vs. Qualitative Usability Testing*, 1 Ottobre 2017, https://www.nngroup.com/articles/quant-vs-qual/.

[43] Roy S. Kalawski, *VRUSE—a computerised diagnostic tool: for usability evaluation of virtual/synthetic environment systems*, Febbraio 1999, https://www.sciencedirect.com/science/article/abs/pii/S0003687098000477.

[44] Merche Gómez Sánchez, *Virtual Reality User Testing*, 26 Gennaio 2018, https://widux.com/virtual-reality-user-testing/.

[45] Ana Pavuna, *A fundamental guide to user testing in Virtual Reality,* 14 Maggio 2019, https://medium.com/ostmodern/a-fundamental-guide-to-user-testing-in-virtual-reality-a71c85c764c0.

[46] Tímea Falmann, *VR In UX Research: All You Need To Know About VR User Testing*, 4 Luglio 2018, https://uxstudioteam.com/ux-blog/vr-in-ux-testing/.

[47] Dionysios Georgios Papadimitriou, *User Experience Evaluation in Virtual Reality*, Novembre 2019, https://trepo.tuni.fi/bitstream/handle/10024/118553/PapadimitriouDionysios.pdf.

[48] Ministro per la Pubblica Amministrazione, *Gruppo di Lavoro per l'Usabilità (GLU),* ultimo aggiornamento 3 Febbraio 2021, http://www.funzionepubblica.gov.it/glu.
Designers Italia, *Usability test,* https://designers.italia.it/kit/usability-test/.

[49] boraso, *8 modi per misurare la customer satisfaction (e migliorare la UX),* 17 Ottobre 2017, https://blog.boraso.com/8-modi-per-misurare-la-customer-satisfaction-e-migliorare-la-ux.

[50] Hannah Walter, Ruixuan Li, Justin Munafo, Christopher Curry, Nicolette Peterson, Thomas Stoffregen, *APAL Coupling Study 2019,* 3 Aprile 2019, https://conservancy.umn.edu/handle/11299/201892.

[51] ivi 47

[52] ivi 49

[53] Francesco di Nocera, *Usability Evaluation 2.0: Una descrizione (s)oggettiva dell'usabilità,* Dicembre 2013, https://www.researchgate.net/publication/268743694_Usability_Evaluation_20_Una_descrizione_soggettiva_dell%27usabilita.

[54] ivi 50

[55] Pixel Streaming Documentation: https://docs.unrealengine.com/en-US/SharingAndReleasing/PixelStreaming/index.html.

APPENDICE A - QUESTIONARIO PRELIMINARE

Questionario preliminare

Ti stiamo chiedendo di partecipare ad un Test di Usabilità che è parte del nostro progetto di tesi. Aderire a questo studio è su base volontaria. Partecipando a questo Usability Test ci aiuterai a valutare l'usabilità e la User Experience del nostro applicativo sia nella sua versione Desktop che VR.

Con questo questionario preliminare vorremmo conoscere le tue esperienze pregresse con la Realtà Virtuale, così da poter tarare l'intera fase di test che seguirà. Pertanto ti invitiamo a rispondere alle domande nel modo più sincero possibile in quanto non sarai tu ad essere valutato, ma piuttosto noi e l'applicazione.

I risultati del test e dei vari questionari saranno riportati anonimamente e non vi sarà alcuna correlazione tra te e i dati rilevati. Per fare ciò durante l'intera sessione ti sarà assegnato un ID Partecipante che utilizzerai per compilare questionari e interviste che ti proporremo.

*Campo obbligatorio

1. ID Partecipante: *

_____

2. Età: *

_____

3. Sesso: *

Contrassegna solo un ovale.

- ◯ Uomo
- ◯ Donna
- ◯ Altro: _____

4. Professione: *

Specificare il relativo campo di studi e/o ambito di lavoro

_____

UTILIZZO DELLA TECNOLOGIA

5. Quante ore al giorno in media spendi davanti ad un computer, telefono, tablet? *

Contrassegna solo un ovale.

- ◯ 0-2 ore
- ◯ 3-4 ore
- ◯ 5-8 ore
- ◯ Più di 8 ore

6. Sei solito giocare a videogiochi per PC? Se Sì, quante volte? *

Contrassegna solo un ovale.

- ◯ No
- ◯ No, ma gioco a videogiochi per altre piattaforme (console, mobile, etc...)
- ◯ Sì, qualche volta al mese
- ◯ Sì, una volta a settimana
- ◯ Sì, 2/3 volte a settimana
- ◯ Sì, 4/5 volte a settimana
- ◯ Sì, tutti i giorno o quasi
- ◯ Altro: _____

Passa alla domanda 7.

ESPERIENZE PREGRESSE CON LA VR

7. Qual è il tuo livello di interesse nei confronti della Realtà Virtuale? *

Contrassegna solo un ovale.

- ◯ Non so cosa sia
- ◯ Non mi interessa perché non mi piace
- ◯ Indifferente
- ◯ Mi incuriosisce
- ◯ Molto interessato/a

8. Hai mai provato un'esperienza in Realtà Virtuale? *

Contrassegna solo un ovale.

- ◯ Sì *Passa alla domanda 9.*
- ◯ No *Passa alla domanda 18.*

FREQUENZA DI UTILIZZO DI UN VISORE

9. Quante volte hai utilizzato un visore per la VR? *



Contrassegna solo un ovale.

- 1
- 1-5
- 5-20
- 20+

10. In base alle tue esperienze pregresse, quanto sei in confidenza con i controller VR? *



Contrassegna solo un ovale.

	1	2	3	4	5	
Poco	◯	◯	◯	◯	◯	Molto

11. In base alle tue esperienza pregresse, quanto trovi semplice interagire con un ambiente VR? *

Contrassegna solo un ovale.

	1	2	3	4	5	
Molto complicato	◯	◯	◯	◯	◯	Molto semplice

12. In generale come ti reputi rispetto a tale tecnologia? *

Contrassegna solo un ovale.

