POLITECNICO DI TORINO

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

Sessione di Laurea Aprile 2023

Tesi di Laurea Magistrale

Virtual Set Framework Design for Independent Movie Production

Integrating DMX Lighting, Compositing and Camera Tracking within a Real-Time Rendering Engine

Relatori
prof. Riccardo Antonio Silvio Antonino

Candidato
Federico Trento

firma del relatore

. . . .

firma del candidato

. . . .

Anno Accademico 2022-2023

† Alla mia mamma

Sommario

Risale agli anni precedenti il 2009 il primo uso di una tecnologia che permettesse di ottenere un feedback in tempo reale di una ripresa live-action trasposta in ambiente virtuale. Per girare *Avatar*, James Cameron ed il suo team ebbero l'idea di tralsare il sistema di tracciamento ottico utilizzato per la *performance capture* direttamente sulla cinepresa. In tal modo era possibile misurare posizione e orientamento della camera in tempo reale oltre che a quelle degli attori, ed utilizzare il flusso di dati generato per animare le loro controparti virtuali per mezzo di un *game engine*. Dieci anni dopo, la serie TV *Disney "The Mandalorian"* viene girata in un limbo denominato *LED Wall* predisposto per essere un *digital responsive background*. Ciò rese possibile ridurre al minimo la fase di post-produzione filmica in virtù di una parte di effetti speciali ottenuti direttamente *"in camera"*.

La domanda di questo progetto di tesi è dove si possano porre artisti e produzioni indipendenti all'interno dello scenario attuale dell'industria cinematografica. Da un lato la corsa all'iperrealismo che comporta costi elevatissimi, dall'altro la ricerca di tecnologie sempre più *"opache"* che permettano di gestire l'enorme complessità di questi workflow nel minor tempo possibile.

Il progetto *"Virtual Set Framework Design for Independent Movie Production"* vuole creare un ambiente di lavoro virtuale che permetta di gestire produzioni indipendenti a basso budget con tecnologie simili a quelle utilizzate dall'industria di riferimento. Attraverso il game engine *Unreal Engine 5* sono stati integrati alcuni degli aspetti più importanti di una produzione audiovisiva in un unico ambiente. Una postazione computer

vicino a un *green set* permette di gestire il sistema di lighting, compositing e camera tracking in tempo reale. La parte di lighting viene gestita per mezzo di una console *Chamsys* da cui vengono inviati dei comandi alle *fixture DMX* sia reali (sul set) sia virtuali (nel game engine). Essendo i segnali identici, le coppie di luci avranno un comportamento similare, producendo una concordanza cromatica nell'immagine finale compositata. Attraverso il plugin *"Composure"* si è predisposto un algoritmo di *chroma keying* necessario a "bucare" il *green screen*. Inoltre, grazie all'ausilio di un *garbage matte* dinamico è possibile muoversi con la cinepresa nel set in qualsiasi posizione, avendo sempre un mascherino correttamente eseguito. Infine, per il fulcro concettuale del virtual set, si è optato per un sistema di tracciamento più economico rispetto a quello ottico usato dall'industria filmica. Attraverso la tecnologia della *HTC Vive* è possibile rilevare posizione e orientamento di un oggetto tramite un tracker posto su di esso ed un minimo di due stazioni ad infrarossi. Basandosi su calcoli trigonometrici, il tracker è in grado di ricostruire la propria posizione e rotazione rilevando ciclicamente la posizione delle stazioni.

Attraverso il *framework* oggetto di questa tesi è quindi possibile ottenere un feedback visivo del compositing finale su un display ausiliario come avvenuto durante le riprese del film *"Avatar"*. Inoltre con una successiva implementazione, e disponendo di alcuni *LED panels*, sarebbe anche possibile elidere la fase di compositng in software per ottenerla direttamente in camera durante le riprese.

Indice

Elenco delle figure

Come vivremmo in un mondo
senza il tempo
senza un parametro così rotondo
che dell'esistenze scandisce il pendolo?
La tragicità del nulla
cadrebbe nello schianto zittito
che un turbinio le vite rifrulla
ed imperterrito tende all'infinito.
Qui
al centro del ciclone
sul pelo del bianconiglio
solo un artista si avvicina tanto
da sperimentarne questo consiglio:
"regista
porrai tu sul campo maschere sformate
di uomini, tempo e spazio
nel tuo personale flusso
di un mondo mai creato;
regista
diventarai tu un nuovo io
ponte tra l'uomo e Dio."
Spezzando il sottile filo
dei compromessi dell'esistenza
egli funambola verso un'uscita.
Una via per scampare
alla legge universale
della gravità esistenziale.

[F. Trento, Il Regista]

Parte I

Design del Framework

Nei prossimi capitoli esploreremo e analizzeremo le tipologie di risorse necessarie al fine di implementare un set di virtual production per produzioni indipendenti. La ricerca e l'approfondimento tecnico delle strumentazioni non vogliono fermarsi alle sole utilizzate concretamente durante lo sviluppo del framework. L'obiettivo è di dare una visione completa piuttosto che parziale delle offerte sul mercato e di ciò che si rende necessario ad un ipotetico set meglio strutturato. Per questo motivo tra la strumentazione analizzata troveremo anche dispositivi alternativi ed eventualmente anche costosi che potrebbero ottimizzare il processo successivamente descritto. Per quanto riguarda il software verrà preso in analisi solamente Unreal Engine e non altri competitors (ad esempio Unity) che non sono stati utilizzati. Questa scelta è stata presa a fronte del fatto che i diversi game engine si differenziano più per i tecnicismi che nella sostanza, ed il taglio di questa parte di tesi è di tipo teorico-generale. Per la stessa ragione il capitolo dedicato al software Unreal Engine è in questa parte una panoramica dell'interfaccia e dei tool a disposizione, poiché gli aspetti tecnico-pratici verranno affrontati in seguito in un'altra parte. Si tratta infine come già detto di capitoli di carattere più teorico che pratico volti a proporre un piccolo "manuale d'uso" per quei produttori o artisti indipendenti che si affacciassero sul mondo delle produzioni virtuali.

Capitolo 1

Introduzione alla Virtual Production

1.1 Breve Storia dei VFX

1.1.1 Prime Tecnologie

Fin dagli albori del cinema i registi hanno sempre cercato di combinare riprese in *live action* con tecniche di effetti visivi. Molti sono stati i precursori effettuando i primi tentativi, dalle tecniche miste di inchiostrazione su pellicola (come nel caso di Walt Disney con la serie *Alice Comedies*) alle tecniche di combinazione di riprese in *stop motion* con riprese *live action* (come in *King Kong* del 1933). Il primo effetto visivo che prevedeva un procedimento più innovativo, precursore dei moderni algoritmi di *chroma keying*, fu il *traveling matte process*. La sua storia risale al cosiddetto *black-backing matting process* che si realizzava riprendendo un soggetto illuminato davanti a uno sfondo nero. Dopodiché la pellicola andava copiata su un'altra ad alto contrasto in modo da ottenere uno sfondo bianco con una silhoutte nera. Questa pellicola poteva quindi essere impressa con un background e successivamente compositata fisicamente con la pellicola contenente il soggetto. Dunn Linwood fu uno dei pionieri della stampante ottica e durante il suo lavoro presso la RKO Pictures sviluppò la tecnica *double exposure matte process* per il musical

Flying Down to Rio. Le location del film venivano riprese davanti ad uno schermo per retroproiezioni. Petro Vlahos perfezionò queste tecniche con un nuovo processo che faceva uso di luci a vapore di sodio. Questo nuovo sistema installava sulle cineprese Technicolor un prisma multi-strato in grado di separare la luce di sodio dal colore della pellicola direzionando la differenza su una pellicola in bianco e nero che avrebbe funzionato da mascherino. Sempre Petro Vlahos brevettò anche la tecnica del *Color Difference Traveling Matte System*. Quando fu ingaggiato per gli effetti speciali del film *Ben-Hur* infatti MGM stava girando il film su pellicola a 65mm (poi stampata sulla 70mm) e perciò la tecnica con le luci a vapore di sodio non poteva funzionare. Vlahos sapeva dei tentativi di utilizzo del *blue screen* con scarsi risultati (capelli, fumo, motion blur [1] , ecc. erano poco gestibili) ma sapeva che c'era una banda della frequenza della luce blu che poteva essere divisa con ottimi risultati utilizzando il principio della sua invenzione precedente. Da allora tutte le tecnologie su cui si basavano le chiavi cromatiche per gli effetti speciali con *blue screen* o *green screen* si sono basate sulla sua invenzione.

Funzionamento dei Matte System

Nel suo libro *The Green Screen Handbook*, Foster [2010] approfondisce il funzionamento dietro ai processi di realizzazione di queste tecniche. Il termine *traveling matte* significa letteralmente che un mascherino si muove con le immagini filmate di frame in frame. Il **Sodium Vapor Traveling Matte** usa appunto le luci al sodio e produce un mascherino ad elevato contrasto se esposto attraverso un prisma posta all'interno della cinepresa e una pellicola in bianco e nero. Per usare questa tecnica si allestisce un fondale bianco dietro ai soggetti che riflette la luce al vapore di sodio ridirezionandola verso la camera. Questa specifica luce giallastra ha una temperatura colore di circa 975°K e si può separare facilmente dalle altre frequenze della luce andando a creare il mascherino su pellicola in bianco e nero. Inoltre grazie alla sua banda molto stretta non influisce sul colore

[1] Il *motion blur* è la scia lasciata sulla pellicola o in una sequenza di immagini da oggetti in movimento. Questo effetto appare quando l'immagine ripresa cambia durante l'intervallo di esposizione della pellicola (dovuto ad un lungo tempo di esposizione oppure ad un immagine che cambia molto rapidamente).

Figura 1.1. Dimostrazione della *pipeline* di produzione di una pellicola compositata attraverso il processo con tecnica *Sodium Vapor Traveling Matte*. Illustrato in *The Green Screen Handbook*

dei soggetti. Il funzionamento può essere riassunto nel seguente modo (figura 1.1): ci sono due pellicole nella cinepresa, una technicolor e una in bianco e nero. Le lente della cinepresa cattura l'immagine mentre il prisma devia la frequenza della luce di sodio di un determinato angolo per impressionare la pellicola in bianco e nero (risulta quindi una silhouette nera con uno sfondo grigio chiaro). La restante luce colorata continua invece dritta verso la pellicola technicolor impressionandola, e lasciando lo sfondo dietro ai soggetti di un colore bronzeo scuro. Le due pellicole vengono perciò impressionate simultaneamente durante tutte le riprese. Successivamente la pellicola in bianco e nero viene processata ed invertita per ottenere un mascherino con uno sfondo completamente nero ed una silhouette bianca. A questo punto questa nuova pellicola può essere usata per ricreare lo sfondo e gli elementi animati prima di essere riunita a quella con i soggetti ripresi in *live action*. Dal momento che come si può immaginare dalla descrizione il processo era molto lungo e dispendioso, pian piano si è abbandonata questa tecnica per passare a quella dei *blue o green screen*. Il processo di produzione dei **Blue/Green Screen Traveling Matte** è costituito fondamentalmente da tre elementi (figura 1.2): il soggetto in primo

piano, lo sfondo colorato e lo sfondo su cui vogliamo compositare il nostro soggetto. Invece di processare due diverse pellicole contemporaneamente durante la fase di riprese per creare il *traveling matte*, il *matte* è generato in post-produzione estraendo il colore di sfondo tramite soluzioni hardware o software direttamente dalla pellicola o dal video digitale.

Figura 1.2. Dimostrazione della *pipeline* di produzione di una pellicola compositata attraverso il processo con tecnica *Green Screen Matte*. Illustrato in *The Green Screen Handbook*

1.1.2 Motion Capture

La *motion capture* è una moderna tecnica per l'animazione di personaggi digitali che si basa sul tracciamento dei movimenti degli attori ma affonda le sue radici nel concetto di *rotoscoping* [2] . Il suo inventore fu Max Fleischer nel 1915 che usò le riprese di una performance di suo fratello travestito da clown per dipingervi sopra l'animazione di *Koko il Clown*. Lo sviluppo e l'evoluzione di questa tecnica ha portato concretamente alla

[2] Il *rotoscoping* è una tecnica che mira a produrre un movimento realistico di un personaggio animato usando delle riprese di un soggetto in *live action* per dipingere sopra di esso frame per frame.

creazione della *motion capture* con il personaggio di Gollum nel *Il Signore degli Anelli - Le Due Torri*. Infatti per questa tipologia di tracciamento l'attore Andy Serkis dove vestire una tuta *MoCap* mentre particolari camere riprendevano i suoi movimenti sia del corpo che facciali. Sebbene al tempo i dati ottenuti dalla *motion capture* venivano utilizzati solamente come punto di partenza per il lavoro degli animatori che lo avrebbero finito a mano frame per frame, il personaggio di Gollum ha segnato un passaggio storico nella storia dei VFX, poiché per la prima volta è stato possibile ottenere i dati della *motion capture* direttamente nella location (figura 1.3). Andy Serkis poteva quindi recitare ed interagire con gli altri attori invece di dover recitare successivamente in uno studio.

Tipologie di Tracking

Esistono quattro principali tecnologie per effettuare il tracciamento di un oggetto:

- *Tracker Meccanico*, dispositivo composto da una struttura cinematica, costituita da componenti meccanici consessi tra loro (in serie o in parallelo); la connessione di questi elementi si crea utilizzando dei giunti che al loro interno hanno dei sensori che servono per misurare l'angolo tridimensionale di riferimento. Nel caso di soggetti umani sono stati fatti alcuni tentativi di implementare complicate strutture dette *Gipsy Motion Tracker*, un esoscheletro da indossare che cattura il movimento di tutto il corpo umano. Per via della sua rigidità produce movimenti poco fluidi.

- *Tracker Magnetico*,sono dispositivi non collegati ad un sistema di riferimento fisso. Utilizzando un campo magnetico generato da una sorgente fissa, detta *Transmitter*, all'interno dello spazio di tracciamento, si è in grado di misurare la posizione e l'orientamento di un determinato elemento, detto *Receiver*, in movimento all'interno dello spazio stesso. Nel caso della *motion capture* i sensori sono integrati in tute particolari ed equipaggiate con uno zainetto che contiene la componentistica elettronica di gestione dei sensori e la batteria e l'unità di calcolo. Lo zainetto comunica tramite Wireless con il computer che gestisce la simulazione.

- *Tracker Ottici*, vengono utilizzate delle telecamere in posizioni diverse per poter tener traccia di ciò che sta accadendo in un determinato ambiente. I *Tracker Ottici*

hanno come punto di forza l'utilizzo della luce. Ci sono due tipologie di tracking ottico: *marker passivi* (marker che riflettono la luce e dei quali si vuole ricostruire la posizione) e *marker attivi* (emettono luce a LED). I sistemi di *Tracking Ottico* basati sui *marker* sono i più utilizzati nel cinema poiché è possibile creare tute di *MoCap* snelle e poco ingombranti.

- *Tracker Markerless*, sono sistemi in fase di sviluppo ed ancora lontani da poter essere considerati standard nell'industria. Non richiedono che l'attori indossi sensori o tute particolari, inoltre hanno tempi di setup minori senza porre vincoli sull'utente. Sono basati sull'uso di telecamere RGB e sfruttano algoritmi di Computer Vision molto complessi.

Fino agli anni 2000 non si vedrà comunque la tecnologia dei *Tracker Ottici* utilizzata su larga scala con una elaborazione dei dati in tempo reale direttamente sulla location, fino all'avvento di *Avatar*.

Figura 1.3. Andy Serkis interpreta Gollum nel film *Il Signore degli Anelli* indossando una tuta *MoCap* direttamente sul set.

1.2 Le Grandi Produzioni

Come descrivono Okun and Zwerman [2021], la *Virtual Production* vede la sua origine nell'adattamento di tecniche di motion capture che hanno permesso ai registi ed in generale agli artisti dietro le produzioni cinematografiche di approcciarsi con immagini di computer grafica in tempo reale in modo del tutto simile a quello delle riprese *live action* [3] tradizionali. La pratica partì infatti in riferimento alle performance degli attori, per poi estendersi alla cosiddetta *Virtual Camera* e alla *Simulcam*. Attualmente la sua più recente evoluzione è stata la virtual production che ha riscosso particolare successo nella fase di *previsualizzazione* [4] sul set e nella fase di shooting virtuale dove si va a riprendere un film con tecniche di computer grafica in *real-time*. Con il suo sviluppo è stata prestata molta attenzione all'implementazione di asset condivisi creati sin dall'inizio per le diverse fasi della pipeline produttiva successiva. ciò implica che debbano essere seguendo linee guida ben precise in accordo con le necessità tecniche e artistiche di produzione. Tuttavia i vantaggi sono considerevoli: infatti viene rimossa la necessità di ricreare gli asset ad ogni set utilizzato nel processo di creazione filmica e inoltre preserva l'integrità dell'intento creativo siccome le decisioni vengono prese durante la produzione anziché a posteriori reinterpretandole e modellandole sulle esigenze. Ripercorreremo quindi le tappe essenziali del cammino che ha portato alle tecniche odierne e a tutt'oggi se e come queste si possano declinare nel caso di produzioni indipendenti.

1.2.1 Avatar 2009

Prima della produzione di *Avatar* la tecnologia che dominava l'industria degli effetti speciali era la *motion capture*. Questa tecnica di animazione consiste nel riprendere attraverso particolari camere le performance degli attori che indossano apposite tute con marker che

[3] Film nei quali le riprese vengono fatte con attori in carne ed ossa.

[4] La *previsualizzazione* (anche conosciuta come *previs*) è la pratica di visualizzare idealmente o concretamente scene o sequenze di un film prima di riprenderlo. Si usa spesso per descrivere tecniche come lo *storyboarding*, sia 2D che 3D, usato per la pianificazione e la concettualizzazione delle scene del film.

servono per riconoscere specifici punti da tracciare utili alle camere. Questa tecnica permette quindi di creare *nuvole di punti* che verranno poi usate per animare personaggi umanoidi o creature. Uno dei difetti di questa tecnologia tanto quanto del green screen è che né gli attori né il regista e le altre figure creative hanno la possibilità di vedere ciò con cui stanno interagendo gli attori e possono solo immaginarselo. Così Cameron decise di collaborare con *Weta Digital* per portare i suoi colleghi e gli attori dentro quel mondo virtuale che tradizionalmente avrebbe preso vita solo in fase di post produzione, Thompson [2010]. *Weta* aveva gà creato alcuni dei migliori personaggi animati del tempo come Gollum del *Signore degli Anelli*. Come spiegato poc'anzi per gli attori era molto difficile interpretare la parte di creature fantastiche dentro a tute attillate e senza sapere dove fosse la camera dal momento che i sistemi di motion capture riprendono l'ambiente a 360°. Anche per i registi risultava una grande sfida poiché dovevano dirigere gli attori senza cinepresa e poi avrebbero scelto gli angoli di ripresa e le ottiche utilizzate in fase di post produzione davanti al computer ed alla scena ricreata. Proprio per questo motivo molta dell'immediatezza della performance veniva persa. Cameron voleva quindi vedere i propri attori nell'ambiente virtuale (chiamato *the volume*) già nel momento esatto in cui riprendeva. La sfida per Glenn Derry, supervisore della nuova tecnologia di virtual production, fu quindi quella di creare una *virtual camera* che mostrasse in bassa risoluzione l'ambiente di Pandora insieme ad una soluzione *real-time* della *motion capture*. Questa camera fu inizialmente chiamata *swing camera* poiché poteva essere posizionata in qualsiasi angolazione permettendo a Cameron la maggior libertà possibile. Una *virtual camera* nella pratica è un *rig* tradizionale di sul quale però non è montata la cinepresa con la lente o altro ma solo uno schermo LCD che simula ciò che si vedrebbe su quello di una vera cinepresa. Una *virtual camera* in pratica è un'estensione reale della camera virtuale interna a un *game engine* e mostra su un display esattamente quello che viene mostrato da quella presente nell'ambiente virtuale. Per funzionare correttamente i movimenti di queste due camere devono essere collegati e per farlo è necessario quindi usare sistemi di tracciamento sulla *virtual camera* fisica con metodi simili a quelli usati per la *motion capture* (un gruppo di sfere riflettenti era posto sulla camera). In tal modo ogni movimento della *virtual camera* manovrata dal regista corrisponde ad un movimento identico di quella

virtuale all'interno del game engine e perciò sullo schermo LCD della *virtual camera* viene renderizzato esattamente quello che viene inquadrato nel *game engine*. Infine Cameron poteva quindi riprendere i suoi attori dal vero e loro potevano avere un riferimento di ciò che stava facendo il regista e dei movimenti della camera sebbene fosse in realtà tutto simulato. Nonostante questo fosse un passo importante per la realizzazione del film era però solo una parte delle molte innovazioni che servivano per rendere il tutto credibile e fotorealistico. Senza le sottili espressioni facciali e micro-movimenti oculari i personaggi animati sembravano infatti senza vita. Cameron non voleva che fosse una *motion capture* ma che diventasse una *performance capture*, il che semanticamente rimandava a quello che era il suo obiettivo. In tutti i film precedenti ad *Avatar*, la *motion capture* serviva infatti solamente come punto di partenza per gli animatori che successivamente avrebbero finito il lavoro manualmente. Così per riuscire ad ottenere più dati possibili dai volti degli attori, Cameron riprese una sua vecchia idea che consisteva in una piccola camera posta davanti al volto e montata su un caschetto che poteva permettere di cogliere e tracciare ogni singolo movimento delle espressioni facciali e degli occhi. Le informazioni ottenute da queste camere produceva quindi un *framework digitale*, un *rig*, del volto dell'attore elaborato secondo una serie di regole che lo associavano ai muscoli facciali del personaggio rappresentato.

In ultimo un altro sistema innovativo vedeva arrivare la sua nascita, quella della *SimulCam*. La domanda che probabilmente si pose Cameron fu: se la *virtual camera* è solamente un oggetto che si muove e viene tracciato nello spazio, potremmo anche tracciare una vera camera *live action* ed avere il risultato finale già compositato in tempo reale su uno schermo simile a quello delle *virtual camera*? A parole il concetto funziona, ma realizzarlo non era per niente semplice. Infatti il sistema dei marker riflettenti utilizzato dalla *motion capture* funziona bene in ambienti bui ma se si provasse ad utilizzare questo sistema con un'intensa illuminazione del set o addirittura in esterno non funzionerebbe più. Questo avviene perché dal punto di vista ingegneristico, i sistemi di tracciamento ottico funzionano con delle soglie luminose oltre le quali si decide se un determinato punto è un marker o meno, perciò l'allestimento delle luci durante le sedute di motion capture dev'essere rigorosamente controllato. La soluzione di Glenn Derry fu quella di

Figura 1.4. James Cameron con una *Virtual Camera* sul set di *Avatar*.

realizzare un sistema di illuminazione a LED ad elevata frequenza. Questi apparecchi di illuminazione a LED potevano sincronizzarsi in fase con le camere della *motion capture* e con la cinepresa *live action*. Il risultato fu quindi che il sistema d'illuminazione a LED si accendeva in modo sincrono con il tempo di cattura ed esposizione della cinepresa per spegnersi contemporaneamente al lasso temporale di 20 microsecondi di esposizione in cui camere di *motion capture* catturavano la scena. Nel concreto, le camere di *motion capture* potevano vedere i marker, ma non l'illuminazione del set *live action* né tantomeno la luce solare (grazie alla bassa esposizione). Un aspetto specifico della camera *live action* era quello di estrarre dalla cinepresa informazioni come *focus*, *iris*, *zoom*, *distanza interoculare* e *convergenza* in tempo reale. Tali informazioni venivano quindi utilizzate per gli algoritmi di *camera-solve* dei software utilizzati per il compositing in tempo reale e in modo che le due camere si comportassero esattamente nello stesso modo. Era così appena stata creata la prima tecnologia che permettesse di realizzare un set di virtual production in cui il regista e gli altri artisti potessero vedere il risultato del compositing con green screen in tempo reale.

Figura 1.5. Setup con green screen e *SimulCam* sul set di *Avatar*.

1.2.2 The Mandalorian 2019

La tecnologia dietro le quinte di questa serie TV è probabilmente una delle più innovative dopo *Avatar* ed ha creato un nuovo standard e paradigma per l'industria dei media. Cosa succederebbe se nella tecnologia creata dal team di James Cameron al posto del green screen ci fosse uno schermo LED gigantesco che proietta ciò verrebbe compositato nella *Simulcam* direttamente sul set? Così è nata una nuova definizione del concetto di *virtual production*, una produzione virtual *full live* che ha come obiettivo quello di ottenere tutti gli effetti speciali non *come se* fossero registrati dalla cinepresa *live action* ma *realmente* registrati da essa. Sono così nati i *ICVFX* [5] . Il problema già anticipato dei *green screen* è che comportano una difficile post produzione, ed il loro allestimento è statico, complesso e bisogna coprire (oltre ad illuminare correttamente) vaste aree affinché

[5] L'*ICVFX* è una brillante nuova metodologia di ripresa cinematografica live action che si basa su una combinazione di illuminazione LED, tracciamento della camera e rendering real-time con la proiezione del background su pannelli appositi per creare un'integrazione *seamless* tra i soggetti in primo piano e lo sfondo virtuale. L'obiettivo principale è quello di eliminare la necessità del compositing del green screen e produrre l'immagine risultante finale direttamente in camera.

sia utilizzati. Inoltre se Cameron aveva risolto il problema per i registi di vedere ciò che circondava gli attori, l'ambiente virtuale, questo non si poteva dire per gli attori, che nei limbi verdi sono costretti a recitare lavorando con l'immaginazione per immedesimarsi nell'ambientazione della scena. Al posto del green screen è stato allestito un grande volume chiamato *Stagecraft* della *ILM*. Questo "muro" comunemente chiamato *LED Wall* è costituito da più pannelli a LED connessi tra loro che circondano il set cinematografico e possono riprodurre un'unica grande immagine del background. Un *LED Wall* è quindi modulare ed adattivo e può essere spostato, ricollocato e riutilizzato per diversi scopi essendo composto da tanti moduli. Questa tecnica di per sé non è una novità. Infatti le tecniche di proiezione del background risalgono ai tempi della pellicola. Tuttavia erano background statici e la cinepresa era ferma. Invece in questo caso l'interno sistema ideato per *Avatar* viene ripreso e riutilizzato, ma invece di mostrare in output solo la parte di scena inquadrata dalla camera i *LED Wall* devono riprodurre tutto l'ambiente virtuale, che deve muoversi sui pannelli LED in maniera sincrona alla camera. Ciò deve avvenire per evitare l'errore di parallasse, ciò quell'effetto ottico che ci permette di riconoscere che un ambiente non è davvero tridimensionale. Ma se l'immagine sullo sfondo ed il movimento della camera si muovono insieme per garantire la coerenza, allora il *LED Wall* visto dalla cinepresa *live action* non sembrerà più uno sfondo bidimensionale bensì uno sfondo reale. La sensazione ricreata per tutti i soggetti coinvolti nella produzione è quella di star girando su un vero set con innumerevoli vantaggi dal punto di vista dell'immediatezza e della credibilità delle performance, della libertà tecnico-creativa dei *filmmakers* e dei tempi di produzione che tagliando considerevolmente la fase di post-produzione si restringono in modo importante. Una delle sfide per produrre effetti speciali di elevata qualità in tempo reale è sincronizzare le diverse tecnologie in campo affinché eseguano il tutto simultaneamente. Per fare ciò è stato utilizzato un *game engine* ovvero un software per la realizzazione di videogiochi che permette di renderizzare scene virtuali in tempo-reale (ogni frame in meno di un trentesimo di secondo). In particolare è stato usato *Unreal Engine* di *Epic Games*. Inoltre anche la tecnologia hardware utilizzata per la sincronizzazione delle immagini sui *LED Wall* è stata precisamente curata in modo che tutto il sistema potesse funzionare senza intoppi. È importante ricordare che lo

Stagecraft non è il primo esempio di LED Wall, ma sicuramente è il primo esempio di questa tecnologia usata alla sua massima espressione dimostrando ciò che davvero si poteva realizzare con essa. Nel capitolo 8 parleremo più approfonditamente di questa tecnologia dal punto di vista tecnico e del software utilizzato.

Figura 1.6. *Stagecraft* usato in *The Mandalorian*.

1.3 Produzioni Indipendenti

Negli ultimi anni sono molti i filmmakers indipendenti che hanno cercato di sfruttare le potenzialità di queste tecnologie utilizzando workflow economici ed amatoriali. Anche le restrizioni dovute al Covid-19 hanno infatti portato allo sviluppo di tecnologie casalinghe ma che permettessero comunque agli artisti di continuare il proprio lavoro. Tra questi è spiccata la figura di Matt Workman, youtuber americano di discreta fama, programmatore e direttore della fotografia con una rilevante carriera cinematografica alle spalle. Attraverso l'uso degli strumenti hardware messi a disposizione da *HTC Vive* ha implementato un proprio sistema casalingo che gli permetteva di avere sia una *virtual camera*

attraverso la quale poter lavorare a distanza e fare le riprese per film e videogiochi in set lontani centinaia di chilometri, sia una *simulcam* con la quale girare shot compositati in tempo reale in casa propria. Oltre all'uso standard dei *Vive Tracker* è anche riuscito a ricreare un sistema che permettesse di ottenere le *FIZ* dai tracker stessi, collegandone meccanicamente la rotazione al movimento delle ghiere del diaframma e dello zoom. Questi sono solo alcuni esempi di come attualmente artisti indipendenti si stanno muovendo nel mondo della *virtual production* per sperimentarne le potenzialità e per poter lavorare ad alti livelli con tecnologie a basso costo.

Capitolo 2

La strumentazione Hardware

2.1 Allestimento del Set

Il primo aspetto da tenere in considerazione nella preparazione di un workflow di lavoro che tenga conto di un set di virtual production è il set stesso. Non solo gli algoritmi di rendering e compositing funzionano in tempo reale, ma chiaramente anche tutti quelli antecedenti come l'algoritmo di chiave cromatica usato per eliminare il background dall'immagine catturata. Questo è il cambiamento di paradigma centrale di tutta la tecnologia che stiamo analizzando. Nella pipeline produttiva tradizionale, la realizzazione degli effetti speciali è interamente prevista nella fase di post-produzione. Questo significa avere lunghi tempi a disposizione non solo per le maestranze che ci lavorano ma per i calcolatori stessi che possono "permettersi" algoritmi molto pesanti e lenti a livello di velocità di calcolo computazionale. Lavorando con un game engine, appunto, tale paradigma viene capovolto e il tempo di calcolo dell'algoritmo viene dettato dal frame-rate [1] a cui è impostato l'output video. Vista l'elevata qualità grafica che ci si aspetta da un prodotto audiovisivo è quindi

[1] La frequenza dei fotogrammi (in lingua inglese frame rate) è la frequenza di cattura o riproduzione dei fotogrammi che compongono un filmato. Un filmato, o un'animazione al computer, è infatti una sequenza di immagini riprodotte ad una velocità sufficientemente alta da fornire, all'occhio umano, l'illusione del movimento. La frequenza dei fotogrammi viene misurata in hertz (Hz), nei monitor a scansione progressiva, oppure espresso in termini di fotogrammi per secondo (fps). Wikipedia [c]

consigliabile preferire un frame-rate tradizionale come i 24 o 25 fps rispetto ai meno convenzionali ma altresì sfruttati 50 o 60 fps. Un dimezzamento del frame-rate comporterà infatti il raddoppiamento del tempo utile di calcolo per i diversi algoritmi che confluiscono nel workflow di rendering.

Per un risultato ottimale bisognerà sempre tenere a mente questo presupposto e predisporre sin dalla fase di allestimento un set-up che faciliti il software nel suo compito. È fondamentale che durante tutto il processo la strumentazione hardware funzioni in sintonia con quella software in quanto gli obiettivi nell'uso di un game engine sono ottimizzazione ed efficienza.

2.1.1 Il Green Screen

Nell'allestimento del set con green screen bisogna tenere conto di diversi aspetti che limitano o dettano le scelte progettuali:

1. Il frame-rate di output dell'immagine compositata

2. Lo spazio a disposizione permesso dal sistema di tracking

3. L'illuminazione necessaria per l'ottimizzazione dell'algoritmo di chroma key

Tutti questi aspetti se opportunamente gestiti permettono di giungere a un buon risultato tecnico durante lo sviluppo della fase di compositing. Ogni dettaglio va correttamente curato in virtù del fatto che l'etica del "*fix it in post*" [2] non può esistere per definizione in questo contesto e viene sostituito dal paradigma del "*fix it in pre*". Esso diventa principio filosofico della virtual production riferito alla preparazione di assets e alla pianificazione in fase di pre-produzione; ponendosi all'opposto del paradigma tradizionale degli effetti speciali del *fix it in post*. Glossary

[2] Frase di uso comune tra i cineasti ed i registi che vogliono avere un ciak "buono" ma delegano la correzione di eventuali errori alla fase di post-produzione per non incorrere nella dispendiosità di rigirare l'inquadratura.

Frame-Rate di Output

Come abbiamo considerato poc'anzi, è buona norma usare frame-rate ridotti nonché i classici della tradizione cinematografica. La cinepresa usata andrà impostata anch'essa alla velocità prescelta in software (24 o 25 fps). Vista questa scelta sarà importante usare uno shutter a 180° per ridurre la possibilità di avere *motion blur* difficilmente recuperabile durante le riprese. Sarà inoltre importante tenere ISO adatti a un contesto studio (sebbene preferibilmente nativi).

Spazio a Disposizione

Come vedremo successivamente il sistema di tracking HTC Vive mette a disposizione un'area massima di 10x10 metri quadrati. La cinepresa deve muoversi all'interno di quest'area per poter essere tracciata ed il green screen allestito non deve interferire con le base stations che necessitano di non essere occluse rispetto al tracker per poterne leggere i dati di locazione. Il green screen può quindi essere allestito adiacentemente al quadrato immaginario descritto dalla posizione delle 4 stazioni base che ne riempiono i vertici. Per diverse motivazioni è necessario che il background verde sia di dimensioni adatte al set. Si può dire che la richiesta minima è che la larghezza corrisponda al lato del quadrato e che l'altezza sia circa quella delle stazioni da terra, in modo da coprire adeguatamente tutte le prospettive utilizzabili dalla cinepresa. Innanzitutto le produzioni di realtà virtuale nascono per dare libertà artistica e di movimenti di camera alle produzioni audiovisive, motivo per cui è raro che la camera sia statica. In secondo luogo, l'opportunità che offre questa tecnologia è di avere un background sempre coerente al foreground senza l'errore di parallasse. Diventa perciò interessante l'esplorazione dell'ambiente circostante al personaggio, inquadrandone diverse prospettive o facendo dei piani sequenza. Un green screen di dimensioni ridotte sarebbe poco adattabile a questa esigenza di versatilità che una produzione anche piccola e indipendente necessita. Infine, porre il green screen al di fuori del quadrato delle base station rende più versatile le scelte tecniche del VFX supervisor come la distanza dell'attore dal background e la conseguente tipologia di illuminazione. Un green screen posto all'interno di questo quadrato sarebbe infatti controproducente,

riducendo notevolmente lo spazio utile per le riprese, obbligando l'attore a starvi molto vicino e rischiando inoltre l'incorrere nell'occlusione delle stazioni base.

Illuminotecnica

Gli elementi fondamentali per ottenere delle riprese tecnicamente buone di un green set sono i materiali usati e l'illuminazione sia del background digitale che del soggetto nel set. I giusti materiali, la corretta illuminazione e posizionamento del soggetto genereranno un risultato ottimizzato per essere processato dall'algoritmo del nostro game engine. Per capire come ottenere questa efficienza computazionale bisogna ragionare dal punto di vista del calcolatore e di come la camera cattura l'immagine. Il background verde dev'essere uniformemente illuminato poiché le aree in ombra sono più difficili da "bucare" *. Questo problema diventa ancora più significativo per background posti dietro a capelli molto fini o ricci oppure dietro a oggetti trasparenti come il vetro o i liquidi. La posizione ottimale del soggetto è lontana dal green screen, viceversa se si trovasse troppo vicino nell'immagine si otterrà un effetto di "spill" verde a causa dalla luce riflessa dal background. Particolarmente sensibili a questa problematica sono gli oggetti riflettenti come i metalli o gli specchi che vanno maneggiati con cura o se possibile evitati. In ultimo, il soggetto deve avere un'illuminazione separata dal resto del set in modo da poter controllare senza vincoli esposizione e direzione delle sorgenti luminose necessarie per la scena che si deve riprendere. Prenderemo in esame i due set-up proposti da Foster [2010] in *"The Green Screen Handbook"* per approfondire come ultimare al meglio l'allestimento del set.

La figura 2.1 mostra un setup standard per l'allestimento di un green screen, ottimale per il contesto della virtual production dove è importante la facilità di elaborazione algoritmica. Questa tipologia di setup funziona a patto che il soggetto sia ben distanziato dal background, generalmente un minimo di 2,5 metri, e che non faccia ombra su di esso

*In questo contesto acquista il significato di elaborare il mascherino (*matte*) di un'inquadratura. Dal momento che quest'ultimo è un'immagine binaria con sfondo bianco e soggetto nero, la percezione visuale è quella di avere un rettangolo bucato dall'impronta del soggetto.

per evitare lo spill di ritorno causato dalla componente di luce riflessa del materiale da cui è composto lo sfondo. Per quanto concerne il soggetto può essere utilizzato il modello di illuminazione standard a 3 punti [3] . La *key light*, la sorgente luminosa predominante, dev'essere posizionata in corrispondenza alla direzione della sorgente luminosa più potente dell'ambiente virtuale. Temperatura colore e intensità sono inoltre ugualmente importanti e devono anche avere un match con la controparte digitale. La *fill light* è necessaria a sostenere l'illuminazione globale, creando l'illusione della presenza di una luce ambientale diffusa. Infine, la *back light* è fondamentale durante l'allestimento di un set di virtual production poiché crea un leggero "effetto *halo*" sui capelli e sui bordi del soggetto. Ciò fa si che il soggetto sia ben definito rispetto allo sfondo facilitando il compito svolto dal calcolatore nel rimuovere quest'ultimo.

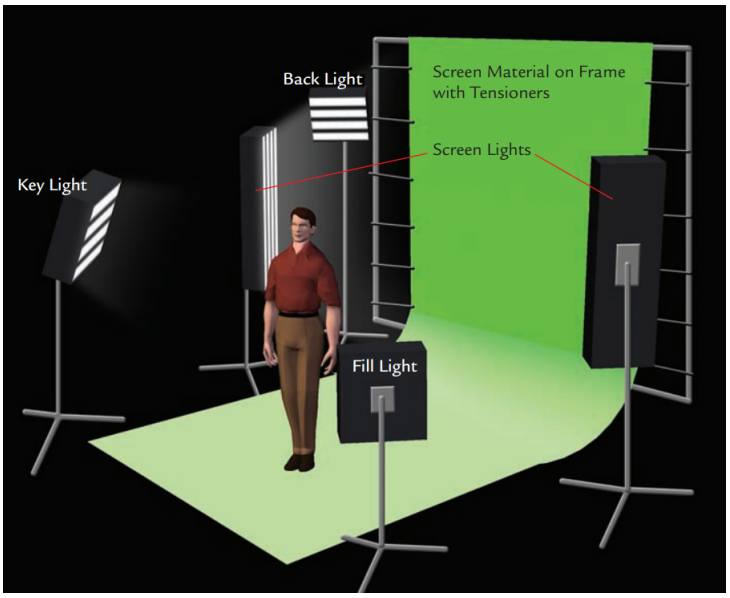

Figura 2.1. Setup standard di un green screen

La figura 2.2 un secondo modello di illuminazione che comprende sia il soggetto che lo sfondo. Esso è basato sull'uso di due luci laterali dette *spill suppressors* poiché hanno

[3] È un modello di illuminazione fotografica che assume che le tre luci essenziali siano la *key light*, la *fill light* e la *back light*.

l'obiettivo di eliminare proprio lo spill prodotto dallo sfondo sul soggetto. Questo modello viene utilizzato quando il soggetto per diversi motivi deve stare vicino al background verde eventualmente interagendo con esso. Risulta infatti particolarmente utile per sfondi come i muri su cui va mantenuta l'ombra. Per questo motivo la *key light* dev'essere particolarmente intensa in modo da proiettare un'ombra definita sullo sfondo verde. Per compensare lo spill riflesso si adoperano come accennato le *side spill-suppressors* da entrambi i lati del soggetto per scaldare * e pulire lo sfondo dallo spill creando una chiave pulita.

