Politecnico di Torino

Collegio di Ingegneria Informatica, del Cinema e Meccatronica

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA (INDIRIZZO GRAFICA E MULTIMEDIA)

Tesi di Laurea Magistrale

"Paesaggi – Landscapes": un viaggio immersivo nell'antico Egitto

Anno accademico 2024-2025

Candidato: Claudio Di Maida

Relatore: Prof. Riccardo Antonino

Agli aviatori di ambizioni,
a chi ha creduto in me,
a chi è sempre stato
nel mio cuore e sempre lo sarà.

INDICE

INTRODUZIONE

La seguente Tesi di Laurea Magistrale discute la creazione di un'esperienza immersiva a carattere museale partendo dalle sue fasi iniziali di studio, passando per la produzione e per i punti salienti che hanno portato al suo sviluppo per poi concludersi con i risultati ottenuti nella sua rappresentazione finale.

Attualmente la computer grafica riveste un ruolo rilevante in settori che spaziano dal mondo dell'intrattenimento a quello industriale, dall'ambito medico a quello artistico. Nel campo dei Beni Culturali, la grafica 3D compie un passo in avanti in un contesto di utilità funzionale, offrendo delle soluzioni alternative alla fruizione di contenuti e all'acquisizione di una conoscenza che altrimenti potrebbe risultare preclusa o difficilmente accessibile. Con la moderna tecnologia, inoltre, è possibile rendere più accattivante l'informazione realizzando attività interattive e installazioni immersive che veicolano concetti in un formato dinamico in cui l'utente non è più un semplice spettatore, bensì diviene attore e parte attiva dell'interazione.

In tale ambito culturale, l'installazione *Paesaggi – Landscapes*, realizzata per le Gallerie d'Italia, si colloca come vetrina di *Egitto Immersivo*, un progetto di ampia scala commissionato dal Museo Egizio di Torino in occasione del bicentenario che prevede, tra le tante cose, la realizzazione di due sale immersive all'interno del polo museale. L'obiettivo è quindi quello di utilizzare un contesto immersivo per esplorare vari aspetti della cultura egizia in appositi spazi che faranno da telaio ai contenuti del museo stesso. La prima sala avrà come focus il fluire della conoscenza e sarà suddivisa in tre segmenti principali. Il primo si occuperà dell'evoluzione degli strumenti di analisi degli oggetti prendendo come riferimento un vaso predinastico. Il secondo riguarderà il paesaggio naturale egiziano, le sue trasformazioni, la toponomastica e la geografia religiosa. Il terzo esaminerà il percorso di conoscenza ponendo l'attenzione sui dati e le pubblicazioni scientifiche prodotte. La seconda sala incentrerà l'attenzione sulle stagioni dell'antico Egitto particolarmente influenzate dal fiume Nilo. Gli egizi, infatti, dividevano l'anno in

tre stagioni basate sul ciclo del Nilo. L'inondazione (*Akhet*), la crescita (*Peret*) e il raccolto (*Shemu*) determinavano il calendario agricolo e condizionavano direttamente le attività di semina, irrigazione e raccolta. Nelle pareti della sala verranno descritte con cura tali fasi e l'impatto nella vita quotidiana di ogni stagione tramite dei *rendering*[1] in *real-time*[2] che terranno conto di dati metereologici e metteranno in relazione i manufatti presenti nel museo.

L'apertura delle sale immersive del Museo Egizio, prevista inizialmente per l'autunno del 2024, a causa di problemi di natura organizzativa è stata posticipata per il 2025. Tale ritardo ha comportato la nascita dell'installazione *Paesaggi – Landscapes* che si prefigura come una *preview* di ciò che sarà contenuto all'interno del museo. Tale progetto, nato da una collaborazione tra il Museo Egizio e le Gallerie d'Italia, è stato reso fruibile in maniera gratuita al pubblico dal 13 giugno al 12 settembre 2024 nella sala immersiva situata al terzo piano ipogeo delle Gallerie d'Italia a Torino. Il lavoro per la realizzazione del progetto è stato affidato al team di Robin Studio che ne ha curato la parte creativa e implementativa.

L'esperienza immersiva *Paesaggi – Landscapes* mira a presentare uno spaccato della cultura egizia cercando di offrire una visione storica del processo che ha portato alla sua evoluzione. La dicotomia che vi è tra paesaggio reale e immaginario nella ricerca archeologica e fotografica è il tema centrale dell'installazione. Il visitatore viene trasportato nell'antico Egitto, attraverso un flusso di conoscenza di immagini e suoni che indagano i limiti e la fragilità delle attuali ricostruzioni storiche e paesaggistiche, per riflettere sulle nuove frontiere dell'archeologia moderna[3].

[1] Il *rendering*, in italiano renderizzazione, è il processo di generazione di un'immagine 2D a partire da una scena composta da modelli 3D.
[2] Fare dei *rendering* in tempo reale vuol dire eseguire la conversione da 3D a 2D in maniera istantanea. Ciò ha tendenzialmente come conseguenza una minor qualità del prodotto finale rispetto ad un *rendering* offline, col vantaggio però del fattore interattività particolarmente importante nell'ambito videoludico e nella realtà aumentata.
[3] La presentazione completa dell'installazione si può esaminare nel sito web delle Gallerie d'Italia, nella sezione "Mostre e iniziative a Torino".

Nella prima parte della presente tesi sarà proposta una breve descrizione relativa a *Paesaggi – Landascapes*, illustrando gli aspetti implementativi e le scelte che hanno definito la realizzazione della sua forma finale. Verrà offerta una panoramica generale dell'intera mostra, dei suoi elementi costitutivi e dei concetti chiave che animano i diversi quadri immersivi che la compongono.

Il primo capitolo sarà dedicato alla presentazione di Unreal Engine, il motore grafico adottato come piattaforma principale per la realizzazione della maggior parte delle scene in computer grafica. Si passerà quindi a delineare i vari studi preliminari condotti in una fase anticipatoria del progetto, con l'obiettivo di esplorare le potenzialità applicative del motore grafico rispetto a possibili soluzioni da poter adottare per *Egitto Immersivo*.

Nel secondo capitolo, l'attenzione sarà rivolta alla tecnica di fotogrammetria, esplorando i principali fondamenti teorici e le modalità con cui è stata impiegata all'interno del progetto. Verrà illustrato nel dettaglio il *workflow* completo per la realizzazione dei modelli digitali, basato su rilievi fotogrammetrici sia aerei che ravvicinati di interi siti archeologici. A tal proposito, troveranno spazio anche delle brevi panoramiche sui software impiegati per la buona riuscita delle operazioni. Infine, sarà proposta una breve analisi dedicata a una tecnica innovativa di ricostruzione tridimensionale con particolare attenzione alle sue potenzialità applicative.

Il terzo capitolo sarà invece incentrato sulla realizzazione dei *time-lapse* presenti nell'installazione immersiva. In questa sezione verranno illustrati gli strumenti e le funzionalità offerte da Unreal Engine per l'ottenimento di risultati fotorealistici. Dopo un'analisi dettagliata sulle soluzioni operative adottate per la costruzione di una scena completa in grafica computerizzata, si procederà a descrivere le varie ambientazioni create con l'integrazione dei modelli digitali fotogrammetrici.

Nel quarto capitolo si darà spazio alla vegetazione tipica dell'ecosistema egiziano. Si esamineranno possibili modalità di implementazione di animazioni di crescita delle piante utilizzando degli approcci procedurali. Una particolare attenzione sarà

posta all'animazione dell'aloe vera, realizzata mediante l'uso del software Houdini, presentando le fasi più significative del processo di produzione.

Il quinto capitolo affronterà gli effetti particellari inseriti nella parte finale dell'esperienza immersiva. Dopo una breve introduzione della piattaforma TouchDesigner, sarà analizzato il "flusso di conoscenza", illustrando i passaggi necessari alla composizione del sistema di particelle e le sue finalità narrative all'interno dell'installazione.

Nella parte conclusiva sarà dato spazio ad una riflessione sui risultati conseguiti nell'ambito dell'esperienza formativa offerta dal progetto *Paesaggi – Landscapes*, concedendo spunti di analisi sul riscontro mediatico dell'iniziativa, per poi terminare con alcune considerazioni sul valore culturale, artistico e tecnologico del progetto.

Figura 0.1. Locandina ufficiale di *Paesaggi – Landscapes*.

L'istallazione immersiva *Paesaggi – Landscapes*, la cui locandina ufficiale è riportata in Figura 0.1, è stata fruibile gratuitamente presso le Gallerie d'Italia a Torino, durante i lavori di riallestimento del Museo Egizio, svolti in occasione delle celebrazioni del suo bicentenario.

SINOSSI DI *Paesaggi – Landscapes*

L'esperienza immersiva *Paesaggi – Landscapes*, oggetto della presente tesi, è un progetto culturale ideato dal Museo Egizio in collaborazione con Intesa Sanpaolo e Gallerie d'Italia nell'ambito di un accordo triennale, che prevede iniziative di sviluppo e diffusione della cultura e dell'arte nonché la trasformazione architettonica del Museo Egizio con il riallestimento e rinnovamento della Galleria dei Re, in occasione del bicentenario dalla fondazione del Museo.

A partire dai primi mesi del 2024, alcuni componenti della squadra di Robin Studio si sono recati nei siti archeologici in Egitto per produrre del materiale audiovisivo utile ai fini dello sviluppo. I lavori di messa a punto dell'installazione immersiva sono poi perdurati fino alla prima settimana di giugno, mese in cui è stata presentata e resa visitabile gratuitamente al pubblico durante tutte le mattine dei mesi estivi. Tra le attività parallele che hanno accompagnato la mostra si annovera un ciclo di quattro dialoghi che ha coinvolto egittologi, fotografi ed artisti internazionali in dibattiti sul cambiamento del paesaggio nell'Egitto moderno, sulla visione dell'Egitto presente nei musei archeologici e sugli allestimenti dei paesaggi all'interno delle gallerie e le relative connessioni.

La disamina dell'esperienza *Paesaggi – Landscapes* è da individuare in:

> *«[…] una traduzione visiva delle ricerche e delle riflessioni multidisciplinari degli archeologi sui reperti, sulle loro connessioni spazio-temporali e sul paesaggio, in una sorta di percorso a ritroso dall'Egitto moderno fino alle schegge di memoria, ai frammenti di reperti archeologici e alle loro interpretazioni. Il video si muove dalla rappresentazione oggettiva della campagna egiziana oggi, per virare verso il passato. Il filtro digitale ricostruisce e rimodella lo sguardo, rappresenta in maniera simbolica il passato. Il suono della natura diventa musica, in un crescendo che porta il visitatore quasi*

> *ad immagini allucinatorie, metafora del dubbio e della pluralità di discipline che oggi sovraintende alla ricerca archeologica»*[4].

Il *concept* di *Paesaggi – Landscapes* è stato realizzato dal team di Robin Studio, in collaborazione col Museo Egizio, sotto la supervisione di otto egittologi. Sin dal principio sono state sintetizzate varie idee che hanno condotto il progetto finale a sfidare la concezione tradizionale dell'Egitto immutabile, eterno nello spazio e nel tempo, con l'eredità culturale e territoriale sopravvissuta dalla fisionomia di quelle civiltà. Andando indietro nel tempo, infatti, risulta sempre più complesso ottenere integralmente una visione d'insieme. Occorre procedere con supposizioni ed interpretazioni degli elementi che creano le culture e definiscono i paesaggi naturalistici, linguistici e sonori che continuano a mutare nel tempo. La necessità degli studiosi di setacciare i territori contemporanei per ripercorrere la ricerca mediante congetture nel modello di offerta di una interpretazione precisa del patrimonio culturale egiziano, è il tema principale dell'esperienza immersiva. Ciò che si vuole trasmettere al visitatore è il pretesto per instaurare un dibattito interiore, una riflessione sul vincolo di poter decifrare la conoscenza procedendo per ipotesi e continue riscritture.

Entrando più in dettaglio, l'installazione si articola su tre fasi:

- Osservazione del paesaggio: fase incentrata sull'esplorazione del paesaggio egiziano con riprese panoramiche e dettagliate.
- Analisi del contesto: le immagini vengono esaminate in relazione all'iconografia e alla scrittura dell'antico Egitto indagando sulla mutazione del paesaggio con l'ausilio dell'IA generativa e di *time-lapse* realizzati in computer grafica. Si cerca di rintracciare dei legami tra i dati raccolti per formulare ipotesi sulla natura del panorama osservato.
- Sintesi: ultima parte che si focalizza sulla sintesi delle conoscenze acquisite offrendo una riflessione sulla complessità e l'inaccessibilità della stessa.