	1	2	3	4	5	
Incapace	◯	◯	◯	◯	◯	Esperto

13. Possiedi un qualsiasi tipo di visore? *

Contrassegna solo un ovale.

- ◯ Sì *Passa alla domanda 14.*
- ◯ No *Passa alla domanda 18.*

VALUTAZIONE DELLE TUE ESPERIENZE CON LA VR

14. Quale modello di visore possiedi? *

Contrassegna solo un ovale.

- ◯ HTC VIVE
- ◯ HTC VIVE PRO
- ◯ LENOVO MIRAGE SOLO
- ◯ OCULUS QUEST
- ◯ OCULUS QUEST 2
- ◯ OCULUS GO
- ◯ OCULUS RIFT
- ◯ OCULUS RIFT S
- ◯ SAMSUNG GEAR VR
- ◯ HUAWEI VR2
- ◯ GOOGLE CARDBOARD
- ◯ SONY PLAYSTATION VR
- ◯ Altro: _____

15. Quali sono i principali difetti del tuo visore? *

Seleziona tutte le voci applicabili.

- [ ] Nessuno, lo trovo perfetto
- [ ] È troppo costoso
- [ ] Non ci sono abbastanza contenuti disponibili nello Store
- [ ] Nessuno dei miei amici lo possiede e non posso giocare con loro
- [ ] Penso che il visore sia scomodo
- [ ] Mi provoca nausea, vertigini o mal di testa
- [ ] L'esperienza di gioco non è migliore di un qualunque altro tipo di videogioco
- [ ] La qualità dell'esperienza offerta non ha rispettato le mie iniziali aspettative

Altro: ☐ _____

16. Alla luce dei difetti da te esposti, ne consiglieresti comunque l'acquisto? *

Contrassegna solo un ovale.

- ◯ Sì
- ◯ No

17. Se hai risposto Sì, motiva la tua risposta:

_____

_____

_____

_____

_____

PROBLEMI DI ACQUISTO DI UN VISORE

18. C'è un motivo particolare che non ti permette di acquistare un visore?

Seleziona tutte le voci applicabili.

- [ ] Nessuno, semplicemente non mi interessa così tanto
- [ ] Nessuno, ma ho intenzione di farlo prima o poi
- [ ] È troppo costoso
- [ ] Non ci sono abbastanza contenuti disponibili per la VR (giochi, applicazioni)
- [ ] Nessuno dei miei amici lo possiede e non potrei giocare con loro
- [ ] Penso che il visore sia scomodo
- [ ] Mi provoca nausea, vertigini o mal di testa
- [ ] L'esperienza di gioco non è migliore di un qualunque altro tipo di videogioco
- [ ] Sto aspettando che tecnologicamente si evolva perché ora non mi convince pienamente
- [ ] Non ho abbastanza spazio in casa
- [ ] Altro: _____

Questi contenuti non sono creati né avallati da Google.

Google Moduli

APPENDICE B - DEFINIZIONE TASK, SCENARIO E TEMPI DI ESECUZIONE

SEQ (Single Easy Question): da 1 a 10:

Quanto ritieni difficile questa task?

Definizione delle task per fase user testing

- *tutto ciò che si può fare nell'app*
    a. spostarsi nell'ambiente
    b. aprire tutti i menu
    c. Chiudere l'applicazione
    d. Spostare a piacimento le interfacce (Grab)
    e. Settings: Cambia modalità di navigazione
    f. Tool:
        i. Meter: prendere e cancellare misure nell'ambiente
        ii. X-Ray: rendere invisibili cose nell'ambiente e resettare quanto fatto
        iii. Paint: disegnare nell'ambiente e cancellare segni fatti
        iv. Move: spostare oggetti nell'ambiente e resettare quanto fatto
        v. Teleport: spostarsi in punti chiave definiti
        vi. Environment: Cambia ambientazione con luce X
        vii. Aprire cassetto
    g. Configuratore:
        i. Cambiare qualcosa nella macchina/veicolo
        ii. Visualizzare le informazione relative ad una variante **(sia tablet che asset dashboard)**
        iii. visualizzare varianti sia mediante Tablet che Asset Dashboard

TASK LIVELLO BASE *TASK LIVELLO MEDIUM* *TASK LIVELLO HIGH*

DESKTOP ***VIRTUAL REALITY***

D.a.1. Spostarsi nell'ambiente essendo a conoscenza che i pulsanti di interazione sono WASD e Mouse. (modalità FLY e WALK) (Base) (20 sec)

- *L'utente trova quali pulsanti usare su tastiera e mouse*
- *L'utente si muove e ruota nell'ambiente*

VR.a.2 Spostarsi e orientarsi nell'ambiente mediante il laser teleport (essendo a conoscenza di B,Y e rotazione controller) (Base) (20 sec)

- **L'utente preme un pulsante B o Y per attivare il laser teleport**
- **L'utente sceglie dove teletrasportarsi muovendo il controller e sceglie la rotazione ruotando il controller**
- **L'utente rilascia il tasto premuto**

D.b.1 Aprire i menu Configurator, Tools, Settings mediante le rispettive icone (Base) (30sec)

- L'utente clicca sulle icone agli angoli dello schermo per aprire i rispettivi menù

VR.b.2 Aprire i menu Configurator, Tools, Settings mediante il menu radiale sul controller sinistro, con i rispettivi Tablet e Dashboard. (Base) (40 sec)

- **L'utente ritrova il menu sul dorso della mano sinistra**
- **L'utente, con il controller destro, punta il laser sull'icona del menu e preme il pulsante di Trigger**
- **L'utente punta a turno tutti gli altri pulsanti e preme Trigger per aprire i relativi menu.**

D.c.1. Chiudere l'applicazione dall'apposito pulsante (Base) (20 sec)

- L'utente trova l'icona di logout in alto a destra e la clicca

V.c.2. Chiudere l'applicazione dall'apposito pulsante (Base) (20 sec)

- **L'utente trova l'icona di logout nel menu home e la clicca. Clicca Yes alla domanda.**