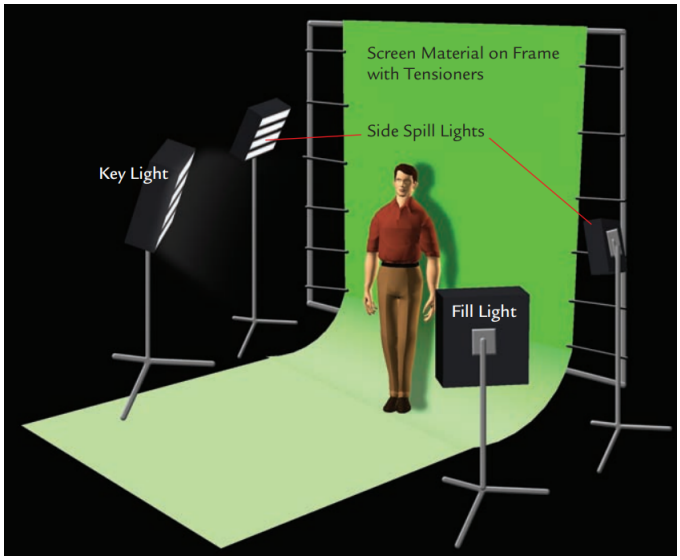

Figura 2.2. Setup di un green screen illuminato con il modello a spill-suppressors laterali

2.2 Tracking System

Il sistema di tracciamento prescelto è lo SteamVR Tracking 2.0 che ha un ottimo rapporto qualità prezzo e si pone quindi a sistema ideale per produzioni indipendenti low-cost. Infatti il setup essenziale con due Valve Index Base Station 2.0 e un solo HTC Vive Tracker

*In questo contesto è riferito alla temperatura colore. Significa traslare verso il rosso lo sfondo rispetto al punto di bianco di riferimento.

3.0 per la camera si può acquistare agevolmente per circa 500€, mentre il l'hardware di punta di questa tecnologia, HTC Vive Mars CamTrack, ideato appositamente per i set virtuali viene venduto oggi a 5.000€. Sebbene non siano cifre irrisorie, lo diventano quando confrontate agli standard dell'industria cinematografica, a causa dei quali spesso le piccole produzioni vengono tagliate fuori dal mercato. La tecnologia più famosa e con prestazioni migliori è probabilmente quella di OptiTrack™ [4] (usata anche per la produzione de "*Il Re Leone*" nella sua versione live-action e di *Star Wars: The Mandalorian* come mostrato in figura 2.3). Confrontando le specifiche si può evincere che l'opzione tecnologica più economica di OptiTrack che rende disponibile un'area paragonabile al setup base di SteamVR [5] ha un prezzo di circa 5.000€, mentre il il sistema più accurato nel setup paragonabile all'area massima dello SteamVR Tracking ha un costo di 50.000€. Si può quindi stimare che a parità di necessità tecniche e di spazio, utilizzare un sistema SteamVR abbatte il prezzo di circa 10 volte, con un risparmio del 90%. Nonostante risultati certamente meno accurati di sistemi più performanti, rimane comunque la prima scelta delle produzioni indipendenti dando loro l'opportunità di mettersi in gioco laddove sarebbe altrimenti impossibile economicamente.

Dal punto di vista tecnico sono entrambe tecnologie che si basano su una tipologia di tracciamento ottico e ricostruiscono la posizione del soggetto tramite la triangolazione. Nonostante ciò i due sistemi funzionano in modo profondamente diverso, ed in particolare la tecnologia SteamVR propone una soluzione innovativa nel settore.

[4] OptiTrack™ è il maggiore provider di motion capture, offre sistemi di tracciamento ottico a performance elevate al prezzo più conveniente nel mercato delle grandi produzioni. La linea di prodotti OptiTrack include Virtual Production, Movement Science, Realtà Virtuale e Robotica.

[5] Il setup essenziale con due stazioni ha un'area attiva di 5x5 metri quadrati, mentre il setup con 4 stazioni copre fino ad un massimo di 10x10 metri quadrati

Figura 2.3. Le camere Primex 41 di OptiTrack sul set di *Star Wars: The Mandalorian* poste sul set di virtual production principale delle riprese

2.2.1 OptiTrack™

OptiTrack™ si base su una tecnologia di tracciamento ottico di tipo *Outside Looking In* [6] . I marker utilizzabili possono essere sia attivi (emettono luce) che passivi (riflettono la luce incidente). In entrambi i casi la luce viene ripresa dalle telecamere che attraverso la triangolazione ricostruiscono la posizione fisica tridimensionale del marker. Un esempio di marker passivo sono le tipiche sfere riflettenti poste sulle tute di motion capture degli attori. Le camere in dotazione con la tecnologia OptiTrack™ possono emettere segnali luminosi a infrarossi che vanno a incidere sul marker creando hotspot luminosi. Tramite algoritmi di computer vision si possono identificare le zone dove l'intensità di luce è più elevata e ricostruirne la posizione tramite la triangolazione. Posizione e orientamento vengono così ricavati:

[6]Esistono due tipologie di tracking ottico (figura 2.4). La *Outside Looking In* si base su telecamere fisse che captano il movimento che avviene all'interno dell'ambiente di sviluppo, mentre nella *Inside Looking Out* le telecamere sono posizionate sull'oggetto in movimento e i marker nell'ambiente circostante.

1. Attraverso la calibrazione delle telecamere utilizzate all'interno del sistema di riferimento comune ne si conosce orientamento e posizione del piano immagine di ognuna

2. Si può quindi ricostruire la retta di tutti i punti appartenenti allo spazio tridimensionale hanno come loro proiezione un punto bidimensionale sul piano immagine

3. Conoscendo la posizione nel piano immagine del marker in almeno due immagini diverse (almeno due sensori) possiamo tracciare due rette tridimensionali di proiezione e la loro intersezione corrisponderà alla posizione (approssimata [7]) del marker nel sistema di riferimento (figura 2.5)

4. Per ottenere le informazioni relative all'orientamento del marker si compongono le posizioni di un cluster di tre marker per calcolare il piano da essi definito. La normale uscente da ogni piano indica l'orientamento del cluster in esame. Ponderando i risultati per diversi terzetti si ottengono risultati attendibili per ogni marker.

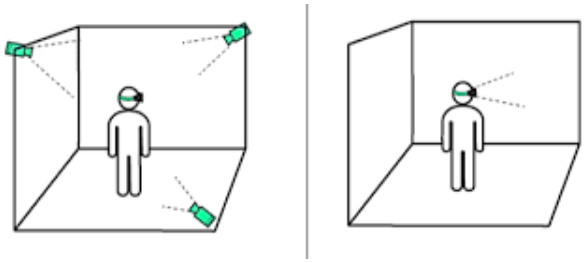

Figura 2.4. A sinistra un sistema *Outside Looking In* e a destra uno *Inside Looking Out*

Come anticipato questa tecnologia supporta anche i marker di tipologia attiva e consiste di una base station che emette un segnale radio sincronizzato con la frequenza su cui sono settati i markers attivi permettendo una sincronizzazione precisa tra l'esposizione delle

[7] In realtà è altamente improbabile che le rette siano incidenti a causa della natura empirica soggetta ad errori di qualsiasi tecnologia. Si sceglie quindi come posizione del tracker il punto medio corrispondente al segmento di distanza minima tra le due rette. Da qui nascono i principali errori di precisione e accuratezza dei sistemi di tracciamento.

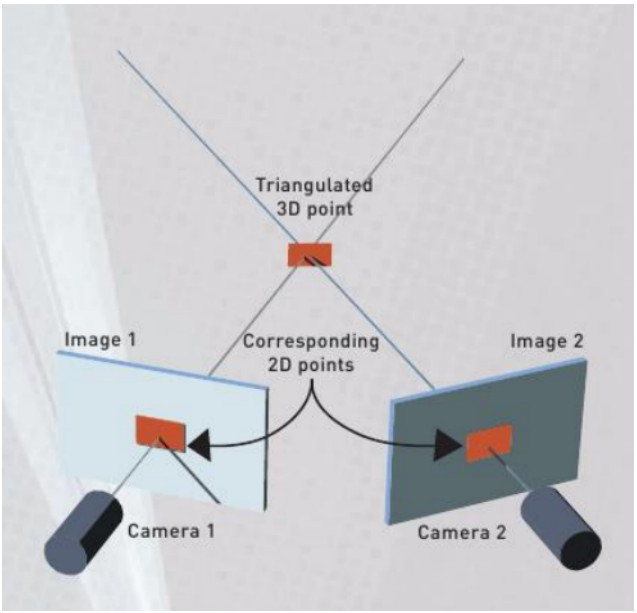

Figura 2.5. Esempio di triangolazione nel caso di tracciamento ottico

camere e l'illuminazione dei LEDs (che emettono luce a infrarossi con una lunghezza d'onda pari a 850 nm). OptiTrack

2.2.2 SteamVR™

Lighthouse Tracking System è un sistema di tracciamento posizionale *Outside Looking In* basato su raggi laser sviluppato da Valve per SteamVR e HTC Vive. Il fulcro concettuale di questa tecnologia sono le Base Stations che servono come punti di riferimento per ogni device che necessita di essere tracciato. Le stazioni di base di SteamVR scansionano la stanza con diversi fasci laser e pulsazioni di sincronizzazione, arrivando a coprire fino a circa 5 metri di distanza (figura 2.6). Calcolando attentamente l'intervallo tra le pulsazioni e le scansioni, il sistema di tracciamento di SteamVR utilizza della semplice trigonometria per trovare la posizione di ciascun sensore con precisione millimetrica. Combinando diversi sensori, 2 stazioni di base e un'unità di misura inerziale (IMU), SteamVR è anche in grado di calcolare l'orientamento, la velocità e la velocità angolare dell'oggetto tracciabile, il tutto ad una frequenza di aggiornamento pari a 1000Hz. Steam

Invece di LEDs a infrarossi, Lighthouse incorpora nei propri devices degli array di fotodiodi capaci di filtrare i raggi infrarossi (figura 2.7). Ogni Base Station è dotata di un Sync Blinker e due fasci laser che ruotano velocemente disegnando un piano X e uno Y. 60 volte al secondo, il Sync Blinker emette una pulsazione di sincronizzazione e uno dei due laser getta un fascio laser intorno alla stanza. Quando il ricevitore rileva la pulsazione di sincronizzazione cronometra il tempo utile affinché uno dei suoi diodi sia colpito da uno dei fasci laser. Il sistema è quindi in grado di calcolare *quando* il fotosensore viene colpito e *dove* è posizionato il diodo rispetto al tracker per trovare l'esatta posizione di quest'ultimo in riferimento alla Base Station.

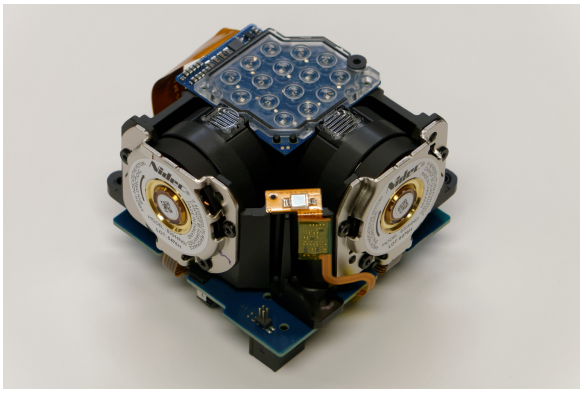

Figura 2.6. Interno di una Base Station 1.0 con Sync Blinker e una coppia di motori sincroni tri-fase a magnete permanente senza spazzole che generano i fasci laser rotanti

Valve ha pubblicato che è necessario che almeno 5 diodi debbano essere colpiti e presi in considerazione dall'algoritmo del sistema per poter stimare posizione e orientamento del device. Conoscendo quindi la configurazione dei diodi, il tempo in cui ognuno viene colpito e quindi gli scarti temporali tra i diversi eventi, SteamVR è in grado di calcolare i 6 DOF [8] del device. Prendendo in esame il Tracker 3.0, esso ha un'origine conosciuta che è il punto in cui si trova la filettatura per la vite alla base del tracker. Nello specifico possiede

[8] In inglese è l'acronimo di *Degree Of Freedom* (gradi di libertà) e si riferisce alla libertà di movimento nello spazio tridimensionale. Ci sono 3 gradi per le direzioni di traslazione e altri 3 gradi per i tipi di rotazione lungo gli assi ortogonali.

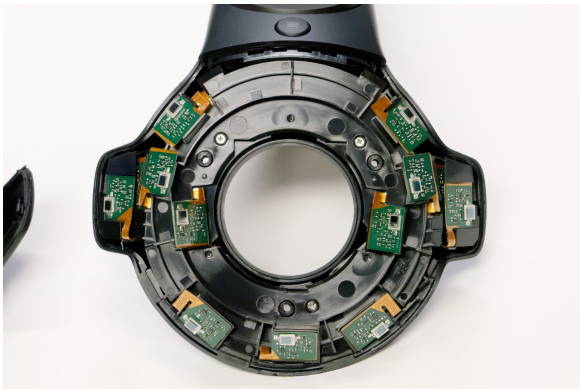

Figura 2.7. Interno di un device HTC Vive (controller) con fotodiodi

22 diodi con filtro dei raggi infrarossi. Bauer et al. [2021] Il sistema di riferimento comune viene calcolato a partire dalla posizione delle base station. I fasci laser vengono generati nell'ordine dettato dai canali su cui sono impostate le Base Station, partendo dal minore. Sarà quindi il fascio laser della stazione con numero di canele minore a colpire per primo il tracker, facendo in modo che questo la elegga come origine del sistema di riferimento. L'asse dimensionale Y verrà in seguito calcolato come il versore del vettore unente la prima e la seconda stazione, il versore Z viene ricavato dagli accelerometri presenti nel tracker ed infine quello X sarà calcolato come versore ortogonale al piano YZ. Come si può facilmente evincere questo è un sistema altamente sensibile all'occlusione e che necessita che le Base Station abbiano una retta ideale che le colleghi al tracker senza interruzioni affinché esso possa essere tracciato correttamente.

2.3 Input Video

Le sequenze di immagini catturate dalla cinepresa devono essere trasmesse al computer affinché possano essere riutilizzate dal game engine in tempo reale. A questo proposito Unreal Engine mette a disposizione degli sviluppatori due media framework per agevolare il flusso di elaborazione video. I due brands che hanno stretto la partnership con Unreal sono Blackmagic Design e AJA. Nella pratica vengono resi disponibili dei plugin

che permettono la compatibilità di alcuni components specifici con le schede di acquisizione video [9] dei due brand. Anche non avendo a disposizione alcun materiale di questi marchi è comunque possibile proseguire nel flusso di lavoro con una scheda di acquisizione video diversa dai brand sopra citati tramite i bundle standard di *file media input* del game engine. Come vedremo nel capitolo 5, nel caso di questo progetto è stata utilizzata la scheda di acquisizione video *Cam Link 4K* di Elgato, una scheda dal design semplice leggero dotata in estremità con uscita USB ed una con un entrata HDMI. Ad ogni modo passare attraverso i framework pensati e dedicati per la trasmissione di dati video rivela importanti vantaggi. Blackmagic ha pubblicato che dalla versione 4.21 di Unreal Engine di Epic Games il software è diventato compatibile con le schede di acquisizione e riproduzione DeckLink 8K Pro, DeckLink Duo 2 e DeckLink 4K Extreme 12G (Figura 2.8) di Blackmagic Design e che tra i vantaggi associati i seguenti sono i più significativi:

- Il supporto per video, audio e timecode [10] in entrata in tempo reale
- Il supporto per l'invio di segnali di chiave e di riempimento, per acquisire immagini Unreal Engine renderizzate e distribuirle tramite SDI
- La lettura di file YUV e la loro conversione a 8 o 10 bit o in RGB a 8 o 10 bit
- La sincronizzazione di Unreal Engine su un ingresso video o il suo Genlock [11] su un'uscita video
- Il supporto per l'HD progressivo e interlacciato fino al 1080p60

[9] Un dispositivo di acquisizione video (ad esempio una scheda) è uno strumento hardware che si può connettere al computer e che converte il segnale video in uscita dalla camera (con connettore HDMI o SDI per esempio) in un formato digitale riconoscibile dal computer stesso. (Figura 2.9)

[10] Il timecode è una metrica di conteggio del tempo che assegna un numero identificativo unico sequenziale ai frame delle riprese audio e video. Può essere quindi adoperato per identificare il tempo esatto di un evento. I timecode di ogni dispositivo sono però indipendenti l'uno dall'altro e perciò possono dover essere sincronizzati.

[11] Il *Generator Locking*, anche detto Genlock, è la tecnica di sincronizzazione di sorgenti video che funziona grazie a un segnale di sync riferimento emesso da un generatore di segnali. Il Genlock abilita la corretta sincronizzazione di più video in contesti come gli studio broadcast che usano set-up multi-camera.

Figura 2.8. Scheda di acquisizione video Blackmagic Design DeckLink 4k extreme 12G

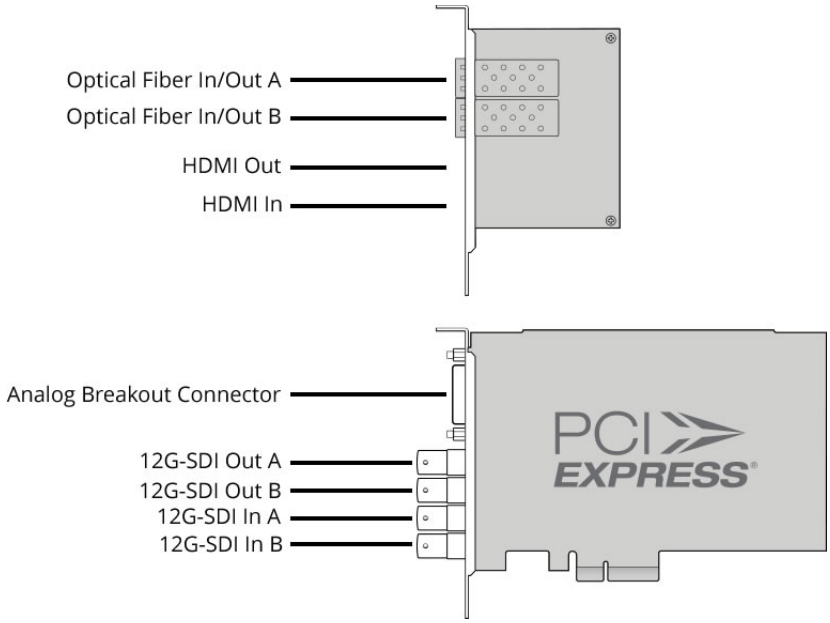

Figura 2.9. Schema descrittivo delle tipologie di ingressi ed uscite disponibili sulla scheda di acquisizione video Blackmagic Design DeckLink 4k extreme 12G

2.4 Controllo DMX dell'Illuminazione

Nella progettazione del desgin di un set di virtual production diventa cruciale la gestione delle luci. Abbiamo analizzato in precedenza l'importanza che durante l'allestimento di un green screen tutta l'illuminazione sul soggetto sia coerente con quella del background

digitale. Tuttavia se nella catena produttiva tradizionale questo è un aspetto che coinvolge le professionalità che lavorano nella fase di post-produzione, nella virtual production la gestione delle luci diventa un problema da affrontare sul momento delle riprese stesse. La via maestra è quella di utilizzare apparecchiature per l'illuminazione del set comandabili via DMX. Usando una console DMX per generare dei segnali (ovvero dei comandi), adibire nell'ambiente digitale delle copie virtuali dei nostri apparecchi fisici permetterà di inviare una copia del segnale in parallelo al game engine, con l'automatica sincronizzazione temporale e degli attributi degli apparecchi fisici e virtuali. Per riassumere, aumentare o diminuire l'intensità della luce sul set corrisponderà alla stessa azione anche nell'ambiente virtuale per esempio, e similmente per qualsiasi altro attributo compresi quelli di movimento delle apparecchiature. Inoltre, grazie alla tipologia di comunicazione che si basa su cavi *ethernet* e XLR, l'assenza di latenza è intrinseca al sistema permettendo sia di usare *Show DMX* registrati sia di modificare gli attributi delle luci *live* e in modo sincrono su hardware e in software.

2.4.1 Lo Standard DMX

Il DMX (Digital Multiplex) è uno standard per le reti di comunicazione digitale ed è usato in tutta l'industria dell'audiovisivo per controllare diversi devices durante eventi live come apparecchi di illuminazione, laser, macchine del fumo, strumenti meccanici o pirotecnici. Inizialmente era stato pensato come metodo standardizzato per il controllo dei *dimmer* [12] delle luci, ma presto diventò il metodo principale per collegare controllers come le console per l'illuminazione ai dimmer e agli altri apparecchi per effetti speciali. Epic Games ha deciso di mettere a disposizione del sistema di comunicazione dati DMX sia la variante *Artnet* che quella *sACN*. *Artnet* e *sACN* sono dei protocolli di rete che permettono di aggregare e inviare dati attraverso cavi ethernet (quindi con indirizzo IP). Il protocollo *Artnet* ha 32.768 universi disponibili per ogni singolo cavo di rete. *sACN* (streaming

[12] Piccolo reostato (resistore a resistenza variabile) per regolare l'intensità luminosa delle lampade

architecture for control networks), attualmente è più popolare grazie alla sua modernità ed alla possibilità di far girare 63.999 universi per dati DMX attraverso un unico cavo.

I Dati

Il DMX può essere pensato come un pacchetto di informazioni digitali che dev'essere inviato da una sorgente a una destinazione. Ogni pacchetto è creato in una certa sorgente con informazioni specifiche che dovrebbero essere ricevute e lette da altri ricevitori. Ogni pacchetto contiene un vettore di 512 bytes ognuno con un valore compreso tra 0 e 255.

$$byte\_1, \ldots, byte\_512$$

Un *universo* può essere pensato come identificazione di un gruppo di apparecchi fisici o virtuali connessi insieme e che leggono gli stessi dati. Infatti un *universo* corrisponde al contenuto singolo pacchetto e perciò contiene 512 bytes di informazioni cosicché il numero di apparecchiature in quella linea dipenda da quanti canali [13] sono necessari per indirizzare ogni fixture.

Il Network

I controllers DMX (anche detti "nodi") agiscono come sorgente dove il segnale DMX viene generato. Viceversa le fixtures DMX sono i devices realmente responsabili per la ricezione e l'esecuzione dei comandi sulla base dei dati ricevuti. Ogni controller è addetto ad uno o più universi, ognuno dei quali ha una fila di più fixture collegate in serie (dette *daisy-chained*, un esempio in figura 2.10). Al fine di inviare i dati agli apparecchi opportuni, oltre ad inviarli all'universo corretto bisogna anche quindi assicurarsi che le diverse fixture siano settate ognuna sui rispettivi canali di ascolto per non sovrapporsi a vicenda. Una volta che un controller ha ricevuto il comando di distribuire un determinato pacchetto di dati DMX, esso alloca l'opportuno universo e invia il pacchetto di dati lungo la linea di fixture

[13] Un canale corrisponde al singolo byte, ovvero ad un valore tra 0 e 255. Una semplice luca RGB per esempio ha bisogno di 3 canali per essere indirizzata, un valore 0-255 per ogni componente primaria della luce.

collegate in serie affinché possano leggere ed interpretare il messaggio. Ogni fixture riceve quindi lo stesso pacchetto di dati ed esegue un comando interno nel caso in cui ci fossero dati significativi per quella fixture in particolare (ovvero se uno dei canali dell'universo ricoperti da quella fixture è incluso nel messaggio). Una volta letto, il pacchetto avanza lungo la catena alla fixture successiva ripetendo il processo. Affinché avvenga la corretta corrispondenza tra struttura del contenuto del messaggio e impostazioni di modalità di ascolto delle singole fixture si utilizza la procedura di *patching*.

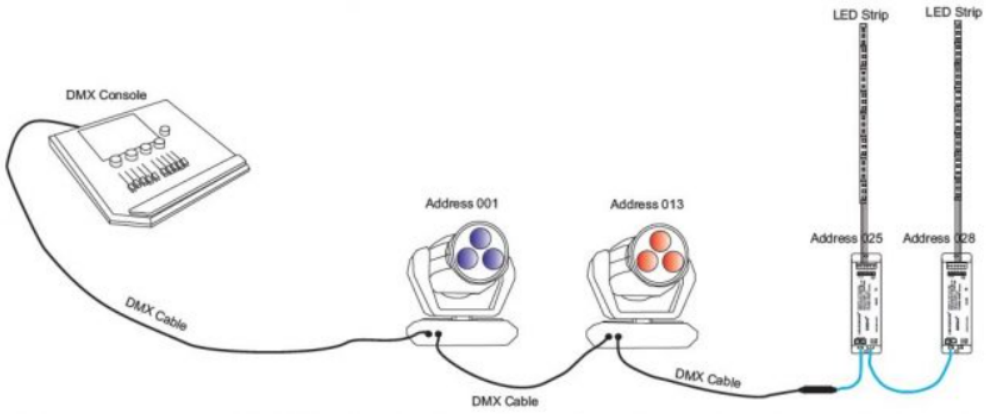

Figura 2.10. Schema esemplificativo di una connessione daisy-chained di fixture DMX

Patching

Il *patching* serve per poter posizionare virtualmente le fixtures lungo una catena di comunicazione (un universo) assicurandosi che i dati siano ricevuti nel modo opportuno. È importante avere un modo univoco per identificare esattamente quali bytes nel pacchetto debbano essere letti e decodificati e quali ignorati. Come mostrato nella tabella 2.1, per farlo si assegna ad ogni fixture uno specifico indirizzo di partenza all'interno dell'universo dal quale iniziare ad interpretare i dati. Assegnando un determinato indirizzo di partenza, la fixture in considerazione occuperà di conseguenza un intervallo di indirizzi la cui ampiezza è definita dal numero di attributi che possiede la fixture stessa.

	Universo 1		
Fixture	Attributi	Indirizzo Iniziale	Range
1	7	1	1-7
2	8	8	8-15
3	12	16	16-27
4	5	28	28-32
5	32	33	33-64
6	4	65	65-68

Tabella 2.1. Patching e posizionamento di 6 fixture nel primo universo in base ai rispettivi attibuti.

2.4.2 I Protocolli DMX

Come mostrato in figura 2.11, i controllers (anche detti "nodi") agiscono come distributori di dati a un gruppo di fixture collegate in serie. Un controller DMX può essere un dispositivo di interfaccia di rete oppure una console per lo standard DMX. Epic [b]

Interfacce di rete

Un'interfaccia di rete converte pacchetti IP in segnali DMX che sono trasmessi a gruppi di fixtures connesse in serie. I protocolli con tecnologia ethernet sono stati sviluppati per gestire universi DMX multipli attraverso un unico cavo *Cat5*. Esistono due principali protocolli ethernet largamente utilizzati e sono *Art-Net* e *sACN*:

- *Art-Net* è un protocollo di comunicazione royalty-free [14] per la trasmissione tramite UDP [15] del protocollo di controllo dell'illuminazione DMX512-A e del protocollo di gestione remota dei dispositivi (RDM). Viene utilizzato per comunicare tra i *nodi* (dispositivi intelligenti di controllo dell'illuminazione) e un *server* (una console luci o un computer su cui giri un software per il controllo dell'illuminazione).

- *sACN* è l'acronimo di Streaming Architecture for Control Networks, ed è il protocollo standard sviluppato da *ESTA* [16] per la trasmissione efficiente di universi DMX in una rete. Permette di allocare quasi il doppio degli universi del protocollo *Art-Net* ed inoltre implementa un'ottima gestione per le trasmissioni multicast tramite una semplice configurazione.

[14] Royalty-free è un tipo di licenza che permette l'utilizzo di una risorsa (foto, video, audio, eccetera) con limitate restrizioni sul suo utilizzo e pagando una cifra iniziale estremamente contenuta. Wikipedia [d]

[15] Lo User Datagram Protocol (UDP), nelle telecomunicazioni, è uno dei principali protocolli di rete della suite di protocolli Internet. È un protocollo di livello di trasporto a pacchetto, usato di solito in combinazione con il protocollo di livello di rete IP. Wikipedia [f]

[16] Entertainment Services and Technology Association. È un'associazione di categoria senza scopo di lucro con base nel Nord America che si occupa della promozione della crescita dell'industria dell'intrattenimento anche dettandone gli standard.

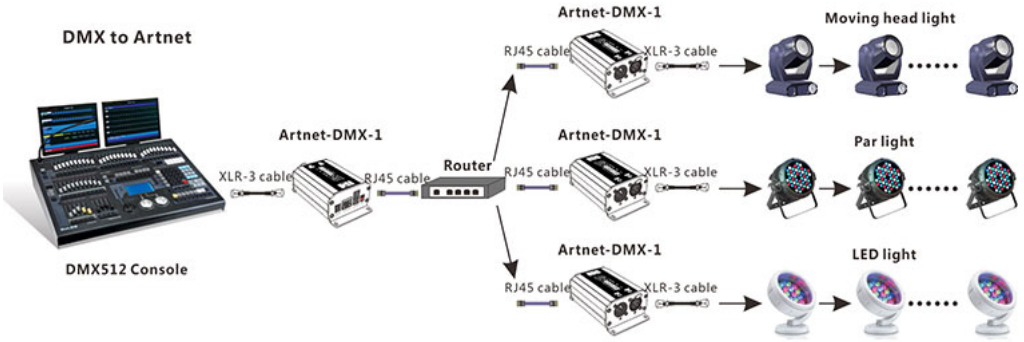

Figura 2.11. Un esempio di network DMX che gira su più universi con una console collegata ad un nodo Art-Net

Console

Una console DMX (figura 2.12) è simile ad un mixer audio esteticamente, ma a differenza di quest'ultimo essa è una sorgente. Tramite il display incorporato è possibile "patchare" determinati apparecchi per l'illuminazione. In genere sono presenti delle librerie che si possono aggiornare e che contengono già buona parte dei dispositivi di illuminazione sul mercato pre-impostati con i rispettivi attributi. Una volta scelto l'apparecchio che si vuole controllare e aver impostato il corretto numero di attributi si deve proseguire al *patching*, allocando lo spazio opportuno per quel determinato ricevitore (come già mostrato in tabella 2.1). Allo stesso tempo bisognerà impostare manualmente il singolo dispositivo in ascolto sul canale corretto (il primo indirizzo occupato). Tramite i fader ed i knobs della console è quindi possibile controllare i diversi attributi della fixture (variando il valore espresso da ogni byte). Grazie alle console è anche possibile registrare degli show (sequenze di comandi) e farli riprodurli a proprio piacere in qualsiasi momento.

Figura 2.12. Una console DMX Chamsys QUICKQ-10 512 Canali

Capitolo 3

Il Software - Unreal Engine 5

Tra i vari game engine è stato scelto come software a cui appoggiarsi per lo sviluppo di un framework per la virtual prodcution *Unreal Engine*. Ciò è avvenuto per due motivazioni principali. Innanzitutto è uno standard a livello globale e tutti i più grandi players dell'industria cinematografica si affidano ad esso. Infatti offre numerose suite sviluppate *ad hoc* per la gestione ed il controllo di set per produzioni virtuali sia *full live* che ibride con framework dedicati ad appositi strumenti di brand leader nell'industria dell'audiovisivo. In secondo luogo è *open source*, e quindi disponibile a costo zero anche per le produzioni indipendenti che vedono nello scoglio economico uno dei maggiori ostacoli ai loro progetti. Un game engine adatto alla Virtual Production deve offrire diversi tool che semplificano il workflow del progetto invece di renderlo eccessivamente complesso. Senza frameworks adatti non sarebbe infatti impossibile sviluppare un proprio plugin ma diventerebbe un vero e proprio onere che in pochissimi potrebbero supportare. Tra i tool necessari elenchiamo plugin adibiti al reindirizzamento interno al software di segnali video in ingresso nel calcolatore, plugin per l'elaborazione delle immagini e video con algoritmi adatti keying ed al compositing in tempo reale, plugin per la calibrazione della cinepresa virtuale, del colore nell'immagine compositata finale e infine delle comunicazioni DMX all'interno del sistema software e hardware. Insomma, le sfaccettature da tenere in conto sono molte e serve un sistema solido e robusto affinché il tutto possa funzionare in maniera sincrona ed ottimale.

3.1 L'Interfaccia

La figura 3.1 mostra come si presenta l'interfaccia di default di Unreal Engine. Di seguito un'analisi delle diverse finestre da cui è composta:

1. **Toolbar Principale**. Questo menù permette di accedere velocemente alle impostazioni generali ed alle preferenze del sistema (*edit*), ai tool forniti dal software tra cui una sezione dedicata alla virtual production (*tool*), ad actor comuni da istanziare con opzionalità *drag and drop*.

2. **Barra del Menù**. Questo menù contiene *shortcut* per alcuni degli editor e dei tool più comuni del software. Inoltre dispone anche del bottone di *play mode* per la preview di rendering real-time impostabile come preview del singolo livello, preview di tutti i livelli oppure simulazione. Permette inoltre di selezionare la modalità di interazione con l'interfaccia dal menù a tendina sulla sinistra che riporta le seguenti opzioni:

   - *Select Editing.*
   - *Landscape Editing.*
   - *Foliage Editing.*
   - *Mesh Editing.*
   - *Fracture Editing.*
   - *Brush Editing.*

   I bottoni di shortcut aprono invece tre diverse comuni interfacce per la generazione, gestione o piazzamento di:

   - *Actors.* Genericamente qualsiasi elemento/oggetto che popoli la scena virtuale.
   - *Blueprints.* Script programmabili a nodi eventualmente annessi ad un oggetto ed instanziabili all'interno della scena.
   - *Cinematic Sequence.* Crea livelli per sequenze di animazioni o azioni registrabili all'interno del software.

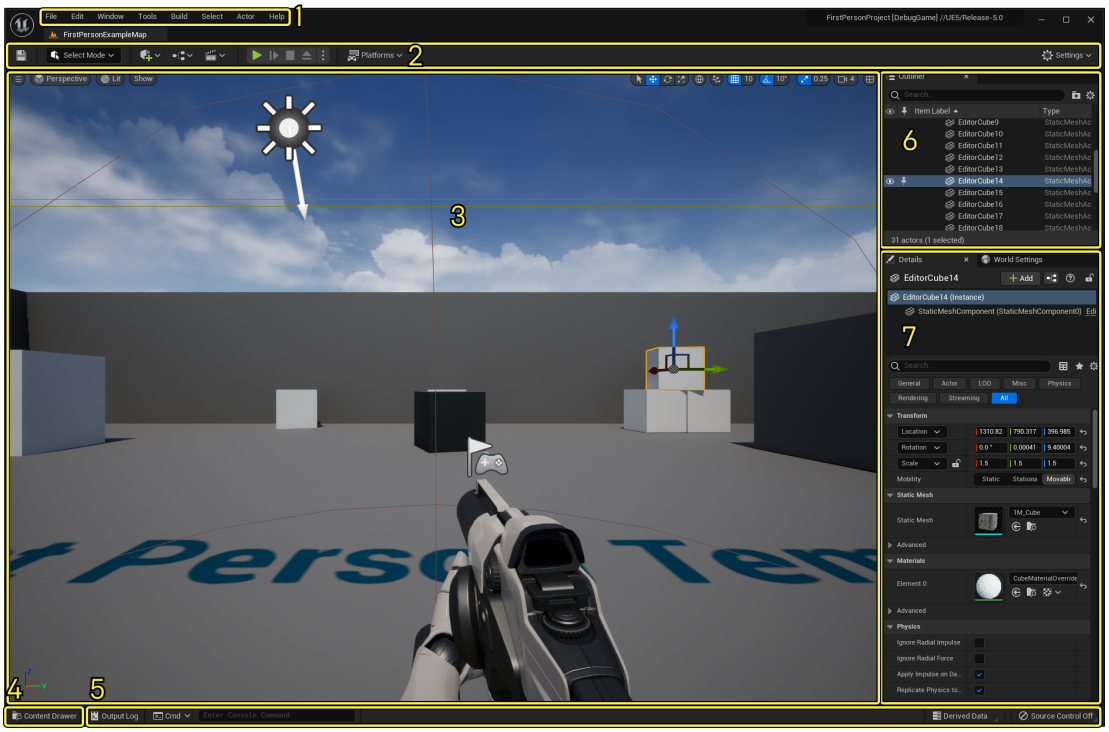

Figura 3.1. Interfaccia default di *Unreal Engine*.

3. **Level Viewport**. Questa finestra mostra sullo schermo il contenuto del mondo virtuale del livello corrente all'interno del quale si possono modificare actors esistenti o istanziarne di nuovi. La visualizzazione è disponibile sia in modalità prospettica che ortogonale oltre a disporre della vista *lit*, *unlit* o *wireframe*.

4. **Content Drawer**. Questa finestra è un *file explorer* interno alla cartella del software memorizzata nel nostro computer. Si può quindi utilizzare per l'organizzazione gerarchica di tutti i file all'interno delle cartelle di progetto. È anche possibile importare file esterni come texture, materiali o oggetti con estensione .fbx del file.

5. **Toolbar Inferiore**. Questo menù mette a disposizione elementi di controllo del software e di debug come l'*output log* e la *linea di comando*.

6. **Outliner**. Questo pannello mostra una vista gerarchica di tutto ciò che è contenuto all'interno del livello. Dà inoltre la possibilità di abilitare/disabilitare la visibilità di

tali elementi.

7. **Details**. Questo pannello si apre ogni qualvolta viene selezionato un *actor* mostrandone impostazioni e proprietà che possono influire o modificare il suo comportamento.

3.2 Terminologia

Al fine di chiarire ai lettori il contenuto e la trattazione che avverrà in questa tesi si propone une breve elencazione della terminologia più comune propria del game engine Unreal Engine.

Project

Un progetto ha estensione .uproject e contiene tutti i contenuti del gioco che si sta sviluppando. La stessa struttura che possiede all'interno della cartella memorizzata sul disco del computer è mostra dal *content browser*.

World

Il mondo è un container di tutti i livelli che costituiscono l'applicazione che si sta sviluppando.

Level

Un livello è un'area di gioco/interazione definita dallo sviluppatore. Ogni livello ha estensione .umap e contiene tutti gli oggetti che l'utente può vedere.

Class

Una classe definisce il comportamento di un actor o un oggetto specifico all'interno di Unreal Engine. Le classi sono gerarchiche e perciò i figli ereditano informazioni dai propri padri. Le classi possono essere definite attraverso codice *C++* o grazie ai *Blueprints*.

Object

Gli oggetti sono la classe basica in Unreal Engine, sono mattoni elementari che contengono funzioni essenziali per gli asset. Quasi tutto infatti in Unreal Engine eredita da una classe C++ *UObject*.

Actor

Un *actor* è qualsiasi oggetto venga posizionato in un livello come una camera o una mesh. Tutti gli *actor* supportano le trasformazioni tridimensionali. Possono essere creati o distrutti per mezzo dei Blueprint. La classe C++ basica di tutti gli *actor* è *AActor*.

Blueprint

Lo scripting visivo fornito dai *blueprint* è un sistema di programmazione per videogiochi con un'interfaccia a nodi che permette di creare elementi per il mondo direttamente dall'editor di Unreal Engine. Attraverso questo sistema si definiscono classi o oggetti tramite la programmazione ad oggetti.

Component

Un componente è una funzionalità che può essere addizionata ad un *actor*, fornendo la possibilità al target actor di usufruire delle sue funzioni. (Un esempio che vedremo è il *LiveLinkComponent* che dà la possibilità ad un *actor* di essere controllato da remoto).

3.3 Virtual Production Tools

Nella sua versione 5.0, Unreal Engine mette a disposizione una suite di 28 plugins dedicati alla virtual production. Tutti insieme creano il bagaglio d'attrezzi indispensabile per il design di un framework per le produzioni virtuali. Nella trattazione ci dilungheremo nella spiegazione dell'utilizzo e funzionamento di quelli contemplati nel progetto di tesi. Perciò ora ci limitiamo ad una panoramica generale spiegandone l'importanza.