[4] Dal comunicato stampa: *Intesa Sanpaolo sostiene il riallestimento della Galleria dei Re del Museo Egizio per il Bicentenario. Il Museo Egizio e le Gallerie d'Italia presentano "Paesaggi – Landscapes"*, Torino, 12 giugno 2024, p.2. Si rimanda alla bibliografia per consultarlo online.

Nella prima sequenza vengono mostrati una serie di panorami a 360°, con immagini dai siti di El-Hammamiya, Gebelein, Deir el-Medina ed Aswan. La narrazione è guidata dagli effetti sonori e lo spettatore viene immerso in un ambiente rurale, dove la storia è percepibile come insieme di sensazioni.

Figura 0.1. Due frame tratti dalla prima sequenza di panorami.

Nella seconda fase vengono mostrati, con la tecnica dello *split screen*, in cornici di diverse dimensioni e proporzioni, le immagini degli elementi che compongono il paesaggio: piante, animali, formazioni rocciose. I *frame* in *split screen* si alternano ad altri paesaggi che riempiono pareti intere e che contestualizzano le immagini degli *split screen*. Ad ogni elemento che viene presentato sullo schermo corrisponde un effetto sonoro. Il ritmo del montaggio è regolare e la comparsa del girato segue una sequenza dal centro di una parete verso l'esterno, per direzionare lo sguardo.

Figura 0.2. Due frame di *split screen* sugli elementi dei paesaggi.

Immagini zenitali riprese dai droni mostrano i *pattern* del territorio, scolpito dal Nilo e rifinito dal lavoro dell'uomo. Sempre con la tecnica dello *split screen*, vengono mostrate le mani di una persona che costruisce un muro in mattoni di fango nonché la gestualità ed il risultato.

Figura 0.3. Fotogrammi su riprese zenitali e lavorazione del fango.

Nella seconda parte della fase di analisi viene rappresentato in *split screen* e a schermo intero il passaggio del tempo. Una serie di *time-lapse* mostra come la luce del giorno cambia in vari scenari e siti storici ricreati in computer grafica.

Figura 0.4. Due frame su *time-lapse* di scenari in computer grafica.

Ad un certo punto i *time-lapse* accelerano, vengono mostrate piante che crescono e fiori locali che sbocciano. Una diga di particelle si mostra coprendo tutto lo scenario e viene decostruita. Segue l'innalzamento del livello del Nilo che sommerge il Tempio di Philae e delle rocce. Queste, erodendosi, denotano il fluire del tempo.

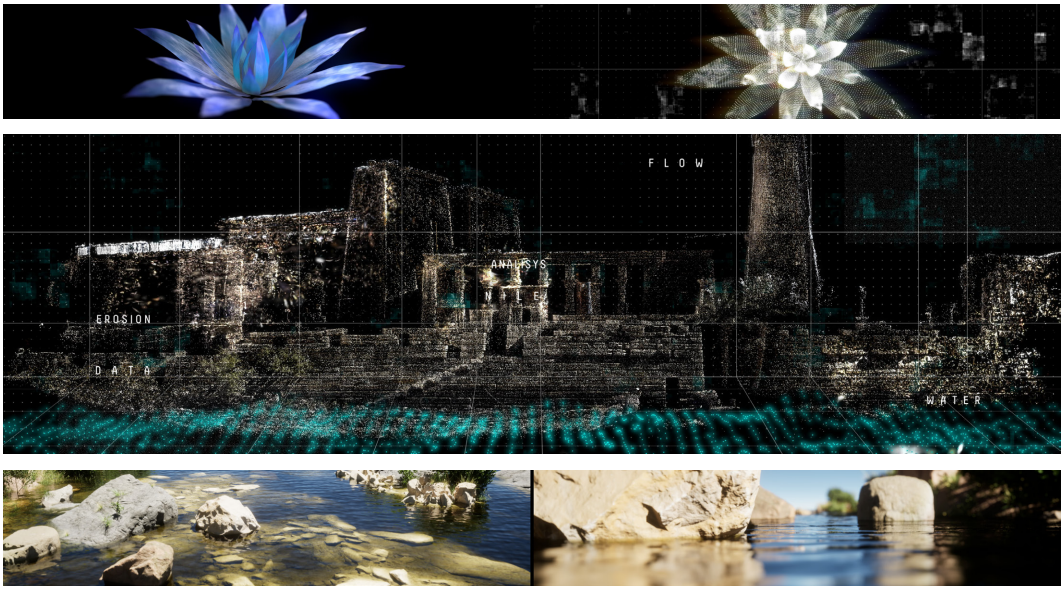

Figura 0.5. Frame di un loto che sboccia, del Tempio di Philae e delle rocce.

In questa fase di analisi entra poi in gioco la riflessione egittologica, ossia la ricerca. Utilizzando il sistema dello *split screen*, vengono affiancate immagini attuali e segni predinastici, dell'iconografia e della scrittura egizia. Il tutto viene esaminato e scandito con tavole della *Description de l'Égypte*[5], considerato uno dei primi testi sulla classificazione della civiltà egizia studiandone i reperti, la flora e la fauna.

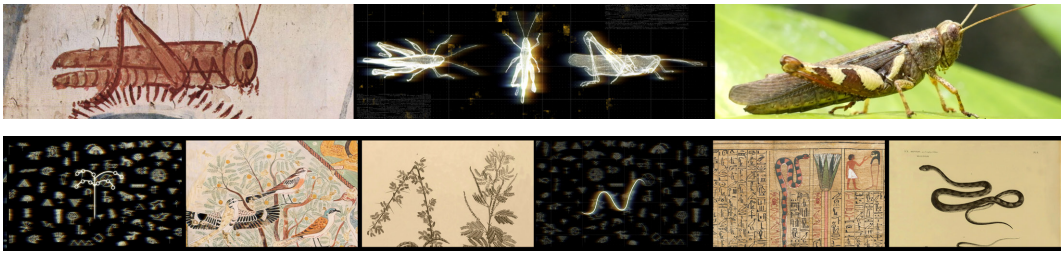

Figura 0.6. Frame con segni predinastici e tavole della *Description de l'Égypte*.

Infine, da uno stadio di interpretazione si arriva alla fase di sintesi. Attraverso l'IA generativa, allenata con un database di immagini di flora, fauna e ambienti egiziani, si vedono dettagli e paesaggi mutare e cercare una definizione con effetti *morphing*. Il processo di interpretazione è trasposto metaforicamente con idee che si formano e modellano in continuazione, come se si venisse condotti nella mente di un ricercatore. L'allucinazione generata dall'IA arriva a concepire immagini sempre più granulari finché il disturbo prende la forma di un flusso di particelle che esprime la conoscenza ultima, inarrivabile, riportando l'osservatore al punto di partenza.

Figura 0.7. Frame sugli effetti di *morphing* dell'IA e sul flusso di particelle.

[5] La *Description de l'Égypte* è una serie di pubblicazioni iniziata nel 1809 che contiene una descrizione scientifica completa dell'Egitto antico e moderno, e della sua storia naturale. Si tratta di un'opera scritta da circa 160 studiosi e scienziati che accompagnarono Napoleone durante la campagna d'Egitto dal 1798 al 1801 nel corso delle guerre rivoluzionarie francesi.

1. STUDI PRELIMINARI

L'avvio dei lavori per *Paesaggi – Landscapes* è stato preceduto da una fase di *brainstorming* su possibili applicazioni che potevano essere incluse all'interno del progetto *Egitto Immersivo*. L'idea iniziale era proprio quella di svolgere dei compiti preparatori in funzione delle esigenze che richiedeva l'ambizioso progetto commissionato dal Museo Egizio di Torino in occasione del bicentenario dell'apertura. Tuttavia, l'intero museo è stato coinvolto da numerose trasformazioni sia dal punto di vista architettonico che per nuovi riallestimenti che hanno generato dei ritardi nella conclusione dei lavori strutturali. Pertanto, visti i rallentamenti logistici, il team creativo di Robin Studio si è focalizzato nell'allestimento di una *preview* dell'esperienza immersiva alle Gallerie d'Italia.

In questo primo capitolo, verrà fatta inizialmente una breve digressione su Unreal Engine, il motore grafico adottato nella sua versione 5.3 con un focus mirato sugli strumenti di cui si è fatto uso. Seguirà una disamina approfondita su alcuni dei primi studi preliminari che sono stati effettuati per *Egitto Immersivo* nel periodo che si antepone all'assegnazione dei lavori per il progetto *Paesaggi – Landscapes*.

1.1. Unreal Engine

Unreal Engine® è un potente motore di rendering 3D e rappresenta attualmente uno degli ambienti di sviluppo più avanzati nel panorama digitale. Nato come strumento per sviluppare videogiochi, ad oggi viene ampiamente impiegato in settori che spaziano dal cinema alla realtà virtuale, dalle installazioni immersive ai progetti di natura architettonica.

Sviluppato inizialmente da Tim Sweeney, uno dei founder di Epic Games, nel 1988 fece il suo debutto come motore grafico per il videogioco *Unreal*, uno sparatutto in prima persona considerato all'epoca uno dei *benchmark* per misurare la grafica dei giochi. Sin dagli albori del suo sviluppo, Unreal Engine introdusse importanti

innovazioni che ben presto lo proiettarono a diventare un riferimento per l'intera industria dell'intrattenimento. Si iniziò così a parlare di effetti grafici avanzati, di una gestione dell'illuminazione più complessa, di *anti-aliasing* e del rilevamento delle collisioni. Nel comparto *multiplayer* le prestazioni del motore grafico risultavano piuttosto scarse rispetto alla concorrenza, ma con lo sviluppo di *Unreal Tournament* la qualità dell'engine iniziò a fare grandi passi in avanti anche dal punto di vista delle ottimizzazioni di rete, aggiungendo il supporto per le Direct3D e OpenGL.

Dopo il grande successo iniziale, il team di Epic Games continuò a migliorare il motore grafico introducendo dei supporti per gli sviluppatori permettendo loro di adattarne le funzionalità in maniera semplice e intuitiva.

La seconda versione di Unreal Engine vide una rivisitazione totale del codice di base e del motore di rendering, l'implementazione della fisica *ragdoll*[6] ed il supporto alle console. Con la settima generazione videoludica, l'engine si adattò agli hardware più performanti migliorando aspetti come la fisica applicabile a più corpi e un sistema particellare potenziato.

Fu a partire da Unreal Engine 3 che si ebbe un notevole balzo tecnologico grazie a *shading*[7] avanzati, illuminazione dinamica, riflessi di luce e texture ad alta risoluzione. Oltre ad introdurre un primo sistema di scripting visuale, Kismet, con l'integrazione del motore PhysX di NVIDIA, Unreal Engine poté contare su una simulazione più realistica per i corpi rigidi, gli oggetti distruttibili e fluidi.

Una rivoluzione in termini di accessibilità avvenne con Unreal Engine 4 in quanto il motore passò ad avere un modello di licenza gratuita con *royalties* che partivano dopo un certo guadagno con le vendite. Con questa versione si introdusse un nuovo paradigma di programmazione visuale, i *Blueprints*, grazie a cui sviluppatori meno

---

[6] La fisica *ragdoll* è una tecnologia per rendere più verosimile il movimento di un essere vivente. Un *ragdoll* (in italiano "bambola di pezza") è un insieme di più corpi rigidi legati da un sistema di vincoli che limitano il modo in cui le ossa possono spostarsi l'una rispetto all'altra.

[7] Nella computer grafica, lo *shading* (in italiano ombreggiatura) indica il modo in cui la luce interagisce con la mesh in relazione a caratteristiche come la riflessione, l'assorbimento, l'angolo di visione e di incidenza.

esperti sul lato codice potevano cimentarsi come veri programmatori. Inoltre, venne migliorata la grafica al punto da renderla quasi cinematografica.

Uno step ulteriore in questa direzione si ebbe a partire dal 2020 con la prima *tech demo* di Unreal Engine 5. Attualmente l'ultima versione del motore grafico, presenta importanti migliorie come *Lumen* e *Nanite*, un progresso nella resa del *ray-tracing*[8] ed una tecnologia capace di renderizzare effetti volumetrici complessi.

Lumen è un sistema innovativo di illuminazione globale che permette di creare ambienti realistici con una maggiore qualità e dettaglio. Grazie ad esso, si possono, inoltre, creare effetti luce dinamici più efficienti, rendendo superflui le gestioni pre-renderizzate del passato.

Nanite è un sistema di rendering che si basa sulle particelle e permette di gestire modelli con alta densità poligonale in *real-time*. In questo modo lo sviluppatore è in grado di utilizzare modelli molto dettagliati senza avere la necessità di doverli adattare in base alla piattaforma di riferimento e di conseguenza non preoccuparsi più dell'aspetto prestazionale. La tecnologia utilizza l'intelligenza artificiale per analizzare i modelli in *real-time* e scegliere automaticamente il livello di dettaglio (LOD) da visualizzare, semplificando eventualmente la mesh in base alla distanza dell'oggetto dalla camera che sta renderizzando la scena.