VR.d.1 Spostare il Tablet/Dashboard (Base) (30 sec)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Tool che apre la relativa Dashboard / L'utente clicca sul menu Configurator che apre la relativo Tablet**
- **L'utente prende la Dashboard / il Tablet tenendo premuto il tasto Grip e la/lo sposta a piacimento**

D.e.1 Cambiare la modalità di navigazione in WALK (Base) (30 sec)

- L'utente clicca sull'icona dei Settings

- L'utente clicca sull'icona Navigation
- L'utente clicca sull'icona WALK

D.e.2 Cambiare la modalità di navigazione in FLY (Base) (30 sec)

- L'utente clicca sull'icona dei Settings
- L'utente clicca sull'icona Navigation
- L'utente clicca sull'icona FLY

D.e.3 Cambiare la modalità di navigazione in ORBIT (Base) (30 sec)

- L'utente clicca sull'icona dei Settings
- L'utente clicca sull'icona Navigation
- L'utente clicca sull'icona ORBIT

D.e.4 Cambiare la modalità di navigazione in VR (Base) (30 sec)

- L'utente clicca sull'icona dei Settings
- L'utente clicca sull'icona Navigation
- L'utente clicca sull'icona VR

VR.e.5 Cambiare la modalità di navigazione in WALK/FLY/ORBIT (Base) (25 sec)

- **L'utente clicca sull'icona Menu Home**
- **L'utente clicca sull'icona Settings che apre il relativo Tablet**
- **L'utente clicca sull'icona Navigation**
- **L'utente clicca sull'icona WALK/FLY/ORBIT**

D.f.i.1 Misura la lunghezza della macchina (Base) (35 sec)

- L'utente clicca sull'icona dei Tool
- L'utente sceglie il tool Meter
- L'utente sceglie l'opzione Add
- L'utente clicca un punto vicino al muso della macchina e uno sul retro

VR.f.i.1 Misura la lunghezza della macchina (Base) (40 sec)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Tool che apre la relativa Dashboard**
- **L'utente seleziona con Trigger il tool Meter**
- **L'utente prende il tool Meter con Grip tenendolo premuto**
- **L'utente con la mano opposta a quella di presa seleziona il pulsante Add**
- **L'utente clicca un punto vicino al muso della macchina e uno sul retro**

D.f.i.2 Misura il diametro dei cerchioni (Medium) (35 sec)

- L'utente si avvicina al cerchione di interesse

- L'utente clicca sull'icona dei Tool
- L'utente sceglie il tool Meter
- L'utente sceglie l'opzione Add
- L'utente clicca un estremo del cerchione e poi su quello opposto

VR.f.i.2 Misura il diametro dei cerchioni (Medium) (35 sec)

- **L'utente si avvicina al cerchione di interesse**
- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Tool che apre la relativa Dashboard**
- **L'utente seleziona con Trigger il tool Meter**
- **L'utente prende il tool Meter con Grip tenendolo premuto**
- **L'utente con la mano opposta a quella di presa seleziona il pulsante Add**
- **L'utente clicca un estremo del cerchione e poi su quello opposto**

D.f.i.3 Misura la pinza dei freni (High) (50 sec)

- L'utente si avvicina alla pinza di interesse
- L'utente clicca sull'icona dei Tool
- L'utente sceglie il tool Move
- L'utente sceglie l'opzione Move
- L'utente sposta il cerchione
- L'utente clicca sull'icona di back
- L'utente sceglie il tool Meter
- L'utente sceglie l'opzione Add
- L'utente clicca un estremo della pinza e poi su quello opposto

VR.f.i.3 Misura la pinza dei freni (High) (1 min)

- **L'utente si avvicina alla pinza di interesse**
- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Tool che apre la relativa Dashboard**
- **L'utente seleziona con Trigger il tool Move**
- **L'utente prende il tool Move con Grip tenendolo premuto**
- **L'utente con la mano opposta a quella di presa seleziona il pulsante Move**
- **L'utente sposa il cerchione**
- **L'utente rilascia il Grip**
- **L'utente seleziona con Trigger il tool Meter**
- **L'utente prende il tool Meter con Grip tenendolo premuto**
- **L'utente con la mano opposta a quella di presa seleziona il pulsante Add**
- **L'utente clicca un estremo della pinza e poi su quello opposto**

D.f.i.4 Cancella una delle misure prese (Base) (20 sec)

- Partendo dal tool attivo
- L'utente clicca l'opzione Delete
- L'utente clicca su una misura da cancellare

VR.f.i.4 Cancella una delle misure prese (Base) (20 sec)

- **Partendo dal tool attivo**
- **L'utente clicca sul bottone dell'opzione attiva per ri-visualizzare tutte le opzioni**
- **L'utente clicca sull'opzione Delete**
- **L'utente punta il laser sull'opzione da cancellare e clicca**

VR.f.i.5 Seleziona e prendi lo strumento Meter (Base) (25 sec)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Tool che apre la relativa Dashboard**
- **L'utente seleziona con Trigger il tool Meter**

D.f.ii.1 Rendere invisibile un oggetto a piacimento (Base) (25 sec)

- L'utente clicca sull'icona dei Tool
- L'utente sceglie il tool X-Ray
- L'utente sceglie l'opzione Apply
- L'utente clicca su un oggetto a piacere

VR.f.ii.1 Rendere invisibile un oggetto a piacimento (Base) (30 sec)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Tool che apre la relativa Dashboard**
- **L'utente seleziona con Trigger il tool X-Ray**
- **L'utente prende il tool X-Ray con Grip tenendolo premuto**
- **L'utente con la mano opposta a quella di presa seleziona il pulsante Apply**
- **L'utente punta e clicca un oggetto a piacere**

D.f.ii.2 Rendere invisibile la carrozzeria della macchina (Medium) (25 sec)

- L'utente si avvicina alla macchina
- L'utente clicca sull'icona dei Tool
- L'utente sceglie il tool X-Ray
- L'utente sceglie l'opzione Apply
- L'utente clicca sulla carrozzeria della macchina

VR.f.ii.2 Rendere invisibile la carrozzeria della macchina (Medium) (30 sec)