- Plugin di tipologia generale per la **Virtual Production**. Includiamo in questa famiglia il plugin generico *Virtual Production Utilities*.
- Plugin per le trasmissioni **media I/O**. Fanno parte di questa famiglia quei plugin necessari importare o esportare in tempo reale flussi di segnali video dalla camera o verso la camera. Sono: *Media IO Framework*, *Stage Monitor*, e *Virtual Camera*.
- Plugin per il **compositing real-time**. Fanno parte di questa famiglia tutti quei plugin che insieme permettono l'attuazione e la gestione di algoritmi e script per l'elaborazione di immagini e video, in particolare facendo riferimento alla fase di compositing. Sono: *Composite Plan* e *Composure*.
- Plugin per il **tracciamento real-time**. Fanno parte di questa famiglia tutti quei plugin come Live Link che permettono la connessione a dispositivi di tracciamento in tempo reale per il controllo di oggetti virtuali. Sono: *LiveLinkCamera*, *LiveLink-FreeD*, *LiveLinkLens*, *LiveLinkVRPN*, *LiveLinkXR* e altri ancora.
- Plugin per la **calibrazione**. Fanno parte di questa famiglia tutti quei plugin che permettono la calibrazione degli attributi degli strumenti fisici sul set nonché della conservazione del principio di simultaneità tra il game engine ed il set fisico. Sono: *Timed Data Monitor*, *Camera Calibration* e *OpenCV Lens Distortion*.
- Plugin per le comunicazioni e la gestione del **protocollo DMX**. Fanno parte di questa famiglia tutti quei plugin che svolgono un ruolo nella pipeline di gestione delle comunicazioni DMX e nel controllo dell'illuminazione dello stage fisico e virtuale. Sono: *DMX Protocol*, *DMX Engine*, *DMX Fixtures* e *DMX Pixel Mapping*.
- Plugin per la **registrazione real-time**. Fanno parte di questa famiglia tutti quei plugin utili alla registrazione cinematografica in tempo reale dell'immagine finale compositata su Unreal Engine. Sono *Take Recorder*, *Take Recorder Multi-User Synchronization* e *Playlist*.
- Plugin per la gestione di **LED Wall**. Fanno parte di questa famiglia tutti quei plugin che svolgono un ruolo nel processazione e nella calibrazione e sincronizzazione dei display *LED Wall*. Sono: *ICVFX*, *LED Wall Calibration* e *DMX DisplayCluster*.

Parte II

Implementazione del Framework

Nei prossimi capitoli affronteremo direttamente lo sviluppo del framework implementato durante questo progetto di tesi. Ogni capitolo affronta singolarmente una fase della progettazione di un set adatto ad una produzione che utilizzi una *SimulCam*, ma solo se prese tutte insieme le parti queste si coordineranno per dare vita ad un set funzionante ed efficiente. Si vogliono quindi ripercorre le tappe cruciali che hanno reso possibile la realizzazione fisica del framework e per questo motivo verranno analizzati solamente quegli assets e quelle procedure che hanno trovato posto all'interno del progetto di tesi. Nella conclusione verrà invece dato spazio a tutte quelle implementazioni ed integrazioni che potrebbero occorrere in uno sviluppo futuro di questo framework. Il carattere descrittivo dei prossimi capitoli vuole rendere la dimensione pratica del lavoro svolto e fornire questa volta appunto un "manuale pratico" per i futuri studenti che si approcceranno allo studio delle virtual production su Unreal Engine od i tecnici delle produzioni indipendenti che vorranno addentrarsi nell'approfondimento di queste tecnologie.

Capitolo 4

Camera Tracking

4.1 Steam VR

La tecnologia scelta per l'implementazione del framework del progetto di tesi è *SteamVR* nella sua declinazione più economica e con setup essenziale. Sono state quindi acquistate due Valve Index Base Station 2.0 (il minimo richiesto affinché questa tecnologia possa funzionare) ed un HTC Vive Tracker 3.0 per il tracciamento della sola camera. Come visto in precedenza, alcuni sistemi artigianali più complessi possono prevedere l'utilizzo del tracker per misurare oltre che alla posizione della cinepresa anche la variazione di lunghezza focale, messa a fuoco o diaframma. In verità la soluzione più professionale è quella di utilizzare lenti performanti che tramite un codificatore per lente sia in grado di inviare i dati di rappresentazione della lente al nostro computer [1] , altresì sono soluzioni di alto profilo e decisamente meno economiche. Il sistema SteamVR al completo prevedrebbe l'utilizzo della tecnologia HTC Vive Mars che come vedremo nel capitolo 8 risolve molte delle problematiche di delay e calibrazione in cui si può andare incontro durante l'implementazione del framework grazie al cablaggio delle connessioni e ad un hub di gestione del sistema multi-camera.

[1] Un esempio recente è il codificatore wireless per lenti cinematografiche LDT-V2 lanciato da *DCS* appositamente per la Virtual Production.

Il sistema SteamVR funziona per mezzo di un'app che possiamo scaricare gratuitamente e velocemente dallo store di Steam [2] . Una volta scaricata l'app non ha bisogno di nessuna configurazione particolare per la destinazione d'uso che occorre ad una produzione virtuale. Dopo il lancio, l'applicazione si mette in ascolto di segnali captabili dal dongle USB e apre una piccola finestra dove mostra i dispositivi connessi (figura 4.1).

Figura 4.1. Finestra aperta dall'applicazione SteamVR dopo il lancio. Un tracker e due stazioni base correttamente rilevate

4.1.1 Base Stations

Per procedere all'utilizzo dell'applicativo SteamVR bisogna innanzitutto allestire adeguatamente lo spazio dedicato al set di virtual production. Questo è molto importante per ottenere un sistema altamente adattabile alle diverse circostanze tecniche e artistiche che possono variare di ripresa in ripresa. Andrà utilizzato infatti il massimo spazio a disposizione (conformemente a quello permesso dalla tecnologia) in modo da non dover cambiare allestimento tra riprese diverse. Lo spazio adibito al sistema di tracciamento è potenzialmente un quadrato, ed al suo interno è meglio che non ricada lo sfondo green screen per evitare occlusioni parziali delle stazioni base (come visto nel capitolo 2 è opportuno

[2]Steam è una piattaforma sviluppata da Valve Corporation che si occupa di distribuzione digitale, di gestione dei diritti digitali, di modalità di gioco multi-giocatore e di comunicazione. Viene usata per gestire e distribuire una vasta gamma di giochi (alcuni esclusivi) e il loro relativo supporto. Wikipedia [e]

che esso sia adiacente all'area definita dalle stazioni base). Le regole principali da seguire sono:

- Non eccedere dallo spazio massimo consentito dal materiale a disposizione. Un setup con due stazioni base permette un'area massima di tracciamento di 5x5m quadrati, mentre un setup con 4 permette un'area massima di 10x10m.

- Ad ogni modo ciascuna Base Station raggiunge una distanza massima di 5 metri, perciò anche nei setup a 4 basi bisogna ricordare che nelle zone perimetrali dell'area di tracciamento, esso risulta di qualità inferiore se vengono sfruttati completamente i 10x10m.

- La soluzione migliore non è sempre quella di usufruire di tutto lo spazio a disposizione bensì di sfruttarlo in maniera efficiente. La figura 4.2 mostra come posizionare al meglio le proprie Base Station in base allo spazio disponibile.

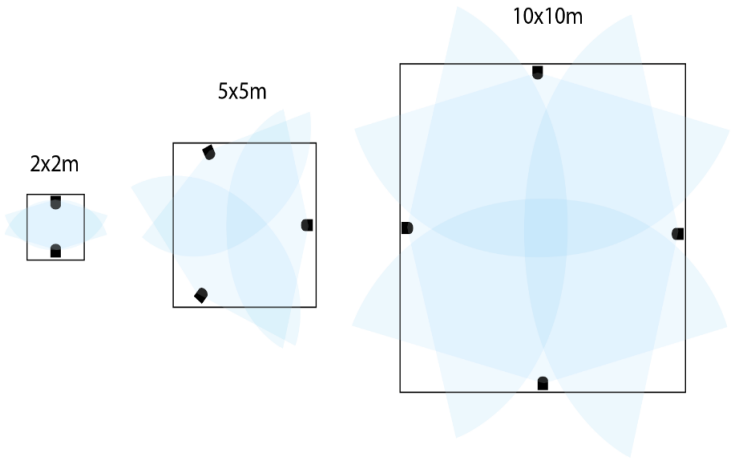

Figura 4.2. Posizionamento stazioni base consigliato da HTC Vive in funzione dello spazio a disposizione

- Ogni stazione base per funzionare in modo ottimale dev'essere posta ad almeno 2 metri di altezza con un'inclinazione compresa nell'intervallo 30°-45°. (Figura 4.3).

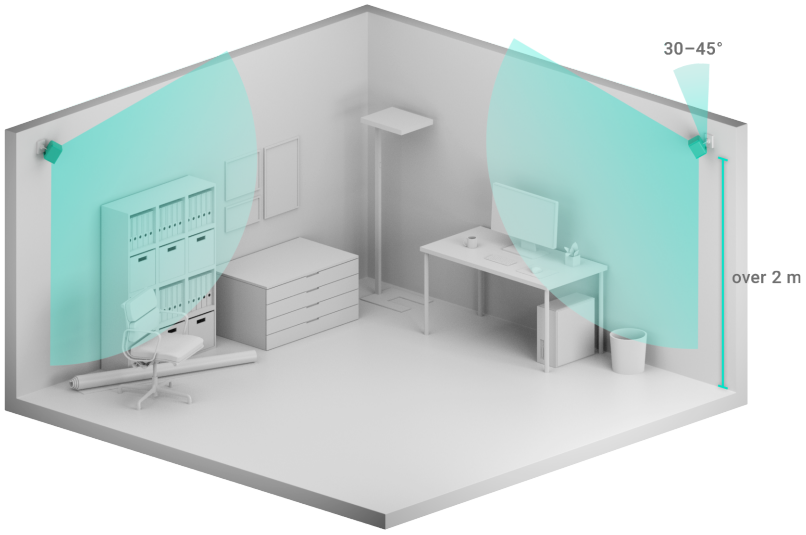

Figura 4.3. Set-up consigliato da HTC Vive per il posizionamento della singola stazione base

- Ogni stazione dev'essere settata su un canale diverso dalle altre affinché l'applicativo per computer possa riconoscerle ed il sistema possa funzionare correttamente. La figura 4.4 mostra come visualizzare questo dato sull'app SteamVR. La modifica del canale va fatta manualmente, premendo con uno spillo a punta piatta *

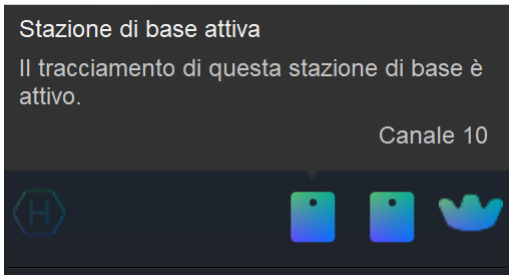

Figura 4.4. Lettura su SteamVR dei canali su cui sono impostate le stazioni base

Alle regole fondamentali si aggiungono quelle di buona prassi:

*Come ad esempio quelli appositi per le SIM dei telefoni.

- Le Base Station vanno costantemente alimentate a corrente. Assicurarsi perciò che siano presenti delle prese elettriche vicino a dove andranno montate le stazioni base, oppure che sia possibile portare una prolunga vicino ad ogni stazione.

- Non inserire all'interno della zona di tracciamento alcun oggetto che possa intralciare od occludere la visione delle Base Stations. Evitare inoltre materiali od oggetti altamente riflettenti in quanto un tracciamento di tipo ottico che fa uso di laser è altamente sensibile a questo tipo di materiali.

- Impostare la Base Station che si vuole adibire ad origine del sistema di riferimento comune sul numero di canale minore. Il sistema la sceglierà quindi come origine ed impostare l'asse immaginario che la unisce alla Base Station successiva in ordine crescente di numero di canale come asse X del sistema di riferimento. Può convenire quindi scegliere con cura il posizionamento di queste due stazioni al fine di ottenere l'allineamento del sistema di riferimento desiderato.

4.1.2 Tracker e Dongle USB

HTC Vive Tracker 3.0 è il dispositivo per il tracciamento di oggetti fisici di ultima generazione di HTC. Il suo design è pensato appositamente per ottimizzare la ricezione dei segnali luminosi lanciati dalle Base Stations ed è inoltre di facile uso. Alla sua base ha un filetto per viti che permette di fissarlo al rig [3] delle camere da presa oppure alla testa degli stativi. Sull'altra faccia, al centro del caratteristico tri-corno, si trova invece il bottone di accensione che abilita la connessione col calcolatore (figura 4.5).

Come accennato, la trasmissione dei dati avviene per mezzo del dispositivo tracciato e non delle stazioni base. Per tale motivo, mentre le Base Stations funzionano in modo indipendente e non necessitano di nessun tipo di connessione con il computer su cui gira SteamVR, il tracker dev'essere invece collegato ad esso. La connessione viene stabilita

[3] È la cosiddetta armatura della cinepresa. Comprende lo scheletro e tutte le apparecchiature su di esso montate che si assemblano intorno ad una camera da presa per poter essere utilizzata efficientemente sul set.

Figura 4.5. HTC Vive Tracker 3.0 montato su Blackmagic Pocket Cinema Camera 4K in posizione ruotata di 90° rispetto all'asse X.

tramite un protocollo Bluetooth tra un ricevitore collegato al computer via cavo (il cosiddetto dongle USB mostrato in figura 4.6) e il tracker che si comporta come sorgente. È il tracker stesso a fornire tutte le indicazioni utili al software, comprese quelle riguardanti le stazioni base come il canale sul quale sono impostate.

È infine importante conoscere come il tracker si percepisce ed individua all'interno dello spazio. HTC [b] ha messo a disposizione delle linee guida per gli sviluppatori dove analizza il tracker e ne esplica il rispettivo sistema di riferimento. Tale definizione si basa sulla normativa ISO5459:2011 [2011] e riporta 3 *Datum Features* [4] (A, B e C). Il sistema di riferimento è costruito rispetto all'anello filettato per la vite (Datum A), alla sua intersezione con la linea mediana dello Standard Camera Mount (Datum B) ed

[4]Il Datum Reference Frame è un sistema di tre piani mutualmente ortogonali che permette di posizionare ed orientare correttamente un componente nello spazio che viene definito da 3 Datum Feature opportunamente scelte.

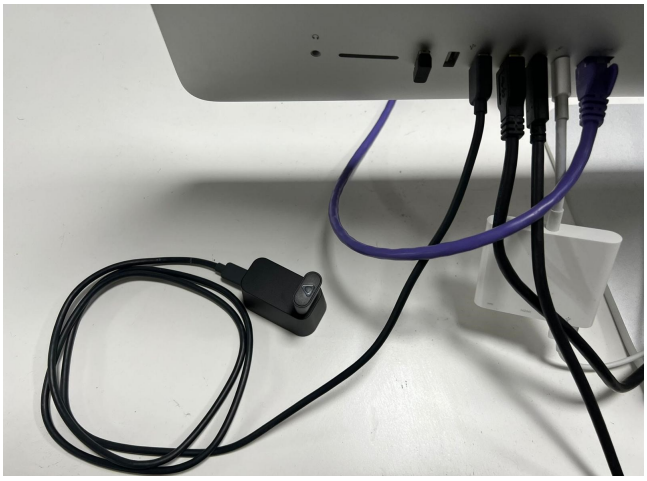

Figura 4.6. Dongle USB HTC Vive Tracker 3.0 connesso al calcolatore su cui è installato l'applicativo SteamVR.

all'intersezione del Datum A e della linea mediana del Pin Stabilizzatore (Datum C). Il risultato mostrato in figura 4.7 è che:

- L'origine degli assi del sistema di riferimento è il punto di intersezione tra l'asse del filetto per la vite di montaggio ed il piano su cui giace la base del dispositivo.

- L'asse Y è lo stesso asse del filetto per lo Standard Camera Mount.

- L'asse Z è la retta ortogonale all'asse Y e passante per il centro della circonferenza che definisce il Pin Stabilizzatore.

- L'asse X è la retta ortogonale al piano YZ passante per l'origine.

Quest'analisi tornerà utile successivamente al fondo di questo capitolo e nel capitolo 5 quando bisognerà calcolare il vettore differenza tra la posizione del centro ottico dell'obiettivo e quella del tracker. Per ora ci limitiamo ad osservare che come mostrato in figura 4.5 il tracker è stato montato con una rotazione di 90° intorno all'asse X rispetto al sistema di coordinate di Unreal Engine che prevede che l'asse Z punti verso l'alto.

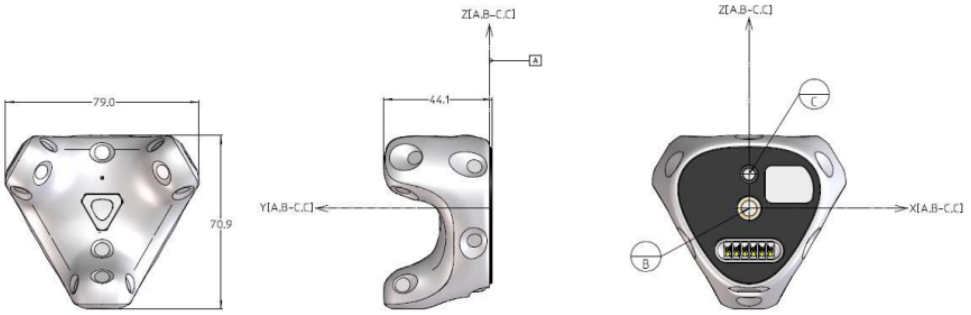

Figura 4.7. Sistema di coordinate del Tracker 3.0 basato sulla normativa ISO5459 che tiene conto del Datum Reference Frame

4.1.3 Uso di SteamVR senza HMD

Se è vero che precedentemente è stato detto che SteamVR è un'applicazione di facile utilizzo che non necessita di alcuna configurazione è altresì vero che non è pensata per essere utilizzata senza i propri caschetti di realtà virtuale * (a.k.a. HDM). Questo comporta la necessità di apportare alcune fondamentali modifiche ai file .vrsettings dell'applicazione. Cambiando poche righe di codice sarà possibile circuire il compilatore che rileverà la presenza di un HDM a priori senza il bisogno che esso sia davvero presente. Innanzitutto bisogna modificare le impostazioni default dei drivers. Il file *default.vrsettings* che ci interessa si trova all'indirizzo:

$$.../SteamVR/drivers/null/resources/settings/$$

Bisogna quindi abilitare la *null reference* per i drivers cambiando il valore booleano del campo **enable** dell'attributo *driver_null* da *false* a *true*. La figura 4.8 mostra il file interessato per i drivers.

*SteamVR nasce infatti come applicazione a supporto dei videogiochi che dispongono delle versione VR in cui l'HDM è il dispositivo che crea la condizione *sine qua non*.

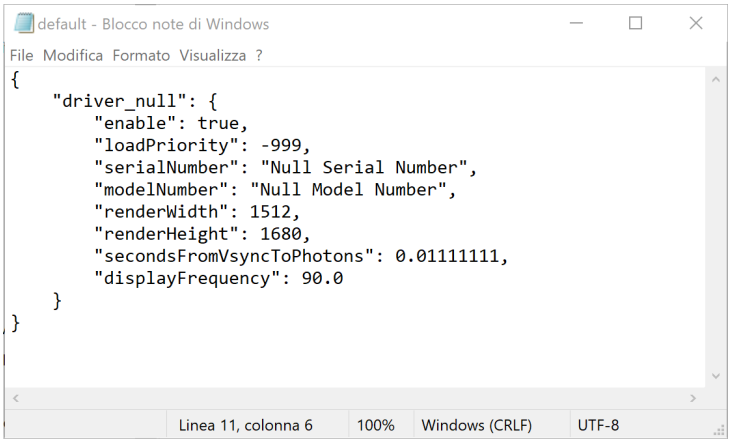

Figura 4.8. File .vrsettings con modifiche da apportare per i driver

Successivamente bisogna modificare il file *default.vrsettings* per le impostazioni dell'applicazione che si trova all'indirizzo:

$$.../SteamVR/resources/settings/$$

Vanno modificati i seguenti campi dell'attributo *steamvr*:

- **RequiredHDM** deve avere come valore *false*
- **forcedDriver** deve avere come valore *null*
- **activateMultipleDrivers** deve avere come valore *true*

Di seguito la figura 4.9 mostra il file interessato ed i campi da modificare.

È buona prassi fare delle copie dei file modificati rinominati "copia" per tenere traccia delle modifiche e non perdere la configurazione originale. Per ultimo segnaliamo che ad ogni aggiornamento di SteamVR i file vengono reinstallati e perciò la procedura sopra descritta va ripetuta.

4.2 LiveLink Tool

Una volta che l'hardware è stato correttamente impostato e che i dati sono opportunamente ricevuti dall'applicazione SteamVR, bisogna procedere alla trasmissione dei dati

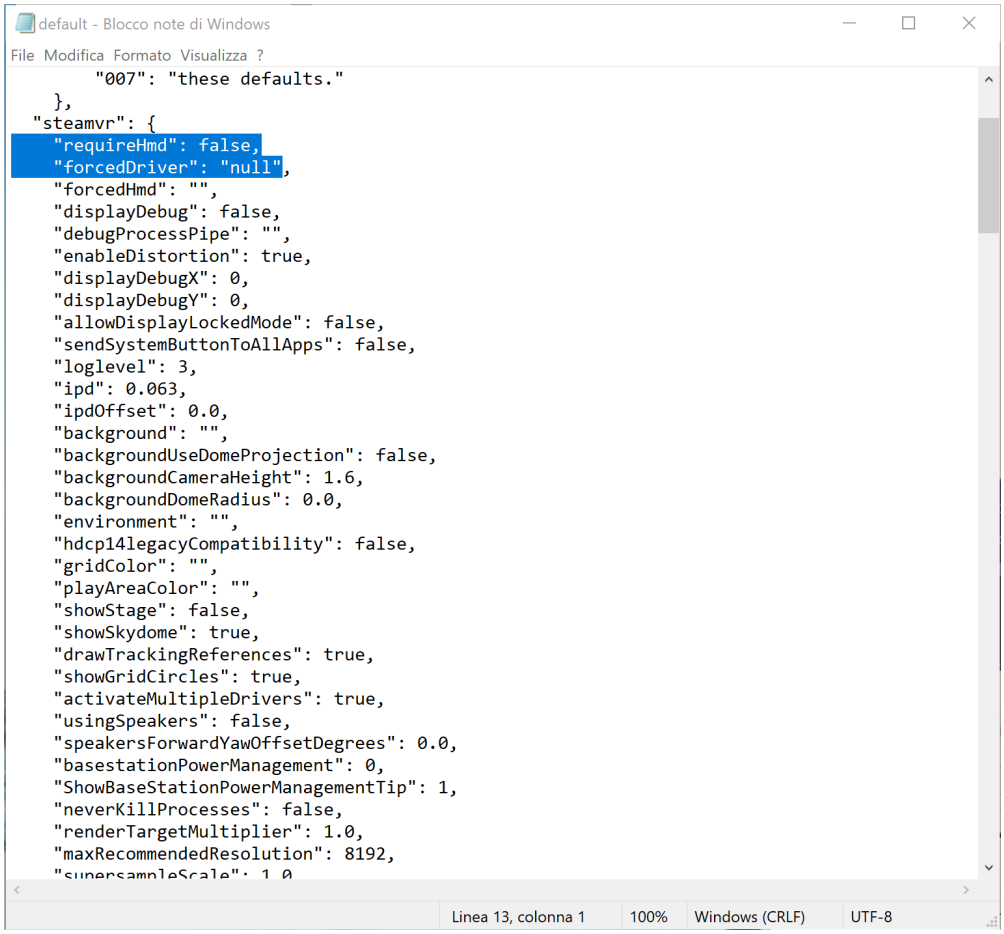

Figura 4.9. File .vrsettings con modifiche da apportare per disabilitare richiesta HDM.

nel game engine. Infatti questo è l'ultimo passaggio che permetterà concretamente di creare un collegamento digitale tra la cinepresa virtuale e quella fisica in modo che esse si muovano all'unisono nello spazio rispetto a punti di riferimento comuni. Sarà proprio quest'ultimo passaggio a realizzare infatti il cosiddetto *tracking real-time* della camera da presa.

L'obiettivo del tool Live Link di Unreal Engine è quello di fornire un'interfaccia comune per la trasmissione e ricezione di dati di animazione provenienti da sorgenti esterne (per esempio server di Mocap o DDC tools) all'interno di Unreal Engine. Questo tool è estendibile attraverso alcuni plugin mostrati in figura 4.10, che permettono a terze parti lo sviluppo di nuove features. I sistemi di motion capture possono inoltre utilizzare Live Link

per trasmettere dati dentro il game engine che possono essere visualizzati come anteprima in tempo reale. Per quanto concerne lo sviluppo del framework di questo progetto, sono stati abilitati i seguenti plugin (oltre al **Live Link** attivo di default):

- **LiveLinkXR** che permette di utilizzare device di tracciamento per la Extended Reality di SteamVR (tra cui il Vive Tracker 3.0).
- **LiveLinkCamera** che aggiunge funzionalità per la gestione delle camere tra cui la possibilità di usufruire dell'interfaccia *Camera Role*.
- **LiveLinkLens** che aggiunge il nuovo *Lens Role* e *Lens Controller* per permettere la trasmissione si dati provenienti da lenti pre-calibrate.

Affronteremo ora l'analisi dei plugin utilizzati che hanno permesso di interpretare ed allocare i dati ricevuti da SteamVR in modo corretto ed efficiente.

4.2.1 LiveLinkXR Plugin

L'aggiunta di una nuova sorgente di dati di animazione va fatta nella finestra delle connessioni del tool LiveLink, che si può aprire dal menù *windows*, cliccando in seguito su *virtual production tools* e poi su *Live Link*. (L'interfaccia della finestra è visualizzabile in figura 4.12). Per aggiungere lo SteamVR Tracker all'elenco delle sorgenti disponibili bisogna aver attivato il plugin *LiveLinkXR* che abilita le API [5] necessarie a riconoscerlo. Si deve quindi cliccare sul bottone in alto a sinistra *Source* e scegliere di aggiungere una *XR Subject*. In questo modo il tracker 3.0 sarà riconosciuto ed importato nell'elenco dei soggetti tracciati sotto la voce *XR* e con un ruolo *Transform* (che significa che è dedicato all'animazione delle trasformazioni elementari). Chiaramente affinché il tutto vada a

[5] Con *application programming interface* (API) si indica un insieme di procedure (in genere raggruppate per strumenti specifici) atte a risolvere uno specifico problema di comunicazione tra diversi computer o tra diversi software o tra diversi componenti di software; spesso tale termine designa le librerie software di un linguaggio di programmazione, sebbene più propriamente le API sono il metodo con cui le librerie vengono usate per sopperire ad uno specifico problema di scambio di informazioni. Wikipedia [b]

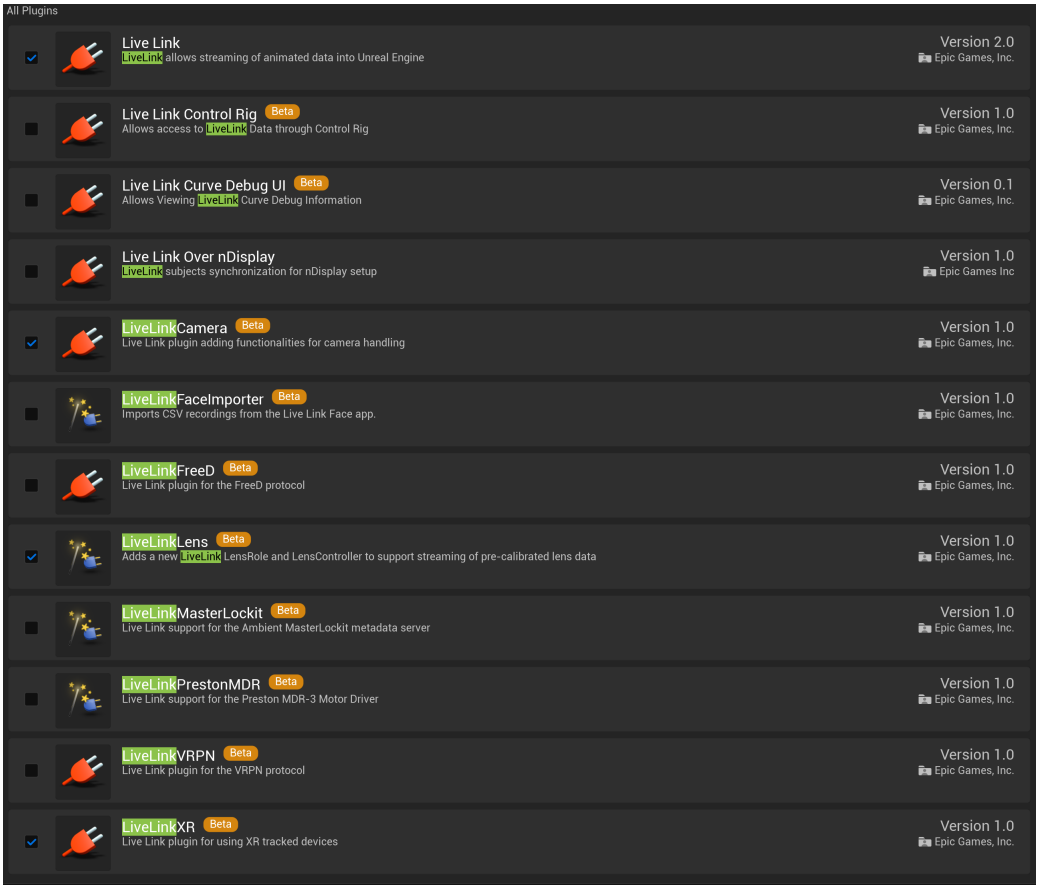

Figura 4.10. Plugins disponibili o attivati per la strumentazione utilizzabile con LiveLink.

buon fine, l'applicazione SteamVR dev'essere attiva e deve mostrare di ricevere almeno un tracker e due stazioni base.

Attualmente i dati sono quindi entrati all'interno del game engine e correttamente interpretati, ma non ancora utilizzati. Per farlo dobbiamo creare una camera virtuale che possa essere comandata dai dati provenienti dal tracker. Dal menù aggiungiamo quindi un nuovo *actor*, nello specifico utilizzando la classe *CineCameraActor*, e lo poniamo figlio di un *Empty Actor* (che nel nostro caso abbiamo denominato *Camera Actor*). Quest'ultimo ha infatti la funzione di creare un punto di riferimento per le coordinate che verranno lette dal sistema SteamVR. La posizione della camera virtuale verrà calibrata rispetto ad esso, che avrà quindi la stessa posizione relativa rispetto alla camera virtuale, che ha

l'origine del sistema di riferimento della camera fisica tracciata. In sintesi, tale *actor* è il corrispettivo virtuale della Base Station eletta ad origine del sistema di coordinate sul set (quella con numero di canale minore). Risulta quindi molto importante scegliere con cura la posizione dell'oggetto padre della camera virtuale in quanto detterà la posizione del nostro set all'interno del mondo virtuale. Infatti per spostare la camera all'interno del mondo al fine di scegliere la posizione iniziale che deve assumere, non potrà essere spostata essa stessa bensì il l'actor padre che funge da origine del suo sistema di riferimento. Per essere controllato, al *CineCameraActor* va aggiunto un *component* specifico che legga le informazioni ricevuto per mezzo del tool Live Link. Esso è il **LiveLinkComponentController** che va opportunamente configurato scegliendo lo SteamVR Tracker tra le *Subject Representation* disponibili nel menù a tendina sotto la voce Live Link del pannello details (figura 4.11). Da questo momento in poi andando nella sezione *Transform* del pannello detail del camera actor vedremo che i parametri di posizione e rotazione non sono più modificabili manualmente, ma sono comandati esclusivamente dal sitema SteamVR.

Il nostro sistema di tracciamento ora funziona concretamente ma ancora diverse configurazioni vanno aggiustate affinché il comportamento della camera virtuale sia completamente assimilabile a quello della camera fisica. La posizione del tracker sulla camera non corrisponde infatti al suo centro ottico, ed una connessione bluetooth comporta una latenza sul sistema non trascurabile.

Impostazione Offset del Tracker

Nel capitolo 5 affronteremo maggiormente nel dettaglio il problema della determinazione del centro ottico della lente. Per ora basta sapere che nel mondo virtuale il camera actor è posizionato in un punto tridimensionale dal quale renderizza un'immagine bidimensionale dell'ambiente digitale rispetto ai parametri sui quali è impostato. Quel punto corrisponde esattamente al centro ottico dell'obiettivo della camera virtuale ed è controllato dalla posizione fisica del tracker. Questo significa che idealmente il nostro tracker dovrebbe trovarsi al centro della lente fisica. Dal momento che è chiaramente un paradosso, nasce la necessità di determinare il vettore di roto-traslazione che sposta il nostro tracker dalla sua posizione ideale-teorica. Per farlo ci serve ricordare innanzitutto due cose:

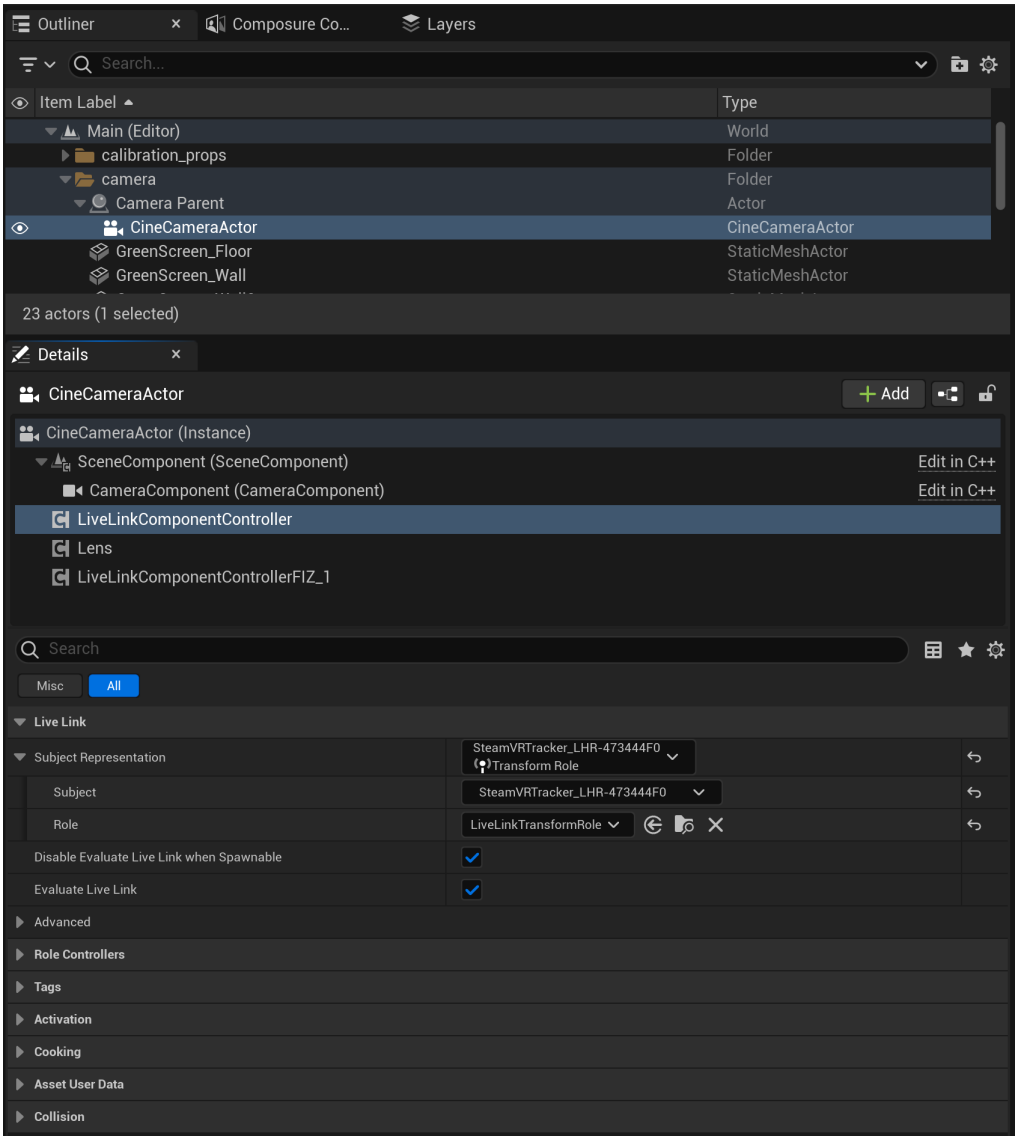

Figura 4.11. Finestra *details* con attribuiti visibili del componente *LiveLinkComponentController*.

1. La posizione naturale del tracker (asse Z rivolto verso l'alto) è con il piano della sua base perpendicolare al pavimento

2. L'origine del sistemo di riferimento del tracker si trova sulla sua base al centro dell'anello che descrive la circonferenza del filetto per lo *Standard Camera Mount*

Come mostra la figura 4.12, aprendo la finestra di connessione del tool *Live Link* è possibile

selezionare il tracker 3.0 e aggiungere un nuovo elemento presso la voce *Pre Processors*. La tipologia da scegliere è *Transform Axis Switch* e dopodiché bisogna mettere un *flag* sui campi **Offset Position e Offset Orientation**. Per quanto concerne la rotazione, come già osservato, rispetto alla lente il tracker ha solo una rotazione rispetto all'asse X di 90°. Per quanto riguarda invece il vettore traslazione, esso va misurato manualmente calcolando le proiezioni lungo i tre assi X, Y e Z della distanza tra la posizione del centro ottico della lente ed il centro dello Standard Camera Mount del tracker (già montato definitivamente sul rig della cinepresa). Nell'esempio mostrato in figura 4.12 ad esempio, il tracker era sopraelevato ed arretrato rispetto al centro ottico della lente, ma perfettamente allineato ad esso lungo l'asse Y (infatti l'offset su Y risulta uguale a 0).

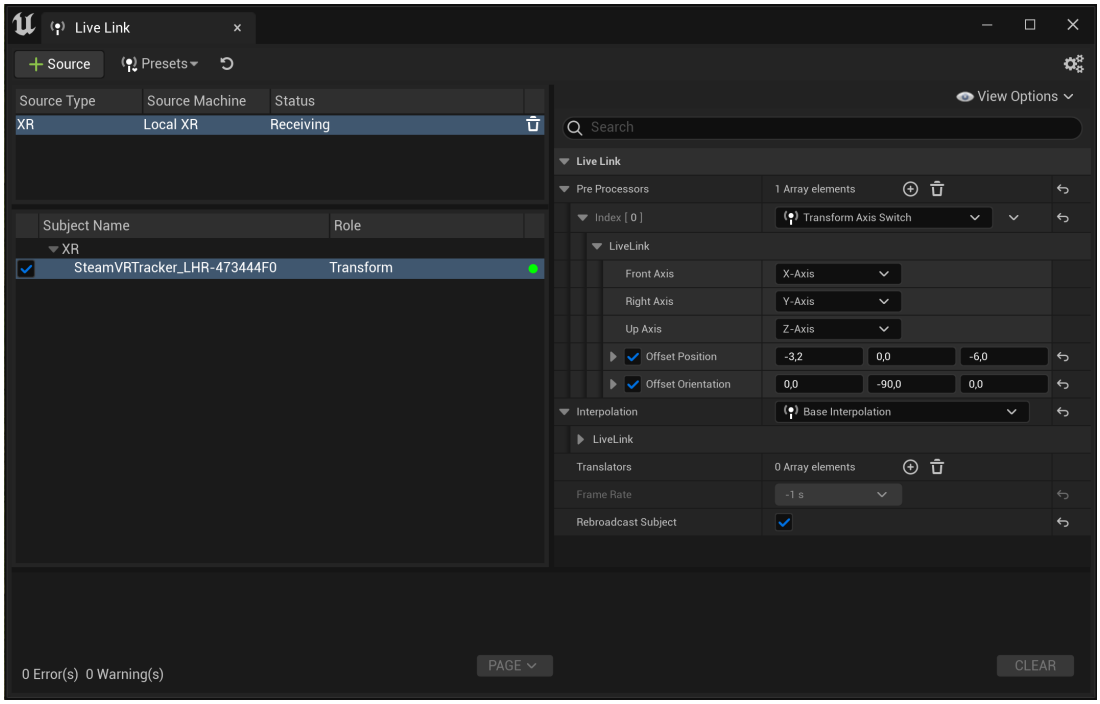

Figura 4.12. Finestra operativa del tool *LiveLink* con riferimento ad un preset per l'offset del tracker 3.0 montato su Blackmagic Pocket Cinema Camera 4K con lunghezza focale reale dell'obiettivo 50mm.