Sempre più sviluppatori e creatori di contenuti si affidano a questo efficiente motore grafico per realizzare i propri lavori, che siano videogiochi, esperienze immersive o produzioni cinematografiche. Grazie al suo costante supporto alle innovazioni e per la sua capacità di evolversi e adattarsi a qualunque esigenza, Unreal Engine viene considerato ad oggi un pilastro e uno standard nell'industria grafica.

[8] Il *ray-tracing* è una tecnica di rendering che riproduce il percorso della luce all'interno di una scena virtuale, calcolando con precisione la sua interazione con gli oggetti presenti e gestendo in modo realistico ombre, riflessi e rifrazioni, a fronte di un elevato costo computazionale. Rispetto al più tradizionale algoritmo di *ray casting* che prevede l'invio di un raggio per ogni pixel ed il rilevamento della geometria e del colore quando questi collide, con il *ray-tracing*, quando un raggio colpisce una superficie, esso può generare fino a tre nuovi tipi di raggi (riflessione, rifrazione ed ombra) che possono interagire con altri oggetti della scena.

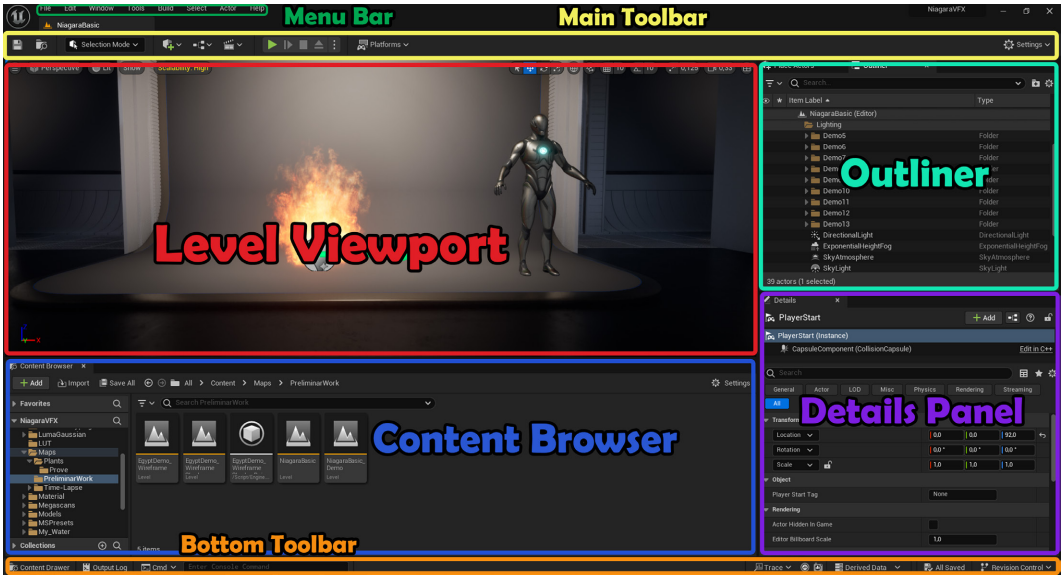

Figura 1.1. User Interface di Unreal Engine.

Come è possibile notare dalla Figura 1.1, l'interfaccia principale dell'editor di Unreal Engine è composta da:

- *Menu Bar*: barra dei menu che consente di accedere ai comandi ed alle funzionalità specifiche dell'editor.
- *Main Toolbar*: contiene *shortcut* per alcuni degli strumenti e comandi più usati su Unreal Engine. È suddivisa nelle seguenti aree: pulsante di salvataggio, modalità di selezione, *shortcut* per inserire nuovi contenuti (*Actor*[9], *Blueprints*[10] e *Cinematics*[11]), controlli per avviare il gioco direttamente dall'editor, opzioni per configurare il progetto per diverse piattaforme ed altre impostazioni.

[9] In Unreal Engine si parla di *Actor* come un qualunque oggetto che può essere posizionato all'interno di una scena (una mesh, una telecamera, la posizione di partenza di un giocatore). Gli *Actor* supportano le trasformazioni 3D e possono essere creati (*spawnati*) o distrutti tramite codice. L'equivalente in Unity (motore grafico molto diffuso) è il GameObject ma a differenza di questo, l'*Actor* è una classe C++ che può essere estesa e personalizzata tramite ereditarietà.
[10] Con il termine *Blueprint* si indica un sistema di scripting visivo che utilizza un'interfaccia basata su nodi per creare elementi di gioco. Viene utilizzato per definire classi o oggetti orientati agli oggetti. Gli oggetti definiti tramite Blueprint sono colloquialmente chiamati "Blueprints".
[11] Le cinematiche permettono di creare sequenze animate e cinematografiche. È possibile pilotare le telecamere per creare effetti *fly-through*, animare luci, spostare oggetti, animare personaggi. Al centro di questi flussi di lavoro c'è il *Sequencer*, una potente suite di editing non lineare usato per creare contenuti cinematografici in Unreal Engine.

- *Level Viewport*: mostra il contenuto del livello (la scena) attualmente aperto. Permette la navigazione nella scena offrendo diverse modalità di visualizzazione (ad esempio *Lit*, *Unlit*, *Wireframe*[12]) e specifiche per le collisioni ed i livelli di dettaglio[13]. Fornisce gli strumenti necessari ad interagire con gli *asset* del livello permettendo operazioni di traslazione, rotazione e scala. Ha dei tool per allineare in maniera precisa gli oggetti e consente di visualizzare i contenuti con una vista che può essere prospettica oppure ortografica. Inoltre, è possibile configurare layout multipli per ottimizzare il flusso di lavoro.
- *Content Browser*: finestra che mostra tutti gli *asset*, i Blueprints e gli altri file contenuti nel progetto. Si possono sfogliare i contenuti, trascinare gli *asset* nel livello, migrare questi tra vari progetti e altro.
- *Bottom Toolbar*: contiene *shortcut* per l'*Output Log* (strumento di debugging che stampa informazioni utili mentre l'applicazione è in esecuzione), la *Command Console* (interfaccia a riga di comando utilizzata per attivare comportamenti specifici dell'editor) e le funzionalità dei dati derivati. Visualizza inoltre lo stato del controllo sorgente.
- *Outliner*: visualizza in maniera gerarchica tutti i contenuti della scena. Se ne possono avere fino a quattro diversi, ognuno con il proprio layout di colonna e la propria configurazione di filtro.
- *Details Panel*: mostra le impostazioni e le proprietà dell'Actor selezionato.

Le modalità di selezione modificano il comportamento primario del *Level Editor* per svolgere delle attività specifiche, come spostare e trasformare risorse nel mondo[14], scolpire paesaggi, generare fogliame, creare volumi e pennelli geometrici

[12] La modalità *Wireframe* mostra solo gli spigoli (lo scheletro) dei modelli, utile per analizzarne le geometrie. La modalità *Unlit* rimuove tutta l'illuminazione dalla scena, mostrando solo il colore di base. Infine, la modalità *Lit* mostra il risultato finale della scena una volta applicati tutti i materiali e l'illuminazione.

[13] I livelli di dettaglio, in inglese *Level Of Details* (LODs), servono a migliorare le prestazioni della scena. Quando una mesh si trova in una posizione lontana rispetto al punto di vista corrente, tramite i LODs questa viene visualizzata con una sua versione meno dettagliata ma che diventa sempre più complessa man mano che si avvicina la camera.

[14] In Unreal Engine con il termine "mondo" si intende l'ambiente virtuale globale in cui si svolgono le interazioni di gioco. Un "livello" è una sezione specifica del mondo, una mappa delimitata che può contenere *asset*.

nonché dipingere su mesh. I pannelli delle modalità contengono una selezione di strumenti su misura per la tipologia di modifica selezionata. Le modalità sono:

- *Select mode*: simile per natura al Content Browser, si concentra solo sugli Actor, ovvero le risorse che possono essere posizionate direttamente nel tuo livello. Quando è attiva, Select Mode permette un accesso immediato a tutti gli oggetti che sono posizionati nel livello, velocizzando notevolmente i tempi di progettazione.
- *Landscape mode*: è una raccolta di strumenti che consente di creare e modificare vasti ambienti esterni in maniera realistica.
- *Foliage mode*: consiste in un set di strumenti per "dipingere" o cancellare rapidamente set di mesh statiche o fogliame su degli Actor e geometrie abilitate. Tramite questa modalità, è possibile popolare un vasto ambiente esterno con fogliame in un breve lasso di tempo. Ne verrà approfondito l'utilizzo in seguito, nel capitolo che tratta gli strumenti di Unreal.
- *Mesh Paint mode*: permette di dipingere i colori dei vertici su mesh statiche. Si possono dipingere più istanze di una singola mesh con valori colore/alfa univoci. I dati del colore possono essere visualizzati direttamente, consentendo di modificare più mesh contemporaneamente.
- *Modeling mode*: fornisce un set di strumenti per creare, scolpire e modificare la geometria 3D direttamente in Unreal Engine. Questi tools offrono flussi di lavoro per modifiche di topologia, creare mappe UV, assegnare materiali multipli, modificare le collisioni e le texture.
- *Fracture mode*: contiene un'ampia varietà di strumenti per creare, frammentare e manipolare collezioni di geometrie. È uno strumento essenziale per simulare la frammentazione in tempo reale in Unreal Engine.
- *Brush Editing mode*: è uno strumento utilizzato per la costruzione dei livelli. Prima di essere sostituito da meccanismi più efficienti come l'uso di mesh statiche[15] e la modalità di modellazione *in-engine*, era considerato lo

[15] Sono considerate mesh statiche i modelli 3D creati tramite un'applicazione esterna di modellazione come Blender, Maya, 3DS Max ed importati sull'Unreal Editor attraverso il Content Browser. Poiché le mesh statiche sono memorizzate nella cache della memoria video, possono essere traslate, ruotate e ridimensionate e risultare più complesse di altri tipi di geometria.

strumento costruttivo primario nella progettazione delle scene. Tuttavia, può risultare ancora utile nelle fasi iniziali di un prodotto per la prototipazione rapida di livelli e oggetti.

- *Animation mode*: espone strumenti, pannelli e comportamenti dell'editor che assistono i flussi di lavoro incentrati alle animazioni.

Verranno di seguito approfonditi tre aspetti fondamentali di Unreal Engine ovvero i *Blueprints*, i materiali ed il *Niagara System*.

1.1.1. I Blueprints

Negli ultimi anni l'industria dell'intrattenimento ha vissuto una transizione verso strumenti sempre più intuitivi e accessibili. Tra questi, il *Blueprint Visual Scripting* di Unreal Engine si distingue come una delle innovazioni più rivoluzionarie. Introdotti con la versione di Unreal Engine 4, i Blueprints consentono ad artisti, sviluppatori e designer di realizzare logiche complesse senza la necessità di dover scrivere codice, migliorando in questo modo la scalabilità del lavoro tra gruppi multidisciplinari. Tale sistema di scripting visuale è, infatti, costruito su di un'interfaccia grafica basata sui nodi che rappresentano azioni, funzioni o variabili. I nodi contengono al loro interno del codice nativo C++ che è anche il linguaggio di programmazione utilizzato dagli sviluppatori per fare scripting più avanzato. Unreal Engine consente l'integrazione tra Blueprints e codice nativo. Gli sviluppatori possono creare delle funzioni scritte in C++ e renderle disponibili nei Blueprints, combinando le qualità dei due modi di fare scripting: l'elasticità della programmazione visuale e l'efficacia del codice nativo.

I tipi di Blueprint più comunemente usati sono i *Level Blueprints* e le *Blueprint Classes*. I *Level Blueprints* racchiudono quella che è la logica per eventi specifici del livello. Ognuno di questi contiene, infatti, un *Level Blueprint* che è in grado di manipolare gli Actor all'interno della scena, controllare le cinematiche utilizzando i *Level Sequence Actor* e interagire con altri *Blueprint Classes* che fanno parte di quel livello. Una *Blueprint Class*, generalmente abbreviata come Blueprint, definisce una nuova classe o tipo di Actor che è possibile posizionare nelle mappe come istanze. Modificare un Blueprint in un progetto, comporta l'aggiornamento

di ogni sua istanza. Le *Blueprint Classes* sono ideali per creare risorse interattive come porte, interruttori, oggetti collezionabili e scenari distruttibili. Esistono diversi tipi di Blueprint che è possibile creare ma prima di tutto è necessario specificare la classe genitore da cui questi erediterà le proprietà. In Unreal Engine sono presenti delle classi *built-in*[16] da cui ereditare come:

- *Actor*: un oggetto che può essere posizionato o generato nel mondo;
- *Pawn*: un *Actor* che può essere "posseduto" e che può ricevere degli input da un Controller;
- *Character*: un *Pawn* che include determinate abilità di movimento;
- *Player Controller*: un *Actor* che controlla un *Pawn* usato da un giocatore;
- *Game Mode*: definisce le regole e gli aspetti di un tipo di gioco.