- **L'utente si avvicina alla macchina**
- **L'utente clicca sul Menu Home**

- **L'utente clicca sul menu Tool che apre la relativa Dashboard**
- **L'utente seleziona con Trigger il tool X-Ray**
- **L'utente prende il tool X-Ray con Grip tenendolo premuto**
- **L'utente con la mano opposta a quella di presa seleziona il pulsante Apply**
- **L'utente punta e clicca la carrozzeria della macchina**

D.f.ii.3 Isolare una gomma della macchina (High) (30 sec)

- L'utente si avvicina alla gomma
- L'utente clicca sull'icona dei Tool
- L'utente sceglie il tool X-Ray
- L'utente sceglie l'opzione Isolate
- L'utente clicca sulla gomma

VR.f.ii.3 Isolare una gomma della macchina (High) (30 sec)

- **L'utente si avvicina alla gomma**
- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Tool che apre la relativa Dashboard**
- **L'utente seleziona con Trigger il tool X-Ray**
- **L'utente prende il tool X-Ray con Grip tenendolo premuto**
- **L'utente con la mano opposta a quella di presa seleziona il pulsante Apply**
- **L'utente punta e clicca sulla gomma**

D.f.ii.4 Fai tornare tutto visibile (Base) (20 sec)

- Partendo dal tool attivo
- L'utente clicca l'opzione Reset All

VR.f.ii.4 Fai tornare tutto visibile (Base) (20 sec)

- **Partendo dal tool attivo**
- **L'utente clicca sul bottone dell'opzione attiva per ri-visualizzare tutte le opzioni**
- **L'utente clicca sull'opzione Reset All**

VR.f.ii.6 Seleziona e prendi lo strumento X-Ray (Base) (25 sec)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Tool che apre la relativa Dashboard**
- **L'utente seleziona con Trigger il tool X-Ray**
- **L'utente prende il tool X-Ray con Grip tenendolo premuto**

D.f.iii.1 Disegna qualcosa nell'ambiente (Base) (25 sec)

- L'utente clicca sull'icona dei Tool
- L'utente sceglie il tool Paint

- L'utente sceglie l'opzione Paint
- L'utente disegna liberamente nell'ambiente

VR.f.iii.1 Disegna qualcosa nell'ambiente (Base) (30 sec)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Tool che apre la relativa Dashboard**
- **L'utente seleziona con Trigger il tool Paint**
- **L'utente prende il tool Paint con Grip tenendolo premuto**
- **L'utente con la mano opposta a quella di presa seleziona il pulsante Paint**
- **L'utente disegna a piacere nell'ambiente**

D.f.iii.2 Vai a scrivere il tuo nome sul vetro della macchina (Medium) (40 sec)

- L'utente si avvicina alla macchina
- L'utente clicca sull'icona dei Tool
- L'utente sceglie il tool Paint
- L'utente sceglie l'opzione Paint
- L'utente scrive il suo nome sul vetro della macchina

VR.f.iii.2 Vai a scrivere il tuo nome sul vetro della macchina (Medium) (35 sec)

- **L'utente si avvicina alla macchina**
- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Tool che apre la relativa Dashboard**
- **L'utente seleziona con Trigger il tool Paint**
- **L'utente prende il tool Paint con Grip tenendolo premuto**
- **L'utente con la mano opposta a quella di presa seleziona il pulsante Paint**
- **L'utente scrive il suo nome sul vetro della macchina**

D.f.iii.3 Scrivi di fianco al cerchione modello X il colore con cui vorresti i bulloni (High) (90 sec)

- L'utente si avvicina al cerchione di interesse
- L'utente clicca sull'icona dei Tool
- L'utente sceglie il tool Paint
- L'utente sceglie l'opzione Paint
- L'utente cerchia i bulloni e scrive il nome di un colore

VR.f.iii.3 Scrivi di fianco al cerchione modello X il colore con cui vorresti i bulloni (High) (90 sec)

- **L'utente si avvicina al cerchione di interesse**
- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Tool che apre la relativa Dashboard**

- **L'utente seleziona con Trigger il tool Paint**
- **L'utente prende il tool Paint con Grip tenendolo premuto**
- **L'utente con la mano opposta a quella di presa seleziona il pulsante Paint**
- **L'utente cerchia i bulloni e scrive il nome di un colore**

D.f.iii.4 Cancella i tratti fatti (Base) (30 sec)

- Partendo dal tool attivo
- L'utente clicca l'opzione Delete Stroke
- L'utente clicca sul tratto da cancellare

VR.f.iii.4 Cancella i tratti fatti (Base) (30 sec)

- **L'utente clicca sul bottone dell'opzione attiva per ri-visualizzare tutte le opzioni**
- **L'utente clicca sull'opzione Delete Stroke**
- **L'utente punta il laser sul tratto da cancellare e clicca**

VR.f.iii.5 Seleziona e prendi lo strumento Paint (Base) (20 sec)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Tool che apre la relativa Dashboard**
- **L'utente seleziona con Trigger il tool Paint**
- **L'utente prende il tool Paint con Grip tenendolo premuto**

D.f.iv.1. Sposta un oggetto a tuo piacimento della macchina (Base) (30 sec)

- L'utente clicca sull'icona dei Tool
- L'utente sceglie il tool Move
- L'utente sceglie l'opzione Move
- L'utente clicca e sposta un oggetto a piacimento

VR.f.iv.1. Sposta un oggetto a tuo piacimento della macchina (Base)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Tool che apre la relativa Dashboard**
- **L'utente seleziona con Trigger il tool Move**
- **L'utente prende il tool Move con Grip tenendolo premuto**
- **L'utente con la mano opposta a quella di presa seleziona il pulsante Move**
- **L'utente clicca e sposta un oggetto a piacimento**

D.f.iv.2. Sposta un oggetto a tuo piacimento della macchina nell'area dedicata / passare a master player (Medium) (40 sec)

- L'utente clicca sull'icona dei Tool
- L'utente sceglie il tool Move
- L'utente sceglie l'opzione Move

- L'utente clicca e sposta un oggetto a piacimento verso l'area dedicata / il master player