Impostazione Delay

Un altro aspetto rilevante da considerare è la latenza del sistema di tracciamento. Infatti a differenza del sistema HTC Vive Mars la cui connessione è cablata, il sistema standard come già visto prevede una connessione con protocollo Bluetooth. Dal momento che la latenza massima del sistema è circa 22 ms, nella voce offset bisognerà testare diversi valori fino a trovare quello ottimale in un intorno di quel valore. (Figura 4.13). Questo valore andrà a compensare il ritardo nella trasmissione dei dati del tracker.

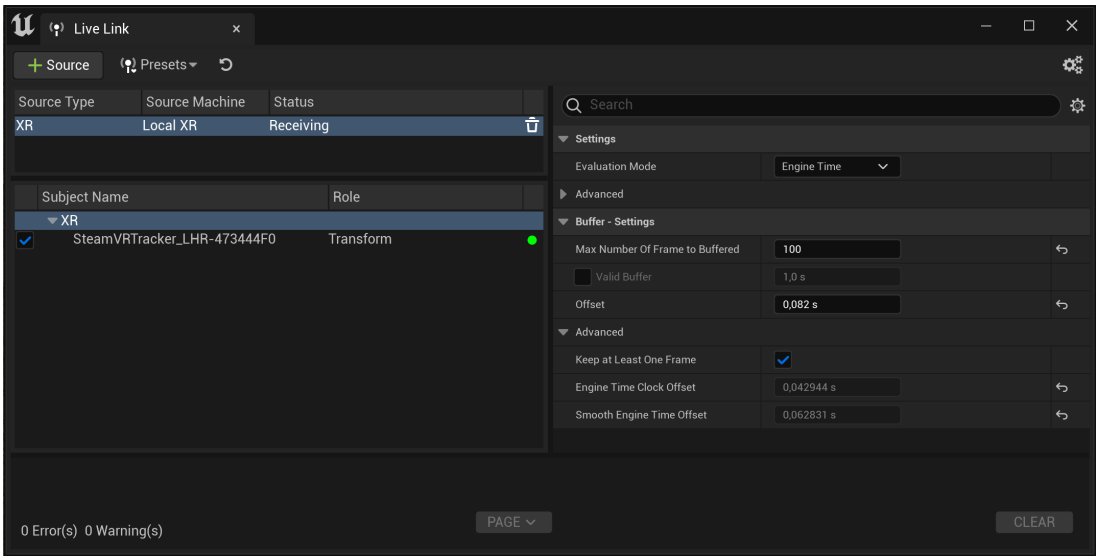

Figura 4.13. Finestra operativa del tool *LiveLink* con riferimento alle impostazioni per la riduzione del delay dei dati di tracciamento.

4.2.2 Virtual Camera FIZ

Come abbiamo visto la suite Live Link oltre a ricevere i dati dentro Unreal Engine si occupa anche del controllo automatico di alcune features. Per calibrare la lente della nostra camera nel capitolo 5 sarà indispensabile avere un soggetto virtuale che acquisisca il controllo di alcuni attributi del nostro *CineCameraActor*. Quindi è stato creato un *LiveLinkComponentController* del tutto simile a quello di prima ma rinominato **LiveLink ComponentController_FIZ**. È proprio qui che andrà inserita una reference a Live Link ed al soggetto virtuale che tra poco mostreremo come istanziare (figura 4.14). Nello

specifico il virtual subject dovrà essere in grado di modificare autonomamente i parametri FIZ della lente della camera virtuale, ovvero:

- **Focus**, la distanza del piano di messa a fuoco.
- **Iris**, l'apertura del diaframma.
- **Zoom**, la lunghezza focale della lente.

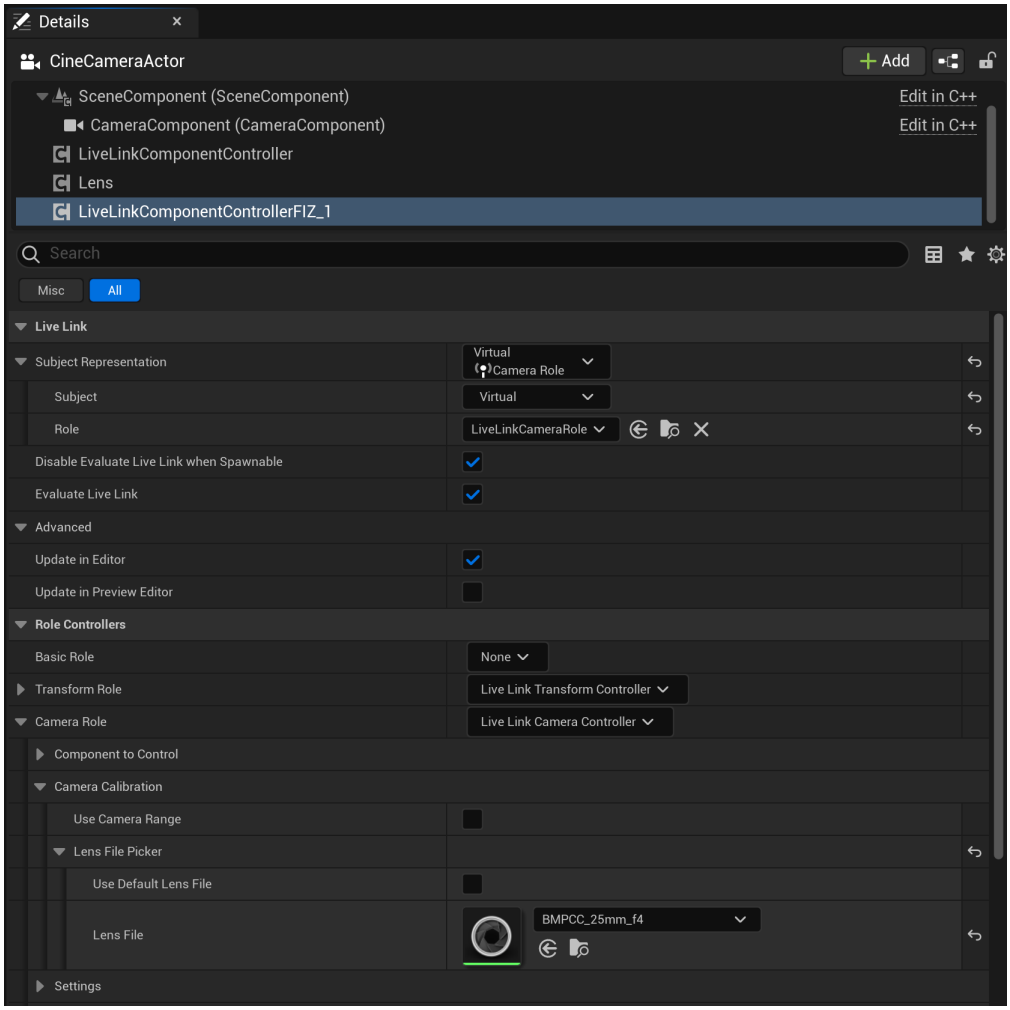

Figura 4.14. Finestra *details* con attribuiti visibili del componente *LiveLinkComponentController_FIZ*.

Senza addentrarci ora nel significato di calibrazione della lente virtuale, ci interessa capire come il tool Live Link ci possa aiutare nell'istanziare un oggetto che sia capace

di quanto sopra descritto. Quello che ci serve è un'interfaccia, perciò creeremo un nuovo blueprint. Una volta finalizzato potremo creare un nuovo *virtual subject*. Cliccando sul bottone *Source* si apre una finestra da dove scegliere l'opzione *add virtual source* e successivamente sceglieremo di utilizzare come script dal menù a tendina il blueprint appositamente preparato. In questo modo verrà visualizzata una nuova sorgente virtuale, con ruolo *Camera Role*, che evidenzierà come parametri controllati quelli la terna FIZ, con i valori assegnati di default (figura 4.15. Da questa finestra è inoltre possibile modificare a proprio piacere questi parametri, cosa che non è invece possibile dal pannello details del CameraActor poiché è ora controllato dal blueprint. Analizziamo infine nello specifico l'implementazione del blueprint sopra citato.

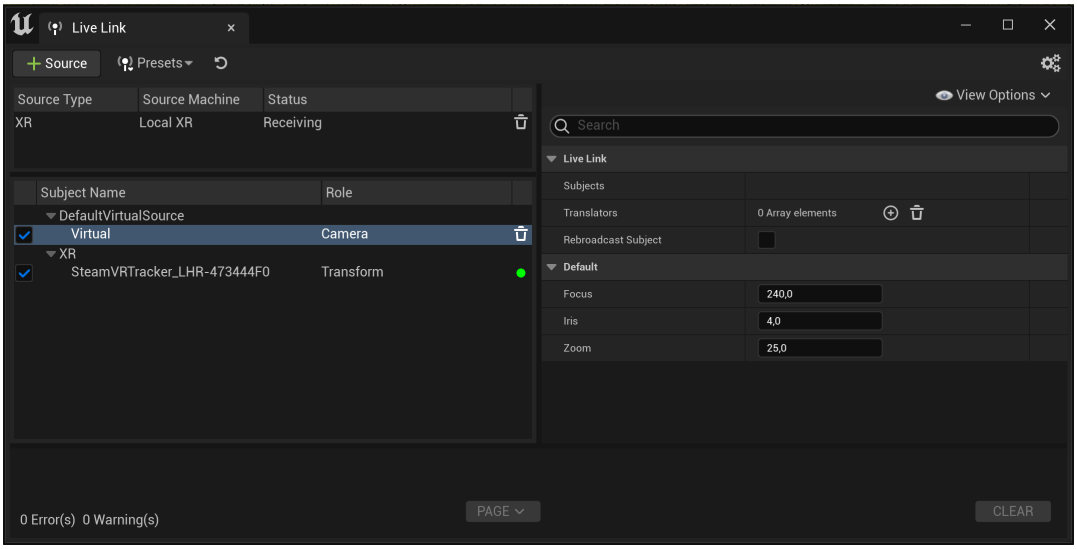

Figura 4.15. Finestra di connessione del tool *Live Link*. Si possono vedere i parametri *FIZ* quando viene selezionata la sorgente virtuale *Virtual*.

Blueprint

L'obiettivo è creare un blueprint capace di controllare i parametri della nostra camera virtuale. Come mostra la figura 4.16, la classe che torna utile al nostro scopo è **LiveLinkBlueprintVirtualSubject**, che andremo a selezionare per creare un nuovo script che ne

erediti i metodi principali. Si aprirà ora una nuova finestra dedicata alla programmazione a nodi di cui abbiamo parlato nel capitolo 3.

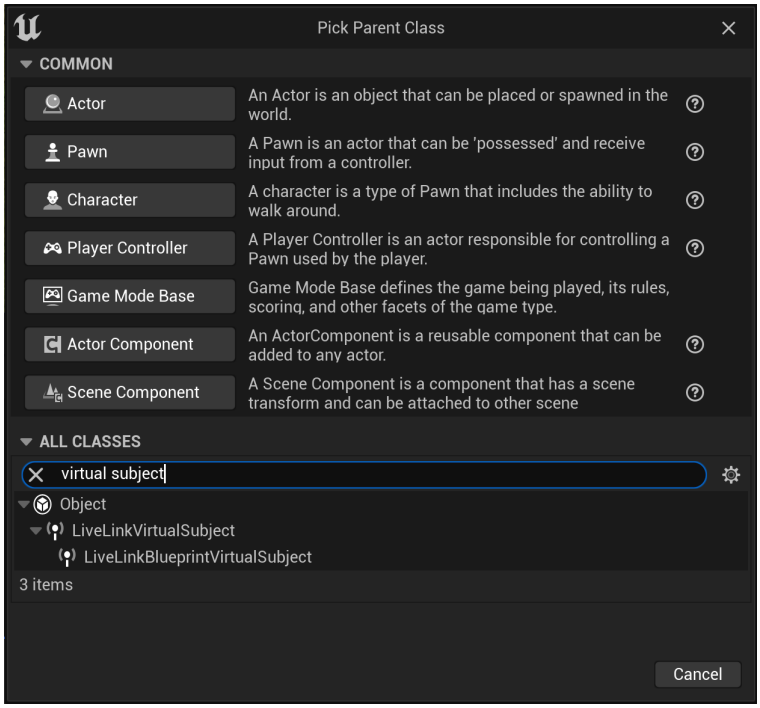

Figura 4.16. Finestra di Unreal Engine per creare un nuovo *Blueprint*. Nello specifico si vuole generare un *LiveLinkBlueprintVirtualSubject*.

La figura 4.17 mostra la visualizzazione del blueprint finalizzato. La finalità è quella di inizializzare i parametri che si vogliono controllare e successivamente aggiornarli in fase di *update*. Analizziamo quindi l'implementazione dello script facendo riferimento alla figura citata. Per semplicità di esposizione riportiamo la procedura da eseguire in modalità cronologica:

1. Cancellare gli eventi presenti di default nello script.
2. Aggiungere due nuovi eventi, *Event On Initialize* e *Event on Update*, tramite il bottone *add* in alto a sinistra.
3. Creare tre nuove variabili sotto la voce *Variables* nominate rispettivamente Focus, Iris e Zoom e settarle sul tipo *float*. Tramite il menù di destra impostare inoltre il default di ognuno sui parametri conosciuti della nostra lente fisica.

4. Collegare il trigger dell'evento *Initialize* alla funzione ereditata **Update Virtual Subject Static Data**. Assicurarsi quindi che le uniche voci da inizializzare, perciò con un *flag*, siano quelle di *Focal Length*, *Aperture* e *Focus Distance*.

5. Collegare l'evento *Update* al metodo funzionale **Update Virtual Subject Frame Data**. Creare successivamente un'istanza di ognuna delle tre variabili tramite il metodo *get* e collegare il loro valore al rispettivo *Frame Data*.

Il blueprint è così completato.

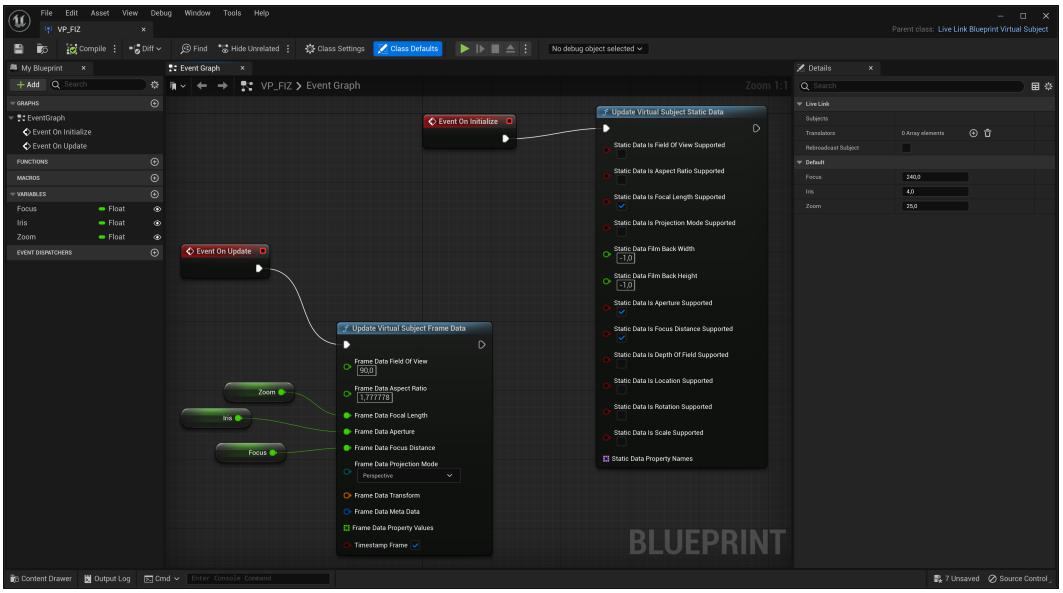

Figura 4.17. Dimostrazione di un *Blueprint* che controlli i parametri (compresi nell'acronimo *FIZ*) della lente della camera virtuale.

Capitolo 5

Calibrazione Camera

5.1 Media Input

Dopo aver analizzato come impostare le controparti software e hardware affinché collaborino al meglio nel tracciamento della nostra camera, ora ci occupiamo della calibrazione di quest'ultima e di come il game engine potrà ricevere il segnale video dalla camera fisica. Partiamo quindi con la dissertazione riguardo alla trasmissione delle immagini riprese dalla nostra camera. A livello hardware la connessione al calcolatore è stata implementata attraverso un cavo HDMI e una scheda di acquisizione *Elgato CamLink 4K* (figura 5.1) che possiede un output USB e un input HDMI. Dal momento che la cinepresa usata durante il progetto di tesi è una *Blackmagic Pocket Cinema Camera 4K* sarebbe stato opportuno utilizzare una scheda di acquisizione DeckLink di proprietà della medesima casa ed abilitare in software il framework dedicato al Blackmagic Media Input. Tuttavia essa non era disponibile, e mostreremo perciò la procedura standard seguita usando una scheda di acquisizione generica che non si differenzia particolarmente dalla procedura necessaria se si fosse utilizzato il framework dedicato.

Figura 5.1. Scheda di acquisizione video *Elgato CamLink 4K* connessa al calcolatore via USB ed alla cinepresa con cavo HDMI.

5.1.1 Media Bundle

All'interno del content drawer bisogna innanzitutto istanziare un oggetto che si ponga in modalità di ricezione rispetto ai dati video e che possa portarli correttamente dentro al software. Per fare ciò scegliamo dal menù a tendina che si apre cliccando sul tasto dentro del mouse la voce *media* e successivamente *media bundle*. Verrà così creato un actor *Media Bundle* ed in aggiunta una cartella contenente gli asset a cui fa riferimento per funzionare correttamente. Come mostra la figura 5.2, la cartella è nominata *MediaBundle_InnerAssets* e al suo interno contiene gli assets:

- **Media Player** per impostare il dispositivo da cui ricevere i dati audio e video.

- **Media Texture** utilizza i dati provenienti dal *media player* per generare una texture video. È l'asset a cui istanziare le referenze negli script collegati agli oggetti che vogliamo utilizzino le immagini catturate dalla nostra cinepresa.

- **Render Target** per la correzione delle aberrazioni ottiche [1] della lente.

- **Material Instance** per la renderizzazione del video su un oggetto virtuale.

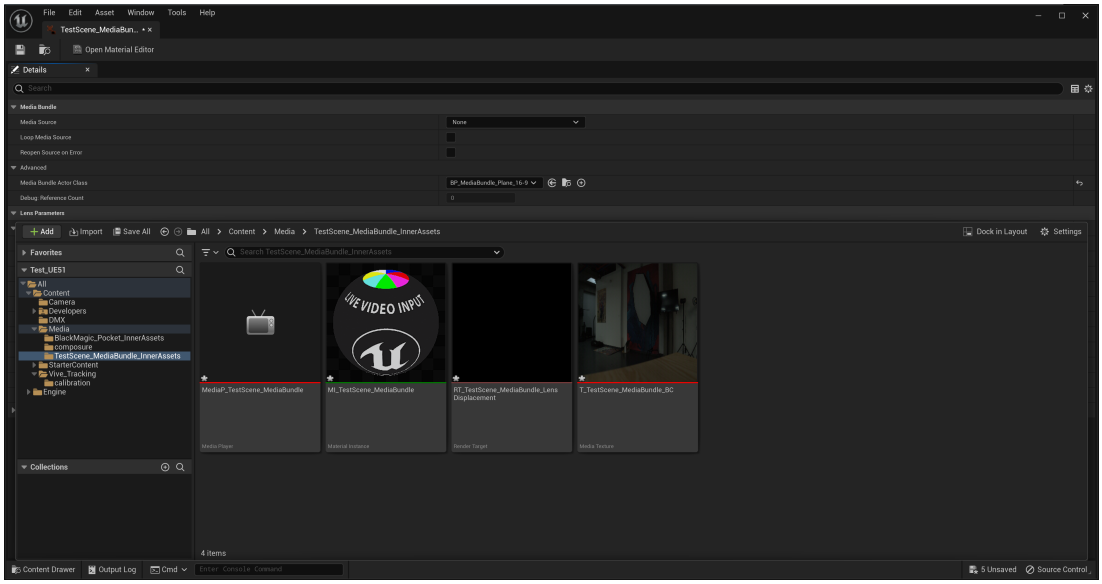

Figura 5.2. Asset presenti nella cartella *MediaBundle_InnerAssets*.

Per quanto concerne l'utilizzo che interessa a noi, sono solo i primi due assets a interessarci. Cliccando sul *Media Player* si apre una nuova finestra. In alto a sinistra si trova l'immagine di una cartella; cliccandovi sopra si apre un menù a tendina per scegliere tra i dispositivi di acquisizione audio e video disponibili quello che desideriamo utilizzare per questo *media player*. Come mostra la figura 5.3, una volta scelta la scheda di acquisizione che stiamo utilizzando, apparirà sul display dell'asset il video registrato dalla cinepresa in tempo reale. Riguardo invece alla *Media Texture*, il secondo asset non necessita di alcun intervento in quanto automaticamente rileva la presenza del target impostato e usa il video acquisito per creare una texture video.

[1] L'aberrazione di un sistema ottico è la differenza tra l'immagine effettiva, reale o virtuale, formata dal sistema e l'immagine che si voleva ottenere, immagine che di solito è bidimensionale e consiste in una proiezione geometrica della scena reale sul piano focale del sistema secondo i principi dell'ottica geometrica ideale.

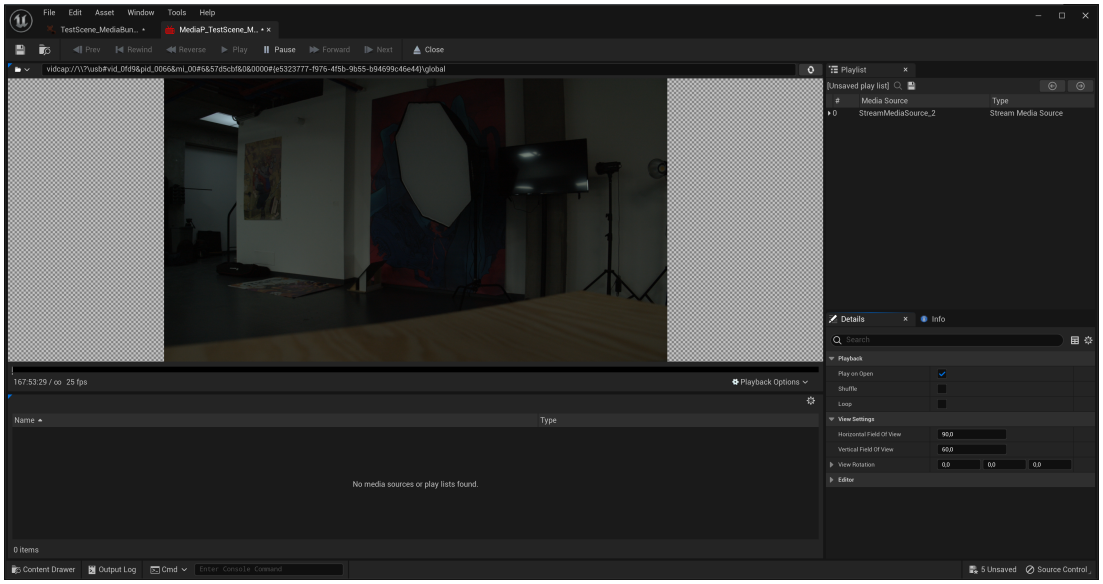

Figura 5.3. Finestra relativa all'asset *Media Player* dalla quale si può scegliere la scheda di acquisizione video.

5.2 Calibrazione della Lente Virtuale

Esaurita la trattazione della procedura di acquisizione dell'immagine video in software è la volta della calibrazione della lente virtuale. Questo delicato procedimento porterà ad ottenere un comportamento dell'immagine virtuale identico a quello dell'immagine reale. Tale corrispondenza risulterà in una perfetta sincronizzazione ottica e spaziale dell'immagine di foreground e di background creando l'illusione di un'immagine unica. Inizialmente trattiamo la questione già anticipata precedentemente di trovare il centro ottico della lente ed in seguito vedremo invece come utilizzare questo dato correttamente e come calibrare la lente virtuale.

5.2.1 Centro Ottico dell'obiettivo

Una lente è un elemento ottico che ha la proprietà di concentrare o di far divergere i raggi di luce. Il centro ottico di una lente è invece il punto posto sull'asse principale (l'asse verticale) che ha la proprietà di non deviare i raggi luminosi che passano per esso. Questo punto è cruciale in quanto definisce l'immagine reale o virtuale creata dalla lente

stessa (figura 5.4). Il centro ottico di un obiettivo va calcolato con tecniche manuali poiché generalmente i dati a disposizione dell'utente non bastano per misurarlo matematicamente. Il metodo più usato è quello che consiste nel trovare il *punto di non parallasse* [2] . Questo metodo si basa sul fatto che la condizione citata si verifica se e solo se l'asse verticale di rotazione della cinepresa interseca il centro ottico della cinepresa stessa.

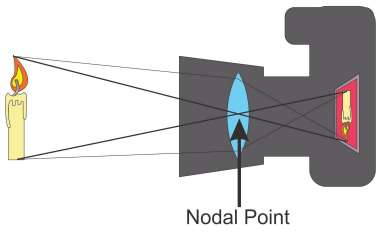

Figura 5.4. Esemplificazione del centro ottico di una lente fotografica

Come mostrato in figura 5.5, nella pratica le situazioni a cui si può andare incontro sono sostanzialmente tre:

1. **L'asse di rotazione precede il centro ottico**: ruotando la camera in senso antiorario gli oggetti più vicini trasleranno verso destra rispetto a quelli più lontani mentre si schiacceranno verso la posizione di quelli più lontani ruotando in senso orario.

2. **L'asse di rotazione prosegue il centro ottico**: ruotando la camera in senso orario gli oggetti più vicini trasleranno verso destra rispetto a quelli più lontani mentre si schiacceranno verso la posizione di quelli più lontani ruotando in senso antiorario.

3. **L'asse di rotazione è sul centro ottico**: ruotando la camera sia in senso orario che antiorario gli oggetti non traslano la loro posizione reciproca.

[2] La parallasse è il fenomeno ottico per cui un oggetto sembra spostarsi rispetto allo sfondo se si cambia il punto di osservazione.

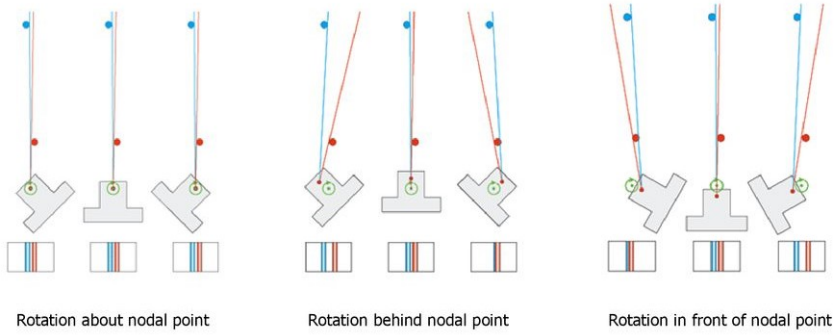

Figura 5.5. Diversi tipi di rotazione di una camera rispettivamente con l'asse di rotazione rispettivamente allineato al centro ottico, antecedente rispetto ad esso ed infine successivo ad esso.

Per effettuare questo test e trovare il centro ottico della lente si possono usare due stativi ed uno slider da montare sopra il treppiede su cui poniamo la camera. Gli stativi andranno posizionati molto lontani tra loro (in particolar modo se la lunghezza focale usata è corta) e dovranno essere allineati all'asse ottico della lente (figura 5.6). Montando uno slider tra il cavalletto e la cinepresa potremo ruotare agevolmente la testa del cavalletto e traslare la camera avanti e indietro in modo da modificare la sua posizione relativa rispetto all'asse di rotazione (5.7). Il test è da effettuarsi con una metodologia di stima. Si pone la camera in una posizione arbitraria e dopodiché si prosegue analizzando il risultato dopo la rotazione. Se si è ricaduti nella prima casistica descritta (ruotando in senso antiorario, lo stativo più vicino è traslato grossolanamente verso destra) allora si sposterà la camera in posizione arretrata viceversa se si è ricaduti nella seconda casistica (ruotando in senso antiorario, lo stativo più vicino si è schiacciato verso quello più lontano) si avanzerà la sua posizione. Una volta rilevata la casistica incorsa si prosegue a spostare la camera finché non si incorre nella casistica opposta. La posizione corretta si trova quindi nell'intervallo tra l'ultima misurazione effettuata e la precedente. Una volta trovato il punto perfetto in cui non avviene la parallasse otterremo la posizione del centro ottica. Per calcolarla dobbiamo tracciare una retta immaginaria che prolunghi l'asse del treppiede passante per il centro della sua testa e intersecarla con l'asse ottico della lente. Trovata quindi la

posizione del centro ottico nella lente ne segniamo la distanza dal punto di montaggio del tracker (4).

Figura 5.6. Allestimento in studio del set predisposto per il progetto di tesi con dimostrazione della strumentazione usato per il test di ricerca del centro ottico della lente.

5.2.2 Lens File

L'asset per la vera e propria calibrazione della lente virtuale virtuale è il *Lens File* che si può creare agevolmente dal menù rapido che si apre cliccando sul tasto destro del mouse nel *content drawer*. Questo andrà specificato referenziato sia nel component *LiveLinkController* dedicato alle *FIZ* che abbiamo visto prima sia al vero componente che permette di realizzare la calibrazione della lente, ovvero il *component* **Lens**. Abilitando questo *component* il *camera actor* acquisisce la possibilità di scambiare i propri dati con il *Lens File*.

Figura 5.7. Cinepresa *Blackmagin Pocket Cinema Camera 4K* posizionato su uno slider per ricercane il centro ottico della lente.

Il meccanismo che avviene è un po' complesso e può essere riassunto nel seguente modo:

1. Attivando il *LiveLinkComponentController_XR*, il *CineCameraActor* è in grado di ricevere e modificare autonomamente i dati di posizione e rotazione della cinepresa reale.

2. Attivando il *LiveLinkComponentController_FIZ*, il *CineCameraActor* è in grado di ricevere e modificare autonomamente i dati FIZ della lente reale.

3. Attivando il *LensComponent*, i dati precedenti possono essere convogliati all'interno del *Lens File* per poter calibrare la cinepresa attraverso appositi strumenti fisici e virtuali di calibrazione.

Per far in modo che tutta la procedura funzioni, nei details del *Lens File* bisognerà inoltre specificare che si vuole utilizzare come *Evaluation Mode* della calibrazione il tool *Live Link* (figura 5.8). Vediamo nel dettaglio come si imposta e quali funzionalità mette a disposizione il *Lens File*. Cliccando due volte sul *Lens File* appena creato e rinominato a piacere si apre una nuova finestra che due tab principali:

Calibration Steps

Il tab *Calibration Steps* apre un'interfaccia con quattro nuove tab che guideranno il percorso di calibrazione, inoltre nella parte inferiore della finestra vengono visualizzate le informazioni riguardo (figura 5.9): la camera trackata, il *Lens Component* usato, i *FIZ* grezzi di default in input, i parametri stimati, di distorsione della lente e di offset del punto nodale. Le quattro tab sono:

- **Lens Information**. Serve per settare i parametri e le informazioni riguardo la lente e il sensore della camera. Affinché la calibrazione avvenga con successo è molto importante conoscere le dimensioni in millimetri del sensore della propria cinepresa. Per impostare la dimensione corretta è consigliabile farlo direttamente dai *details* del *CineCameraActor* alla voce *Filmback - Sensor Width/Height* (figura 5.10).
- **Lens Distorsion**. Serve per elaborare dataset di immagini catturate dalla cinepresa per ottenere i parametri di distorsione della lente che ne definiscono attraverso un procedimento di stima i parametri *FIZ* reali. Gli algoritmi a disposizione rispecchiano i seguenti metodi:
  - *Checkerboard*: sfrutta un pattern a scacchiera con *features* [3] automaticamente rilevabili.
  - *Points Method*: utilizza un qualsiasi pattern che possa funzionare come oggetto calibratore. Le *features* vanno selezionate manualmente.

[3] In *computer vision*, una *feature* è una parte di informazione rigurardo al contenuto di un'immagine (generalmente circa le proprietà di una determinata regione dell'immagine). Le *features* possono essere strutture specifiche e caratteristiche all'interno dell'immagine come ad esempio punti, spigoli od oggetti.

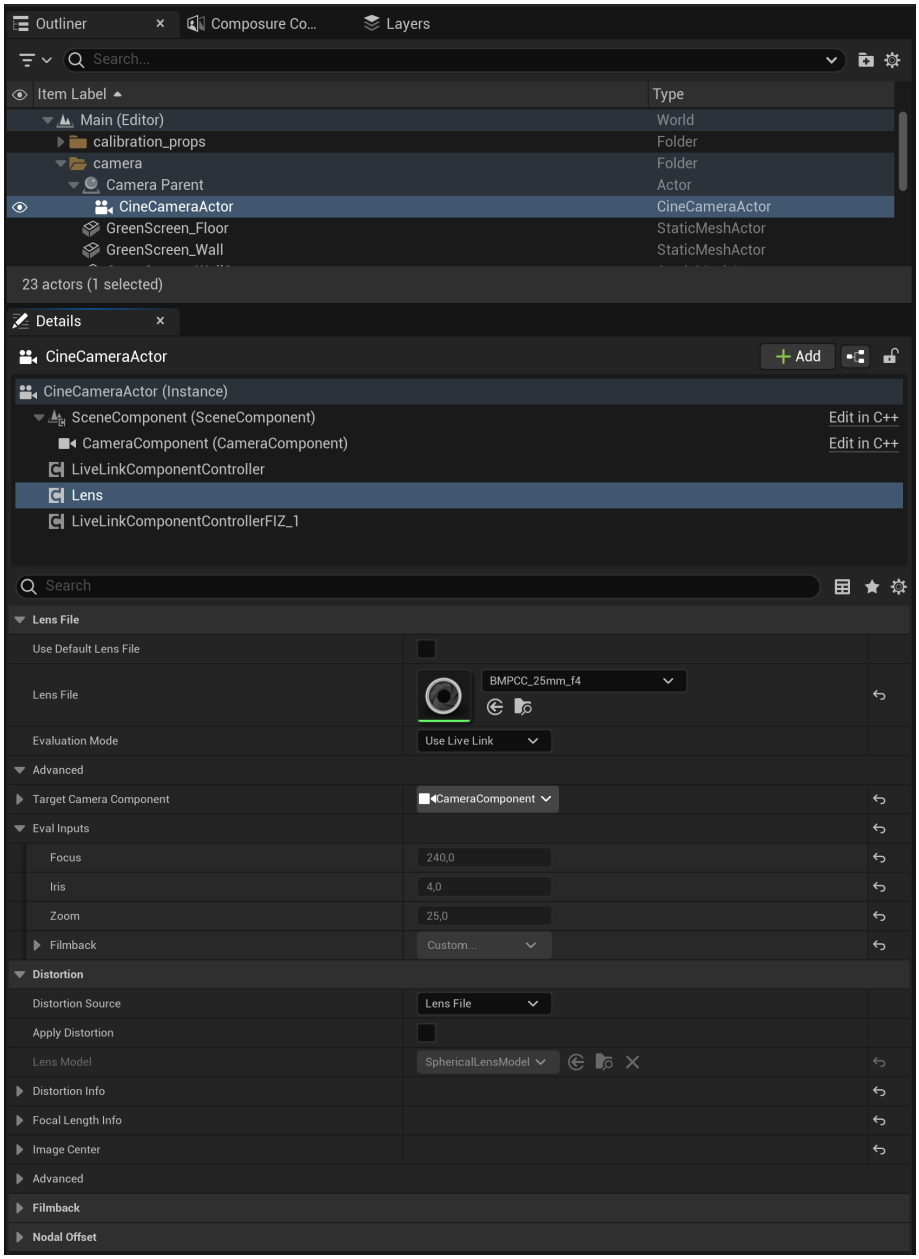

Figura 5.8. Finestra *details* con attribuiti visibili del componente *Lens*.

- **Image Center**. Serve per il riposizionamento manuale del centro dell'immagine dopo aver calibrato la lente. Utile quando alcuni fattori come la temperatura possa influire leggermente sull'immagine prodotta dalla cinepresa.

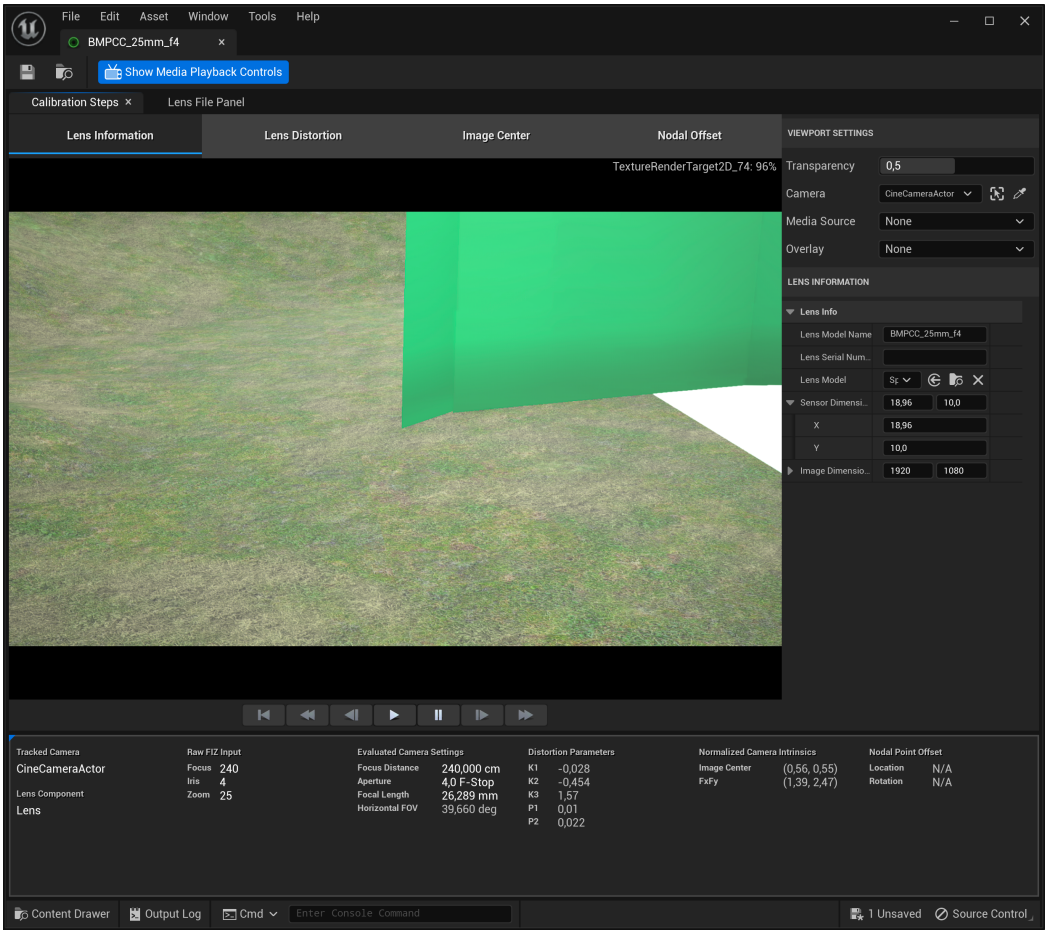

Figura 5.9. Visualizzazione finestra *Lens Information* all'interno del *Lens File* per la calibrazione.