Dopo aver scelto la classe genitore, nel Blueprint è possibile definire funzioni, interfacce o elementi come: *Construction Script* per contenere la logica eseguita prima dell'avvio del livello, *Event Graph* per la logica eseguita all'avvio e *Components* ossia oggetti che definiscono un comportamento modulare riutilizzabile.

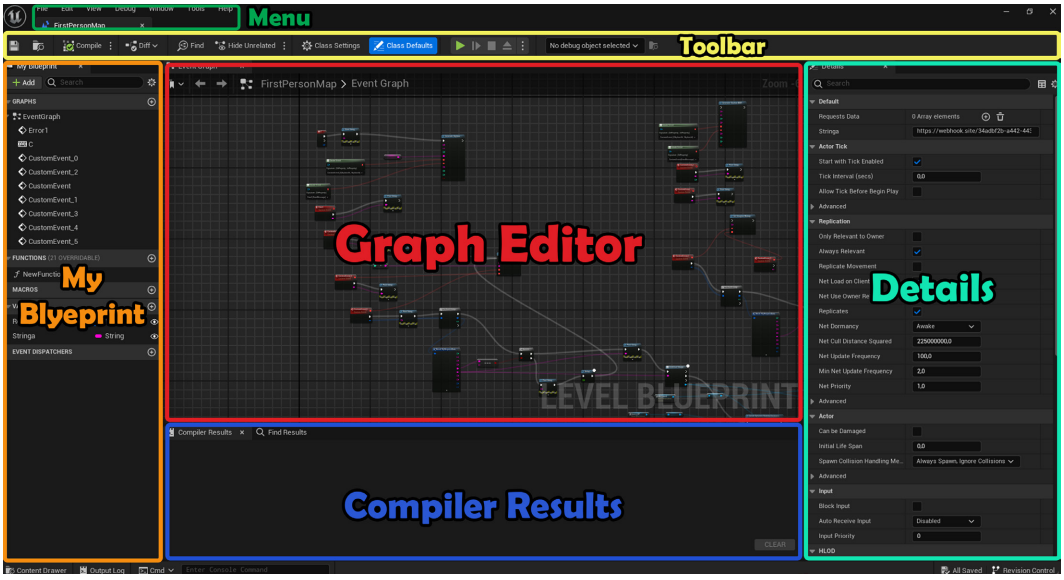

Figura 1.2. User Interface del Level Blueprint.

[16] Unreal Engine segue una struttura ad albero in cui ognuna delle classi eredita ed estende la classe base Object che rappresenta un oggetto generico.

Come detto precedentemente, un *Level Blueprint* è un tipo specifico di *Blueprint* che definisce gli eventi globali di un determinato livello. Nella sua interfaccia, mostrata in Figura 1.2, è possibile distinguere vari pannelli:

- *Menu*: tipico menu che permette di eseguire operazioni di salvataggio, gestione dei file e delle finestre presenti nell'editor.
- *Toolbar*: fornisce un facile accesso ai comandi comuni necessari durante la fase di editing dei *Blueprints*. Oltre a permettere di avviare la simulazione, contiene strumenti per la compilazione ed il debugging, mostrare le differenze con le precedenti compilazioni, visualizzare le impostazioni ed i valori di default della classe.
- *My Blueprint*: mostra un elenco ad albero di grafici, variabili, funzioni e macro contenuti nel Blueprint. Si tratta di una panoramica del Blueprint che consente di visualizzare gli elementi esistenti e di crearne di nuovi.
- *Details*: fornisce accesso alle proprietà dell'elemento selezionato all'interno dell'editor. Tramite tale pannello è possibile gestire attività di modifica delle variabili, aggiungere input e output per le funzioni, implementare interfacce e inserire eventi per i Components selezionati.
- *Graph Editor*: rappresenta il nucleo centrale dell'editor in cui è possibile creare, visualizzare e modificare la rete di nodi che definisce il comportamento del Blueprint.
- *Compiler Results*: offre dei *feedback* durante la compilazione del Blueprint, indicando se questa è riuscita o se sono avvenuti degli errori o *warning* fornendo eventualmente delle informazioni specifiche.

I nodi sono gli elementi fondanti nel sistema di *Blueprints* e rappresentano istruzioni o blocchi logici che vengono collegati tra loro per la costruzione di comportamenti complessi. Il modo in cui essi sono congiunti determina il flusso della logica dello script. I nodi possono avere dei pin su entrambi i lati: sulla parte sinistra vengono chiamati pin di input mentre sulla destra pin di output. Esistono due tipi principali di pin:

- Pin di esecuzione: vengono utilizzati per collegare i nodi creando un flusso di esecuzione. Quando viene attivato un pin di esecuzione di input, il nodo

viene eseguito. Terminata l'esecuzione del nodo, tramite un pin di esecuzione di output viene continuato il flusso di esecuzione.

- Pin di dati: servono per immettere dati in un nodo o per emettere dati da un nodo. Sono specifici per il tipo di dato e possono essere collegati ad altri pin dello stesso tipo o a variabili. Nel caso in cui si vogliano collegare due pin che hanno tipi di dati diversi, viene effettuata, ove possibile, un'operazione di *casting* ovvero di conversione del tipo di dato.

Nei *Blueprints* è possibile trovare varie tipologie di nodi in base al loro scopo e si possono distinguere dal colore. Esistono *Event Node* (in rosso) che rappresentano dei punti di ingresso di un flusso di esecuzione e vengono chiamati al susseguirsi di determinati avvenimenti nella scena. Questi tipi di nodi includono i più comunemente usati: *Event Begin Play* chiamato all'avvio della scena o quando un Actor viene immesso in un livello, *Event Tick* chiamato ad ogni frame e *Custom Event* che rappresenta un insieme di eventi personalizzati definiti dall'utente e che possono essere chiamati in qualsiasi punto della sequenza del Blueprint o in un'altra classe ed a cui è possibile passare dei dati in input.

Altre tipologie di nodi sono *Function Node* (in blu) e *Pure Function Node* (in verde) in cui è possibile definire delle funzioni impure e pure[17]. Sono presenti anche nodi (in viola) ottenuti tramite Construction Script o chiamate a funzioni. Infine, si hanno *Cast Node* (in ciano) per svolgere operazioni di conversione, *Flow Control Node* che forniscono dei meccanismi per il controllo del flusso e nodi per le *Macro* (entrambi in grigio).

1.1.2. I materiali

In Unreal Engine il concetto di materiale è strettamente legato all'aspetto visivo di una superficie di un oggetto della scena. In termini tecnici, i materiali indicano al

[17] La differenza principale è che le funzioni pure promettono di non modificare in alcun modo lo stato o i membri della classe, mentre le funzioni impure sono libere di modificare lo stato. Le funzioni pure sono generalmente utilizzate per funzioni di tipo *getter* o operatori che emettono solo un valore di dati.

motore di rendering come una superficie dovrebbe interagire con la luce definendo il colore, la trasparenza, la rugosità e la riflessività.

Il compito di definire come un pixel viene renderizzato viene svolto dagli *shader* ovvero un insieme di algoritmi capaci di simulare l'aspetto di un materiale virtuale in modo da risultare simile a quello reale.

In Unreal Engine gli *shader* sono scritti in *High Level Shading Language* (HLSL) e convertiti in istruzioni Assembly da far eseguire alla GPU. Tuttavia, non è necessario scrivere codice HLSL per realizzare *shader* bensì si possono creare delle risorse chiamate *Material* tramite un'interfaccia di scripting visuale conosciuta come *Material Editor*. Questo editor consente di definire le proprietà visive delle superfici utilizzando un sistema basato su nodi. La composizione dei nodi prende il nome di *Shader Graph*.

La possibilità di integrare complessi calcoli matematici, effetti dinamici e funzioni personalizzate rende il *Material Editor* uno strumento estremamente potente ed efficace in qualsiasi area di applicazione. Allo stesso modo di quanto visto per i *Blueprints*, ogni nodo del *Material Editor* è costituito da pin di ingresso e di uscita che coincidono agli input e output del codice HLSL presente nel nodo e permettono di creare *shader* complessi.

I nodi possono essere dei *Material Expressions* o *Material Functions*. Le *Material Expressions* sono un set di istruzioni statiche che svolgono operazioni relativamente semplici. Le *Material Functions* consentono di confezionare parti di un *Material Graph* in una risorsa riutilizzabile che è possibile condividere in una libreria comune e inserire facilmente in altri *Material*. Il loro scopo è principalmente quello di semplificare la creazione di *Material* fornendo un accesso immediato alle reti di nodi dei materiali più utilizzati. Un ulteriore vantaggio è dovuto al fatto che le modifiche a una singola *Material Function* si propagano in tutte le reti di *Material* che la utilizzano, senza dover andare a modificare manualmente ogni materiale che impiega quella funzione.

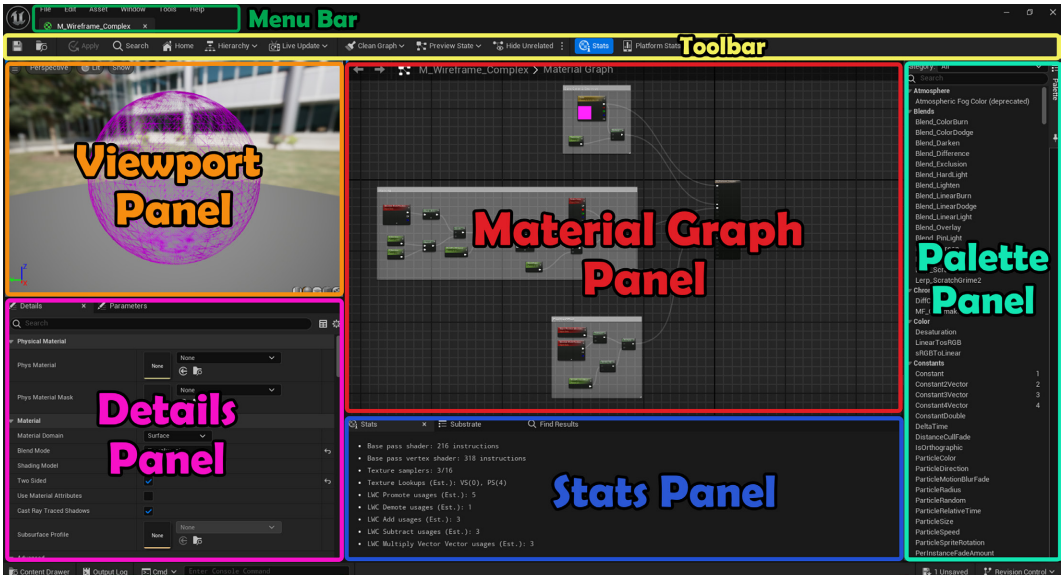

Figura 1.3. User Interface del Material Editor.

Seguendo la Figura 1.3, gli elementi fondamentali che compongono l'interfaccia grafica del *Material Editor* sono:

- *Menu Bar*: barra dei menu che permette di gestire gli *asset*, le finestre mostrate e i tool messi a disposizione dall'editor.
- *Toolbar*: contiene degli *shortcut* per i principali strumenti dell'editor. Tra questi si trova la possibilità di applicare le modifiche al materiale, trovare *asset* nel *Content Browser* e cercare determinate espressioni all'interno del grafo del *Material* corrente.
- *Viewport Panel*: mostra una *preview* in tempo reale del *Material* applicato su di una superficie di prova. È possibile modificare la mesh visualizzata in anteprima utilizzandone una tra quelle di default oppure una *custom* selezionando una *Static Mesh* dal *Content Browser*.
- *Material Graph Panel*: contiene i nodi e la logica del *Material* similmente a quanto accade per i *Blueprints*. Di default, ogni materiale contiene un singolo *Main Material Node* costituito da una serie di input, ciascuno associato ad un particolare aspetto del materiale, a cui altri nodi possono connettersi.
- *Palette Panel*: comprende una lista, ripartita in categorie, di tutti i nodi che è possibile inserire all'interno del *Material Graph Panel*. Si possono filtrare i nodi catalogati scegliendo espressioni o funzioni nell'elenco a discesa.