VR.f.iv.2. Sposta un oggetto a tuo piacimento della macchina nell'area dedicata / passare a master player (Medium) (35 sec)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Tool che apre la relativa Dashboard**
- **L'utente seleziona con Trigger il tool Move**
- **L'utente prende il tool Move con Grip tenendolo premuto**
- **L'utente con la mano opposta a quella di presa seleziona il pulsante Move**
- **L'utente clicca e sposta un oggetto a piacimento verso l'area dedicata / il master player**

D.f.iv.3. Sposta un oggetto a tuo piacimento della macchina nell'area dedicata/ passare a master player senza muoverti (High) (35 sec)

- L'utente clicca sull'icona dei Tool
- L'utente sceglie il tool Move
- L'utente sceglie l'opzione Move
- L'utente clicca e sposta un oggetto a piacimento verso l'area dedicata / il master player utilizzando la rotellina del mouse

VR.f.iv.3. Sposta un oggetto a tuo piacimento della macchina nell'area dedicata/ passare a master player senza muoverti (High) (30 sec)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Tool che apre la relativa Dashboard**
- **L'utente seleziona con Trigger il tool Move**
- **L'utente prende il tool Move con Grip tenendolo premuto**
- **L'utente con la mano opposta a quella di presa seleziona il pulsante Move**
- **L'utente clicca e sposta un oggetto a piacimento verso l'area dedicata / il master player utilizzando il thumbstick**

D.f.iv.4. Resetta la posizione di un oggetto mosso (Base) (20 sec)

- Partendo dal tool attivo
- L'utente clicca l'opzione Reset
- L'utente clicca sull'oggetto da resettare

VR.f.iv.4. Resetta la posizione di un oggetto mosso (Base) (20 sec)

- **L'utente clicca sul bottone dell'opzione attiva per ri-visualizzare tutte le opzioni**
- **L'utente clicca sull'opzione Reset**
- **L'utente punta il laser sull'oggetto da resettare e clicca**

D.f.iv.5. Resetta la posizione di tutti gli oggetti spostati (Base) (20 sec)

- Partendo dal tool attivo
- L'utente clicca l'opzione Reset All

VR.f.iv.5. Resetta la posizione di tutti gli oggetti spostati (Base) (20 sec)

- **Partendo dal tool attivo**
- **L'utente clicca sul bottone dell'opzione attiva per ri-visualizzare tutte le opzioni**
- **L'utente clicca sull'opzione Reset All**

VR.f.iv.6 Seleziona e prendi lo strumento Move(Base)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Tool che apre la relativa Dashboard**
- **L'utente seleziona con Trigger il tool Move**
- **L'utente prende il tool Move con Grip tenendolo premuto**

D.f.v.1. Spostati in una posizione a tuo piacimento usando i tool a disposizione (Base) (30 sec)

- L'utente clicca sull'icona dei Tool
- L'utente sceglie il tool Teleport
- L'utente sceglie un'opzione a piacimento

VR.f.v.1. Spostati in una posizione a tuo piacimento usando i tool a disposizione (Base) (30 sec)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Tool che apre la relativa Dashboard**
- **L'utente clicca sulla sfera per aprire il cassetto**
- **L'utente clicca il tool Teleport**
- **L'utente clicca su un portale a piacimento**

D.f.v.2. Spostati nella posizione chiamata Second Floor (raggiungi il master player) (Medium) (35 sec)

- L'utente clicca sull'icona dei Tool
- L'utente sceglie il tool Teleport
- L'utente sceglie l'opzione Second Floor

VR.f.v.2. Spostati nella posizione chiamata Second Floor (raggiungi il master player) (Medium) (35 sec)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Tool che apre la relativa Dashboard**
- **L'utente clicca sulla sfera per aprire il cassetto**

- **L'utente clicca il tool Teleport**
- **L'utente clicca sul portale Second Floor**

VR.f.v.3 Apri il cassetto e attiva Teleport (Base) (25 sec)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Tool che apre la relativa Dashboard**
- **L'utente clicca sulla sfera per aprire il cassetto**
- **L'utente clicca il tool Teleport**

D.f.vi.1 Cambia una ambientazione/illuminazione a piacimento (Base) (30 sec)

- L'utente clicca sull'icona dei Tool
- L'utente sceglie il tool Environment
- L'utente sceglie un'opzione a piacimento

VR.f.vi.1 Cambia una ambientazione/illuminazione a piacimento (Base) (30 sec)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Tool che apre la relativa Dashboard**
- **L'utente clicca sulla sfera per aprire il cassetto**
- **L'utente clicca il tool Environment**
- **L'utente clicca su un portale a piacimento**

D.f.vi.2 Cambia ambientazione e metti Sunset/Night (Medium) (35 sec)

- L'utente clicca sull'icona dei Tool
- L'utente sceglie il tool Environment
- L'utente sceglie l'opzione Sunset/Night

VR.f.vi.2 Cambia ambientazione e metti Sunset/Night (Medium) (35 sec)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Tool che apre la relativa Dashboard**
- **L'utente clicca sulla sfera per aprire il cassetto**
- **L'utente clicca il tool Environment**
- **L'utente clicca sul portale Sunset/Night**

VR.f.vi.3 Apri il cassetto e attiva Environment (Base) (20 sec)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Tool che apre la relativa Dashboard**
- **L'utente clicca sulla sfera per aprire il cassetto**
- **L'utente clicca il tool Environment**

D.g.i.1 Cambiare una qualsiasi variante del veicolo (Base) (30 sec)

- L'utente clicca sull'icona del Configuratore
- L'utente sceglie la Fase e le sottocategorie a piacimento
- L'utente sceglie una variante tra quelle a disposizione

D.g.i.2 Cambiare una specifica variante del veicolo e torna alle fasi (Medium) (35 sec)

- L'utente clicca sull'icona del Configuratore
- L'utente sceglie la Fase CarPaint e la sottocategoria X
- L'utente sceglie la variante indicata
- L'utente clicca back per tornare alle fasi