- **Nodal Offset**. Serve per il calcolo algoritmico dell'offset del punto nodale della lente. Gli algoritmi disponibili sono:
  - *Points Method*: utilizza un oggetto calibratore tracciato con una *feature* identificabile (ad esempio il LED dei Vive Tracker 3.0). Si può quindi posizionare il tracker in diverse posizioni all'interno del campo visivo della camera ed una volta raccolto un numero sufficiente di punti, l'offset nodale viene stimato minimizzando l'errore di retroproiezione dei punti catturati.
  - *Aruco Markers*: è un perfezionamento del metodo precedente che processa una sola immagine catturata dalla cinepresa per stimare la posizione di un

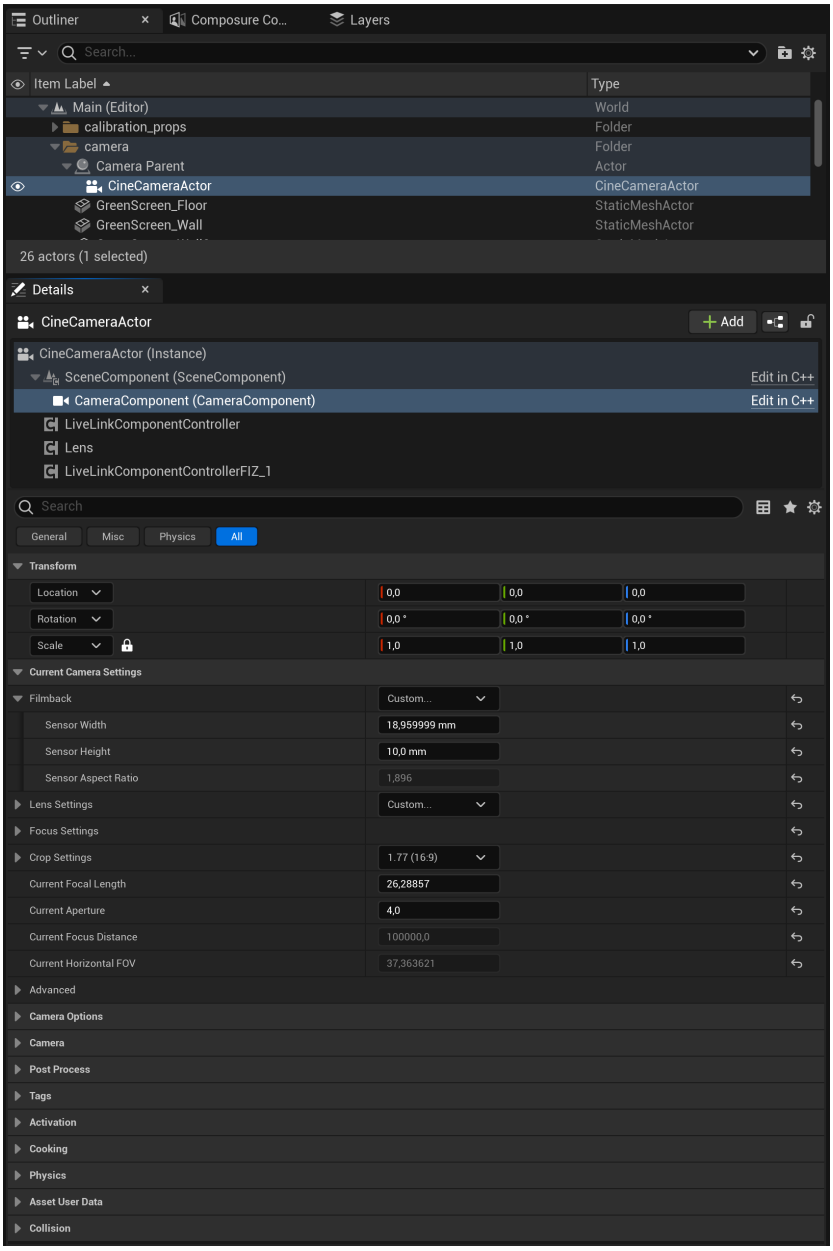

Figura 5.10. Pannello *Details* del *CineCameraActor* visualizzazione della voce *Filmback*.

calibratore con sopra stampato un pattern *ArUco*.

– *Checkerboard*: è una variante del metodo dei punti simile alla precedente ma che utilizza un pattern a scacchiera.

– *Optical Axis*: utilizza un oggetto calibratore che viene catturato a diverse distanze dalla cinepresa che si proiettano tute nel centro esatto della lente. La linea tra questi punti rappresenta l'asse ottico ed il punto nodale si può trovare muovendo manualmente la camera lungo l'asse finché la posizione del centro ottico non è trovato (tecnica mostrata nella sezione precedente).

Lens File Panel

Questa finestra è la prima che va aperta ed impostata quando si vuole calibrare la camera. Infatti essa serve da impostare i parametri della scala sulla quale si baseranno i codificatori dei parametri FIZ (figura 5.11). Il procedimento non è complicato ma va fatto con precisione:

1. Cliccando sul bottone "+" nel menù in alto a sinistra viene creato un nuovo parametro che rinominiamo come *Focus*.
2. Cliccando con il tasto sinistro del mouse sulla retta colorata sulla destra si può creare un nuovo punto di input che va settato al valore di minima distanza del piano focale che può ottenere la nostra lente.
3. Cliccando nuovamente si deve impostare un nuovo valore che corrisponda al massimo.
4. Lo stesso procedimento va ripetuto per l'*Iris*.
5. Lo *Zoom* va implementato vuoto, creando solo un attributo *Focal Length* che abbia due sotto-attributi *Fx* e *Fy*.

Finché non verranno impostati questi parametri il *Lens File* non riuscirà a leggere i *Raw FIZ Input* od a stimare le impostazioni di camera.

Vediamo ora nello specifico i metodi di calibrazione utilizzati durante il progetto di questa tesi: *checkerboard* e *ArUco* markers (figura 5.12).

5.2.3 Stima Parametri di Distorsione - Checkerboard

Unreal Engine mette a disposizione un actor denominato *CameraCalibrationCheckerboard* già predisposto per la funzione che dovrebbe svolgere questo strumento molto comune.

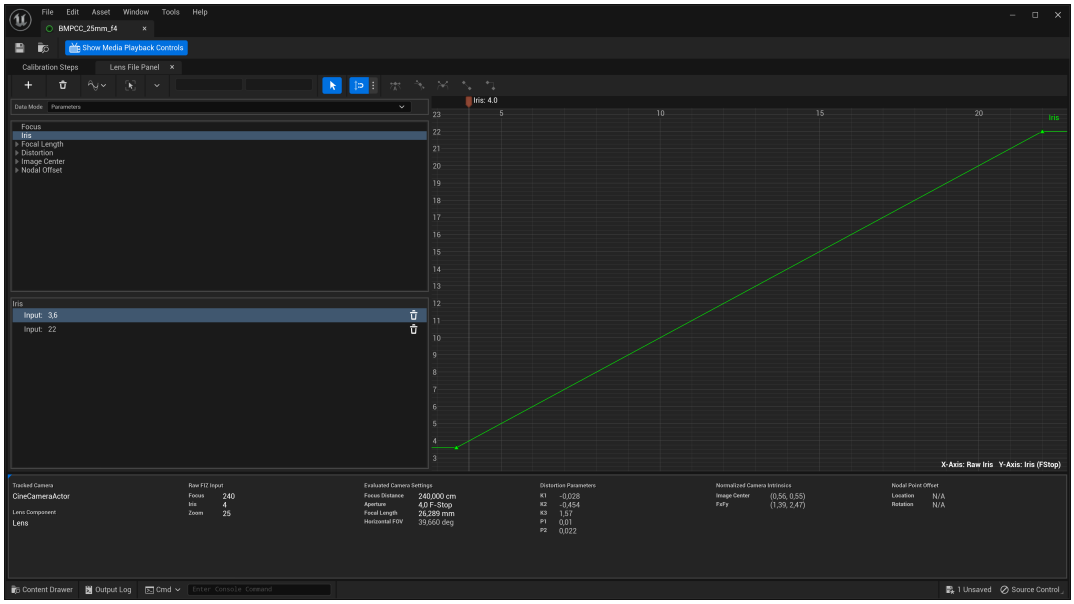

Figura 5.11. Visualizzazione finestra *Lens File Panel* all'interno del *Lens File* per la calibrazione.

come mostra la figura 5.13, questo actor istanziabile dal menù veloce nella parte superiore sinistra dell'interfaccia è al suo interno così strutturato: un elemento *root* e cinque figli che sono rispettivamente 4 *calibration point* per i vertici della scacchiera ed uno per il suo centro. Per utilizzare correttamente questo strumento all'interno del *Lens File* bisogna stampare una scacchiera su un foglio di carta e indicare nel pannello *Details* della *CameraCalibrationCheckerboard* il numero degli angoli per riga e per colonna rispettivamente di fianco alle voci *Num Corner Rows* e *Num Corner Cols* (variabili esposte). Inoltre è fondamentale indicare la lunghezza del lato del singolo quadrato che servirà per calcolare le dimensioni dell'intera scacchiera (di fianco alla voce *Square Side Length*). Infine è consigliato assegnare due materiali di colori diversi per i quadrati pari e dispari della scacchiera (*Odd/Even Cube Material*) in modo che siano facilmente riconoscibili quando bisognerà valutarne la coerenza con la scacchiera fisica ripresa dalla camera (si consiglia il nero per i quadrati bianchi sulla *checkerboard* del set, ed il rosso per quelli neri).

Una volta impostato correttamente questo actor si posizionare nello spazio virtuale. Dalla finestra *Lens Distortion* selezioniamo come metodo l'algoritmo *Lens Distortion*

Figura 5.12. Allestimento studio green per progetto di tesi. *Checkerboard* e *ArUco MArker* dedicati alla calibrazione.

Checkerboard e come checkerboard quella appena creata. Quando istanziamo il *Lens File* legandolo a un *CineCameraActor* specifico, viene generata una nuova composizione in automatico che ha come foreground un *media plate* con l'immagine ripresa e come background un *CG layer* ripreso dalla nostra camera virtuale stessa (si approfondiranno questi concetti nel capitolo 6). Per poter visualizzare all'interno della finestra del *Lens File* l'immagine catturata dalla nostra cinepresa reale bisogna dunque andare nel pannello *details* del *media plate* collegato ed impostare come texture di input il *media player* del *media bundle* generato in precedenza. Ora è possibile visualizzare l'immagine ripresa dalla camera nella finestra del tab *LEns Distortion*. Per procedere alla stima dei parametri di distorsione della lente bisogna conoscere il funzionamento dell'algoritmo. Quest'ultimo funziona bene infatti solo vengono scattate una buona quantità di immagini (circa 10)

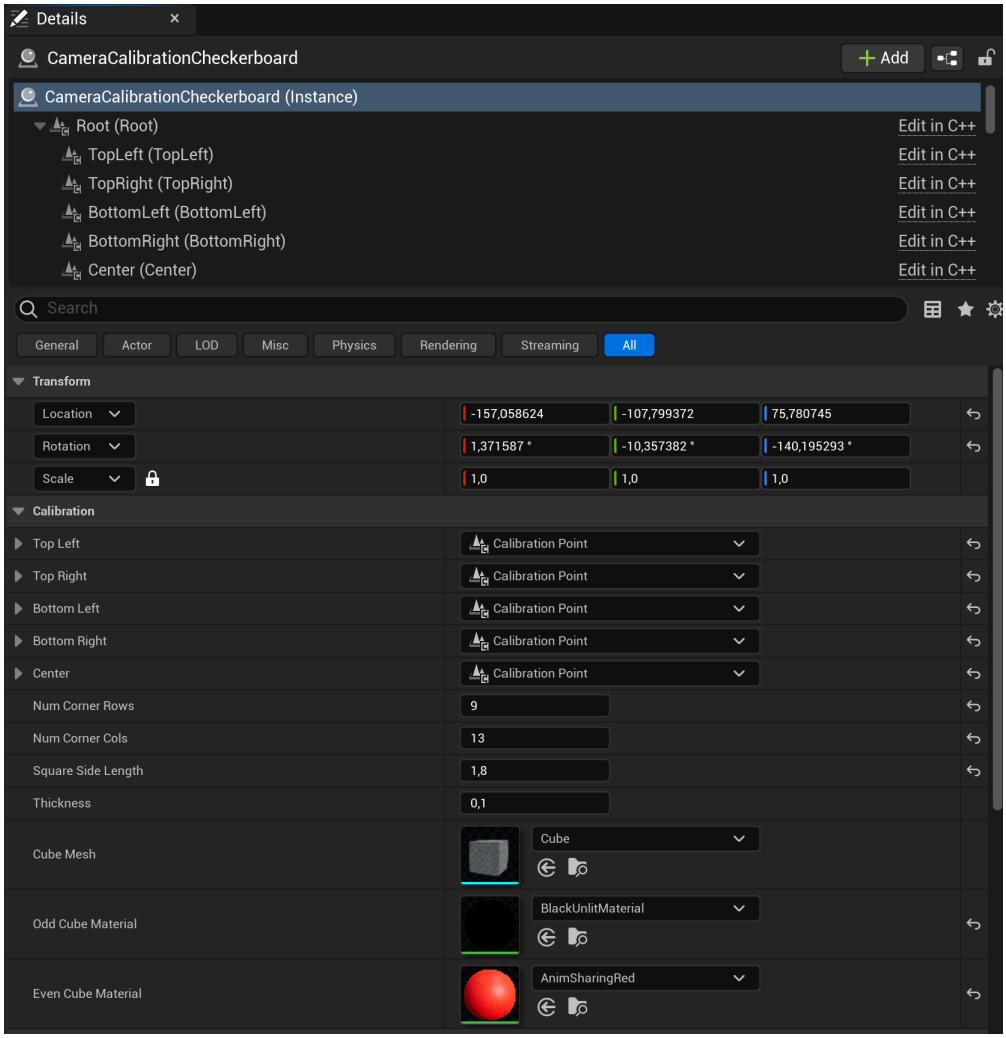

Figura 5.13. Pannello *Details* dell'actor *Checkerboard*.

in cui la checkerboard è in prospettive diverse, occupa spazi diversi, ed è parzialmente sovrapposta a sé stessa in altre immagini. È importante che le immagini vengano scattate con la checkerboard che viene posta con un pattern a zig-zag partendo da destra verso sinistra, in questo modo si aiuterà l'algoritmo a lavorare meglio. Si può procedere sia spostando la scacchiera manualmente, sia tenendola ferma e spostando o ruotando la camera. Per raccogliere le immagini direttamente da Unreal Engine è possibile usare un metodo molto snello che è quello di cliccare col tasto sinistro del mouse sulla checkerboard visualizzata nello schermo per catturare una foto istantanea del riquadro. Nella

figura 5.14 viene mostrato un esempio di questo procedimento. Una volta raccolto un numero buono di immagini si può cliccare sul bottone *Add To Lens Distortion Calibration*. Questa azione eseguirà l'algoritmo che ritornerà in una nuova finestra la stima dell'errore con cui considerare il risultato (è accettabile se inferiore al pixel, ottimo se inferiore al decimo di pixel). Se non accettiamo l'errore l'algoritmo viene interrotto per permetterci di modificare il dataset di immagini che si è creato. Altrimenti accettando l'errore vengono modificate automaticamente le impostazioni della camera virtuale aggiornandola con le corrette *FIZ* stimate e visualizzabili nel tab *Lens Information*.

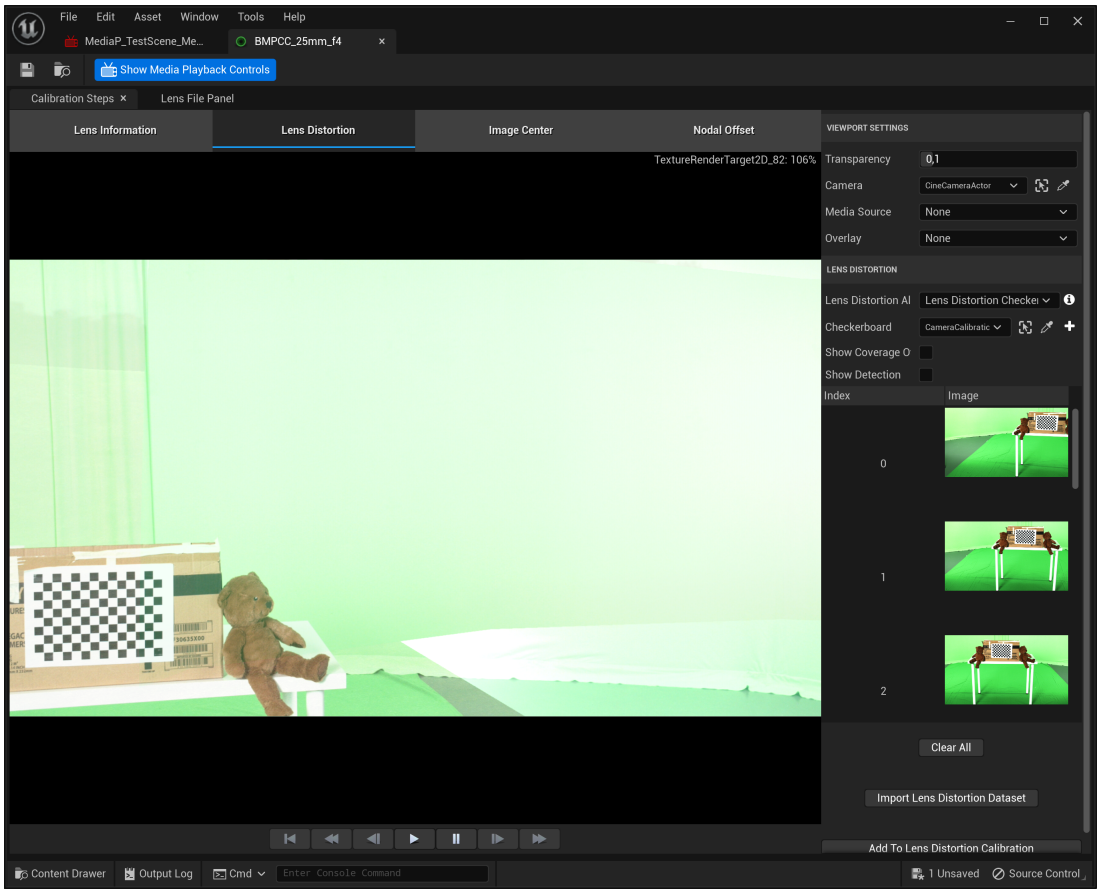

Figura 5.14. Finestra *Lens Distortion* con esempio dimostrativo di creazione di un *dataset* di immagini.

5.2.4 Stima Offset Punto Nodale - Aruco Marker

Unreal Engine mette a disposizione come abbiamo visto poco sopra un algoritmo che funziona con i pattern *ArUco*, tuttavia non esiste un actor già predisposto come nel caso della checkerboard. Si dovrà quindi istanziare una classe *blueprint* che svolga correttamente il ruolo di oggetto calibratore. Come esemplifica la figura 5.15, per farlo bisogna creare una cartella con all'interno tre asset:

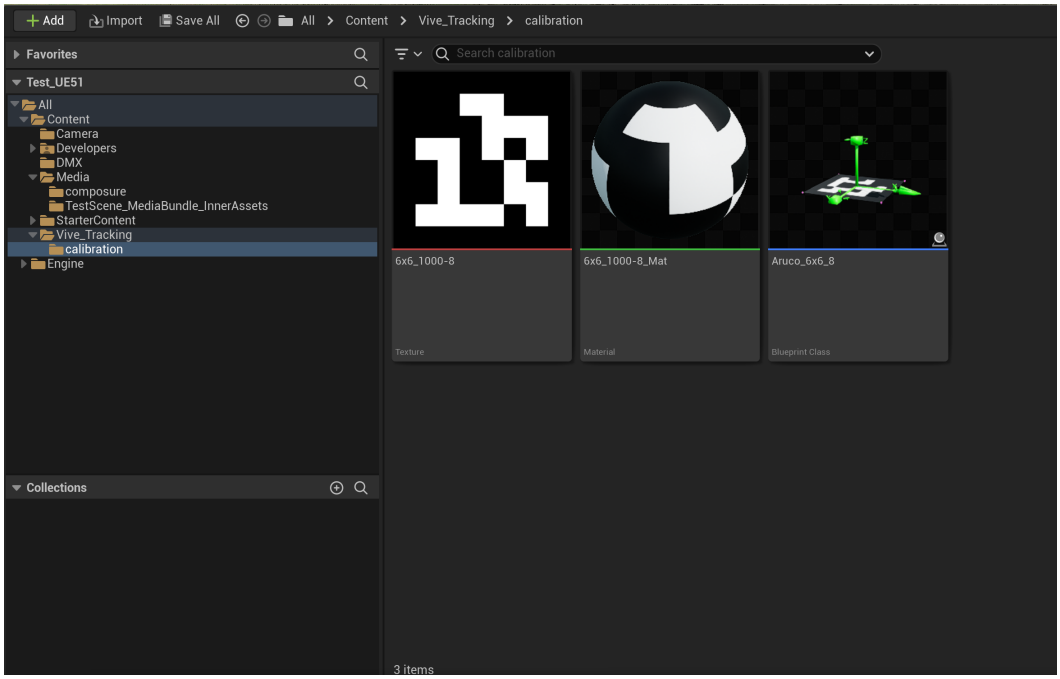

Figura 5.15. Asset presenti nella cartella *ArUco Marker* nel *Content Drawer*.

1. Una *Texture* contenente il pattern Aruco prescelto. Questi pattern sono facilmente scaricabili dal sito https://chev.me/arucogen/. Possiamo selezionare il *dictionary*, il *marker ID* e la lunghezza del lato in millimetri. Si consiglia di segnarsi tutti questi dati perché torneranno utili successivamente. Il sito permette inoltre sia di stampare il marker generato che di salvarne una copia con estensione .png per creare la texture sul game engine.

2. Un *Material* molto semplice che abbia come input del parametro di *shading Base Color* la texture stessa.

3. Una *Blueprint Class* che istanzi l'oggetto marker ed i suoi punti calibratori. Tratteremo al fondo della sezione questo asset nello specifico.

Si può ora istanziare una copia del blueprint e posizionarla in un punto utile della scena virtuale per esempio dove vorremmo che stia il nostro attore. Se sul set in studio posizioniamo lo stesso pattern nel punto in cui reciterà l'attore otterremo l0effetto desiderato. Nel pannello *Nodal Offset* del *Lens File* selezioniamo quindi il metodo che utilizza l'algoritmo *Nodal Offset Aruco Markers* e scegliamo come *Calibrator* la classe *blueprint* che avremo creato (figura 5.16). Se il *blueprint* è impostato correttamente basterà cliccare col tasto sinistro del mouse sull'immagine mostrata su schermo ripresa dalla camera reale nel punto in cui si trova il target per estrarne le posizioni dei 4 *calibrator points* e dopodiché premere il bottone *Apply to Camera Parent* per fare in modo che il padre del *CineCameraActor* possa essere spostato nello spazio tridimensionale virtuale in modo che il marker virtuale e quello reale corrispondano. Risulta importante per il corretto funzionamento dell'algoritmo che il *Calibrator Point* assegnato sia quello inferiore destro e che i punti di calibrazione siano stati nominati in maniera coerente a come sono aspettati dall'algoritmo:

$$DICT\_Dictionary\_100 - MarkerID - PosVertice$$

Setup Blueprint

Per implementare correttamente il design un *Marker Aruco* bisogna innanzitutto strutturarlo correttamente a livello visuale nel tab *viewport* nell'editor che si apre facendo doppio clic sulla classe blueprint del marker. Attraverso il bottone *Add* generiamo come sotto-rami della *DefaultSceneRoot* 3 elementi principali (figura 5.17):

- Un *Plane* che fungerà da supporto fisico per il nostro marker. Nell'attributo *Material* ci assicuriamo che sia referenziato il material già appositamente preparato precedentemente.

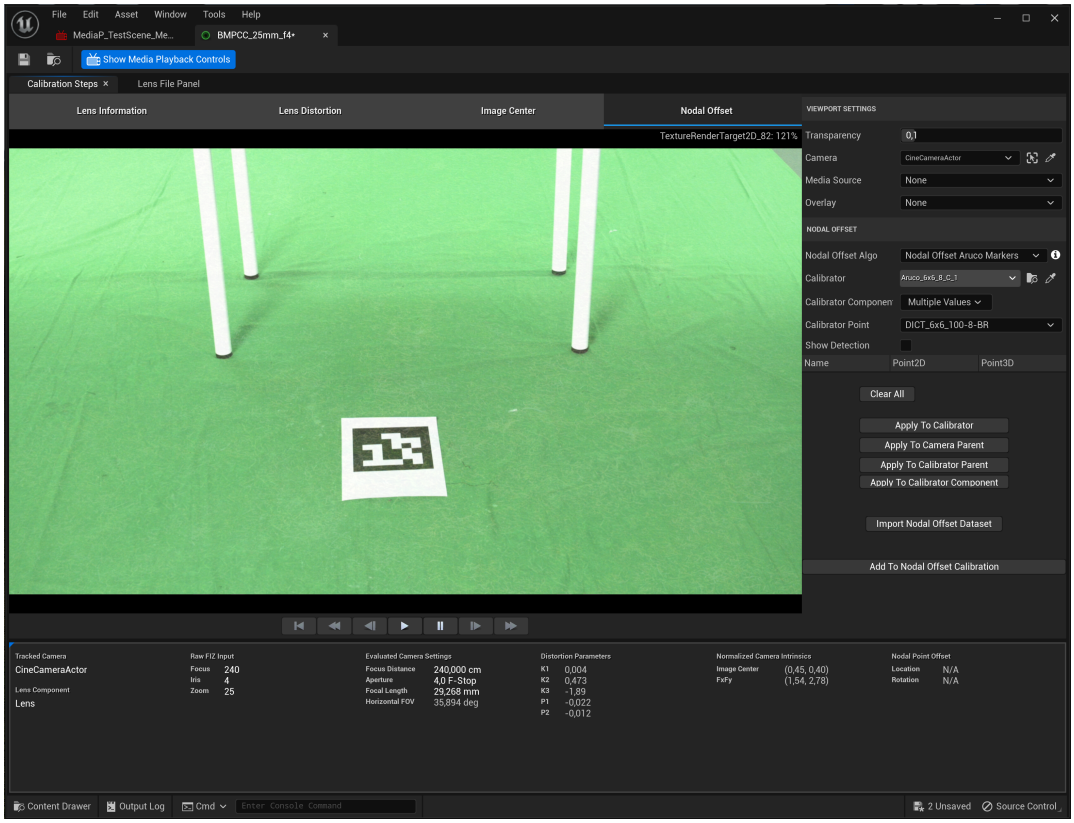

Figura 5.16. Finestra *Nodal Offset* con un esempio dimostrativo di creazione di un *dataset* di punti di calibrazione.

- Una *Static Mesh* di tipo *Grid Axis* per avere un sistema di coordinate del marker. Bisogna infatti prestare attenzione che le coordinate sia coerenti a quelle della scena virtuale e che il marker sia posizionato con lo stesso orientamento sia nella realtà che nell'ambiente virtuale.

- Quattro *Calibration Point* da posizionarsi ai quattro vertici del marker e da nominare accuratamente come visto poc'anzi e indicando la posizione di ciascuno relativamente al sistema di riferimento come *Bottom/Top* e *Left/Right*.

Per quanto riguarda la parte di *scripting* bisogna creare uno script costruttore che istanzi l'oggetto nel mondo quando viene invocato. Per farlo bisogna premere sul bottone "+" di fianco alla voce *Functions* dell'interfaccia e scegliere *ConstructionScript*. Inoltre fin da subito predisponiamo anche una nuova variabile con il bottone "+" alla voce *Variables*

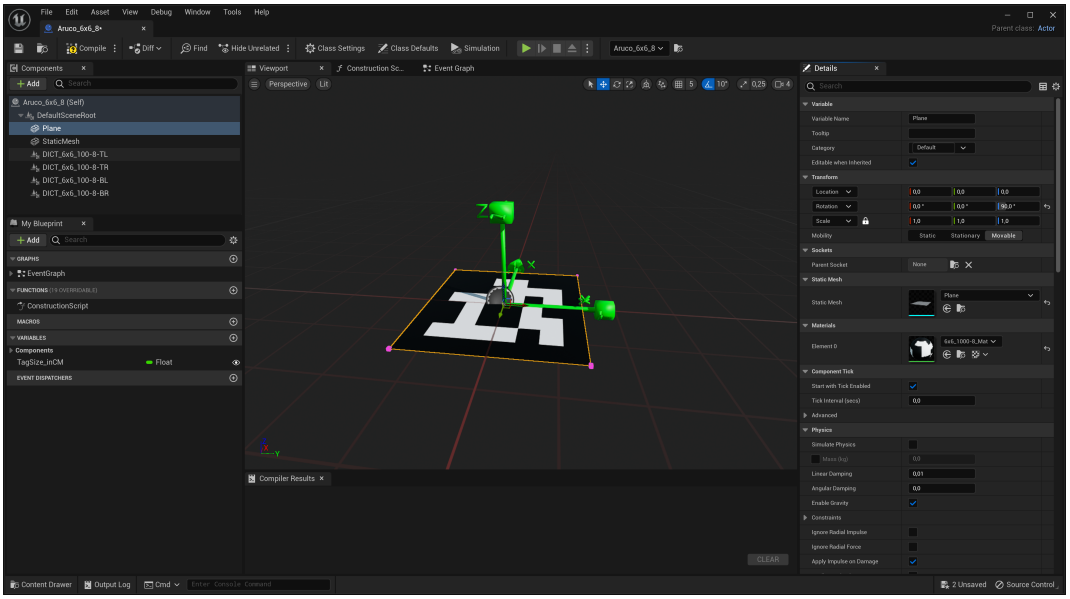

Figura 5.17. Finestra generale e *viewport* del *blueprint* per *ArUco Marker*.

che nominiamo *TagSize_CM* di tipo *Float*. Impostiamo il valore di default alla misura in centimetri della lunghezza del lato del marker *Aruco* stampato e presente sul set. Aprendo quindi la nuova tab dedicata allo script aggiunto possiamo definire la funzione costruttrice. La figura 5.18 mostra il grafo a nodi completo. Si tratta di un grafico molto semplice:

1. Si collega l'invocazione del *Constructor Script* a una nuova funzione, la *Set Relative Scale 3D*.

2. Impostiamo come target di questa funzione la *DefaultSceneRoot* ottenuta attraverso la funzione *Get*.

3. Impostiamo come nuova input della funzione di scalamento la variabile prima predisposta i cui valori vanno opportunamente divisi per 100 poiché l0unità di default è in metri.

Il *Marker Aruco* è così pronto per essere utilizzato.

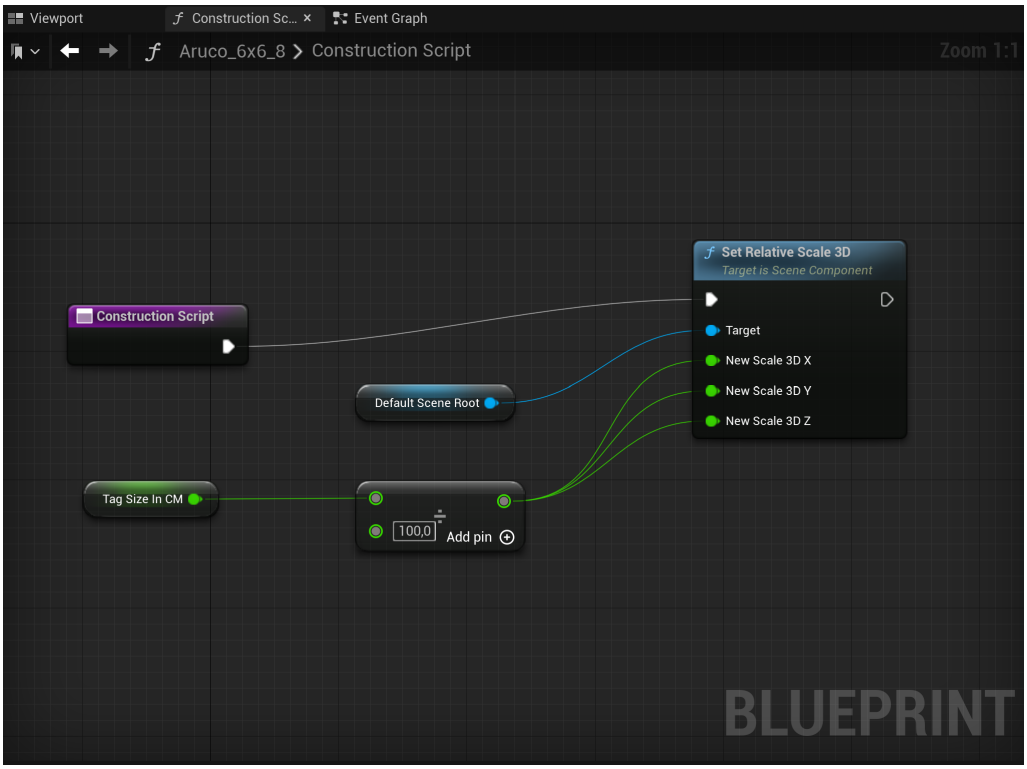

Figura 5.18. *Constructor Script* dell'*ArUco Marker*.

Capitolo 6

Compositing Real-Time

6.1 Composure Tool

Ora che la cinepresa è correttamente tracciata e che, grazie alla calibrazione, le immagini riprodotte dalla nostra camera virtuale corrispondono precisamente alle caratteristiche di quelle catturate dalla cinepresa sul set, dobbiamo affrontare una parte fondamentale delle produzioni di virtual production ovvero il *compositing real-time*. Infatti è attraverso il tool *composure* che potremo proiettare su uno schermo ausiliario o sul nostro computer l'immagine compositata finale nello stesso momento in cui effettuiamo le riprese. Questo è infatti uno degli aspetti più rilevanti di questa tecnologia per gli addetti ai lavori. Avere un feedback in tempo reale del risultato finale permette di risolvere numerosi problemi nell'immediato, senza dover rimandare in post-produzione la loro correzione. Inoltre si può da subito scegliere al meglio le inquadrature ed i movimenti di camera non dovendo immaginare il risultato ottenuto ma vedendolo.

Entriamo nella trattazione tecnica del tool offerto da Unreal Engine. Il *composure tool* si può attivare dalle opzioni in alto a sinistra dell'interfaccia di Unreal Engine cliccando su *windows* e successivamente su *virtual production tools* e *composure*. Si aprirà quindi una nuova finestra simile all'*outliner* nella quale potremo istanziare nuovi asset. Innanzitutto bisogna generare un nuova composizione, per farlo si clicca il tasto destro del mouse all'interno della finestra e si sceglie la voce *new comp*. Si apre quindi una nuova finestra

(figura 6.1) dalla quale scegliere di generare un *Empty Comp Shot*.

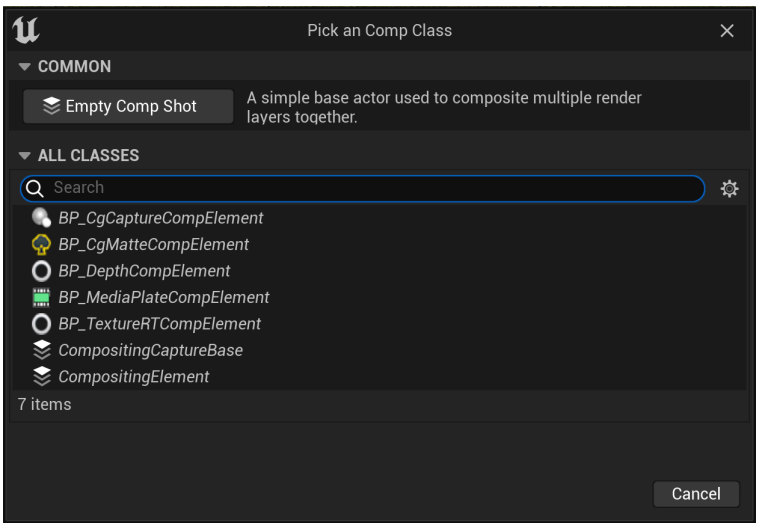

Figura 6.1. Finestra aperta da *Unreal Engine* per la creazione di una nuova composizione.

A questo punto dovremo popolare la composizione con i layer necessari al nostro lavoro, cliccando sul tasto destro del mouse e poi su *add layer*. Come mostra la figura 6.2, i tre layer basilari sono:

- **Media Plate**, un elemento usato reindirizzare l'input video del media bundle all'interno della pipeline di compositing ed elaborarlo.
- **CG Layer**, un elemento di compositing utile a catturare e renderizzzare una porzione della scena virtuale attraverso una determinata camera virtuale.
- **CG Matte**, un elemento il quale, a partire dalla scena virtuale, genera una maschera a canale singolo rosso/nero.

Tratteremo ora nello specifico ognuno dei tre layers principali mostrando come possono essere utilizzati per creare una semplice composizione che necessiti di:

1. Un soggetto ripreso dalla cinepresa su un green set correttamente "bucato".
2. Un background virtuale.

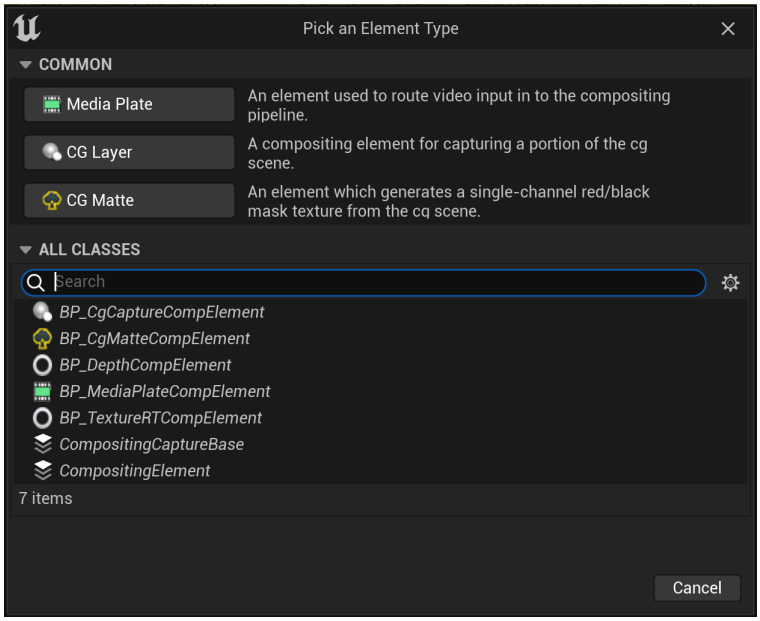

Figura 6.2. Finestra aperta da *Unreal Engine* per aggiungere un nuovo *layer* ad una composizione del *composure tool*.

3. Un garbage matte che escluda ciò che non è necessario dalla ripresa e che sia adattivo/dinamico rispetto ai movimenti della cinepresa.

6.1.1 Media Plate

Il primo layer che analizziamo è il *Media Plate*. Il suo ruolo è quello di reindirizzare i dati della sorgente video all'interno della catena di compositing. Per farlo dobbiamo perciò abilitare la voce *enabled* dell'attributo *composure* sotto le voci in ordine *input, inputs, media source* (figura 6.3). Dopodiché scegliamo dal menù a tendina della variabile *Media Source* la Media Texture presente nella cartella Inner Assets del nostro MediaBundle precedentemente predisposto. Nel menù a tendina della variabile *Material* scegliamo invece la Material Instance presente sempre nella stessa cartella. A questo punto selezionando il *Media Plate* dall'outliner potremo vedere un'anteprima del video trasmesso dalla cinepresa dentro il software. L'elaborazione del video passa invece attraverso a più step sotto la voce *Transform/Compositing Passes* che analizzeremo ora.