- *Details Panel*: contiene una finestra delle proprietà per i nodi di *Material Expressions* o *Material Functions* selezionati. Se non è selezionato alcun nodo, vengono visualizzate le proprietà di base del *Material*.
- *Stats Panel*: include una finestra in cui vengono visualizzate le statistiche del *Material* come il numero di istruzioni *shader* utilizzate e gli eventuali errori di compilazione. Minore è il numero di istruzioni, meno costoso è il *Material*. Le *Material Expressions* che non sono collegati al *Main Material Node* non contribuiscono al conteggio delle istruzioni (costo) del *Material*.

Nel *Material Graph*, i dati fluiscono da sinistra a destra ed il *Main Material Node* è il punto finale di ogni *Material*. Esso contiene i pin di input finali che determinano quali informazioni vengono poi compilate per definire l'aspetto del *Material*. Affinché le *Material Expressions* influenzino il *Material* devono far parte di una catena che si collega al *Main Material Node*. Unreal Engine utilizza l'approccio del *Physically Based Rendering*[18] per quanto concerne la resa dei materiali. Gli attributi principali di un *PBR Material* sono direttamente correlati agli input fondamentali del *Main Material Node* e sono:

- *Base Color*: definisce il colore complessivo del *Material*. È il colore della luce diffusa sul materiale rimuovendo le riflessioni speculari.
- *Roughness*: controlla la ruvidità della superficie di un *Material*. Questo si manifesta in base a quanto nitidi o sfocati appaiono i riflessi. I materiali ruvidi disperdono la luce riflessa in più direzioni rispetto a quelli lisci, il che si traduce in riflessi diffusi.
- *Metallic*: definisce se il *Material* si comporta come un metallo o come un non-metallo. Più il valore tende ad 1 e più il materiale tende ad assumere una caratteristica puramente metallica.

[18] Il *Physically Based Rendering* (PBR) è un approccio alla computer grafica che simula l'interazione della luce con le superfici in modo da approssimarne il comportamento reale. I materiali che seguono i principi del PBR risultano più accurati e generalmente hanno un aspetto più naturale rispetto a un *workflow* di *shading* che si basa sull'intuizione dell'artista per la definizione dei parametri. I *Material* basati sulla fisica funzionano bene in tutti gli ambienti di illuminazione. Inoltre, i valori dei *Material* possono essere meno complessi e interdipendenti, con conseguente flusso di lavoro di creazione più intuitivo. Questi vantaggi sono applicabili anche per il rendering non fotorealistico, come è evidente nei film Disney.

- *Specular*: controlla la quantità di luce speculare riflessa dalla superficie. Non viene utilizzato per le mappe di riflessione o specularità o per aggiungere imperfezioni sulle superfici che dovrebbero essere altresì gestiti dalla *Roughness*.

Oltre a questi, altri importanti input del *Main Material Node* sono:

- *Emissive Color* che controlla le parti del *Material* che emettono luce e la luminosità dell'emissione.
- *Opacity* che controlla l'opacità ed è utilizzata per materiali traslucidi, additivi e modulari.
- *Normal* utilizzata per aggiungere dettagli fisici significativi alla superficie modificando la "normale", o direzione di orientamento, di ogni singolo pixel.
- *World Position Offset* che consente a un materiale di manipolare i vertici di una mesh nello spazio di riferimento globale.
- *Subsurface Color* che permette di aggiungere un colore al materiale per simulare i cambiamenti di colore quando la luce attraversa la superficie.
- *Ambient Occlusion* che viene utilizzato per aiutare a simulare il *self-shadowing*[19] che si verifica nelle fessure di una superficie.
- *Refraction* che simula l'indice di rifrazione (IOR) della superficie, utile per materiali come il vetro o l'acqua che rifrangono la luce che li attraversa.

Nel *Detail Panel* di un *Material* è possibile specificarne le proprietà. Le più importanti sono: *Material Domain* che definisce in modo in cui il materiale verrà usato, *Blend Mode* che specifica come l'output del materiale si fonderà con i pixel dietro di esso e *Shading Model* che definisce come combinare gli input del materiale per determinare il colore finale impostando il modo con cui questo interagisce con la luce in arrivo. Gli *Shading Models* più utilizzati sono: *Unlit* in cui il materiale non risponde alla luce ed è definito solo dagli input *Emissive Color* e *Opacity*,

[19] Il *self-shadowing* è un effetto di illuminazione che consente agli oggetti non statici dell'ambiente di proiettare ombre su sé stessi e gli uni sugli altri. Occorre specificare se l'ombra proiettata è statica o dinamica ed in quest'ultimo caso se ha cambiamenti di geometria all'interno di una scena. I metodi di *self-shadowing* hanno dei compromessi tra qualità e velocità a seconda del risultato desiderato.

Default Lit utilizzato per la maggior parte degli oggetti solidi in quanto interagisce con l'illuminazione diretta e indiretta e supporta le riflessioni, *Subsurface* utilizzato per la *subsurface scattering*[20] dei materiali attivando l'input *Subsurface Color*.

Una delle opzioni di *Material Domain*, sfruttata nel paragrafo 1.4 per il *wireframe shader*, è *Post Process* che serve a realizzare dei *Post Process Materials*. Questi, consentono di impostare dei materiali che possono essere utilizzati durante la post-produzione per creare degli effetti visivi sullo schermo come danni, effetti di tipo area o per modificare completamente l'aspetto generale della scena. Per produrre il nuovo colore, il relativo *Main Material Node* è composto solo dal nodo di *Emissive Color*. I *Post Process Materials* dovrebbero essere usati con parsimonia. Quando possibile, ad esempio per la correzione o le regolazioni del colore, *bloom* e profondità di campo, è preferibile usare le impostazioni inerenti al *Post Process Volume*, che sono ottimizzate e più efficienti.

I *Material*, raramente sono considerate delle risorse monouso. La realizzazione di un *Material* nuovo per ogni *Actor* potrebbe risultare estremamente inefficiente, in particolare se le caratteristiche degli *asset* sono similari e di conseguenza lo sono anche i relativi materiali. Le *Material Instances* forniscono un modo per creare rapidamente più varianti o istanze partendo da un singolo *Material* padre. Una volta che il Material padre è stato compilato, le modifiche apportate alle relative istanze non comportano un ulteriore fase di compilazione e vengono applicate in tempo reale. Le istanze vengono utilizzate quando un gruppo di *asset* correlati richiede lo stesso *Material* di base, ma con caratteristiche superficiali diverse. Queste vengono in genere specificate tramite dei parametri di tipo scalare, vettoriale, statico o texture. Tali parametri possono essere modificati manualmente per ogni istanza oppure cambiati dinamicamente a *runtime* tramite *Blueprints*. Le *Material Instances* presentano notevoli vantaggi sia per quanto riguarda l'ottimizzazione della memoria del progetto che durante il flusso di lavoro degli artisti dediti

[20] La *subsurface scattering* è una proprietà tipica dei materiali traslucidi come foglie, cera, pelle, che a differenza dei materiali completamente opachi si lasciano attraversare dalla luce che viene dispersa più volte ad angoli irregolari all'interno del materiale prima di uscire con un angolo diverso da quello che avrebbe avuto se fosse stata riflessa direttamente dalla superficie.

all'aspetto visivo dell'opera in quanto vengono accorciati i tempi necessari alla realizzazione di materiali simili con la stessa struttura di nodi e caratteristiche parametriche.

1.1.3. Il Niagara System

Il *Niagara System* è un sistema all'avanguardia per realizzare VFX avanzati su Unreal Engine. Uscito in beta nella versione 4.2 di Unreal Engine, ha sostituito il precedente sistema particellare *Cascade* offrendo al suo posto un sistema adattabile ad ogni esigenza. Il *Niagara System* veste un ruolo di primo piano nelle applicazioni immersive, permettendo di realizzare effetti visivi che interagiscono agli input degli utenti, nonché simulazioni naturali per ambienti realistici.

Niagara è in grado di gestire grandi quantità di particelle senza dover sacrificare prestazioni. Inoltre, gli sviluppatori possono integrare il sistema di VFX tramite *Blueprints* o script personalizzati per ottenere un maggior grado di interattività.

Costruito per essere un sistema flessibile che presenta un approccio modulare alla progettazione di VFX, *Niagara* offre il controllo totale sulla simulazione e sul rendering delle particelle. È particolarmente scalabile in quanto i principianti possono modificare modelli predefiniti, mentre gli utenti avanzati possono creare i propri moduli.

Ciò che rende *Niagara* estremamente personalizzabile è che gli emettitori di particelle possono avere attributi unici che sono arbitrariamente collegati tra loro all'interno di un sistema di particelle. Questo consente di gestire diversi aspetti di una simulazione VFX indicizzando gli attributi ad una caratteristica scelta. Si possono utilizzare degli input dinamici per collegare un attributo della particella come accelerazione, colore, dimensione, orientamento a qualsiasi dato arbitrario come la velocità delle particelle, la distanza tra due oggetti o il movimento di un osso su una mesh scheletrica.

Nel sistema *Niagara* si hanno quattro componenti fondamentali:

- *Emitter*: è l'unità base che genera le particelle. Un emettitore controlla il modo in cui nascono le particelle, cosa succede a queste man mano che invecchiano, come appaiono e come si comportano.
- *Module*: blocco che controlla un aspetto specifico delle particelle, come *spawn*, movimento o rendering, e viene creato usando *High-Level Shading Language* o tramite l'interfaccia a nodi. In quest'ultima è comunque possibile inserire del codice HLSL impiegando il nodo *CustomHLSL*.
- *System*: è un insieme di uno o più *Emitter* e definisce comportamenti del VFX a livello di sistema. Può essere posizionato all'interno di una scena. I *Modules* che fanno parte di *System* vengono eseguiti per primi, gestendo il comportamento condiviso con ogni emettitore.
- *Parameters*: astrazione di dati utilizzati all'interno di una simulazione per avere una rappresentazione quantitativa sul comportamento delle particelle.

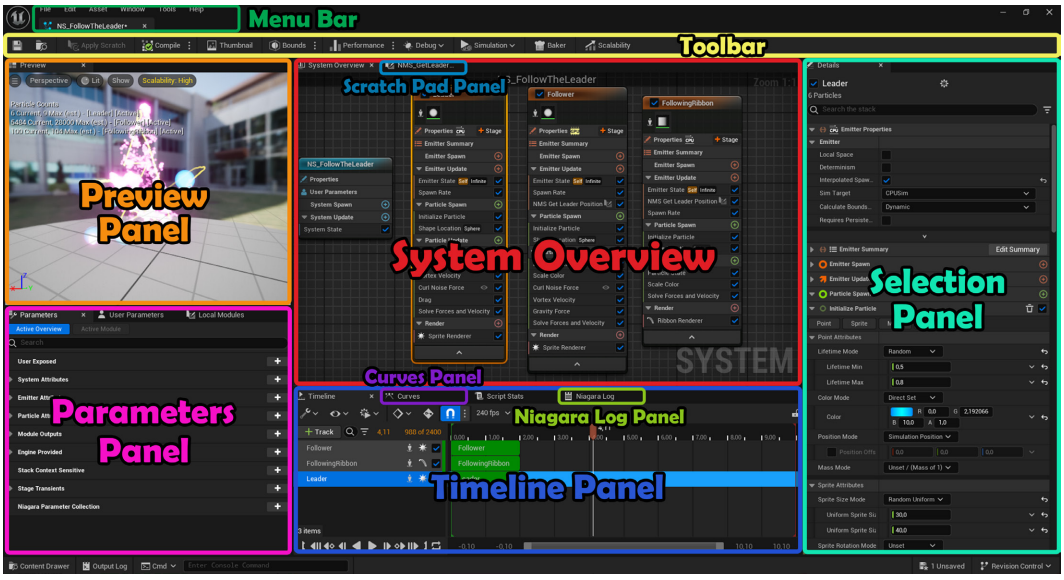

Figura 1.4. User Interface del Niagara Editor.

Dalla Figura 1.4, l'interfaccia grafica del *Niagara Editor* si compone principalmente dei seguenti elementi:

- *Menu Bar*: classica barra dei menu con le principali opzioni di gestione dei file e delle finestre dell'editor.