VR.g.i.3 Cambiare una qualsiasi variante del veicolo da Tablet (Base) (45 sec)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Configurator che apre il relativo Tablet**
- **L'utente sceglie la Fase e le sottocategorie a piacimento**
- **L'utente sceglie una variante tra quelle a disposizione e clicca il pulsante Load**

VR.g.i.4 Cambiare una specifica variante del veicolo Tablet e torna alle fasi (Medium) (35 sec)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Configurator che apre il relativo Tablet**
- **L'utente sceglie la Fase CarPaint e la sottocategoria X**
- **L'utente sceglie la variante indicata e clicca il pulsante Load**
- **L'utente clicca home per tornare alle fasi**

VR.g.i.5 Cambiare una qualsiasi variante del veicolo da Dashboard (Base) (45 sec)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Configurator che apre il relativo Tablet**
- **L'utente sceglie la Fase e le sottocategorie a piacimento**
- **L'utente, raggiunte le varianti, clicca 3D Mode / clicca sul pulsante info di una variante e clicca 3D Mode**
- **L'utente seleziona una variante e clicca su Load / clicca su Load**

VR.g.i.6 Cambiare una specifica variante del veicolo Dashboard (Medium) (55 sec)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Configurator che apre il relativo Tablet**
- **L'utente sceglie la Fase e le sottocategorie indicate**
- **L'utente, raggiunte le varianti, clicca 3D Mode / clicca sul pulsante info della variante di interesse e clicca 3D Mode**
- **L'utente seleziona la variante di interesse e clicca su Load / clicca su Load**

D.g.ii.1 Visualizza le informazioni di una variante a piacere (Base) (30 sec)

- L'utente clicca sull'icona del Configuratore
- L'utente sceglie la Fase e le sottocategorie a piacimento
- L'utente sceglie una variante tra quelle a disposizione e legge le info nel pannello a sinistra

D.g.ii.2 Visualizza le informazioni della variante X (Medium) (40 sec)

- L'utente clicca sull'icona del Configuratore
- L'utente sceglie la Fase CarPaint e la sottocategoria X
- L'utente sceglie la variante indicata e legge le info nel pannello a sinistra

D.g.ii.3 Configura il veicolo a piacere, tenendo conto di avere X Budget per la carrozzeria, Y Budget per i cerchioni e Z Budget per le pinze (High) (3-5 minuti)

- All'utente vengono comunicati i budget per ogni Phase (off game o master play)
- L'utente esegue a piacere la configurazione verificando i prezzi nelle info
- Verifica se l'utente ha rispettato i budget

VR.g.ii.1 Visualizza le informazioni di una variante a piacere da Tablet (Base) (35 sec)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Configurator che apre il relativo Tablet**
- **L'utente sceglie la Fase e le sottocategorie a piacimento**
- **L'utente sceglie una variante tra quelle a disposizione e clicca info per leggerle**

VR.g.ii.2 Visualizza le informazioni della variante X da Tablet (Medium) (40 sec)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Configurator che apre il relativo Tablet**
- **L'utente sceglie la Fase CarPaint e la sottocategoria X**
- **L'utente sceglie la variante indicata e clicca info per leggerle**

VR..g.ii.3 Configura il veicolo a piacere, tenendo conto di avere X Budget per la carrozzeria, Y Budget per i cerchioni e Z Budget per le pinze, tutto da Tablet (High) (3-5 min)

- **All'utente vengono comunicati i budget per ogni Phase (off game o master play)**
- **L'utente esegue a piacere la configurazione verificando i prezzi nelle info del Tablet**
- **Verifica se l'utente ha rispettato i budget**

VR.g.ii.1 Visualizza le informazioni di una variante a piacere da Dashboard (Base) (35 sec)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Configurator che apre il relativo Tablet**

- **L'utente sceglie la Fase e le sottocategorie a piacimento**
- **L'utente, raggiunte le varianti, clicca 3D Mode**
- **L'utente seleziona una variante e clicca sul bottone Info e le legge nel mini Tablet**

VR.g.ii.2 Visualizza le informazioni della variante X da Dashboard (Medium) (40 sec)

- **L'utente clicca sul Menu Home**
- **L'utente clicca sul menu Configurator che apre il relativo Tablet**
- **L'utente sceglie la Fase CarPaint e la sottocategoria X**
- **L'utente, raggiunte le varianti, clicca 3D Mode**
- **L'utente seleziona la variante indicata e clicca sul bottone Info e le legge nel mini Tablet**

VR.g.ii.3 Configura il veicolo a piacere, tenendo conto di avere X Budget per la carrozzeria, Y Budget per i cerchioni e Z Budget per le pinze, visualizzando le variante da Dashboard (High) (3-5 min)

- **All'utente vengono comunicati i budget per ogni Phase (off game o master play)**
- **L'utente esegue a piacere la configurazione verificando i prezzi nelle info della Dashboard**
- **Verifica se l'utente ha rispettato i budget**

QUESTIONARIO SU USABILITY E UX

*Campo obbligatorio

1. ID Partecipante: *

_____

Simulator Sickness Questionnaire

In questa sezione ti chiederemo di valutare il tuo stato fisico a seguito della fase di test da te svolta. Qui sotto sono stati elencati i sintomi più comuni riportati dai tester dopo un'esperienza in Realtà Virtuale.

2. Quali di questi stintomi presenti? A che livello? *

Contrassegna solo un ovale per riga.

	Nessuno/a	Leggero/a	Moderato/a	Acuto/a
Malessere Generale	◯	◯	◯	◯
Nausea	◯	◯	◯	◯
Fastidio allo stomaco	◯	◯	◯	◯
Eruttazione	◯	◯	◯	◯
Sudorazione	◯	◯	◯	◯
Incremento della salivazione	◯	◯	◯	◯
Affaticamento	◯	◯	◯	◯
Vertigini	◯	◯	◯	◯
Vertigini ad occhi aperti	◯	◯	◯	◯
Vertigini ad occhi chiusi	◯	◯	◯	◯
Pressione alla testa	◯	◯	◯	◯
Mal di testa	◯	◯	◯	◯
Difficoltà di concentrazione	◯	◯	◯	◯
Difficoltà di messa a fuoco	◯	◯	◯	◯
Vista sfocata	◯	◯	◯	◯
Occhi affaticati	◯	◯	◯	◯

Immersione e Presenza

3. Quanto sei stato capace di controllare gli eventi? *

Contrassegna solo un ovale.