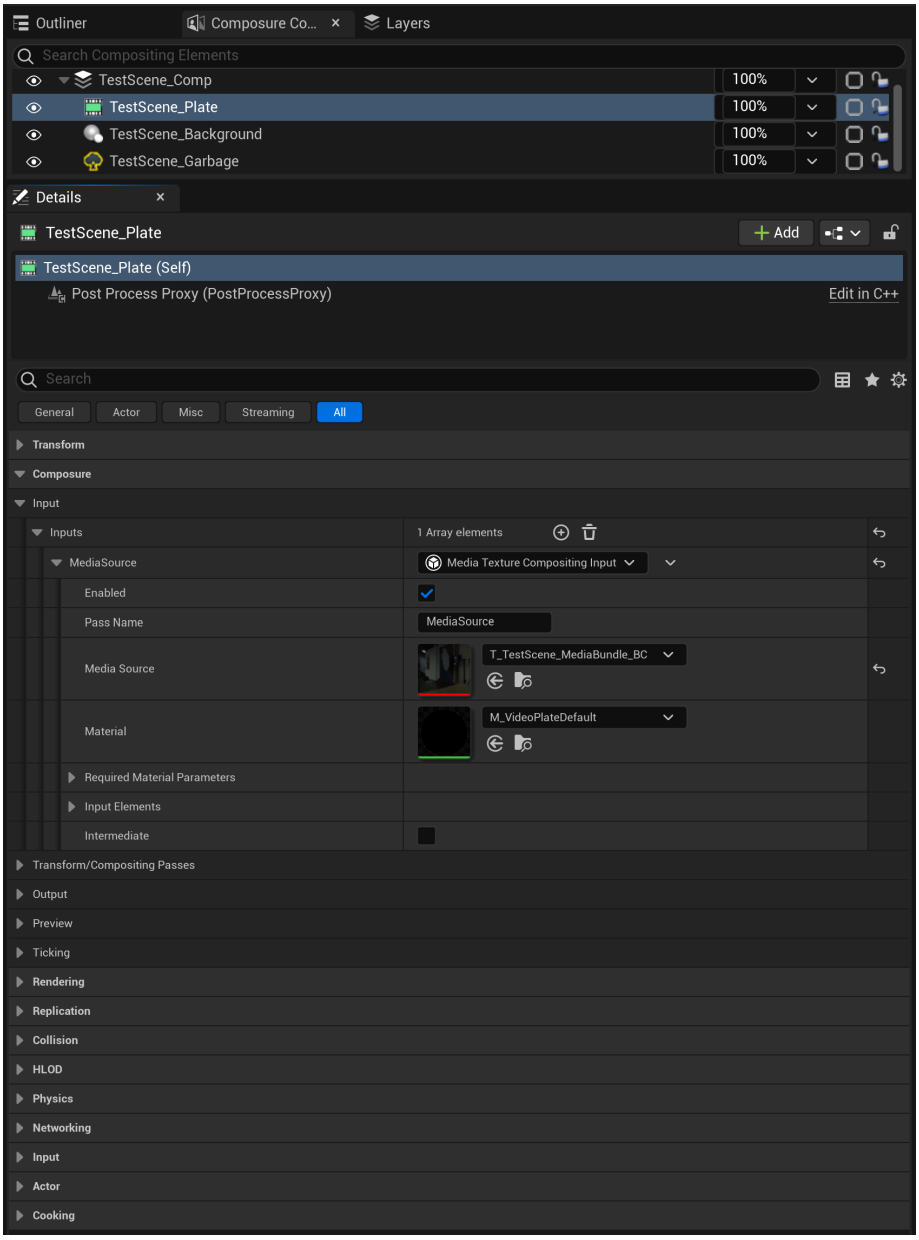

Figura 6.3. Finestra aperta da *Unreal Engine* per aggiungere un nuovo *layer* ad una composizione del *composure tool*.

Chroma Keying

Per utilizzare correttamente l'algoritmo di *chroma keying* messo a disposizione dal tool *composure* impostiamo l'algoritmo su *Multi Pass Chroma Keyer*. Sotto la voce *Transform Passes* scegliamo le "chavi" di campionamento del pigmento da eliminare. Come

mostrato in figura 6.4, si possono impostare chiavi multiple. Per farlo si deve cliccare sul "contagocce" disegnato a destra delle barre riportanti visivamente il coloro memorizzato per ogni indice del vettore *Key Colors*. Si apre una finestra della preview dell'immagine ripresa dalla camera fisica con la possibilità di selezionare il colore in formato RGB di uno specifico pixel e quindi confermare o annullare la scelta corrente. Una volta impostata la chiave bisogna quindi ritoccare i *Materials Parameters*. I parametri più rilevanti sono (figura 6.4):

- **ChromaBound** e **ChromaContrast** che servono a controllare il contrasto e la rifinitura tra le zone bianche e nere del mascherino.
- **DevignetteOuter** è un parametro che permette di regolare l'effetto "vignetta" del nostro mascherino aumento il raggio delle zone cancellate verso gli angoli
- **BlackClip** e **WhiteClip** che servono per regolare le rispettive soglie rispetta alle quali un pixel viene o meno mascherato.
- **AlphaBias** è un parametro da aggiustare nel caso in cui il colore del green screen non sia verde "puro" e stia quindi causando un effetto di trasparenza su alcune parti del foreground.
- **PreBlurKernelSize** che serve per regolare la quantità di blur prima che venga valutato l'algoritmo.

Despill

Una ripresa effettuata in uno studio con green screen avrà una certa quantità della luce ambientale della stessa tonalità del colore del fondo verde a causa della riflessione e diffusione della luce. Questo effetto è conosciuto come *spill*. Usando un algoritmo di *chroma keying* ciò potrebbe apparire innaturale e causare la dissolvenza di alcune parti dei soggetti in primo piano. Anche prendendo tutte le precauzioni necessarie descritte nel capitolo 2, un poco di spill rimane sempre ed è perciò utile avvalersi dell'algoritmo di *Despill* a disposizione. Su Unreal Engine quest'ultimo si chiama *Multi Pass Despill* (come mostrato in figura 6.5) e si può correttamente configurare nel seguente modo:

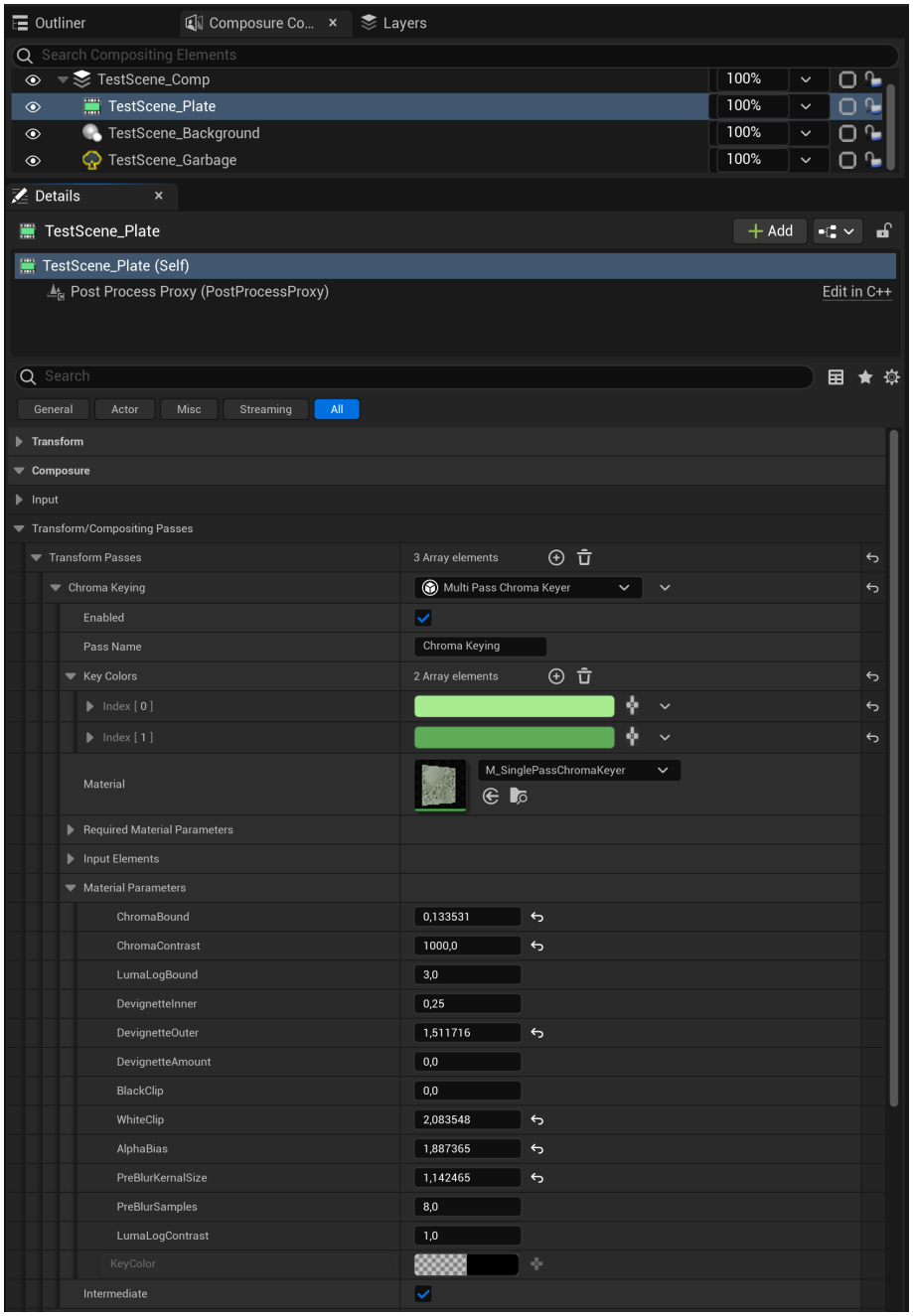

Figura 6.4. Variabili esposte e modificabili dell'attributo *Transform Passes - Chroma Keying* del layer *Media Plate*.

- Scegliere un **Key Color**. Il campione dev'essere selezionato da una regione della figura in foreground che generalmente soffre in particolarmente questo difetto come

ad esempio i capelli.

- Impostare il valore del parametro **DeSpillAmount** su un valore più o meno alto di 1 (che è il default) in base a quanto l'effetto di *spill* è visivamente presente nella nostra immagine.
- Modificare il valore del parametro **HueRange** per amplificare o ridurre l'intervallo cromatico che viene desaturato dall'algoritmo.

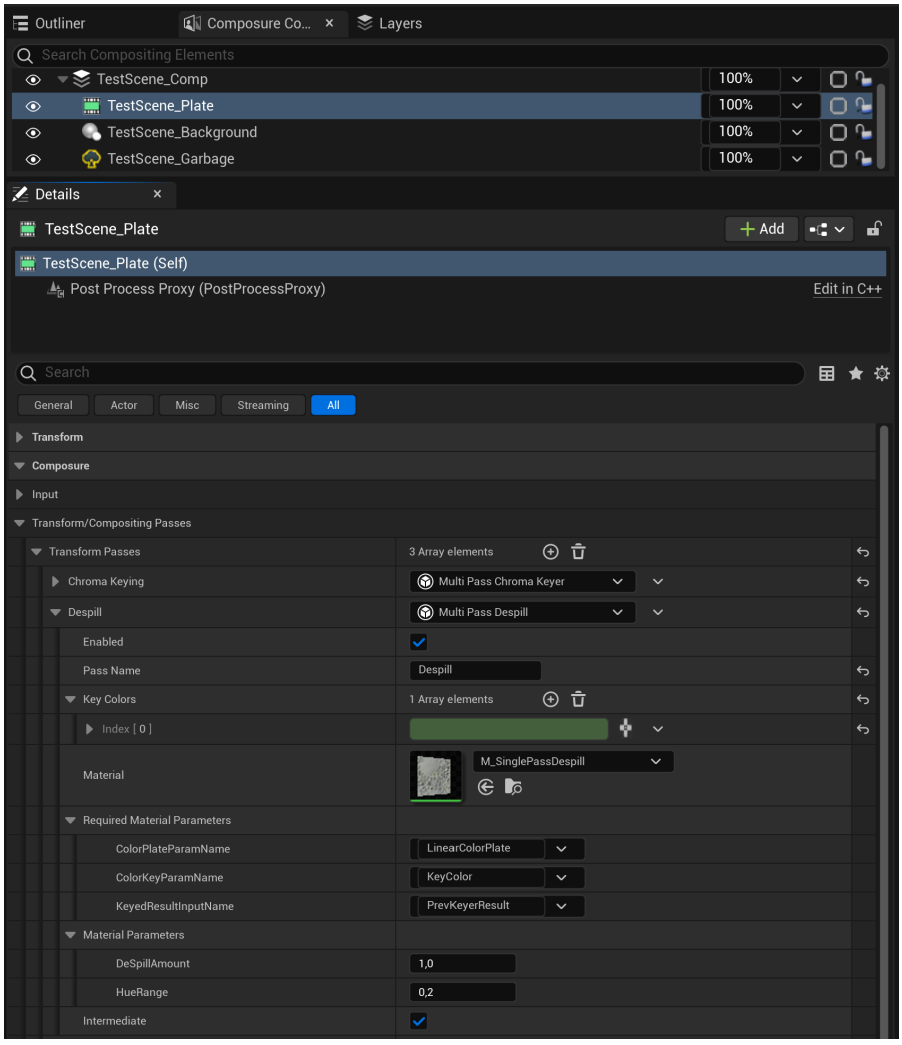

Figura 6.5. Variabili esposte e modificabili dell'attributo *Transform Passes - Despill* del layer *Media Plate*.

Erode

L'algoritmo di *erode* serve per smussare i bordi del mascherino ottenuto attraverso l'algoritmo di chroma keying. Risulta molto semplice da configurare in quanto come mostrato in figura 6.6 gli unici parametri significativi da modificare all'occorrenza sono:

- **ErodeKernelSize** che definisce l'ampiezza della finestra di campionamento
- **ErodeNumSamples** che definisce il numero di campioni utilizzati per il campionamento stesso.

6.1.2 CG Plate

Il secondo layer che analizziamo è il *CG Layer* che si occupa della cattura e renderizzazione di parti della scena virtuale. In base al posizionamento del layer specifico, quest'ultimo potrà risultare davanti o dietro rispetto al soggetto ripreso sul set. Analizziamo però solamente la casistica per creare un background virtuale. All'interno dell'attributo *composure* del layer CG sono di rilevanza pratica le seguenti voci:

1. **Input**, questo tab permette di inserire il *Target Camera Actor* che deve renderizzare una porzione di scena. Permette inoltre di includere od escludere dalla visione della camera selezionata determinati *actors* precedentemente raggruppati in *layers* (tratteremo questa procedura al fondo del capitolo). Per esempio nel caso esposto in figura 6.7 è stato escluso il layer *Garbage* che racchiudeva gli *actors* che componevano il green screen virtuale.
2. **Transform/Compositing Passes**, questo tab permette di gestire la *color correction* [1] e la *color grading* [2] dell'immagine renderizzata nel *CG Layer*.

[1] La *color correction* è una delle ultime fasi della produzione di un video e consiste nel bilanciamento dei suoi colori. Il colorist corregge il colore complessivo delle immagini nel video, così che la sensazione visiva risultante sia di equilibrio cromatico.

[2] La *color grading* avviene dopo la correzione colore e si pone l'obiettivo di dare un look unico e riconoscibile alle immagini che rispecchi lo stato d'animo degli eventi filmati per poter empatizzare meglio con lo spettatore.

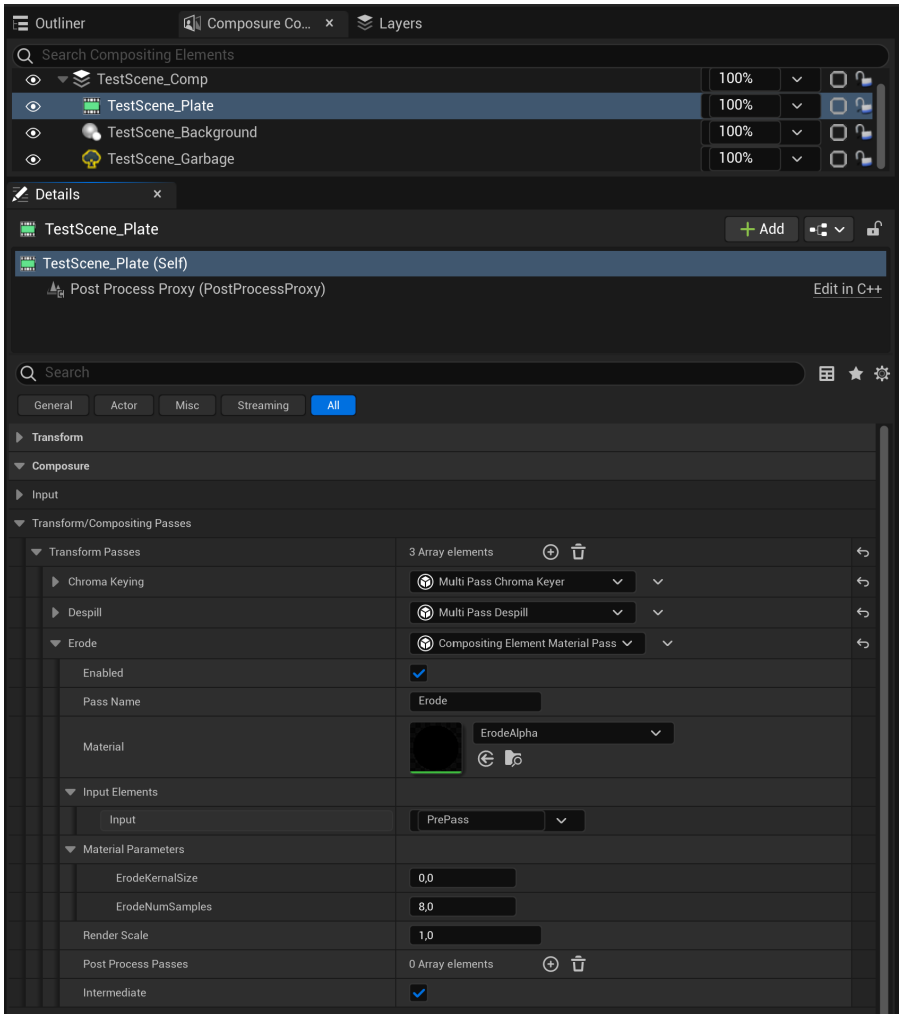

Figura 6.6. Variabili esposte e modificabili dell'attributo *Transform Passes - Erode* del layer *Media Plate*.

3. **LensDistortion**, è utile per dare all'immagine digitale la stessa aberrazione ottica provocata dalla lente della cinepresa reale sulle immagine trasmesse al *Media Plate*. Per questo motivo è stato scelto il componente *Lens* trattato nel capitolo 5 già appartenente al Camera Actor sincronizzato alla camera del set fisico.

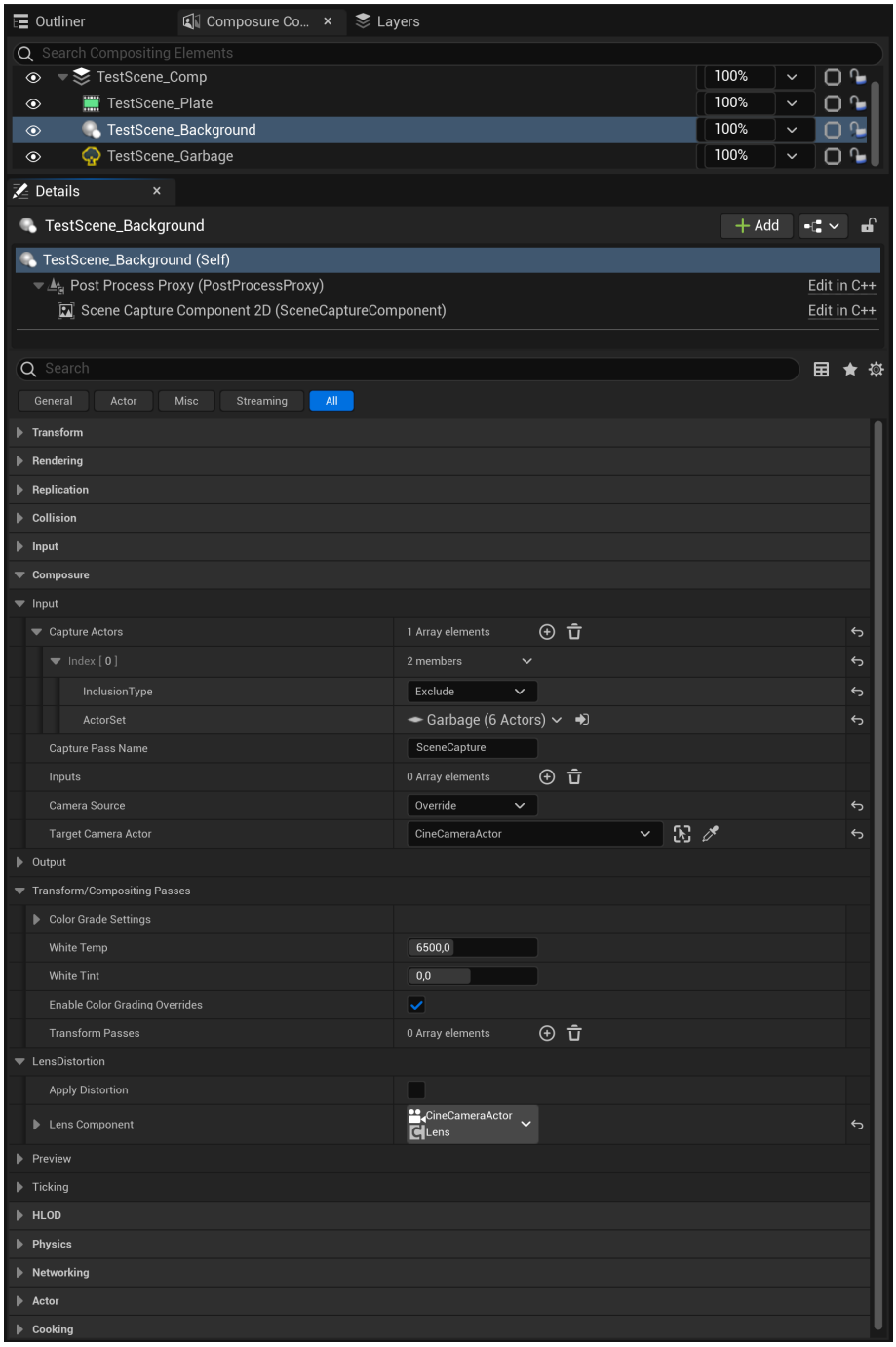

Figura 6.7. Variabili esposte e modificabili dell'attributo *Transform Passes - Erode* del *CG Layer*.

6.1.3 CG Matte

Analizziamo infine il terzo Layer. Esso serve per la creazione di un mascherino (in inglese *matte*) e a titolo esemplificativo dimostreremo come creare un **garbage matte** nello specifico dinamico e adattivo che è l'utilizzo principale di questo tool durante una seduta di virtual production ibrida. È necessario che il *garbage matte* sia reattivo al movimento della camera poiché se il mascherino fosse un'immagine statica maschererebbe appunto sempre la stessa porzione del frame, mentre vorremmo che siano solo determinate zone ad essere mascherate, prevalentemente in un'area circostante i soggetti in primo piano.

Green Screen Virtuale

Per questo motivo il metodo usato è stato quello di creare una controparte del green screen allestito nello studio all'interno del mondo virtuale creato nel game engine. Per farlo ricordiamo quanto visto al capitolo 4, ovvero che la posizione del *parent* del camera actor corrisponde alla posizione sul set della base station impostate sul numero di canale minore. Procediamo per cui in ordine:

1. Misurando le dimensioni del green screen. Ne ricreiamo quindi una copia virtuale opportunamente scalata istanziata come figlia del *parent* del nostro camera actor (figura 6.8).

2. Scegliamo un punto di riferimento comune facilmente utilizzabile, ad esempio uno degli angoli del tappeto verde e ne facciamo l'origine di un sistema di coordinate. Istanziamo quindi un *empty actor* padre degli actors che compongono il green screen nel corrispondente punto virtuale.

3. Calcoliamo il vettore di rototraslazione * tra questo sistema di riferimento e quello del sistema di tracciamento SteamVR (ricordiamo la sua trattazione al capitolo 2) e inseriamo i dati raccolti nell'attributo *transform* della collezione di oggetti.

*Questo procedimento può essere velocizzato e semplificato utilizzando il tracker 3.0 per ottenere in maniera rapida i dati relativi a posizione e rotazione del punto prescelto rispetto al sistema di riferimento dello SteamVR.

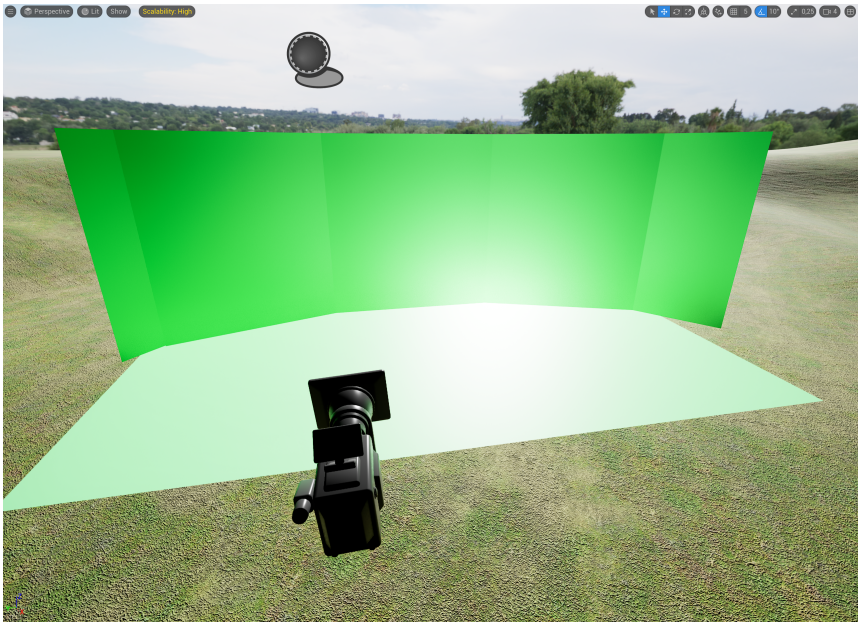

Figura 6.8. Esempio di scena generata su *Unreal Engine* con un *CameraActor* ed una collezione di piani verdi che simulano il green screen presente sul set in studio.

Layers

Successivamente la collezione di actors creata dev'essere inserita all'interno di **layer**. Questo sarà infatti indispensabile per poter comunicare al *CG Matte* gli oggetti da escludere o includere all'interno del mascherino. Per farlo selezioniamo tutti gli actors che vogliamo inserire in un nuovo layer, andiamo all'interno del tab *Layers* e clicchiamo sul tasto destro del mouse. Selezioniamo quindi l'opzione *Create new layer from current selection*. La figura 6.9 mostra il risultato di questa operazione. La figura 6.10 mostra invece come utilizzare il layer appena creato nelle impostazione del *CG Matte*. Sotto l'attributo *Composure*, *Input* selezioniamo come *Matte Type* la modalità *Garbage Matte*. Alla voce *Capture Actors* selezioniamo invece la modalità *Include* e come *ActorSet* il layer Garbage appena creato. Come anticipato in precedenza questa operazione è necessaria inoltre per escludere il green screen istanziato dal campo visivo della camera virtuale.

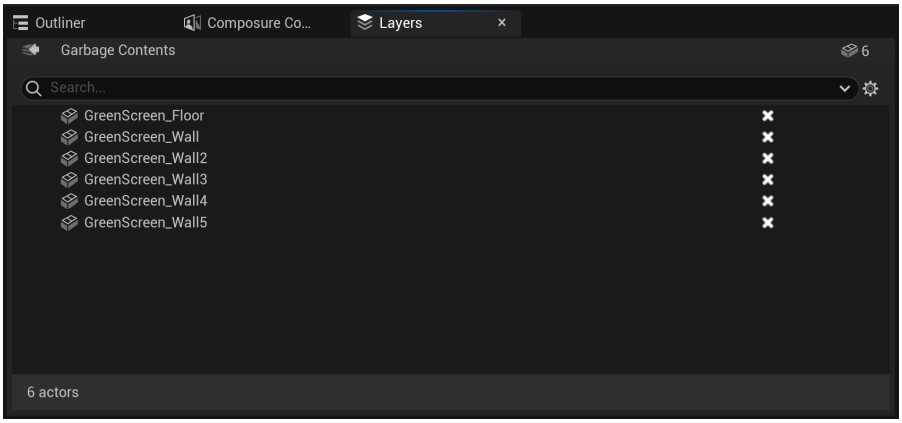

Figura 6.9. Esempio di creazione di un nuovo *layer* con diversi oggetti (*actors*) al suo interno.

6.2 Material per Compositing

Attualmente è tutto predisposto affinché l'immagine trasmessa ad Unreal Engine dalla cinepresa del set possa essere correttamente acquisita ed eleborata. Tuttavia manca uno script che descriva le regole attraverso cui i diversi layer debbano essere compositati. Questo script si genera attraverso il grafo a nodi di un nuovo *material*. Per istanziarlo ed assegnarlo alla composizione creata nel tool di composure, selezioniamo la composizione che dentro di sé racchiude i layers prima definiti, sotto l'attributo *Transform/Compositing Passes*, *Transform Passes* aggiungiamo col bottone "+" un nuovo elemento al vettore e dal menù a tendina selezioniamo *Custom Material Pass*. Cliccando col tasto destro del mouse sulla nuova voce che comparirà chiamata *Material* selezioniamo *create new material*. Si aprirà quindi una nuova finestra per lo *shading* [3] dei materiali il cui output è il materiale appena creato. La figura 6.11 mostra il grafico del *Material* finalizzato. Di seguito la procedura per la sua implementazione:

1. Istanziare 3 **Texture Parameters 2D** e nominarli rispettivamente con le diciture utilizzate per i 3 layers della nostra composizione.

[3] In computer grafica, lo *shading* si riferisce al processo di alterazione del colore di un oggetto o una superficie o un poligono di una scena tridimensionale. Esso si basa su diversi aspetti come l'incidenza della luce sull'oggetto, la normale delle sue facce rispetto ad essa ed altro ancora.

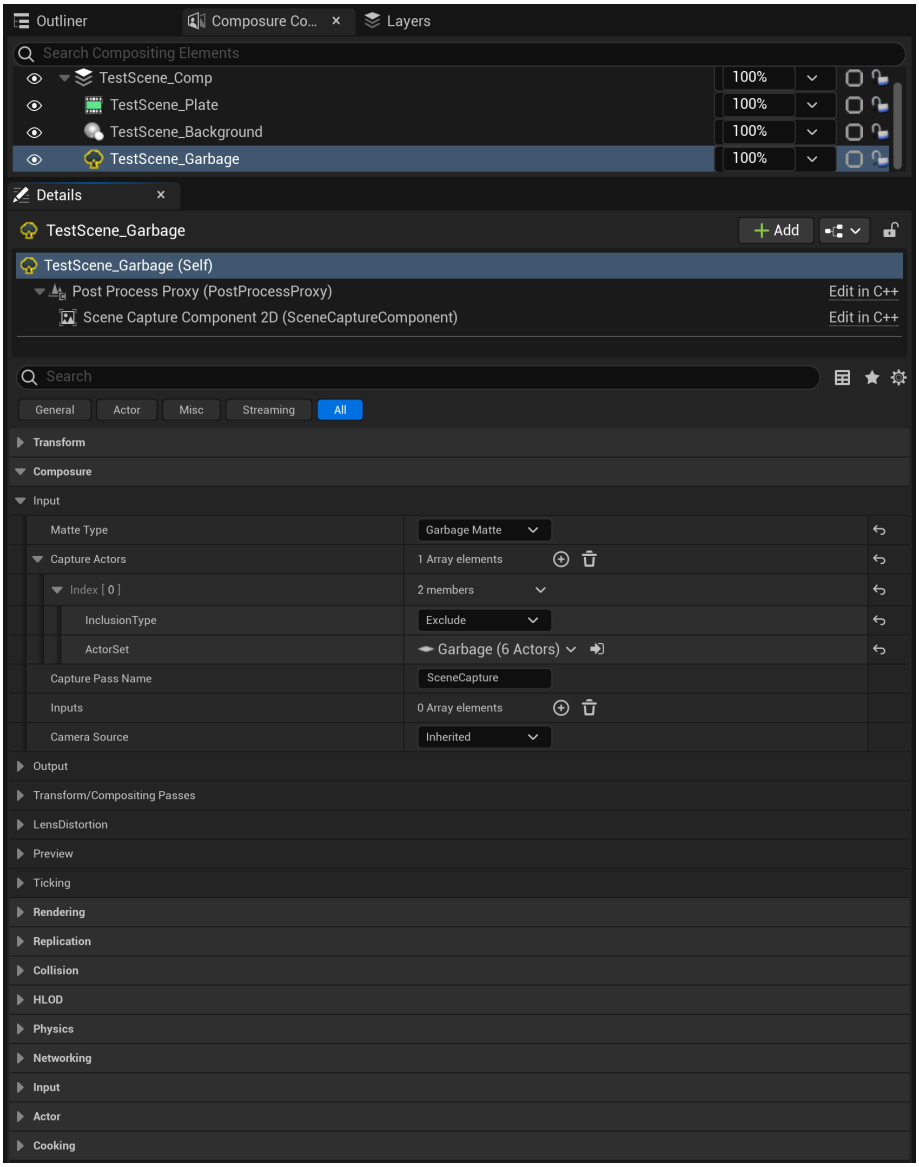

Figura 6.10. Variabili esposte e modificabili dell'attributo *Composure - Input* del layer *CG Matte*.

2. Utilizzare l'operatore **IN** per comporre il *Media Plate* ed il *CG Matte*. Questo operatore ritorna in uscita la porzione dell'immagine di A contenuta nell'immagine B.

3. Utilizzare l'operatore **OVER** per comporre l'immagine di foreground appena ottenuta con il *CG Layer* che funge da background. Questo operatore ritorna in uscita l'immagina A sovraimpressa all'immagine B.

4. Collegare la composizione ottenuta alla componente **Emissive Color** dello shader del materiale che stiamo editando.

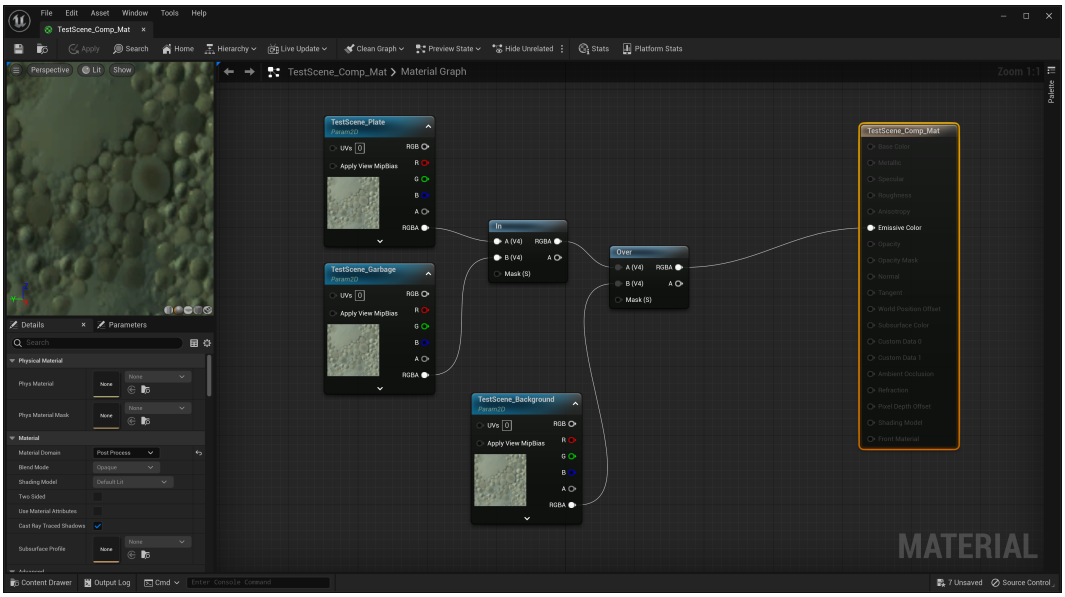

Figura 6.11. Dimostrazione del *blueprint* per un materiale che esprima la procedura attraverso cui dev'essere eseguito il *compositing*.

La procedura di compositing è così completata ed è quindi ora possibile visualizzare un'anteprima dell'immagine finale già compositata in tempo reale. Inoltre il nostro sistema si adatta autonomamente ai movimenti della camera, perciò avremo per ogni movimento un mascherino pulito e correttamente eseguito, e due immagine di foregorund e di background perfettamente concordi tra loro senza errori di parallasse.

Capitolo 7

DMX Lighting

7.1 Comsole DMX

Come visto nel capitolo 2 la soluzione ottimale per la gestione live delle apparecchiature d'illuminazione sia fisiche che virtuali è utilizzare una console per la generazioni di comandi DMX. La console che utilizza il protocollo *Art-Net* ha un indirizzo IP che deve risultare nella *subnet mask* [1] 255.0.0.0 ed avere come primo byte l'indirizzo 2 oppure 10. Utilizzando uno *switch ethernet* [2] portiamo un cavo ethernet dalla console ad esso, mentre ai suoi output colleghiamo due cavi ethernet che rispettivamente sono connessi al computer sul quale gira il game engine ed al *nodo Art-Net* per il controllo delle luci fisiche. In tal modo possiamo quindi assicurare la sincronizzazione di esecuzione dei comandi DMX essendo lo stesso pacchetto di dati inviato parallelamente su entrambi i canali.

Tuttavia andremo ad analizzare ora una soluzione alternativa non meno valida che è quella per cui si è optato durante il progetto di tesi non avendo a disposizione una console

[1] Una *subnet mask* è un numero a 32-bit creato impostando tutti i bit dell'*host* a 0 e tutti quelli della rete considerata a 1 (ottenendo perciò 255 in sistema decimale). In tal modo la *subnet mask* separa/maschera l'indirizzo IP in due parti: indirizzo di rete ed indirizzo di host.

[2] Lo *switch ethernet* serve a sdoppiare il cavo Ethernet o Lan e a gestire il traffico dei dati quando ci sono più nodi collegati, separando i cosiddetti domini di collisione connessi alle sue porte. Zanotti [2022]

DMX fisica. Per tale motivo si è invece adoperata la console virtuale *MagicQ* di *Chamsys*, un software con un'interfaccia che simula una console hardware e può essere lanciato in parallelo all'esecuzione di Unreal Engine.

7.1.1 MagicQ by Chamsys

MagicQ nella sua versione *demo* è un software gratuito il cui download può essere effettuato direttamente dal sito ufficiale di *Chamsys*. A differenza di quanto sopra descritto, in questo caso non sarà chiaramente necessario collegare la console al computer essendo il software già dentro esso. Bisognerà quindi limitarsi ad assicurarsi che il segnale generato venga reindirizzato all'interno del computer stesso oltre che al nodo *Art-Net*, e che tutti i dispositivi sia correttamente settati sui giusti indirizzi IP di ascolto. Vediamo nel dettaglio come configurare la console virtuale *MagicQ* e come eseguire il *patching* dei nostri dispositivi DMX.

Configurazione Impostazioni

Per configurare correttamente la scheda di connessione di rete di *MagicQ* e le principali impostazioni per i dispositivi hardware bisogna tenere conto di alcune importanti specifiche che regolamentano il protocollo *Art-Net* e seguire i seguenti passaggi:

1. **Configurazione interfaccia di rete ethernet**. Cliccando nel tab *Network* dal menù delle impostazioni generali sarà possibile individuare alla voce *IP address* l'indirizzo IP assegnato all'interfaccia di rete della console virtuale. Esso, come anticipato, sarà predisposto agli indirizzi

   $$2.x.x.x/10.x.x.x$$

   con subnet mask

   $$255.0.0.0$$

   poiché segue lo standard dettato dal protocollo *Art-Net* che è impostato di default nella console (figura 7.1). Bisogna quindi procedere aprendo la finestra delle impostazioni del nostro sistema operativo e nella sezione *connessioni di rete* modificare

l'indirizzo IPV4 [3] e la *subnet mask* inserendo quelle mostrato nelle impostazioni di *MagicQ*. Questo passaggio serve ad assicurarsi che il cavo ethernet che connette il nodo *Art-Net* al computer possa ricevere correttamente i dati dalla console virtuale.