- *Toolbar*: contiene i principali strumenti usati all'interno dell'editor come l'operazione di compilazione, la generazione di anteprime, la visualizzazione delle *bounding box*[21] del sistema particellare, le statistiche di performance e gli strumenti per il debug e la simulazione.
- *Preview Panel*: mostra un'anteprima del VFX utile per tenere traccia della resa dell'effetto durante lo sviluppo. Permette inoltre di mostrare statistiche e cambiare la modalità di visualizzazione.
- *Parameters Panel*: elenca tutti i parametri esposti dall'utente o forniti dal sistema, dall'emettitore, dalle particelle o dal motore (chiamati anche attributi) che vengono utilizzati dall'emettitore o dal sistema selezionato mostrando anche il numero di volte in cui questi sono referenziati e dando la possibilità di crearne di nuovi.
- *System Overview*: fornisce una panoramica di alto livello del sistema o dell'emettitore in fase di modifica, abbinando una vista grafica a nodi ad una versione compatta degli *stack* di sistema e dell'emettitore.
- *Scratch Pad Panel*: finestra che permette di costruire dei moduli personalizzati o input dinamici tramite scripting. Consente anche di creare qualcosa di specifico per l'emettitore o per il sistema attivo e non deve essere trasformato in un *asset* a sé stante.
- *Timeline Panel*: consente di gestire il comportamento di ogni *Emitter* al variare del tempo come verificare i loop, conteggiare le ripetizioni, i *burst*, la frequenza di *spawn* e gli arresti casuali.
- *Curves Panel*: fornisce un *Curve Editor* per regolare l'evoluzione dei valori tramite una curva durante il ciclo di vita di una particella o di un emettitore. Affinché una proprietà possa essere modificabile all'interno del *Curve Editor*, deve avere un tipo di distribuzione che utilizza una curva.
- *Niagara Log Panel*: permette di visualizzare eventuali avvisi o errori che possono susseguirsi durante la compilazione di uno script, di un emettitore o di un sistema.

[21] Con il termine *bounding box* si indica la più piccola "scatola invisibile" rettangolare capace di contenere interamente un oggetto in uno spazio di lavoro digitale.

- *Selection Panel*: consente di mostrare i dettagli degli stack di sistema e dell'emettitore selezionato permettendo la modifica dei valori, l'aggiunta di nuovi moduli e l'inserimento di input dinamici o altri attributi.

Nel *Niagara System* gli emettitori seguono un approccio di tipo *stack-oriented* in cui, come in una sorta di pila, vi sono una serie di fasi all'interno delle quali è possibile inserire dei moduli che, impilati uno sull'altro, vanno a costituire il comportamento del sistema di particelle finale. Le fasi (o *stages*) degli emettitori con i rispettivi moduli vengono eseguite in ordine dall'alto verso il basso e quelle comuni a tutti gli *Emitter* sono:

- *Emitter Spawn*: definisce cosa avviene non appena un emettitore viene creato dalla CPU per la prima volta. All'interno compaiono le impostazioni di default nonché quelle iniziali e di rado vengono inseriti i *Modules*.
- *Emitter Update*: regola il comportamento delle particelle durante il loro ciclo di vita, fornendo i *Modules* che vengono richiamati ad ogni frame dalla CPU. Tra i più comuni si trovano *Spawn Rate*, per la creazione di particelle in modo continuo a una velocità specifica, ed *Emitter State* per impostare i comportamenti ciclici dell'emettitore.
- *Particle Spawn*: fase chiamata una sola volta per particella, precisamente quando questa nasce. Qui, all'interno del modulo *Initialize Particle* vengono definiti i dettagli riguardanti l'inizializzazione delle particelle, come la posizione in cui nascono, il colore, la dimensione, il tempo di vita. Tra gli altri moduli che possono essere inseriti si annoverano: *Add Velocity* per assegnare una velocità iniziale alla particella, S*pawn Beam* per posizionare le particelle lungo una curva di Bézier o una linea tra due punti, *Shape Location* per generare particelle dalla superficie di una primitiva come un piano o una sfera.
- *Particle Update*: chiamato ad ogni frame, per ogni particella. Vengono settate le proprietà delle particelle che riguardano il loro ciclo di vita. Tramite il modulo *Particle State* si eliminano le particelle che hanno terminato la loro vita, con *Solve Forces and Velocity* vengono gestite le proprietà fisiche delle particelle e applicate modifiche agli attributi

intrinseci rendendoli indipendenti dal frame rate, con *Collision* si simula la fisica di collisione delle particelle. Altri moduli usati sono: *Curl Noise Force* per aggiungere una forza con un pattern *curl noise*[22] al moto delle particelle, *Drag* usato per limitare la velocità delle particelle simulando una resistenza fluidodinamica, *Gravity Force* per aggiunge una forza gravitazionale alle particelle, *Scale Color* per scalare separatamente le componenti RGB e Alpha delle particelle.

- *Event Handler*: determina come un emettitore risponde agli eventi in arrivo. Questi possono essere utilizzati solo con simulazioni CPU ed occorre che sia presente almeno un *Event Module* nella fase di *Particle Update* dell'emettitore che sta generando l'evento. Ad esempio, se si vuole che le particelle dell'emettitore B seguano quelle dell'emettitore A, bisogna posizionare un *Generate Location Event* nel *Particle Update* dell'emettitore A, un *Event Handler* nell'emettitore B con un *Receive Location Event* per ascoltare l'evento di posizione dell'emettitore A.
- *Render*: viene scelto il modo con cui verranno visualizzate le particelle impostando uno o più *Renderer*. In base al tipo di render scelto, si hanno costi computazionali differenti. Con *Mesh Renderer* si può definire un modello 3D come base delle particelle su cui applicare un materiale mentre con *Sprite Renderer* le particelle vengono renderizzate come fossero uno *sprite* bidimensionale. Si possono combinare più *Renderer* all'interno di un singolo *Emitter*.

Una volta creato un *Emitter*, un ruolo di primaria importanza lo ricopre la scelta della destinazione di calcolo, nota come *Simulation Target*, che determina quale processore (CPU o GPU) debba eseguire la simulazione particellare. Nella maggior parte dei casi, le simulazioni GPU sono più performanti e consentono di gestire un numero maggiore di particelle ma sono più adatte ad effetti visivi statici con calcoli semplici e paralleli. Per gli emettitori con un numero ridotto di particelle, in cui si

[22] Con *curl* (rotore in italiano) si denota un'operazione applicata al rumore di input. Anche vari input come campi vettoriali o texture possono essere sottoposti al trattamento *curl*. Se il rotore del campo vettoriale funge da campo di flusso, è possibile imitare il comportamento delle particelle come se fossero sospese in un fluido turbolento.

necessita al tempo stesso una elevata interattività e gestione di calcoli complessi, potrebbe essere più adatta una simulazione CPU.

Un'ulteriore funzionalità avanzata è data dai *Dynamic Input* ovvero dei moduli che possono essere collegati ai parametri di un *Emitter* o di un *System* fornendo degli input variabili calcolati in tempo reale che aggiungono comportamenti dinamici e personalizzabili alle particelle. Gli utilizzi più frequenti dei *Dynamic Input* includono la generazione di range di numeri casuali, la scomposizione di vettori e operazioni matematiche. Unica accortezza è valutare un *trade off* sulle prestazioni poiché calcoli complessi possono ridurre il frame rate durante la simulazione.

1.2. Skybox dinamico

Il primo dei lavori preliminari che viene analizzato in questo capitolo riguarda la generazione di *skybox*[23] in maniera dinamica tramite prompt collegati ad un'intelligenza artificiale.

L'idea iniziale era quella di realizzare, in una delle sale del Museo Egizio, un'esperienza del tutto interattiva in cui l'utente, con un comando vocale, richiedeva un paesaggio da visionare e, in tempo reale, il software ne generava uno in base alle specifiche e lo proiettava a 360° nelle pareti della sala.

Questo concept, dati i problemi logistici di lavorare sin da subito nelle sale immersive del Museo Egizio, è stato successivamente accantonato in favore di un filmato multimediale renderizzato *offline* confluito nel progetto *Paesaggi - Landscapes*. Ciò nonostante, ne verrà descritta in seguito una sua possibile implementazione sfruttando il motore grafico Unreal Engine, lo strumento Skybox

[23] Uno *skybox* è una tecnica che prevede la proiezione di un ambiente sulle pareti interne di un cubo, creando l'illusione di un ampio spazio tridimensionale. Nell'uso nelle scene 3D, lo skybox rimane fisso rispetto alla prospettiva, creando l'illusione che gli oggetti al suo interno siano situati a una distanza infinita, poiché non manifestano alcun movimento di parallasse.

AI[24] di Blockade Labs® e adattando il plugin Skybox Generator[25] per gli usi che ne conseguono.

Il processo che consente di generare un nuovo skybox lo si può suddividere in quattro passi fondamentali. Il primo di questi riguarda la richiesta http POST che viene fatta al server di Blockade Labs. Alla richiesta vengono specificati i seguenti parametri:

- *API Key*: chiave personale associata all'abbonamento del servizio.
- *Skybox Style ID*: utilizzato per scegliere la versione del modello e influenzare l'estetica generale della generazione dello skybox.
- *Prompt*: descrizione di come deve essere lo skybox che si vuole generare.
- *Webhook URL*: opzionalmente si può inserire un *webhook*[26] per tenere traccia degli stati di avanzamento della richiesta.
- *Negative Text*: descrizione delle cose che non si desiderano nello skybox.
- *Seed*: seme per la generazione dello skybox, con 0 si ha una generazione casuale.
- *Remix Image ID*: identificativo dello skybox originato da cui si vuole partire per la generazione.
- *Return Depth*: flag per richiedere la mappa di profondità.

Tutti questi parametri vengono confezionati in una stringa JSON e spediti al web service di generazione dello skybox. In Figura 1.5 è riportato lo script che ne gestisce la procedura.

[24] Skybox AI è uno strumento per generare skybox a 360° utilizzando l'intelligenza artificiale. Esso consente di creare sfondi panoramici che possono essere utilizzati nello sviluppo di videogiochi, esperienze VR e progetti artistici. Per generare uno skybox, gli utenti inseriscono una descrizione testuale e l'IA produce un'immagine adatta alla descrizione.
[25] Plugin sviluppato da Adem Kilic per la versione 4.27 di Unreal Engine che consente di generare skybox usando il web service di Blockade Labs.
[26] I *webhook* rappresentano un metodo attraverso il quale un'applicazione può fornire informazioni in tempo reale ad altre applicazioni. Essi permettono di inviare una notifica quando si verifica un determinato evento, evitando la necessità di eseguire continue richieste per ottenere nuovi dati. Spesso sono definiti come API "inverse", poiché inviano i dati piuttosto che estrarli.

```
void URestApiBlockadelabs::GenerateSkybox(FString ApiKey, FString Prompt, int32 SkyboxStyleId, FString
WebhookUrl, const FOnGenerateSkyboxSuccess& OnSuccessCallback, const FGenerateSkyboxError& OnErrorCallback,
FString NegativeText, int32 Seed, int32 RemixImagineId, bool bReturnDepth){
    TSharedRef<IHttpRequest, ESPMode::ThreadSafe> Request = FHttpModule::Get().CreateRequest();
    Request->SetURL("https://backend.blockadelabs.com/api/v1/skybox");
    Request->SetVerb("POST");
    Request->SetHeader("Content-Type", "application/json");
    FString JsonString;
    TSharedRef<TJsonWriter<>> Writer = TJsonWriterFactory<>::Create(&JsonString);
    Writer->WriteObjectStart();
    Writer->WriteValue(TEXT("api_key"), ApiKey);
    Writer->WriteValue(TEXT("prompt"), Prompt);
    Writer->WriteValue(TEXT("skybox_style_id"), SkyboxStyleId);
    Writer->WriteValue(TEXT("webhook_url"), WebhookUrl);
    if (!NegativeText.IsEmpty()) Writer->WriteValue(TEXT("negative_text"), NegativeText);
    if (Seed != 0) Writer->WriteValue(TEXT("seed"), Seed);
    if (RemixImagineId != 0) Writer->WriteValue(TEXT("remix_imagine_id"), RemixImagineId);
    if (bReturnDepth) Writer->WriteValue(TEXT("return_depth"), bReturnDepth);
    Writer->WriteObjectEnd();
    Writer->Close();
    Request->SetContentAsString(JsonString);
    Request->OnProcessRequestComplete().BindLambda([OnSuccessCallback, OnErrorCallback]
    (FHttpRequestPtr Request, FHttpResponsePtr Response, bool bWasSuccessful){
        if (!bWasSuccessful || !Response.IsValid()){
            OnErrorCallback.ExecuteIfBound(TEXT("Failed to generate skybox. Network error."));
            return; }
        if (Response->GetResponseCode() != 200){
            FString ErrorMessage = FString::Printf(TEXT("Failed to generate skybox. HTTP status code: %d.
            Error message: %s"), Response->GetResponseCode(), *Response->GetContentAsString());
            OnErrorCallback.ExecuteIfBound(ErrorMessage);
            return; }
        TSharedPtr<FJsonObject> JsonObject;
        TSharedRef<TJsonReader<>>JsonReader = TJsonReaderFactory<>::Create(Response->GetContentAsString());
        FString SkyboxUrl;
        int32 SkyboxId;
        if (FJsonSerializer::Deserialize(JsonReader, JsonObject)){
            SkyboxUrl = JsonObject->GetStringField("obfuscated_id");
            SkyboxId = JsonObject->GetIntegerField("id");
            OnSuccessCallback.ExecuteIfBound(SkyboxUrl, SkyboxId);
            return; }
        else {
            OnErrorCallback.ExecuteIfBound(TEXT("Failed to parse the response JSON.")); }
    });
    Request->ProcessRequest();
}
```

Figura 1.5. Script che gestisce la richiesta di uno skybox al web service.