	1	2	3	4	5	
Per nulla	◯	◯	◯	◯	◯	Completamente

4. Quanto era reattivo l'ambiente rispetto alle tue azioni? *

Contrassegna solo un ovale.

	1	2	3	4	5	
Per nulla reattivo	◯	◯	◯	◯	◯	Molto reattivo

5. Quanto ti sembravano naturali le tue interazioni con l'ambiente? *

Contrassegna solo un ovale.

	1	2	3	4	5	
Per nulla naturali	◯	◯	◯	◯	◯	Totalmente naturali

6. Quanto ti ha coinvolto l'aspetto visivo dell'ambiente? *

Contrassegna solo un ovale.

	1	2	3	4	5	
Per nulla	◯	◯	◯	◯	◯	Completamente

7. Quanto la tua esperienza nell'ambiente virtuale è sembrata coerente con quella nel mondo reale? *

Contrassegna solo un ovale.

	1	2	3	4	5	
Per nulla coerente	◯	◯	◯	◯	◯	Totalmente coerente

8. Sei stato in grado di anticipare cosa sarebbe successo in risposta alle azioni che facevi? *

Contrassegna solo un ovale.

	1	2	3	4	5	
No, mai	◯	◯	◯	◯	◯	Sì, sempre

9. Il campo visivo del visore ti ha permesso di ispezionare facilmente l'ambiente che ti circondava? *

Contrassegna solo un ovale.

	1	2	3	4	5	
No, per nulla	◯	◯	◯	◯	◯	Sì, completamente

10. Quanto vicino hai potuto esaminare gli elementi nell'ambiente? *

Contrassegna solo un ovale.

	1	2	3	4	5	
Solo da molto lontano	◯	◯	◯	◯	◯	Molto vicino

11. Quanto ti sei sentito coinvolto nell'esperienza nell'ambiente virtuale? *

Contrassegna solo un ovale.

	1	2	3	4	5	
Per nulla coinvolto	◯	◯	◯	◯	◯	Completamente coinvolto

12. Hai percepito ritardo tra l'azione da te eseguita e il risultato atteso? *

Contrassegna solo un ovale.

	1	2	3	4	5	
Un ritardo esagerato	◯	◯	◯	◯	◯	Non ho percepito ritardi

13. Quanto velocemente ti sei adattato all'ambiente virtuale? *

Contrassegna solo un ovale.

	1	2	3	4	5	
Non mi sono adattato	◯	◯	◯	◯	◯	Immediatamente

14. Alla fine di questa esperienza, pensi di aver compreso meglio i movimenti e le azioni tipiche di un ambiente virtuale? *

Contrassegna solo un ovale.

	1	2	3	4	5	
No, per nulla	◯	◯	◯	◯	◯	Sì, assolutamente

15. La qualità del display del visore ha interferito o ti ha distratto nell’eseguire le task assegnate o altre attività? *

Contrassegna solo un ovale.

	1	2	3	4	5	
Ha disturbato molto	◯	◯	◯	◯	◯	Non ha disturbato per nulla

16. I controller sono stati d'intralcio durante l'esecuzione delle task assegnate o durante altre attività? *

Contrassegna solo un ovale.

	1	2	3	4	5	
Hanno disturbato molto	◯	◯	◯	◯	◯	Non hanno disturbato per nulla

17. Sei riuscito a concentrarti sulla task/attività richiesta oppure sei stato distratto dall'artificio meccanico necessario all'esecuzione di tale task/attività? *

Contrassegna solo un ovale.

	1	2	3	4	5	
Il meccanismo in sè mi ha distratto	◯	◯	◯	◯	◯	Mi sono concentrato solo sulle Task

UEQ: User Experience Questionnaire

18. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Fastidioso	◯	◯	◯	◯	◯	Piacevole

19. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Incomprensibile	◯	◯	◯	◯	◯	Comprensibile

20. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Priva di fantasia	◯	◯	◯	◯	◯	Creativa

21. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Difficile da apprendere	◯	◯	◯	◯	◯	Facile da apprendere

22. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Di poco valore	◯	◯	◯	◯	◯	Di grande valore

23. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Noiosa	◯	◯	◯	◯	◯	Appassionante

24. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Non interessante	◯	◯	◯	◯	◯	Interessante

25. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Imprevedibile	◯	◯	◯	◯	◯	Prevedibile

26. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Lenta	◯	◯	◯	◯	◯	Veloce

27. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Convenzionale	◯	◯	◯	◯	◯	Originale

28. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Ostruttiva	◯	◯	◯	◯	◯	Di supporto

29. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Scarsa	◯	◯	◯	◯	◯	Buona

30. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Complicata	◯	◯	◯	◯	◯	Facile

31. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Repellente	◯	◯	◯	◯	◯	Attraente

32. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Usuale	◯	◯	◯	◯	◯	Moderna

33. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Sgradevole	◯	◯	◯	◯	◯	Gradevole

34. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Insicura	◯	◯	◯	◯	◯	Sicura

35. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Soporifera	◯	◯	◯	◯	◯	Attivante

36. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Non conforme alle aspettative	◯	◯	◯	◯	◯	Conforme alle aspettative

37. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Inefficiente	◯	◯	◯	◯	◯	Efficiente

38. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Confusa	◯	◯	◯	◯	◯	Chiara

39. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Non pragmatica	◯	◯	◯	◯	◯	Pragmatica

40. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Sovraccarica	◯	◯	◯	◯	◯	Ordinata

41. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Non Invitante	◯	◯	◯	◯	◯	Invitante

42. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Ostile	◯	◯	◯	◯	◯	Congeniale

43. L'applicazione è: *

Contrassegna solo un ovale.