Figura 7.1. Impostazioni *Network* della console virtuale *MagicQ*. [parte 1]

2. **Reindirizzamento interno**. Una volta configurata la connessione esterna bisogna verificare che i comandi DMX vengano correttamente riciclati all'interno del computer per poter essere letti da Unreal Engine. Per fare ciò, sempre nel tab*networks*, serve selezionare l'opzione *Normal + Loopback IP* alla voce *Net Host Options* (figura 7.2). Con questa configurazione la console virtuale non invierà i dati solamente all'indirizzo IP impostato ma anche a quello dedicato all'indirizzo

127.0.0.1

[3] IPv4 (*Internet Protocol Version 4*) è la quarta revisione dell'Internet Protocol. Il protocollo è descritto nell'*RFC 791*, pubblicato dalla *IETF* nel settembre 1981, ed è il più usato a livello di rete, poiché fa parte della suite di protocolli Internet. Wikipedia [a]

ovvero l'indirizzo di *local host* del calcolatore che fa riferimento al proprio server interno.

Figura 7.2. Impostazioni network della console virtuale *MagicQ*. [parte 2]

Inoltre è buona prassi controllare che il default della console virtuale sia correttamente impostato. Ovvero che la modalità di connessione sia impostata su *output* (tab *Ports* - *DMX Mode* in figura 7.3) e che il segnale in uscita sia continuo e non ad intermittenza impostandolo quindi su *continuos* (tab *Hardware* - *Reduced Rate Output* in figura 7.4).

Patch Heads

Configurata correttamente la console virtuale possiamo istanziare nel software i dispositivi d'illuminazione controllati con protocollo DMX utilizzati sul set in studio. Durante il progetto di tesi erano disponibili due teste generiche RGB a soli 3 canali. Negli esempi a seguire saranno però patchate delle fixture con anche il dimmer per l'intensità luminosa. Infatti come vedremo succesivamente nella trattazione di questo capitolo, in Unreal Engine non è possibile aumentare l'intensità luminosa delle fixture virtuali solo attraverso i canali dedicati al colore come avviene invece con le fixture fisiche a disposizione. Per questo

Figura 7.3. Impostazioni *Ports* della console virtuale *MagicQ*.

Figura 7.4. Impostazioni *Hardware* della console virtuale *MagicQ*.

motivo le fixture generate su Unreal Engine richiedono necessariamente un quarto canale per il controllo dell'intensità. La soluzione è stata perciò patchare delle fixture in modo del tutto simile anche sulla console virtuale *MagicQ* ed impostare invece gli indirizzi di ascolto delle fixture fisiche traslati di un byte in modo da interpretare solamente le informazioni dedicate al colore. Consapevoli delle ragioni che motivano l'esempio riportato possiamo ora analizzare il procedimento utile a configurare i dispositivi d'illuminazione (detti "teste") all'interno di *MagicQ*:

1. **Choose Heads**. Cliccando sul bottone *patch* si apre una nuova finestra dedicata ai dispositivi da inserire nello show DMX. Il tab *choose head* permette di scegliere il dispositiva che stiamo utilizzando nella realtà da una libreria di preset preinstallati nel software. La prima griglia offre una panoramica su diverse case produttrici di apparecchi d'illuminazione. Nel nostro caso sceglieremo la testa di tipo *Generic* ad esempio, non avendo un vero e proprio marchio le luci utilizzate (figura 7.5). Effettuata la scelta si apre una seconda griglia con una linea di prodotti della marca selezionata ed evidenziando ogni prodotto si possono visualizzare tutte le modalità possedute. Per i motivi elencati sopra per il progetto di tesi è stata scelta la voce *Dim + RGB* (figura 7.6).

2. **Patch It**. Cliccando su questo tab si apre una nuova finestra che permette di effettuare il *patching* del dispositivo scelto manualmente seguendo una specifica semantica (figura 7.7).

   - Il primo numero indica la quantità di dispositivi della tipologia e modalità scelta che vogliamo istanziare.
   - La chiocciola precede l'indirizzo a cui si vuole istanziare la prima fixture. L'indirizzo viene interpretato nel seguente modo:

     $$Universo - Canale$$

   - Se non vengono specificate altre indicazioni le fixture rimanenti vengono inserite nel primo indirizzo libero dello stesso universo.

Figura 7.5. Finestra dedicata alla scelta di *presets* di apparecchi d'illuminazione standardizzati. [parte 1]

Figura 7.6. Finestra dedicata alla scelta di *presets* di apparecchi d'illuminazione standardizzati. [parte 2]

- Se si vuole procedere al patching di una sola testa è possibile cliccare il tasto *full* pe riempire automaticamente il primo indirizzo libero.

Figura 7.7. Finestra dedicata al *patching* delle apparecchiature selezionate tramite *regular expressions*.

3. **View Heads**. Una volta effettuato il *patching* il software apre automaticamente questo tab da quale si può visualizzare una lista di resoconto di tutte le fixture patchate ed i loro attributi (figura 7.8).

4. **View DMX I/O**. Cliccando sul bottone *setup* è necessario infine tornare alle impostazioni principali di progetto ed abilitare da questo tab gli universi Art-Net utilizzati durante la fase di *patching*. È importante notare fin da ora che come si può apprezzare dalla figura 7.9, ogni universo della lista è in realtà impostato sull'universo *Art-Net* di indice precedente. Infatti l'hardware Art-Net parte dall'indicizzazione 0. Quest'osservazione tornerà utile nelle prossime sezioni di questo capitolo quando bisognerà assicurarsi che Unreal Engine e MagicQ dialoghino correttamente.

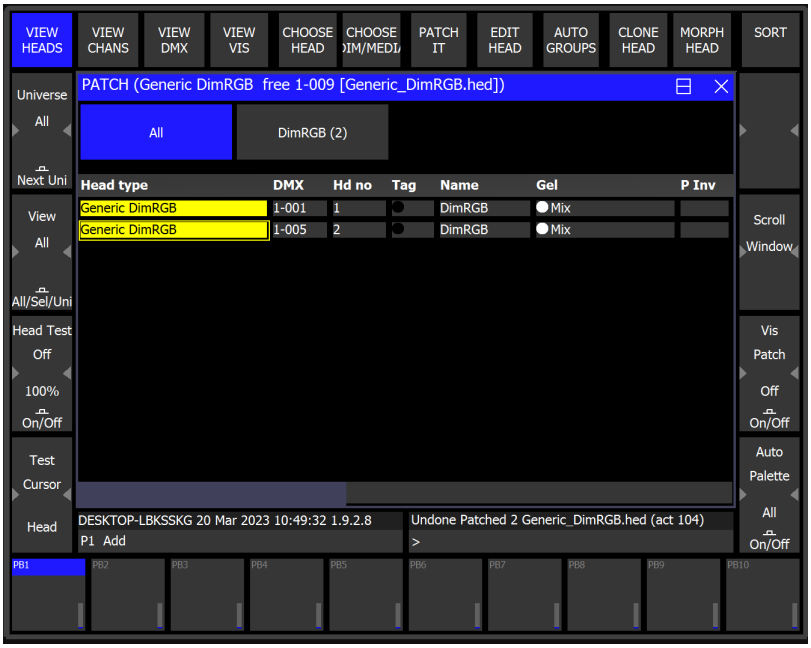

Figura 7.8. Dimostrazione di apparecchiature effettivamente "*patchate*".

Figura 7.9. Finestra per la gestione dello stato degli *universi* DMX.

Layers

Una volta impostato e creato il network per la trasmissione dei dati agli opportuni apparecchi utilizzati si possono inviare i comandi alle luci del set e del game engine. Per variare gli attributi dei dispositivi li si raggruppa in dei layers e poi si utilizzano i knobs e i faders della console. Le modifiche possono essere fatte in tempo reale attraverso le manopole oppure si può impostare la console in modalità *rec* di registrazione e comporre così il cosiddetto *show*. Nel progetto di tesi in analisi è stata presa in considerazione solamente la prima opzione che permette la gestione real-time delle luci sia fisiche che virtuali. Come riportano le figure sottostanti 7.10 e 7.11, le due teste sono state raggruppate nel layer 1, accessibile dal omonimo bottone. I parametri di colore RGB sono controllabili rispettivamente dalle manopole A, B e C sulla sinistra della console. L'intensità luminosa controllata dal dimmer va invece modificata dall'apposita finestra dei faders cliccando sul bottone *intensity*.

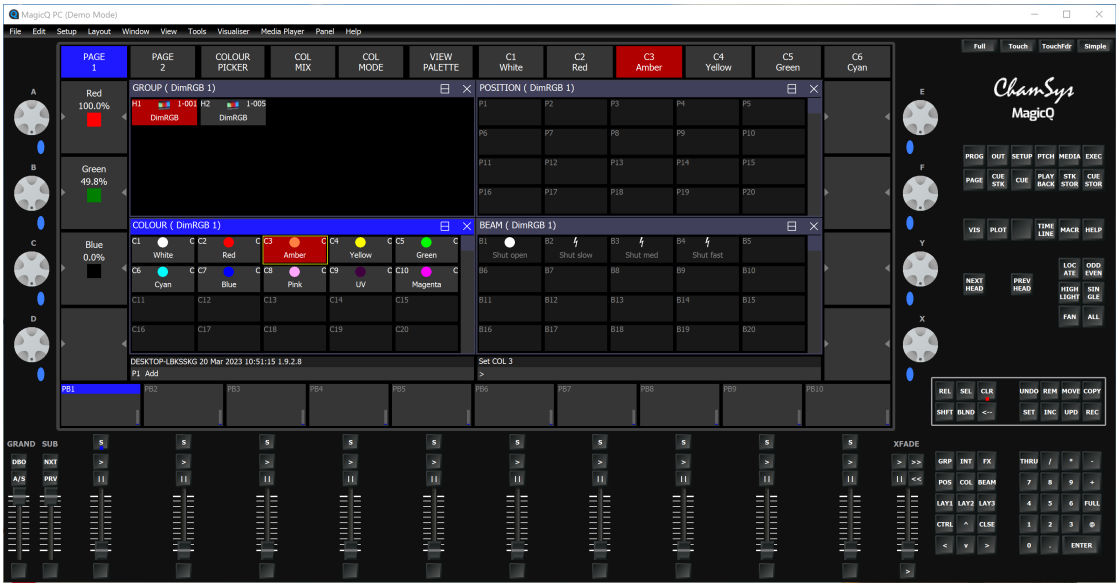

Figura 7.10. Finestra per la visualizzazione di layers contenenti apparecchiature d'illuminazione. Sezione dedicata al controllo cromatico.

Si possono modificare i parametri di una testa alla volta, oppure selezionandole entrambe si possono modificare i parametri contemporaneamente a patto che si voglia ottenere

Figura 7.11. Finestra per la visualizzazione di layers contenenti apparecchiature d'illuminazione. Sezione dedicata al controllo dell'intensità luminosa.

il medesimo risultato.

7.2 DMX Tool

Unreal Engine offre un tool dedicato alla gestione delle comunicazioni DMX in ingresso ed uscita dal software. Come mostra la figura 7.12, i plugins da attivare per poter procedere con la prossima trattazione sono i seguenti:

- **DMX Protocol**, implementa il protocollo DMX all'interno di Unreal Engine.
- **DMX Engine**, rende disponibili assets e funzionalità per la comunicazione con dispositivi abilitati con protocollo Digital Multiplex (DMX).
- **DMX Fixture**, rende disponili alcuni blueprints di modelli di fixture comuni.

Dai tab in altro a sinistra dell'interfaccia di Unreal Engine è inoltre disponibile il tab DMX. Cliccandovi sopra si apre una finestra da quale attivare o meno le interfacce che compongono il tool (figura 7.13):

- **Open Channel Monitor**, apre una finestra dalla quale selezionando un universo si apre una vista di tutti i 512 canali con i dati ricevuti su ognuno.
- **Open Activity Monitor**, simile al precedente ma apre una finestra in cui è possibile vedere i soli canali su cui il software sta ricevendo e leggendo dati.

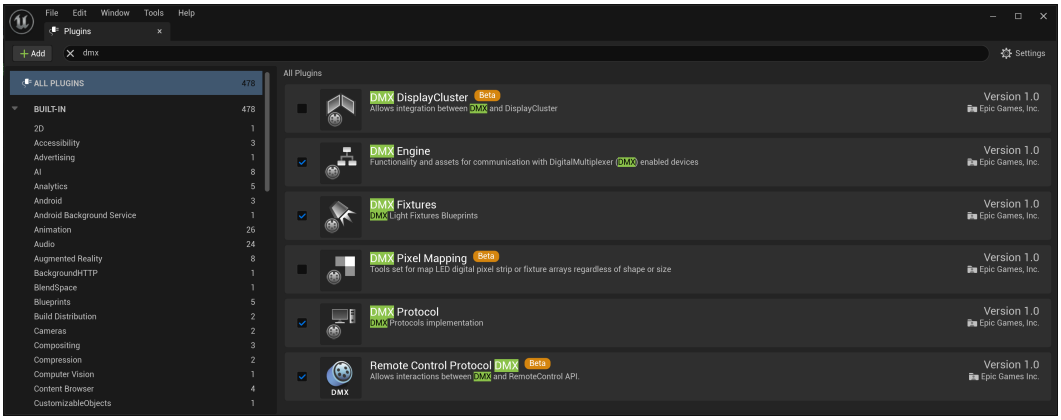

Figura 7.12. Plugins disponibili/da attivare su *Unreal Engine* per la gestione efficiente delle *Fixture DMX*.

- **Open Output Console**, apre una finestra dal quale è possibile generare dei faders associati a indirizzi specifici di uno o più universi per la modifica dei valori assunti quando si è in modalità *DMX Output*.
- **Open Patch Tool**, apre la finestra di patching che permette la gestione visivo/ manuale dell'indirizzamento delle fixture.
- **Send/Receive DMX**, permettere di impostare il software in modalità solo di emissione datti, solo di ricezione oppure entrambe.

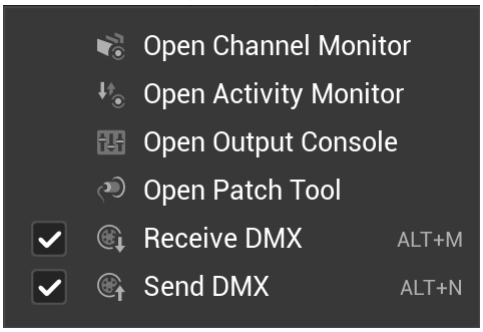

Figura 7.13. Menù principale del tool *DMX*.

Vediamo quindi nello specifico come istanziare delle nuove fixture su Unreal Engine tramite il tool DMX.

7.2.1 DMX Library

Innanzitutto bisogna provvedere a creare una *DMX Library* cliccando il tasto destro del mouse all'interno del *content drawer* e selezionando *DMX - DMX Library*. Questo è l'asset alla base del sistema di gestione della comunicazione DMX su Unreal Engine poiché permette di creare diverse librerie concettualmente simili a quelle preinstallate in *MagicQ* contenenti specifici dispositivi che debbono essere configurati manualmente dall'utente stesso. Inoltre permette anche l'operazione di patching dei dispostivi che si referenziano alle fixture configurate all'interno della libreria e la configurazione di impostazioni di progetto dedicate alla libreria stessa. Analizziamo quindi una ad una le interfacce messe a disposizione da questo asset.

DMX Project Settings

Cliccando sull'asset *DMX Library*, si apre una nuova finestra contenente tre tab. Il primo è quello dedicato ai *Library Settings* e mostra un resoconto delle impostazioni di input ed output DMX per quanto riguarda le configurazioni di rete. Per modificare le impostazioni di default e personalizzarle si può premere sul bottone verde in alto a sinistra *Project Settings* che contiene un link alla pagina *Plugins - DMX* delle impostazioni di progetto generali di Unreal Engine. Come mostrato nella figura 7.14, da tale pagina si possono modificare le impostazioni di comunicazione della rete DMX. Possiamo istanziare una o più nuove porte di ingresso e/o di uscita. Nel caso del setup utilizzato per il progetto di tesi, non è necessario che Unreal Engine generi segnali DMX, in quanto si pone in una posizione di mero ascoltatore. Una porta di ingresso affinché possa dialogare con una console virtuale impostata come quella sopra descritta deve avere le seguenti caratteristiche:

- **Protocol Name** dev'essere impostato su *Art-Net*.
- **Network Interface Card IP Address** dev'essere impostato su 127.0.0.1 . Infatti come visto sopra questo è l'indirizzo di *local host* dove viene rigirato il segnale in uscita dalla console virtuale all'interno del computer stesso.
- **Local Universe Start** settato su 1. Poiché l'indicizzazione degli universi su Unreal Engine parte da 1.

- **Extern Universe Start** settato su 0. Abbiamo infatti già notato prima che nella finestra di gestione degli universi DMX di *MagicQ* l'universo 1 era associato all'indirizzo 0 su Art-Net. Torna utile ora come dato affinché possa essere effettuata la corretta comunicazione degli universi tra la console virtuale e Unreal Engine.
- **All Fixture Patches receive DMX in Editor** dev'essere flaggato affinché i blueprint e gli actors nel campo virtuale possano ricevere i segnali e funzionare correttamente in tempo reale.

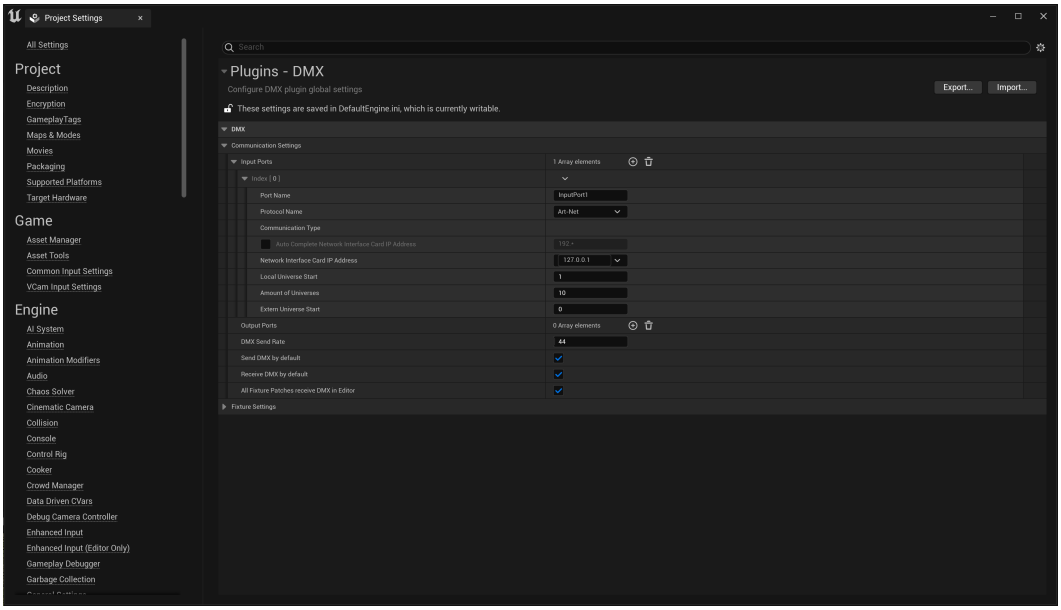

Figura 7.14. Finestra *Library Settings* per la configurazione delle porte di ingresso e uscita per la comunicazione di dati con protocollo DMX.

Fixture Types

Una volta configurate opportunamente le impostazioni di progetto si può passare alla seconda finestra *Fixture Types*. Essa serve per istanziare nuovo dispostivi e creare una libreria di preset di fixture. Le fixture si possono sia aggiungere importando un opportuno file tramite il bottone *import* oppure si possono configurare manualmente. Vediamo come generare manualmente una nuova fixture che sia una controparte su questo software di quelle già create in *MagicQ* (figura 7.15):

1. Cliccare sul bottone **New Fixture Type** per istanziare un nuovo elemento e nominarlo.

2. In basso a sinistra impostare **DMX Category** su *static* in quanto non è una testa mobile.

3. Cliccare sul bottone **Add Mode** per definire una modalità di default della fixture. (Noi l'abbiamo denominata *mode*). Nel caso di apparecchiature complesse che possiedo più modalità possono essere create più *mode*, altrimenti ne basta una di default come in questo caso.

4. Cliccare sul bottone **Add Function** per definire un nuovo attributo della fixture e come essa interpreterà ed eseguirà i dati letti per un determinato canale. Come anticipato, su Unreal Engine l'intensità luminosa delle fixture virtuali vanno necessariamente controllate con l'attributo dimmer, motivo per cui era stato aggiunto anche su *MagicQ* nonostante non fosse presente nelle apparecchiature fisiche (dov'è stato bypassato). Per definire un attributo possiamo modificare il *Function Name* e scegliere l'attributo su cui mappare il canale dal menù a tendina che ci viene messo a disposizione. Per attributi particolari a più di 8 bit si può modificare l'impostazione da *Data Type* per fare in modo di occupare più di un canale. Infine può essere impostato un valore di default tramite *Default Value*.

Fixture Patch

Istanziate le tipologie di fixture necessarie non rimane che fare il *patching* delle fixture virtuali che corrispondono ai nostri dispositivi fisici. Tale deve risultare speculare a quello già fatto sulla console virtuale se vogliamo che tutto il sistema risponda correttamente. Per questo c'è a disposizione l'ultimo tab denominato *Fixture Patch* (figura 7.16). Clicchiamo sul bottone verde **Add Fixture** per assegnare una nuova fixture. In ordine possiamo definire:

- **Fixture Patch**, nome della fixture.
- **FID**, ID della fixture.

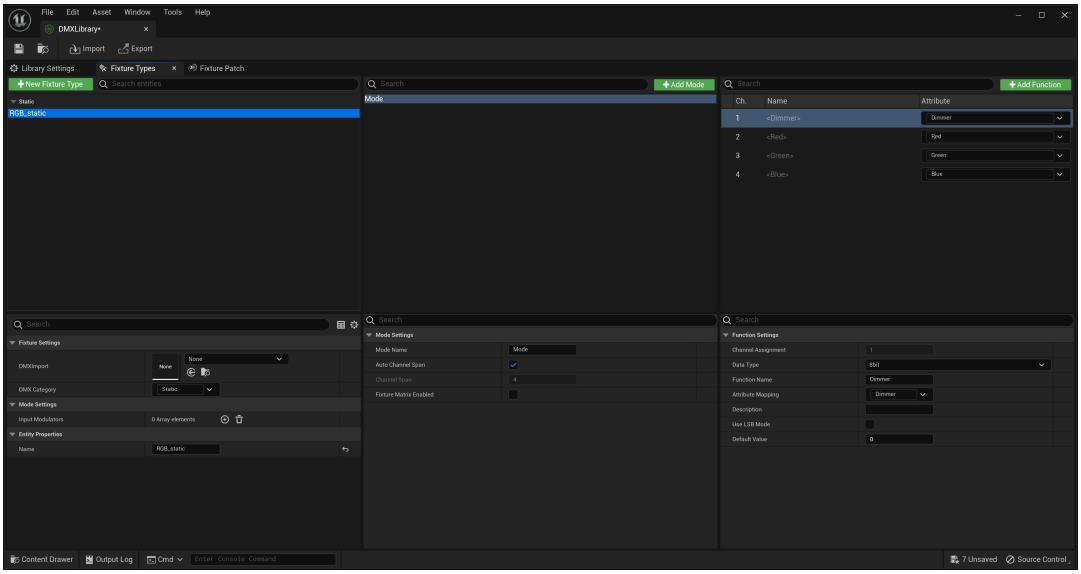

Figura 7.15. Finestra *Fixture Types* dedicata all'istanziazione di nuove tipologie di *fixture*, della loro *modalità* e delle *funzioni* in essa eseguibili.

- **Fixture Type**, tipologia della fixture tra quelle create nella pagina precedente.
- **Mode**, modalità utilizzata.
- **Patch**, indirizzo su cui viene posizionato il primo canale della fixture con la seguente semantica

  $$Universo.Canale$$

- **Editor Color**, colore con cui viene visualizzato il *patch* del dispositivo nell'editor.

Oltre alla configurazione standard, l'editor del tool DMX di Unreal Engine permette anche un posizionamento visivo delle fixture con la possibilità di usare la modalità *drag and drop* del patching. Basterà cliccare sulle forme colorate che occupano porzioni dell'universo e spostare a proprio piacimento dove si preferisce. Automaticamente verranno riaggiornati i canali.

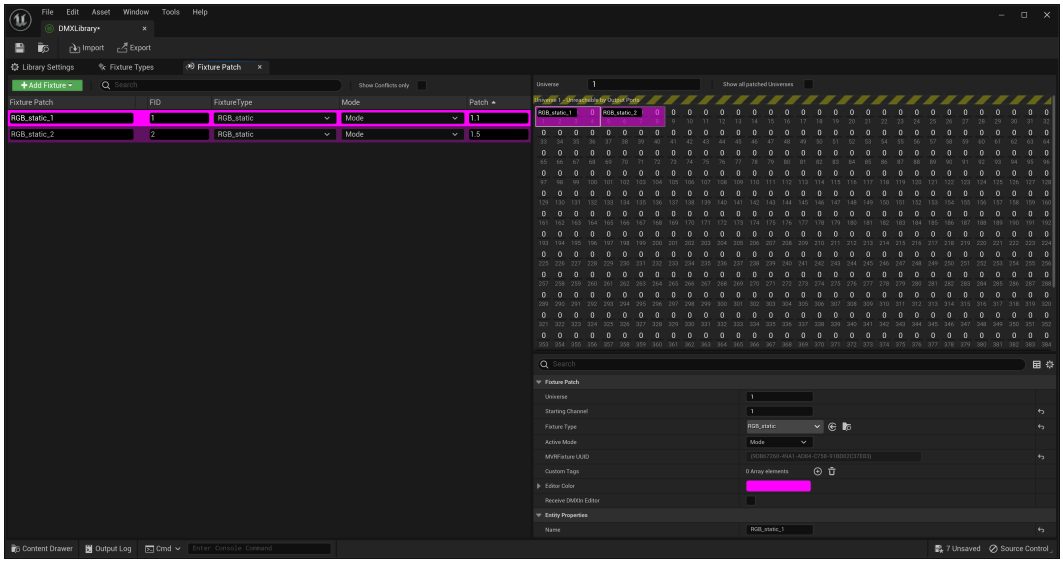

Figura 7.16. Finestra *FixturePatch* che offre una soluzione visuale di tipo *drag and drop* per l'implementazione del *patching* sugli indirizzi interessati.

7.2.2 DMX Fixture

Grazie al plugin *DMX Fixture* nella cartella *Engine - Contents* troviamo una serie di blueprints di actors utilizzabili per i nostri show DMX. Prendiamo in esame il più semplice di tutti, ovvero il blueprint *StaticHead* che è un'ottima interfaccia per i dispositivi analizzati finora e tutti quei dispositivi DMX a testa fissa. Per istanziare una copia basterà utilizzare l'opzione di *drag and drop* nella scena. Nello specifico:

- Il blueprint è formato da tre elementi principali che sono organizzati con legami di parentela padre-figlio uno con l'altro. In ordine: *Base* origine del sistema di coordinate, *Yoke* l'apparecchiatura sulla quale monta la testa (in questo caso fissa), *Head* la testa del dispositivo d'illuminazione.
- La *Head* che è il centro dell'oggetto è quindi così strutturata:
  - Tre *mesh* [4] la compogono: *StaticMeshHead*, la testa esterna, *StaticHeadLens*, la lente della lampada, *StaticMeshBeam*, il fascio luminoso generato.

[4]Il termine *mesh* dall'inglese significa maglia o rete. In questo caso ci si riferisce alle mesh poligonali, ovvero reticoli formati da vertici, spigoli e facce generati in computer grafica che servono nella

- Una *Point Light* serve a simulare l'hotspot luminoso all'interno della lente quando la fixture viene accesa.
- Una *Spot Light* serve a simulare il fascio luminoso generato ed è questo il componente ad essere controllato via DMX nel caso della nostra analisi.
- Una *Occlusion Direction* per indirizzare il fascio luminoso solo delle direzioni opportune evitando che irraggi parti occluse dalla mesh.

Inoltre questo blueprint associa una serie di *components* all'actors che permettono di reindirizzare le comunicazioni DMX all'interno dell'actors e di eseguirle correttamente nel concreto. Questi componenti possono essere aggiunti o eliminati, nel nostro caso si sono utilizzati:

- *DMX*
- *Color Component*
- *Dimmer Component*

Dopo la panoramica generale sulla struttura del blueprint, vediamo brevemente come collegare un certo blueprint (quindi una fixture virtuale presente nella scena) ad il patching della rispettiva istanza virtuale.

Settings

Come mostra la figura 7.17, al menù dettagli clicchiamo su *BP_StaticHead* e apriamo la voce *DMX*. Alla destra dell'opzione *DMX Library* cliccando si apre un menù a tendina dal quale possiamo scegliere una delle librerie già create e memorizzate nel nostro software. Per collegare il blueprint ad una *patch* non ci resterà che selezionare dal menù a tendina dell'opzione *Fixture Patch* il nome con cui abbiamo salvato il dispositivo desiderato e controllare che sia presente il flag di fianco all'opzione *Receive DMX from Patch*. A questo punto è tutto correttamente configurato e tutte le variazioni che andremo ad apportare

modellazione digitale di oggetti tridimensionali.

dalla nostra console appariranno in tempo reale anche sull'oggetto istanziato per mezzo di questo blueprint.

7.3 Apparecchiatura Fisica

Analizzato il workflow per l'implementazione delle comunicazioni DMX in software, proseguiamo ora con quella dell'hardware utilizzato in fase di test del framework realizzato. Come già anticipato sono state utilizzati due dispositivi a testa fissa di tipologia generica a led RGB e un nodo Art-Net. Di seguito viene dimostrato come realizzare una corretta rete di connessione tra i dispositivi.

1. Collegare il computer ad un'interfaccia che funzioni con protocollo Art-Net opportunamente alimentata. Abbiamo utilizzato il dispositivo *LightMouse 3R2* di *Equipson*, un nodo Art-Net con 2 universi DMX disponibili in output e un input per cavo ethernet *RJ-45* (figura 7.18). Nel nostro caso è stato utilizzato un solo universo e perciò un solo cavo XLR in uscita. Essendo l'output del nodo a 5 pin, è stato inoltre utilizzato un adattatore per il passaggio da XLR 5 pin a XLR 3 pin.

2. Collegare il cavo XLR uscente dal nodo Art-Net all'interfaccia di input del primo dispositivo RGB della *daisy-chain* (già opportunamente alimentato). Il dispositivo va impostato sul rispettivo indirizzo di partenza su cui è stato allocato in fase di *patching* * (figura 7.19).

3. Con un altro cavo XLR collegare l'uscita del primo dispositivo della *daisy-chain* al secondo (già opportunamente alimentato). Anche in questo caso il dispositivo va impostato sul rispettivo indirizzo di partenza su cui è stato allocato in fase di patching (figura 7.20).

*Per i motivi sopra spiegati, nel nostro caso è stato necessario settarlo sul secondo indirizzo occupato per leggere solamente i dati relativi al colore e non al *dimmer*.

È così conclusa la dissertazione dell'implementazione di un sistema DMX per il controllo *real-time* dei dispositivi d'illuminazione sia reali che virtuali su un set di Virtual Production per produzioni indipendenti.

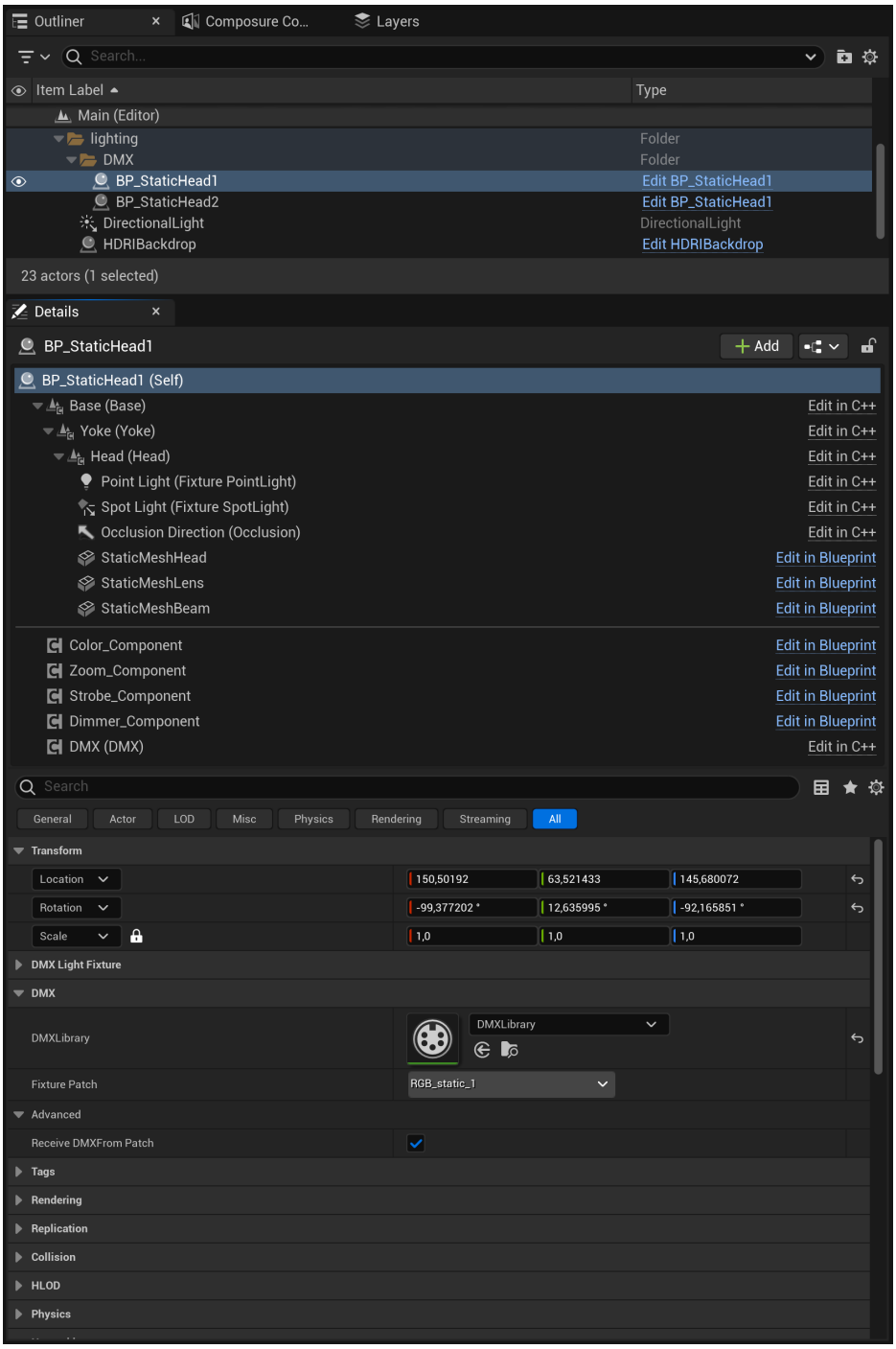

Figura 7.17. Pannello details dell'actor *Blueprint StaticHead* in riferimento alla sua struttura interna ed alla locazione dove indicare la *DMX Library* di riferimento e la rispettiva *fixture*.

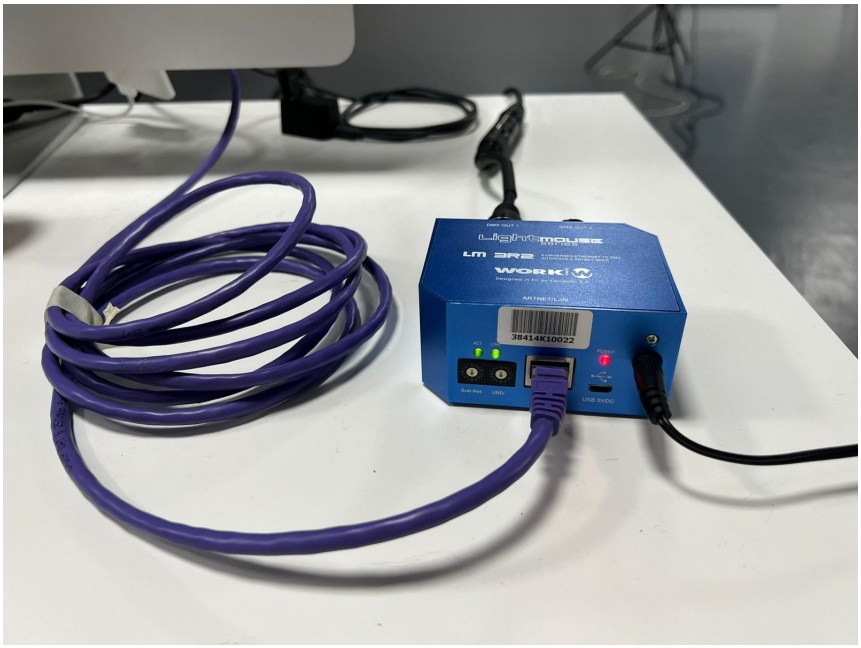

Figura 7.18. Nodo Artnet *LightMouse 3R2* di *Equipson*. Un ingresso per l'alimentazione e un'interfaccia *RJ-45* per connessione *ethernet*. Due uscite XLR (con 5 pins).

Figura 7.19. Primo dispositivo d'illuminazione della catena *daisy-chain* settato sull'universo 1, indirizzo 2. In input il cavo proveniente da nodo *Art Net* mentre in output quello indirizzato alla fixture successiva della catena.

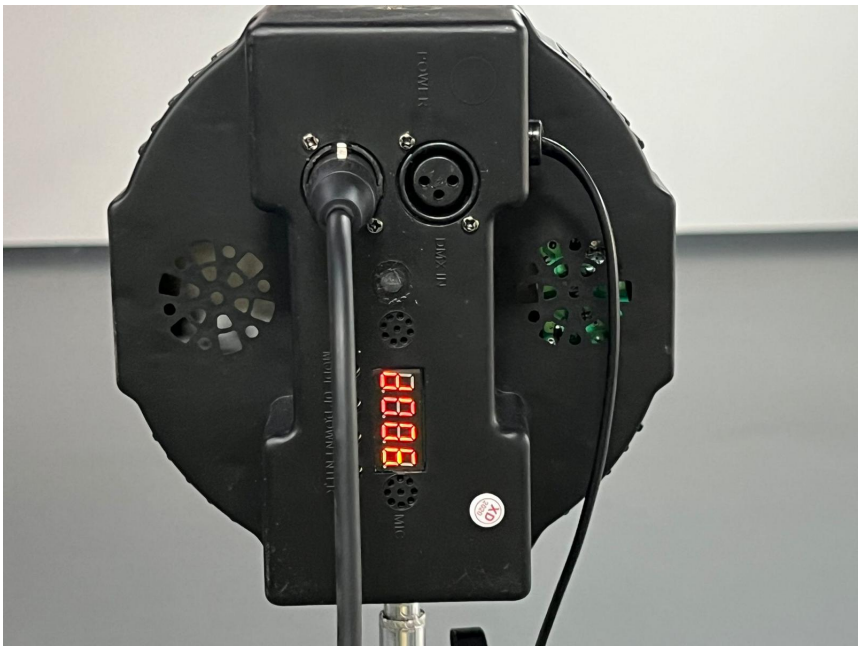

Figura 7.20. Ultimo dispositivo d'illuminazione della catena *daisy-chain* settato sull'universo 1, indirizzo 6. In input il cavo proveniente dalla fixture precedente della catena.