Una volta che il server risponde, si definiscono due *delegate*[27] che fungono da *callback* chiamate in caso di successo o di errore nell'esecuzione della richiesta. Nell'esempio mostrato nel Blueprint in Figura 1.6, tramite la pressione di un tasto si fa partire la procedura di richiesta dello skybox. Come si può notare, il prompt che ne descrive la natura è inserito come input della funzione Generate Skybox in modo manuale. Una ulteriore implementazione per soddisfare le specifiche della sala immersiva sarebbe quello di generarlo tramite un comando vocale e passarlo successivamente alla funzione che si avvia senza la pressione di un tasto.

[27] Particolari tipi di dato in grado di "contenere" un metodo, ossia una procedura o una funzione. In Unreal Engine vengono spesso usati nelle *callback*, chiamate, ad esempio, quando si fa qualcosa sull'interfaccia utente.

Nel caso in cui la procedura dovesse fallire per errore di rete, del server o di formattazione del JSON, viene richiamato l'evento "Error_1" che mostra a schermo il tipo di errore.

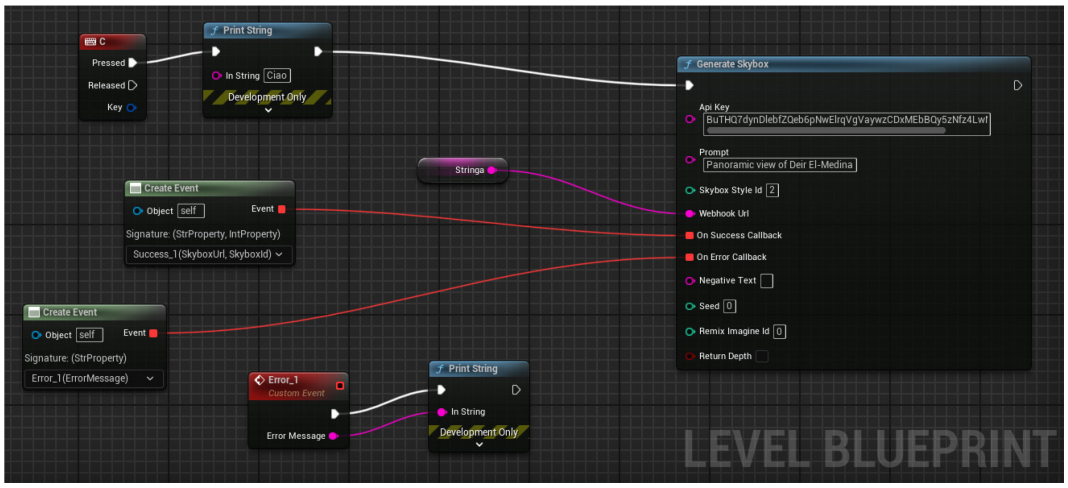

Figura 1.6. Blueprint per la richiesta di uno skybox.

In caso di successo della richiesta, il passaggio successivo riguarda l'acquisizione della URL dell'immagine generata. Nel progetto questa viene implementata con una http GET fatta al web server di Blockade Labs. Alternativamente può essere gestita tramite dei *webhook* che tengono traccia dello stato della richiesta.

Il problema principale di questo step è dovuto al tempo necessario al server per elaborare la richiesta di generazione dello skybox. Questa inizialmente parte da uno stato di "*pending*" e la risposta data in quel momento dal server contiene diverse informazioni ma non l'URL dell'immagine che è ancora in fase di generazione. Tra i dati che possono essere ottenute da questa risposta vi è l'ID della richiesta, utile per tenere traccia del processo di generazione. Una volta trascorso un certo lasso di tempo, la generazione raggiunge lo stato "*complete*" e facendo una richiesta di immagine tramite ID al web server è possibile ottenere una struttura dati che contiene tutte le informazioni necessarie, inclusa l'URL dell'immagine. In Figura 1.7 viene riportato lo script che esegue la richiesta per ID e inserisce in un *array* di dati le informazioni delle immagini che soddisfano un particolare identificativo ed una API key.

```
void URestApiBlockadelabs::GetImagineByID(FString ApiKey, FString ObfID, const FOnGetImagineHistorySuccess&
OnSuccessCallback, const FGetImagineHistoryError& OnErrorCallback){
    TSharedRef<IHttpRequest, ESPMode::ThreadSafe> Request = FHttpModule::Get().CreateRequest();
    Request->SetURL(FString::Printf(TEXT(
   "https://backend.blockadelabs.com/api/v1/imagine/myRequests?api_key=%s&imagine_id=%s"),*ApiKey,*ObfID));
    Request->SetVerb("GET");
    Request->OnProcessRequestComplete().BindLambda([OnSuccessCallback, OnErrorCallback]
    (FHttpRequestPtr Request, FHttpResponsePtr Response, bool bWasSuccessful){
        if (!bWasSuccessful || !Response.IsValid()){
            OnErrorCallback.ExecuteIfBound(TEXT("Failed to get my requests. Network error."));
            return; }
        if (Response->GetResponseCode() != 200){
            FString ErrorMessage = FString::Printf(TEXT("Failed to get my requests. HTTP status code: %d.
            Error message: %s"), Response->GetResponseCode(), *Response->GetContentAsString());
            OnErrorCallback.ExecuteIfBound(ErrorMessage);
            return; }
        TArray<FMyRequestData> RequestsData;
        TSharedPtr<FJsonObject> JsonObject;
        TSharedRef<TJsonReader<>>JsonReader = TJsonReaderFactory<>::Create(Response->GetContentAsString());
            if (FJsonSerializer::Deserialize(JsonReader, JsonObject)){
                const TArray<TSharedPtr<FJsonValue>>* RequestsArray = nullptr;
                if (JsonObject->TryGetArrayField("data", RequestsArray)){
                    for (const TSharedPtr<FJsonValue>& JsonValue : *RequestsArray){
                        if (JsonValue->Type == EJson::Object){
                            const TSharedPtr<FJsonObject>& JsonObj = JsonValue->AsObject();
                            FMyRequestData RequestData;
                            RequestData.Id = JsonObj->GetIntegerField("id");
                            RequestData.Title = JsonObj->GetStringField("title");
                            RequestData.Prompt = JsonObj->GetStringField("prompt");
                            RequestData.Status = JsonObj->GetStringField("status");
                            RequestData.file_url = JsonObj->GetStringField("file_url");
                            RequestData.thumb_url = JsonObj->GetStringField("thumb_url");
                            FString DepthMapUrlValue;
                            if (JsonObj->TryGetStringField("depth_map_url", DepthMapUrlValue)){
                                RequestData.depth_map_url = DepthMapUrlValue; }
                            else {
                                RequestData.depth_map_url = ""; }
                            RequestsData.Add(RequestData); } } }
                OnSuccessCallback.ExecuteIfBound(RequestsData); }
            else {
                OnErrorCallback.ExecuteIfBound(TEXT("Failed to parse the response JSON.")); }
        });
    Request->ProcessRequest();
}
```

Figura 1.7. Script per richiedere un'immagine dato un ID al web service.

Quindi, come descritto nella Figura 1.8, se nella richiesta di generazione dello skybox viene chiamata la "Success_1", dopo aver atteso un tempo pari a 3 secondi viene eseguita la funzione descritta sopra. A questa fanno riferimento due possibili eventi: "Error_2" che segue fedelmente le casistiche di "Error_1" e "Success_2" che copre il caso in cui la richiesta restituisca un *array* di dati del formato corretto.

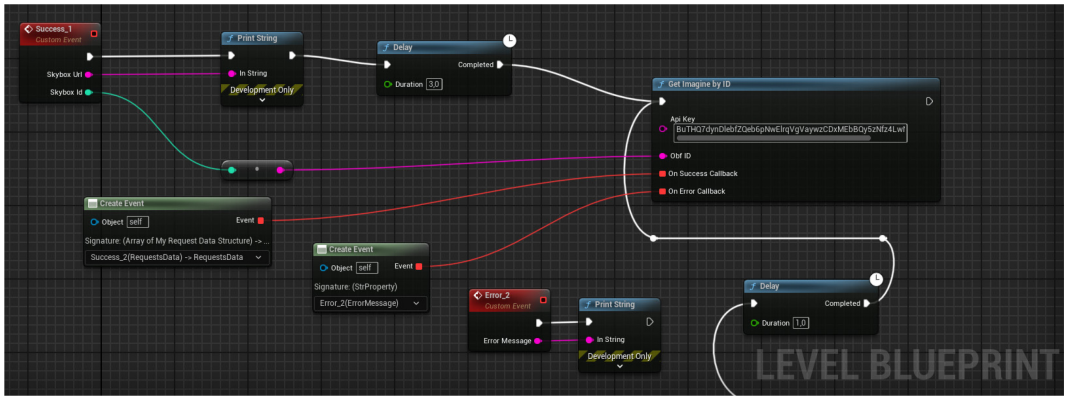

Figura 1.8. Blueprint per la richiesta di un'immagine dato un ID.

A questo punto, dalla lista di dati, si estrapolano le singole strutture e si controlla se qualcuna ha nello stato la dicitura "*complete*". Se nessuna di queste soddisfa la condizione, viene richiamata nuovamente la funzione di richiesta immagine per ID dopo aver atteso un tempo di 1 secondo. Non appena una struttura soddisfa i requisiti, si estrapola l'URL dell'immagine dal blocco dati e si esegue il download dell'immagine. In Figura 1.9 sono evidenziati i nodi necessari per effettuare quanto descritto.

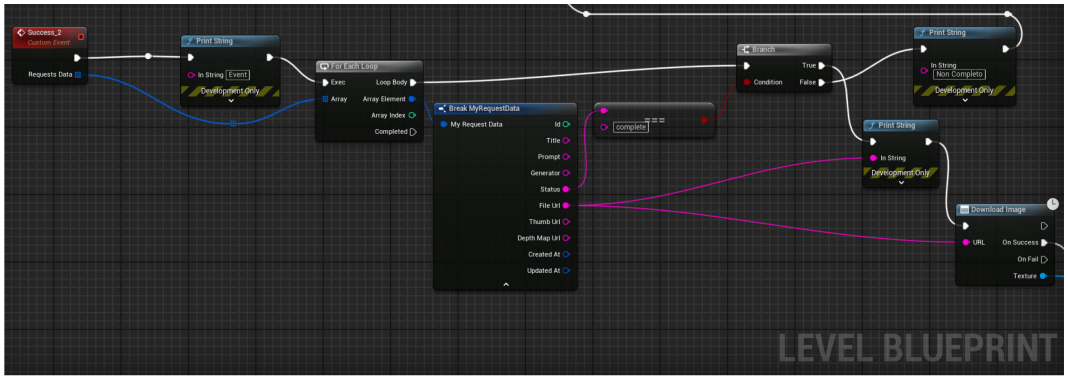

Figura 1.9. Blueprint per il download dell'immagine.

L'ultimo passaggio consiste nel fare in modo che l'immagine venga sostituita allo skybox attuale. Per fare ciò, nella scena viene inserito un Actor *Space_BP* che ha come Component lo skybox utilizzato. Di questo, ne viene preso un riferimento del materiale e sostituita la texture con l'immagine scaricata. Al materiale viene poi conferita una fonte emissiva in relazione al *Linear Color* dell'immagine in modo che l'illuminazione globale possa andar bene per la resa finale della scena.

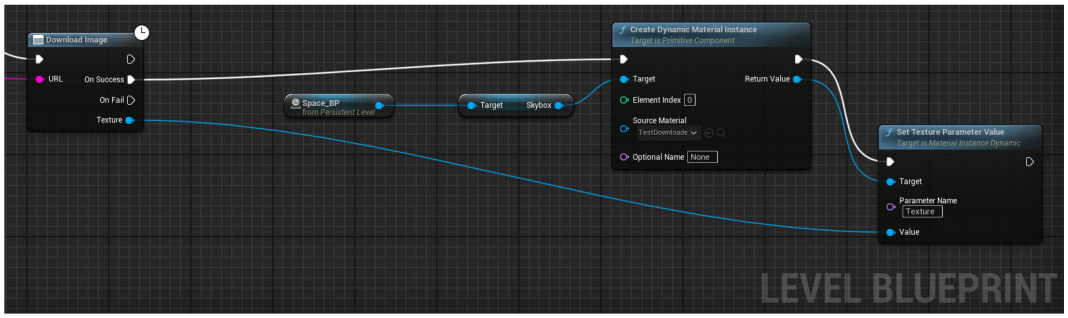

Figura 1.10. Blueprint per l'applicazione dell'immagine come skybox.