	1	2	3	4	5	
Conservativa	◯	◯	◯	◯	◯	Innovativa

Maneggevolezza

Indica per ogni affermazione quanto sei d'accordo con essa, facendo attenzione a distinguere le risposte riguardanti l'applicazione Desktop da quelle relative alla VR

44. DESKTOP *

Contrassegna solo un ovale per riga.

	Per nulla	Poco	Abbastanza	Molto	Totalmente
Esplorando l''ambiente,, capivo in ogni momento dove mi trovavo					
Mi è stato chiaro fin da subito di quante e quali sezioni era composta l''applicazione					
Potevo richiamare e usare il Main Menu con facilità					
Ho trovato facile usare i vari menu a disposizione					
E'' stato facile usare l''applicazione					
Quello che mi interessava era sempre a portata di mano					
Avevo il pieno controllo di quanto succedeva					
Le informazioni presenti nell''applicazione erano sufficienti a comprenderne l''utilizzo					

45. REALTA' VIRTUALE *

Contrassegna solo un ovale per riga.

	Per nulla	Poco	Abbastanza	Molto	Totalmente
Esplorando l''ambiente,, capivo in ogni momento dove mi trovavi					
Mi è stato chiaro fin da subito di quante e quali sezioni era composta l''applicazione					
Potevo richiamare e usare il Main Menu con facilità					
Ho trovato facile usare i vari menu a disposizione					
E'' stato facile usare l''applicazione					
Quello che mi interessava era sempre a portata di mano					
Avevo il pieno controllo di quanto succedeva					
Le informazioni presenti nell''applicazione erano sufficienti a comprenderne l''utilizzo					

Soddisfazione

Indica per ogni affermazione quanto sei d'accordo con essa, facendo attenzione a distinguere le risposte riguardanti l'applicazione Desktop da quelle relative alla VR

46. DESKTOP *

Contrassegna solo un ovale per riga.

	Per nulla	Poco	Abbastanza	Molto	Totalmente
L''applicazione ha soddisfatto le mie aspettative					
Ho trovato l''applicazione particolarmente utile					
Userei frequentemente questa applicazione					
La fase di apprendimento è stata rapida					
Le task erano semplici ed intuitive					

47. REALTA' VIRTUALE *

Contrassegna solo un ovale per riga.

	Per nulla	Poco	Abbastanza	Molto	Totalmente
L''applicazione ha soddisfatto le mie aspettative					
Ho trovato l''applicazione particolarmente utile					
Userei frequentemente questa applicazione					
La fase di apprendimento è stata rapida					
Le task erano semplici ed intuitive					

Attrattiva

Indica per ogni affermazione quanto sei d'accordo con essa, facendo attenzione a distinguere le risposte riguardanti l'applicazione Desktop da quelle relative alla VR

48. DESKTOP *

Contrassegna solo un ovale per riga.

	Per nulla	Poco	Abbastanza	Molto	Totalmente
Questa applicazione è esteticamente gradevole					
Gli elementi grafici sono curati e accattivanti					
E" correttamente equilibrata l''estetica con la semplicità di utilizzo					

49. REALTA' VIRTUALE *

Contrassegna solo un ovale per riga.

	Per nulla	Poco	Abbastanza	Molto	Totalmente
Questa applicazione è esteticamente gradevole					
Gli elementi grafici sono curati e accattivanti					
E" correttamente equilibrata l''estetica con la semplicità di utilizzo					

50. Con quale probabilità consiglieresti questa applicazione a un amico o a un collega? *

Contrassegna solo un ovale.

	1	2	3	4	5	6	7	8	9	10	
Nessuna											Sicuramente

INTERVISTA

Ti faremo diverse domane a risposta chiusa o aperta. Sentiti libero/a di dare qualunque tipo di opinione, consiglio o critica e sii completamente sincero/a nelle risposte.

51. Come valuti l’esperienza col visore per la Realtà Virtuale? *

Contrassegna solo un ovale.

	1	2	3	4	5	
Pessima	◯	◯	◯	◯	◯	Ottima

52. Useresti di nuovo un visore? Se no, perché? *

53. Puoi descrivere in poche parole come è stato indossare il visore? *

54. C’è qualcosa che ha attirato la tua attenzione durante l’esperienza? Se sì, cosa? *

55. Tre cose che ti sono piaciute di più dell’esperienza. *

56. Tre cose che non ti sono piaciute dell'esperienza. *

57. C’è qualcosa che cambieresti nell’applicazione? Se sì, cosa? *

58. C’è qualcosa nell’applicazione che hai ritenuto particolarmente non chiaro? Se sì, cosa? *

59. Riterresti utile inserire un tutorial per utilizzare l’applicazione? *

Seleziona tutte le voci applicabili.

- [ ] No, è già chiarissima così
- [ ] Sì, nella parte VR e Desktop
- [ ] Sì, nella parte in VR, sotto forma di video visionabili quando serve
- [ ] Sì, nella parte in VR, sotto forma di testo consultabile quando serve
- [ ] Sì, nella parte in VR, sotto forma di video la cui visione è obbligatoria ad inizio esperienza
- [ ] Sì, nella parte in VR, sotto forma di testo la cui lettura è obbligatoria ad inizio esperienza
- [ ] Altro: _____

60. L’utilizzo del Main Menu era semplice e intuitivo? Se no, per quali ragioni? *

61. L’utilizzo della Tool Dashboard era semplice e intuitivo? Se non lo era, per quali ragioni? *

62. L’utilizzo del Tablet era semplice e intuitivo? Se non lo era, per quali ragioni? *

63. L’utilizzo della Configurator Dashboard era semplice e intuitivo? Se non lo era, per quali ragioni? *

64. Pensando alla Tool Dashboard e ai relativi Tool: ritieni complesso il loro utilizzo? Se sì, perché? *

65. Pensando alla Configurator Dashboard e alle relative varianti in 3D: ritieni complesso il loro utilizzo? Se sì, perché? *

66. Hai altri commenti o consigli in merito all’applicazione? *

Questi contenuti non sono creati né avallati da Google.

Google Moduli