Capitolo 8

Conclusioni

8.1 Limiti e Opportunità

Infine si vorrebbe concludere delineando limiti ed opportunità delle tecnologie proposte nell'implementazione del progetto di tesi. È infatti evidente come i risultati video ottenuti grazie al framework realizzato non siano ottimali sotto l'aspetto di precisione e accuratezza derivanti dal tracciamento. Il progetto di tesi è infatti stato realizzato con gli strumenti a disposizione del sottoscritto in forma di proprietà personale, di Robin Studio e del Visionary Lab del Politecnico di Torino. Sebbene il framework sia dedicato alle piccole produzioni cinematografiche indipendenti, ciò non implica che il sistema non possa essere ottimizzato con un investimento maggiore rispetto al cosiddetto "minimo indispensabile". In particolare la qualità hardware influisce in modo diretto sulla qualità del risultato senza dipendere dall'implementazione del framework stesso. Un maggiore investimento sull'hardware contribuisce necessariamente ad un miglioramento delle prestazioni e dell'efficienza degli algoritmi eseguiti dal game engine. Avere ad esempio una console DMX hardware permette una migliore reattività nel controllo delle fixture DMX oppure disporre del sistema di tracking HTC Vive Mars permette l'aumento di precisione nell'elaborazione della posizione del tracker grazie al cablaggio delle connessioni, risultando in una maggiore stabilità e quindi credibilità dell'immagine finale. Vedremo perciò di seguito il limite più grande del framework proposto e le motivazioni dietro ai limiti della

tecnologia di tracciamento appunto, oltre che alle soluzioni. Successivamente passeremo in rassegna alcuni tool forniti da Unreal Engine non considerati e tecniche supplementari o alternative che possano migliorare l'efficienza del framework proposto.

8.1.1 Precisione e Accuratezza

Precisione e accuratezza sono due dati molto importanti per la comprensione dell'efficienza e della robustezza di un sistema di tracciamento. L'accuratezza è quanto la misurazione empirica di un dato si distanzia da quella vera/assoluta di un sistema ideale. La precisione invece riguarda le misurazioni effettuate e indica quanto ci possiamo aspettare che due misurazioni siano identiche. La figura 8.1 dimostra bene il significato di queste due metriche ed il loro impatto sui risultati attesi o analizzati.

Figura 8.1. Schema dimostrativo del significato delle metriche di precisione e accuratezza

Verso la fine del 2022 un gruppo di ricercatori coordinato da Kuhlmann de Canaviri et al. [2023] ha condotto un interessante studio riguardo all'accuratezza delle misurazioni di tracciamento effettuate dal sistema SteamVR confrontando i risultati con un setup a 2 base station e a 4 base station. Sono state valutate 3 diverse analisi:

- Traslazione e Rotazione Statica
- Traslazione Dinamica

- Robustezza del Sistema all'Occlusione

Quelle che più interessano l'ambito di questo progetto sono le prime due analisi. La prima è stata condotta fissando un punto di partenza e di fine dello spostamento del Vive tracker 3.0 in modo tale che essi fossero statici rispetto alle base station. Sono quindi state effettuate delle traslazioni di ± 10 cm lungo gli assi X, Y e Z e delle rotazioni di ± 30° intorno agli assi X e Y e ± 60° intorno all'asse Z. Per l'esperimento riguardante la traslazione dinamica si è invece ricreato un moto a spirale conica. I movimenti di precisione sono stati effettuati per mezzo di un braccio robotico Universal Robots UR10 CB-Series. Per la misura dell'accuratezza della posizione rilevata dal sistema SteamVR rispetto a quella realmente conosciuta tramite l'utilizzo del braccio robotico è stata usata la metrica RMSE [1] mentre per la precisisione si è usata la deviazione standard delle diverse misurazioni effettuate. La figure 8.2, 8.3 e 8.4 riportano i risultati ottenuti dalla ricerca di Kuhlmann de Canaviri et al. [2023] presa in analisi in merito alla qualità di tracciamento del sistema SteamVR comparando due diversi setup (due e quattro base station).

RMSE	Axis	Two Base Stations mm (*sd*)	Four Base Stations mm (*sd*)
Movement direction	X	0.393 (0.363)	**0.085** (0.083)
	Y	0.345 (0.347)	**0.199** (0.117)
	Z	0.551 (0.499)	**0.158** (0.112)
	All	0.429 (0.403)	**0.155** (0.104)

Figura 8.2. Errori (mm) nel tracciamento della traslazione statica, accompagnati da deviazione standard (sd) delle misurazioni. Cinque misurazioni effettuate per ogni movimento.

Come evidenziano i risultati di questa ricerca, il sistema SteamVR 2.0 ha un'elevata accuratezza del tracciamento, sia nello studio della traslazione che della rotazione, con

[1] La metrica RMSE (Root Mean Square Error) è data dall'equazione

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(p_{r_i} - p_{t_i})^2} \tag{8.1}$$

Dove N è il numero dei campioni e $(p_r - p_t)$ è la differenza tra la posizione definita dal braccio robotico (p_r) e la posizione misurata dal tracciamento SteamVR (p_t).

RMSE	Axis	Two Base Stations ° (*sd*)	Four Base Stations ° (*sd*)
Movement direction	X	0.827 (0.029)	**0.375** [a] (0.297) [a]
	Y	0.474 (0.126)	**0.086** (0.066)
	Z	0.125 (0.053)	**0.084** (0.076)
	All	0.574 (0.069)	**0.227** [a] (0.146) [a]

Figura 8.3. Errori (°) nel tracciamento della rotazione statica, accompagnati da deviazione standard (sd) delle misurazioni. Cinque misurazioni effettuate per ogni movimento.

RMSE	Axis	Two Base Stations Slow mm (*sd*)	Two Base Stations Medium mm (*sd*)	Two Base Stations Fast mm (*sd*)	Four Base Stations Slow mm (*sd*)	Four Base Stations Medium mm (*sd*)	Four Base Stations Fast mm (*sd*)
Movement direction	X	6.130 (5.872)	11.320 (9.215)	14.377 (15.217)	**0.316** (0.444)	2.020 (1.998)	5.537 (5.421)
	Y	0.899 (0.912)	1.990 (2.146)	4.727 (5.231)	**0.637** (0.513)	1.994 (1.876)	2.144 (1.965)
	Z	1.165 (1.231)	13.910 (12.731)	15.432 (15.144)	**0.823** (0.781)	2.508 (2.322)	4.376 (4.451)
	All	2.824 (2.672)	8.664 (8.031)	11.091 (11.864)	**0.574** (0.579)	2.164 (2.065)	4.237 (3.946)

Figura 8.4. Errori (mm) nel tracciamento della traslazione dinamica durante il movimento a spirale conica eseguito dal braccio robotico a tre diverse velocità: 12.5 mm/s (slow), 25 mm/s (medium), and 50 mm/s (fast)

errori nei campi submillimetrico e del subgrado. In particolare nel caso di quattro stazioni base si riescono ad ottenere risultati con errori all'unità del decimo di millimetro e al decimo di grado registrando miglioramenti fino a tre volte più ottimali nella traslazione statica e fino a otto volte nella rotazione. Riguardo alla traslazione dinamica i risultati sono buoni, sebbene siano particolarmente influenzati dalla velocità del moto. Nel caso con setup a due stazioni base si ottengono scarsi risultati per la modalità medium e fast infatti (le più utilizzate nel cinema rispetto ai 12.5 mm/s della modalità slow) mentre con il setup a 4 stazioni base l'accuratezza e la precisione dei risultati migliorano di 3 volte ottenendo ottimi e nettamente migliori anche nelle modalità medium e fast. La figura 8.5 aiuta a visualizzare meglio l'impatto e l'importanza di questi dati nell'approccio pratico alla tecnologia SteamVR. Utilizzare un setup a quattro stazioni base (con un moderato investimento aggiuntivo rispetto al setup con due) può migliorare visivamente la qualità del tracciamento del Vive tracker 3.0 delineando traiettorie molto più pulite e realistiche.

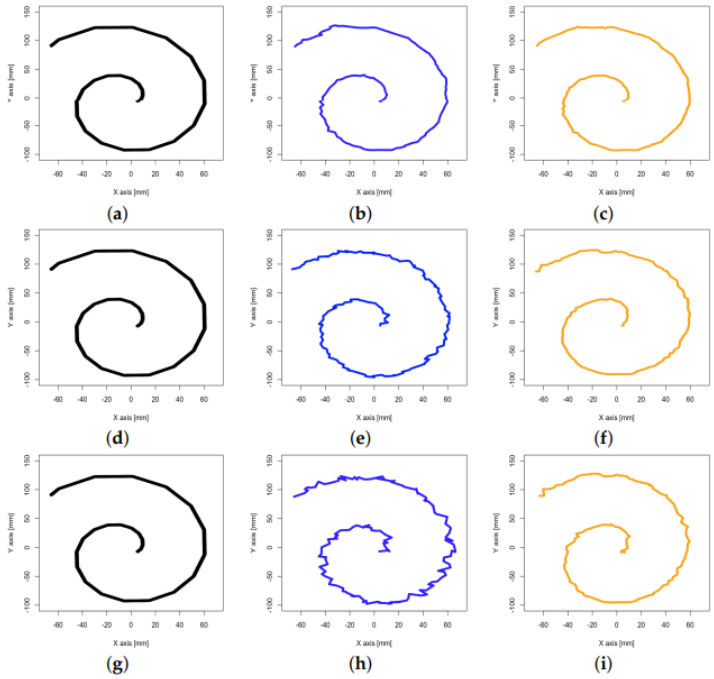

Figura 8.5. Traiettoria media della spirale alle velocità 12.5 mm/s (slow - prima riga), 25 mm/s (medium - seconda riga), and 50 mm/s (fast - terza riga). Per la traiettoria descritta dal braccio robotico Universal Robots UR10 è stato usato il colore nero mentre per i setup con due base station il blu e con quattro base station l'arancione.

8.1.2 HTC Vive Mars

Questa tecnologia presentata per la prima volta nella primavera del 2022, integra e ottimizza il già esistente sistema SteamVR in un pacchetto hardware dedicato alle virtual production. È proprio la sua specificità a renderlo il candidato ottimale per le produzioni indipendenti che possano permetterselo. Il suo costo da listino è 5.000€ e può essere acquistato direttamente dal sito ufficiale HTC Vive. Partendo dal setup ideale con 4 base station, integra il sistema di tracciamento già esistente con il cosiddetto *Mars*, un hub per la gestione dei flussi di dati tra i tracker ed il computer, e i *Rover*, dispositivi di interfaccia tra il tracker o la lente e l'hub centrale. È proprio grazie a questi due nuovi elementi che viene permesso un azzeramento del delay di comunicazione dei dati di tracciamento grazie al cablaggio ethernet e importanti e il controllo di timecode e genlock. Vediamo uno per uno gli elementi integrativi che compongono questa suite e quali sono i vantaggi annessi

facendo riferimento alla stessa guida messa a disposizione da HTC [a].

Rover

I *Rover* (figura 8.6) inviano i dati di tracciamento dal *VIVE Tracker 3.0* al *Mars*. I dati di tracciamento dopo essere stati raccolti non vengono memorizzati ma subito trasmessi in modo da poter prevenire la degradazione del segnale grazie a un continuo *refresh*. Il *Rover* è inoltre responsabile per la trasmissione dei dati FIZ (visti al capitolo 5) che un codificatore lenti può convertire in dati che possono essere usati appunto dal game engine. Nello specifico *HTC Vive Mars* supporta i codificatori lenti della linea *LOLED Indiemark*.

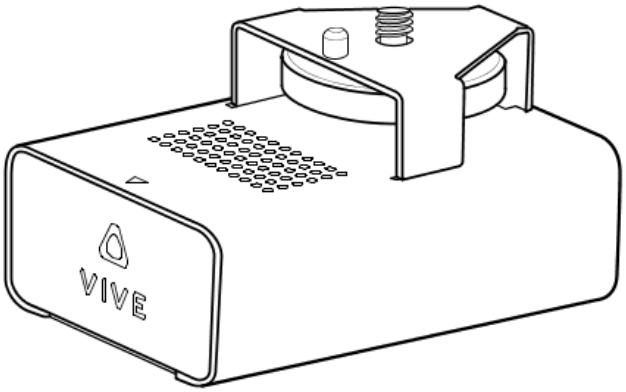

Figura 8.6. *Rover* dell'hardware *HTC VIVE Mars* per la Virtual Production.

Come mostra la figura 8.7, un *Rover* dispone delle seguenti interfacce input/output:

1. Porta di connessione primaria per *VIVE Tracker 3.0* (USB Type-A).
2. Porta attraverso cui possono essere trasmessi i dati relativi allo *Zoom* (USB Type-A).
3. Porta attraverso cui possono essere trasmessi i dati relativi al *Focus* (USB Type-A).
4. Porta attraverso cui possono essere trasmessi i dati relativi all'*Iris* (USB Type-A).
5. Porta per la trasmissione di dati dal *Rover* al *Mars* (Ethernet).

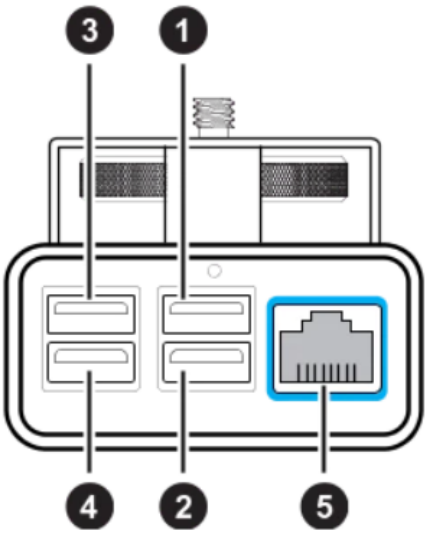

Figura 8.7. Schema delle interfacce input e output di un *Rover*.

Mars

Il *Mars* (figura 8.8) raccoglie i dati di tracciamento provenienti da ciascun *Rover* (massimo 3 per ogni hub) e li invia al computer sul quale gira il game engine tramite un indirizzo IP e un router. Il *Mars* permette di gestire ogni dispositivo connesso e controllare rilevanti informazioni grazie alla sua *dashboard touchscreen* posta sulla faccia superiore del hub. Grazie alla dashboard è possibile modificare ed impostare l'indirizzo IP e le impostazioni di connessione di rete, visualizzare dati come il timecode, il genlock ed il numero di dispositivi attivi, resettare le coordinate del tracker ad una posizione originale.

Come mostra la figura 8.9, un *Mars* dispone delle seguenti interfacce input/output:

1. Tasto di accensione, per spegnere o accendere il *Mars*.
2. Porta di alimentazione del *Mars* (DC input).
3. Porta di connessione al router IP del PC (Ethernet).
4. Porta per la ricezione di dati da un generatore di timecode (ad esempio una cinepresa).

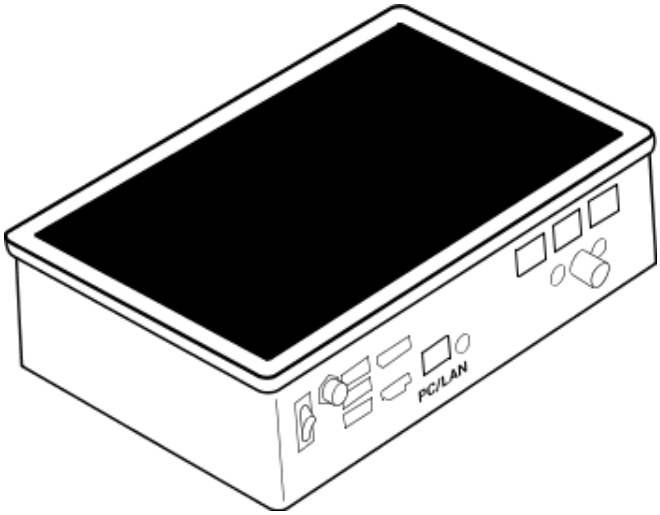

Figura 8.8. Hub hardware *HTC VIVE Mars* per la Virtual Production.

5. Porta per ricevere i dati trasmessi dai *Rover* (Ethernet).

6. Porta per ricevere un segnale di *synch* del timecode da parte di un generatore genlock (necessita un dispositivo specifico che attui questo ruolo).

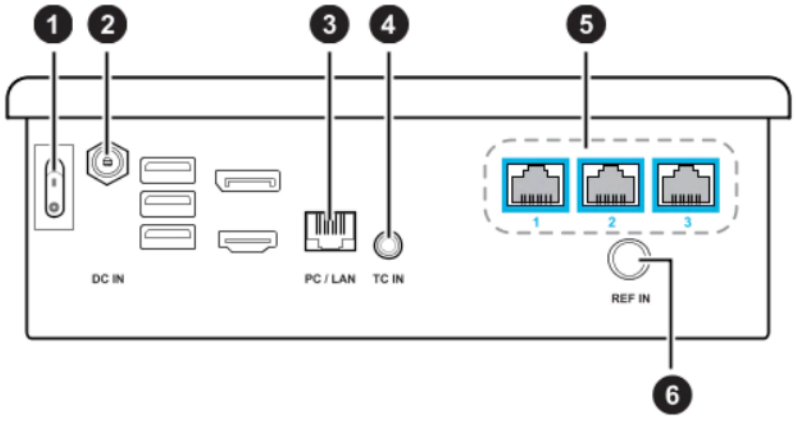

Figura 8.9. Schema delle interfacce input e output di un *Mars*.

Calibration Kit

Il kit di calibrazione (figura 8.10) include alcuni oggetti per assemblare una *calibration board* che verrà posta all'interno dell'area di tracciamento durante la fase di calibrazione della camera. Il kit contiene una *calibration board* di materiale acrilico con il classico pattern a scacchiera, una base di supporto per la tavola ed un foro filettato per il fissaggio di un modulo *Rover* e un *VIVE Tracker 3.0* che verrà utilizzato per determinare la posizione del *calibration target*. Per effettuare la calibrazione è disponibile un'applicazione per PC chiamata *Camera Calibration Tool* dalla quale si può impostare l'indirizzo IP del *Mars* e la porta TCP/UDP e il device di acquisizione video (per esempio Camlink). Grazie al tool è quindi possibile effettuare una rapida calibrazione seguendo le istruzioni e salvare il dataset ottenuto importando successivamente nel game engine e abbreviando notevolmente la procedura precedentemente mostrata al capitolo 5. Infatti grazie al tracciamento della *checkerboard* otterremo sia i dati relativi al posizionamento reciproco di modo reale e virtuale (che ottenevamo grazie agli *Aruco Markers*) sia i dati relativi alla distorsione e focale della lente trovati grazie alla checkerboard stessa.

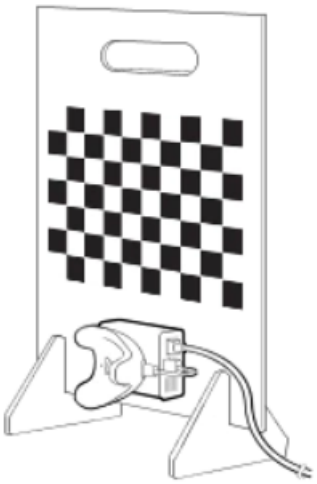

Figura 8.10. Kit a disposizione con l'hardware *HTC VIVE Mars* per la calibrazione della camera.

8.1.3 Altri Usi e Implementazioni

Per concludere infine la trattazione di questo progetto di tesi, si vuole passare in analisi quali potrebbero essere ulteriori utili supplementi al framework realizzato, in base alla disponibilità economica e agli interessi della singola casa di produzione cinematografica. Infatti il framework proposto funge da cornice basilare ad una produzione virtuale ed il suo set, realizzando i tasselli indispensabili o fondamentali ad una corretta realizzazione. Altresì le esigenze di ogni produzione sono diverse, come i campi artistici che si vogliono esplorare. In una produzione fantasy potranno essere fondamentali i sistemi di motion capture per gli attori al fine di ricreare personaggi 3D oppure per la simulazione di un concerto sarà invece necessaria una massiccia implementazione del sistema DMX. Questi sono solo alcuni esempi delle molteplici funzioni che di volta in volta potrebbe essere necessario integrare.

BlackMagic o AJA Bundles

Dal momento che abbiamo poc'anzi analizzato il sistema HTC VIVE Mars è doveroso citare l'utilità di schede di acquisizione supportate da Unreal Engine. Utilizzare per esempio una *Deck Link* di Blackmagic Design come già anticipato nel capitolo 5 comporta evidenti vantaggi tecnici come la possibilità di generare il segnale di genlock, importarlo nel *Mars* e sincronizzare tutti i dispositivi su un medesimo timecode. Questo permetterebbe quindi di avere tutti i dispositivi legati ai *Rover* sincronizzati tra loro ed una rapida gestione delle impostazioni di tutta la rete di camere controllate grazie alla disponibilità di utilizzare il software *Media Express* compatilbile con il media framework dedicato a Blackmagic. Simili ragionamenti varrebbero anche per i media bundle dedicati ad AJA.

Sequencer Tool

Non è stato tenuto in considerazione nel nostro progetto di tesi in quanto specifico per la creazione di un prodotto finale o di previs, mentre questa tesi si concentra sul design di un framework generico per le produzioni audiovisive che può avere destinazioni d'uso molto differenti. Tuttavia Unreal Engine contiene tool particolarmente robusti per la produzione

cinematografica real-time che permettono di creare animazioni e sequenze filmiche. Risulta possibile pilotare la camera , animare le luci, renderizzare la sequenza generata e altro ancora. Il cuore di questo workflow è appunto il *Sequencer*, una suite per l'editing non lineare. Come mostra la figura 8.11, l'editor del *Sequencer* si presenta come un classico editor di montaggio audio video. Si possono inserire le varie clip o animazioni precotte al suo interno per creare sequenze cinematografiche. Ma oltre a questo offre un potente strumento per le produzioni di Virtual Production ovvero il tool *Take Recorder*. Grazie ad esso è possibile registrare le performance live che generiamo in Unreal Engine grazie al tool *Live Link*. L'interfaccia utente si presenta così:

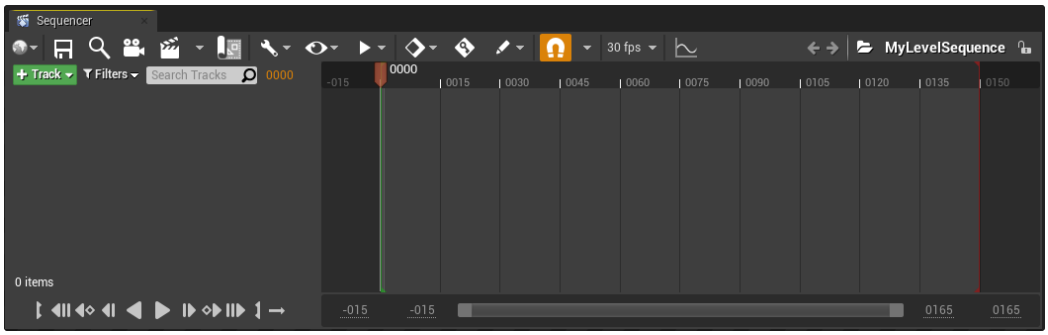

Figura 8.11. Interfaccia dell'editor del tool *Sequencer*.

1. Una *Toolbar* per il controllo di impostazioni generali e di display
2. Una sezione *Slate* che contiene le informazioni riguardanti il take in sospeso e al timecode. È anche dove si trova il bottone per iniziare e fermare la registrazione.
3. Una sezione *Source* che contiene le informazioni da specificare per registrate le performance di determinate sorgenti nella nostra sequenza. (le sorgenti possono anche essere actor Live Link).
4. Una sezione dettagli contenente le impostazioni di progetto del tool *Take Recorder* e quelle delle sorgenti coinvolte nella sequenza.

Una volta registrate le sequenze è possibile inserirle sotto forma di clip come un qualsiasi altro elemento all'interno dell'editor del *Sequencer*.

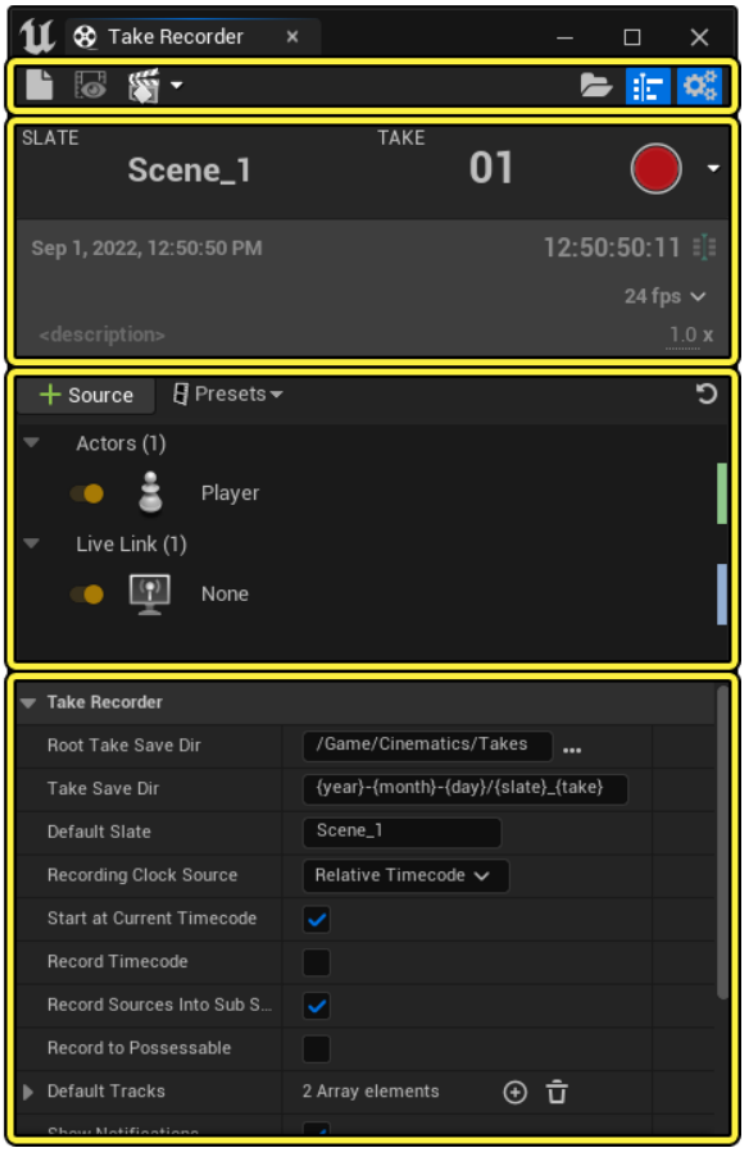

Figura 8.12. Interfaccia del tool *Take Recorder*.

DMX

Come nell'esempio riportato prima un altro aspetto da tenere in considerazione potrebbe essere quello di una rete DMX molto più complessa dovuta per esempio all'esigenza di importanti effetti speciali come condizioni ambientali estreme (un temporale) oppure situazioni molto specifiche come quella di un palcoscenico od una discoteca. In tutti questi

casi, le luci in gioco sono molte e diverse e vanno gestite con un workflow più complesso, facendo innanzituto affidamento ad una console DMX fisica e non virtuale ed a più di un nodo Art-Net in modo da utilizzare non un solo universo ma tanti. Le fixture che mette a disposizione Unreal Engine grazie al plugin *DMX Fixture* sono molte e coprono tutte le casistiche in cui si potrebbe incorrere. Ecco un elenco dei blueprint messi a disposizione dal software game engine usato:

- **BP_WashLed**, luci a led con movibili.
- **BP_StaticHead** e **BP_MovingHead**, luci a incandescenza o a fluorescenza statiche o movibili.
- **BP_StaticMatrix** e **BP_MovingMatrix**, matrici di luci a led o apparecchi pixel mappabili, statici o movibili.
- **BP_StaticStrobe**, luci stroboscopiche ovvero provenienti da fonti intermittenti ad elevata frequenza.
- **BP_FireworkLauncher**, modulo per il lancio telecomandato di fuochi d'artificio.
- **BP_PyroModule**, modulo per il controllo a distanza di effetti pirotecnici.
- **BP_LaserModule**, modulo per il controllo di luci a laser.
- **BP_WaterSource**, modulo per il controllo di fontane d'acqua controllabili a distanza.

Motion Capture

All'interno del framework proposto non è stato previsto un sistema di *motion capture* degli attori. Tuttavia può risultare fondamentale in molte produzioni votate all'animazione 3D. Acquisito l'hardware necessario ed il sistema di tracciamento (ad esempio le classiche tute di motion capture ed il sistema visto nel capitolo 2 OptiTrack), grazie nuovamente al tool Live Link è possibile reindirizzare i dati generati dal tracciamento degli oggetti o degli attori all'interno del game engine. Dopodiché Unreal Engine mette a disposizione principalmente due tool che servono per la gestione dell'animazione dei personaggi virtuali:

- **Skeletal Mesh Animation System**. Questo sistema comprende molte interfacce all'interno di Unreal Engine e si può suddividere idealmente in:
  - *Animation Editors* che una della modalità che si possono attivare dal menù principale. Comprende gli editor per lo scheletro, per la mesh e per le sequenze animate.
  - *Animation Blueprints* che racchiude la famiglia dei blueprint dedicati all'animazione.
  - *Animation Assets and Features* che racchiude tutti gli asset forniti da Unreal Engine per la gestione pratica delle animazioni come Skeletons, Animation Sequence, IK Rig, Animation Pose Assets, Skin Weight Profile, ecc..
- **Control Rig**. Unreal Engine fornisce questa suite di tool di animazione che permettono di riggare e animare un personaggio direttamente *in-engine*. Usando la suite *Control Rig*, si può evitare la necessità di effettuare il rigging e l'animazione in tool esterni ed animare direttamente nell'editor di Unreal Engine. Grazie a questo sistema si possono creare controlli personalizzati sul rig di un personaggio, animato all'interno del *Sequencer* e usare un'ampia varietà di altri tool di animazione per aiutarsi nel processo stesso di animazione.

LED Wall

Un altro metodo alternativo alla produzione virtuale ibrida esposta in questo progetto di tesi è la cosiddetta virtual production *full live* già descritta al capitolo 1. La fase di compositing viene anticipata ed eliminata dal flusso di elaborazione dell'immagine trasmessa dalla cinepresa al computer tornando idealmente ad un cinema tradizionale dove gli effetti speciali sono prodotti sul set e registrati *in-camera*. L'obiettivo principale è quello di eliminare la necessità del compositing del green screen e produrre l'immagine risultante finale direttamente in camera (*In-Camera VFX*). Una delle sfide per produrre effetti speciali di elevata qualità in tempo reale è sincronizzare le diverse tecnologie hardware in campo affinché eseguano il tutto simultaneamente. Come riportato nella documentazione Unreal Engine di Epic [a], questo game engine supporta l'uso dei *LED Wall* per mezzo

di diversi sistemi come *nDisplay*, *Live Link*, *Multi-User Editing* e il *Web Remote Control*. Come abbiamo visto nell'introduzione della tesi in un limbo con *LED Wall* esistono due componenti principali:

- L'*inner frustum* che rappresenta il campo visivo della camera. L'immagine mostrato sul display è legato al tracciamento della camera in modo che i loro movimenti siano sempre concordi creando un effetto di parallasse che genera l'illusione di star riprendendo in una location del mondo reale quando si guarda attraverso la cinepresa fisica.

- L'*outer frustum* che comprende tutte le immagini sui display LED al di fuori del campo visivo della camera. I LED dell'*outer frustum* fungono da sistema e sorgente d'illuminazione e riflessione dinamica per il set fisico dal momento che lo circondano emettendo il mondo virtuale e illuminandolo come se fosse una vera location del mondo reale.

Ogni setup di ripresa può essere posto nella location desiderata all'interno di un ambiente creato su Unreal Engine e dettando cosa debba essere renderizzato dall'*outer frustum* per illuminare la scena. Il design di un set con *LED Wall* dipende direttamente dall'uso che ne si vuole fare e dal tipo di setup *In-Camera VFX* che si vuole ottenere. Il numero di pannelli necessari in un volume LED ed il modo in cui sono piazzati influisce il restante setup hardware. I pannelli LED possono essere posizionati ad arco intorno al set dove recitano gli attori per ricreare una miglior illuminazione ambientale. Spesso risulta anche utile usufruire di un soffitto LED per contribuire all'illuminazione globale di tutta la scena. Le produzioni più grandi che volessero creare un ambiente *full virtual* dovrebbero allestire un volume con una chiusura minima di 270° in modo da raggiungere una buona accuratezza del setup d'illuminazione e riflessione. Data l'importanza del sistema hardware e in gran parte fisico di una produzione con LED Wall, sono molti i fattori che possono influenzare il design di un *LED set*. Tra questi si annoverano il budget di produzione, i limiti spaziali e la disponibilità di pannelli dai produttori. Un *LED Volume* è costituito da cluster di blocchi detti *cabinets* (figura 8.13). Ogni *cabinet* possiede una risoluzione in pixel prefissata che può andare da bassissime risoluzioni come 92x92 pixels a risoluzioni molto elevate superiori

ai 400x400 pixels. Attraverso un processore detto *LED Processor*, hardware e software combinano i diversi *cabinets* in array che infine riproducono l'intera immagine. È anche possibile disporre i *cabinets* in qualsiasi configurazione permessa dai canvas all'interno di Unreal Engine controllati dal processore. Nei casi di grandi studi LED Wall è anche possibile che ci siano diversi cluster ognuno con un proprio processore ma tutti insieme riescono a produrre un LED Wall *seamless*.

Figura 8.13. Esempio di cluster di *cabinet* che si possono distinguere esaminando la parte posteriore di un *LED Wall*.

Un aspetto rilevate da tenere conto riguardo agli schermi utilizzati durante una produzione con *LED Wall* è il cosiddetto *Pixel Pitch*. Esso rappresenta la funzione di densità dei pixel in un certo *cabinet* ed è correlato alla risoluzione globale. Il *pixel pitch* è definito come la distanza che intercorre tra ogni luce LED e viene misurato in millimetri. Più sono quindi vicine tra loro le luci LED, minore è il *pixel pitch* e quindi maggiore è la densità dei pixel stessi. Chiaramente un'elevata densità di pixel equivale a un ragguardevole aumento della risoluzione/qualità ma anche ad un aumento del costo del singolo *cabinet*. Per determinare la distanza minima dal *LED Wall* alla quale si dovrà riprendere il soggetto affinché non si creino artefatti visibili serve conoscere e il *pixel pitch* e la dimensione del sensore della camera. Se questa condizione non dovesse essere soddisfatta si incorrerà

infatti in un tipico artefatto ottico che è conosciuto come *pattern di Moiré* (figura8.14). Questo artefatto compare sul immagine ripresa dalla cinepresa quando la distanza inter-pixel (*pixel pitch*) del sistema LED e la dimensione del sensore della cinepresa stessa sono comparabili. Questo particolare pattern può quindi apparire quando il piano di messa a fuoco della camera si allinea a quello del *LED Wall*. Un altro caso è invece quello dove si raggiungo angoli di incidenza molto acuti dell'asse focale della cinepresa rispetto ai pannelli LED (l'angolo ottimale risulta infatti quello perpendicolare al *LED Wall*).

Figura 8.14. Esempio di *pattern di Moiré* su un *LED Wall*.

Infine è importante fare alcune considerazioni riguardo alla consistenza del colore nelle riprese. Risulta infatti importante testare il risultato dell'immagine finale usando l'*outer frustum* come sorgente d'illuminazione. Alcune delle *best practice* che sono consigliate durante le riprese:

- Non tutte le cineprese producono lo stesso output. Per evitarlo è importante usare camere identiche all'interno di un *LED Volume*.
- Testare sempre l'effetto dell'illuminazione del LED Wall sopra agli oggetti di scena in quanto i pannelli LED hanno un comportamento particolare rispetto ad altre fonti di luci.

- Assicurarsi che il *tonemapper* [2] sia disabilitato in modo che i contenuti di Unreal Engine non abbiano curve tonali assegnate e siano nello spazio colore lineare *sRGB* così come da input nei *cabinet*.

[2] Il *Tone Mapping* è una tecnica usata nell'elaborazione di immagini e in CG per mappare un gruppo di colori su un altro vicino in modo da approssimare la percezione di avere un'immagine con un *high dynamic range* (HDR) su un display che possiede invece una gamma dinamica limitata.

Ringraziamenti

Il sottoscritto ringrazia il prof. Riccardo Antonino per la disponibilità mostrata nell'accogliere questo progetto di tesi e per il materiale e lo spazio messo a disposizione da Robin Studio. Un ringraziamento speciale va a Matteo Morato per la supervisione e i consigli dispensati nei mesi di sviluppo del progetto che mi hanno insegnato la concretezza del lavoro su un set cinematografico. Un grazie anche a tutti i colleghi nell'ambiente di lavoro con cui si sono condivisi pranzi, risate e passioni. Il ringraziamento più grande va a mio padre che mi ha spronato durante questo percorso universitario ad ottenere sempre il meglio ed alla mia fidanzata che con amore mi ha supportato negli innumerevoli momenti bui di questi mesi mostrandomi il faro laddove non lo vedevo. Infine alla mia mamma che mi ama incondizionatamente sempre.

Bibliografia

P. Bauer, W. Lienhart, and S. Jost. Accuracy investigation of the pose determination of a vr system. *Sensors*, 21(5):1622, 2021. URL https://doi.org/10.3390/s21051622.

Blackmagic. Unreal engine compatibile con i prodotti decklink. URL https://www.blackmagicdesign.com/it/media/release/20181116-01.

Epic. Unreal engine documentation - in-camera vfx overview, a. URL https://docs.unrealengine.com/5.1/en-US/in-camera-vfx-overview-in-unreal-engine/.

Epic. Unreal engine documentation - dmx overview, b. URL https://docs.unrealengine.com/4.27/en-US/WorkingWithMedia/CommunicatingWithMediaComponents/DMX/Overview/.

J. Foster. *The Green Screen Handbook. Real-World Production Techniques*. Wiley Publishing, Inc., 2010.

The Virtual Production Glossary. Fix it in pre. URL https://www.vpglossary.com/vpglossary/fix-it-in-pre/.

HTC. Vive mars camtrack user guide, a. URL https://dl4.htc.com/Web_materials/Manual/CamTrack/HTC_VIVE_Mars_CamTrack_User_Guide.pdf.

HTC. Vive tracker 3.0 developer guidelines, b. URL https://developer.vive.com/documents/850/HTC_Vive_Tracker_3.0_Developer_Guidelines_v1.1_06022021.pdf.

ISO5459:2011. *Geometrical product specifications (GPS) — Geometrical tolerancing — Datums and datum systems*. International Organization for Standardization, 2011.

L. Kuhlmann de Canaviri, K. Meiszl, V. Hussein, P. Abbassi, S.D. Mirraziroudsari, L. Hake, T. Potthast, F. Ratert, T. Schulten, M. Silberbach, Y. Warnecke, D. Wiswede, W. Schiprowski, D. Heß, R. Brüngel, and C. M. Friedrich. Static and dynamic accuracy and occlusion robustness of steamvr tracking 2.0 in multi-base station setups. *Sensors*, 23(2):725, 2023. URL https://doi.org/10.3390/s23020725.

J. A. Okun and S. Zwerman. *The VES Handbook of Visual Effects*. Routledge Taylor & Francis, 2021.

OptiTrack. Optitrack documentation. URL https://docs.optitrack.com/.

Steam. Steamvr tracking. URL https://partner.steamgames.com/vrlicensing#:~:text=The%20SteamVR%20Tracking%20Basestations%20sweep,a%20fraction%20of%20a%20millimeter.

A. Thompson. How james cameron's innovative new 3d tech created avatar, 2010. URL https://www.popularmechanics.com/culture/movies/a5067/4339455/.

Wikipedia. Ipv4, a. URL https://it.wikipedia.org/wiki/IPv4.

Wikipedia. Application programming interface, b. URL https://it.wikipedia.org/wiki/Application_programming_interface.

Wikipedia. Frequenza dei fotogrammi, c. URL https://it.wikipedia.org/wiki/Frequenza_dei_fotogrammi.

Wikipedia. Royalty-free, d. URL https://it.wikipedia.org/wiki/Royalty-free.

Wikipedia. Steam (informatica), e. URL https://it.wikipedia.org/wiki/Steam_(informatica).

Wikipedia. User datagram protocol, f. URL https://it.wikipedia.org/wiki/User_Datagram_Protocol.

L. Zanotti. Come funziona uno switch di rete e a cosa serve: storia e vantaggi della tecnologia ethernet, 2022. URL https://www.zerounoweb.it/techtarget/searchdatacenter/configurazione-di-rete-e-significato-dello-switch-tecnologia-storia-e-vantaggi/.