In generale, facendo tutte le ottimizzazioni del caso, il tempo per eseguire tutta la procedura è di circa 5-10 secondi e dipende molto dal carico del server e dalla durata

in cui la richiesta rimane in coda. Ne consegue che, usando il servizio di IA generativa, la velocità di risposta è per certi versi vincolata da vari fattori che inficiano sull'efficienza del processo di generazione. Usare pipeline e modelli ottimizzati per il rendering riduce i tempi di calcolo necessari per ottenere immagini senza artefatti o distorsioni ma comunque si è ancora lontani dall'avere una risposta *real-time* o rapida come si pensa. Ciò nonostante, non si può fare a meno di notare come con questo tipo di tecnologie generative, ancorché nelle fasi iniziali, si riescano a raggiungere risultati sorprendenti e di grande impatto visivo. Oltre ad ampliare le possibilità espressive e concettuali, le IA generative possono offrire nuovi strumenti agli artisti e creativi, contribuendo a rendere l'arte digitale più accessibile e ad essere un supporto essenziale per il restauro del patrimonio artistico.

In Figura 1.11 viene mostrato un esempio di skybox generato partendo dal prompt "*Vista di Deir el-Medina*". Come si può notare, l'immagine non è propriamente fedele al villaggio di Deir el-Medina per come lo si conosce attualmente bensì è una libera interpretazione creativa del prompt dato in input al modello di IA allenato con le informazioni presenti sul web.

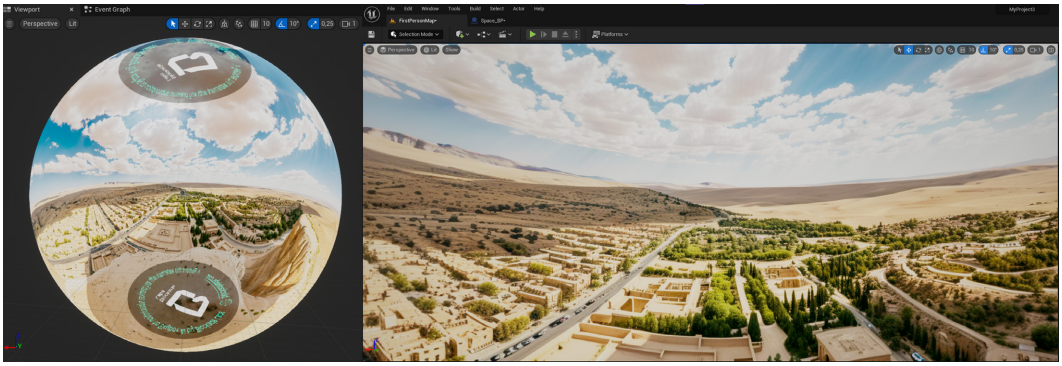

Figura 1.11. Skybox risultate usando il prompt "Vista di Deir el-Medina".

1.3. Rain FX

Nelle fasi iniziali di *brainstorming* del progetto si era discusso di come implementare effetti particellari all'interno di quegli scenari che includevano i fenomeni meteorologici. In particolare, per la seconda sala immersiva del Museo

Egizio, che riguardava le stagioni dell'antico Egitto influenzate dal fiume Nilo, vi era la necessità di integrare i dati climatici provenienti da un database alle effettive scene che mostravano in tempo reale il mutamento delle stagioni.

Uno degli aspetti che poteva manifestarsi nella rappresentazione del cambiamento climatico, riguardava l'eventuale simulazione delle precipitazioni più o meno intense collegate in via parametrica ai dati forniti dall'esterno. Si descriverà di seguito una possibile implementazione pensata per il VFX della pioggia usando Unreal Engine ed in particolare il Niagara System.

Per la realizzazione dell'effetto della pioggia si parte dall'analisi della singola goccia d'acqua. In generale, per creare un materiale da applicare ad un liquido trasparente si ha bisogno di una *normal map* in modo da simulare l'aspetto tridimensionale di una particella sferica a causa della tensione superficiale dell'acqua e di una *mask map* per distribuire maggiormente la rifrazione nei bordi delle particelle. Entrambe le due texture sono state recuperate da un *asset pack*[28] per VFX in Unreal Engine e vengono di seguito citati con l'appellativo di "LiquidNormalMap" e "LiquidMask".

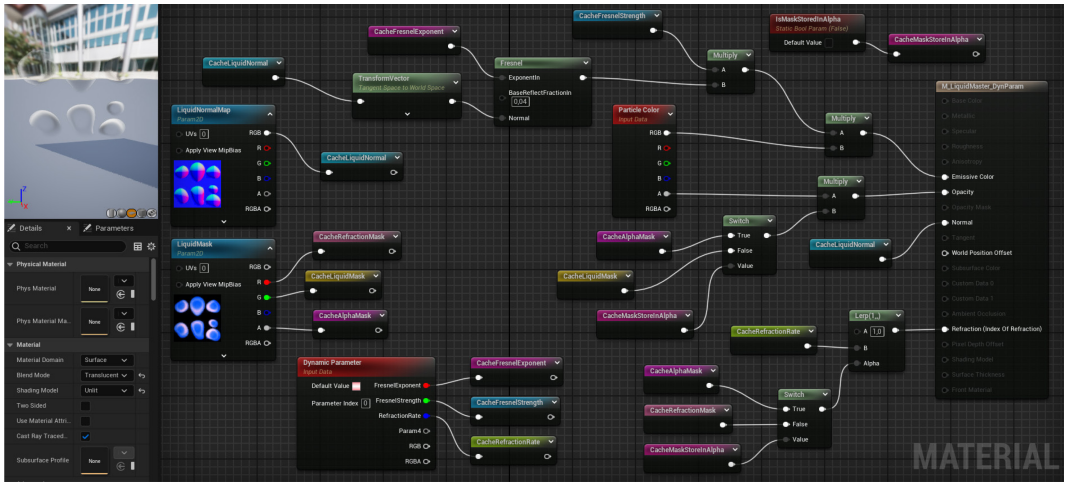

Figura 1.12. Material Graph della goccia di pioggia.

[28] Il pacchetto di *asset* si chiama "Realistic Starter VFX Pack Vol. 2" ed è disponibile in forma gratuita ad uso personale sul marketplace di Epic Games.

Come mostrato in Figura 1.12, al Material si è scelto un metodo di fusione *translucent* ed un modello di *shading*[29] di tipo *unlit*. Inoltre, nella sua composizione si è preferito di utilizzare dei parametri dinamici per avere la possibilità di gestire i dati in modo che ad ogni goccia di pioggia si possano attribuire dei valori casuali.

Per gestire i dati relativi al colore delle gocce d'acqua è stato inserito un nodo di tipo *Fresnel*[30] controllandolo mediante parametri dinamici. In questo modo è possibile modificare il valore *ExponentIn*[31] del nodo direttamente all'interno del Niagara System. Nel valore *Normal* viene collegata, tramite trasformazione vettoriale, la LiquidNormalMap. Impiegando una *normal map* si influenza il modo in cui l'effetto Fresnel viene distribuito sul materiale. Tecnicamente, il nodo *Fresnel* funziona verificando se la normale alla superficie è perpendicolare alla telecamera ed in quel caso l'effetto è visibile. Ciò significa che l'effetto Fresnel su una sfera liscia, si verifica solo sui bordi. Tuttavia, quando viene introdotta una *normal map*, le normali alla superficie vengono modificate in modo tale che possano esserci ondulazioni e contorni all'interno della silhouette di una mesh. Ciò significa che l'effetto Fresnel può evidenziare o enfatizzare dettagli che non sarebbero visibili se la superficie apparisse liscia.

Per fare in modo che ogni goccia di pioggia avesse un tasso di rifrazione diverso, è stato utilizzato un parametro dinamico con un valore predefinito di 1,33. È stato ottenuto un effetto di rifrazione più intenso nei bordi della goccia eseguendo un'interpolazione lineare di questo parametro con il valore 1[32] e usando come

[29] Nella computer grafica, uno *shader* è un insieme di algoritmi che conferiscono al materiale a cui viene applicato i livelli appropriati di luce, oscurità e colore durante il rendering di una scena 3D.
[30] Fresnel è il termine utilizzato per descrivere il modo in cui la luce si riflette a diverse intensità in base all'angolo di visuale. In Unreal Engine, la *Fresnel Material Expression*, racchiusa nel nodo *Fresnel*, calcola un *falloff* in base al prodotto scalare tra la normale alla superficie e la direzione verso la camera. Quando la normale alla superficie punta direttamente la camera, viene emesso un valore pari a 0, il che significa che non dovrebbe verificarsi alcun effetto Fresnel. Quando la normale alla superficie è perpendicolare alla camera, viene emesso un valore pari a 1, il che significa che dovrebbe verificarsi l'effetto Fresnel completo. Utilizzando Fresnel, è possibile simulare l'illuminazione dei bordi nel materiale di un oggetto ottenendo un maggiore controllo sull'aspetto e sulla sensazione dell'effetto.
[31] Il valore *ExponentIn* della *Fresnel Material Expression* controlla lo smorzamento dell'effetto Fresnel mentre *BaseReflectFractionIn* specifica la frazione di riflessione speculare quando la superficie è vista di fronte.
[32] Nei punti in cui l'interpolazione dà come esito "1" non si ottiene alcun effetto di rifrazione.

maschera di rifrazione il canale rosso della LiquidMask. Tramite il canale verde della stessa texture è stata invece regolata l'opacità del materiale.

Una volta ottenuto il materiale di base, sono state realizzate cinque istanze che simulano, oltre alle gocce, le increspature dell'acqua e varie tipologie di schizzi prodotti dalla pioggia quando arriva a collidere con il terreno. Le texture parametrizzate sono state sostituite con altre ottenute dallo stesso *asset pack* usato in precedenza ed è stato utilizzato un valore booleano per specificare di volta in volta se per la maschera di rifrazione e di regolazione dell'opacità vengono impiegati i canali di default del materiale di base o il canale alpha della LiquidMask.

Figura 1.13. Niagara System dell'effetto di pioggia.

Le istanze del materiale sono state successivamente sfruttate nel Niagara System per generare l'effetto della pioggia. Dalla Figura 1.13, si nota come questo sistema particellare si compone di ben cinque *Emitter*:

- *Main Rain Drops*: serve a rappresentare le singole gocce d'acqua. Avendo uno *sprite* di partenza composto da 3x2 variazioni di gocce, all'emettitore è stata applicata una *Sub UV* ed un modulo di *Sub UV Animation* in modo da ottenere un'animazione lineare in cui ad ogni frame viene switchata una variazione della goccia. Il tempo di vita delle singole particelle è stato settato in maniera randomica nel range tra 1 e 2 secondi mentre il tasso di generazione delle gocce è stato reso dipendente dal parametro *SpawnRate* impostato dall'utente. Seguendo gli scopi preposti per l'esperienza immersiva che verrà riprodotta in una delle nuove sale del Museo Egizio, tale parametro lo si può collegare a valori ottenuti da un database esterno di

informazioni climatiche. Un secondo valore parametrico è *PlaneSize*, un vettore bidimensionale che conserva la dimensione del piano presso cui le particelle si generano. Per conferire al flusso di particelle una forza gravitazionale che le attiri verso il basso è stato inserito il modulo *Gravity Force*. Inoltre, per far somigliare le particelle a delle gocce di pioggia, per ognuna di queste è stata modificata la dimensione iniziale attribuendo una forma più allungata ed è stato aggiunto un fattore di scala in modo che la grandezza aumenti con l'invecchiamento. Tramite il modulo *Dynamic Material Parameters* sono stati modificati la forza e l'esponente di Fresnel e il tasso di rifazione, assegnando loro un intervallo casuale di possibili valori. Per conoscere l'istante in cui le gocce di pioggia entrano in collisione con la superficie, nella fase di *Particle Update* è stato inserito il modulo *Collision* e, per evitare che queste particelle possano rimbalzare sulla superficie, si fa terminare il loro ciclo di vita col modulo *Kill Particles*, una volta che la collisione ha dato esito positivo. Tramite un *Generate Collision Event* si fa in modo che gli altri *Emitter* possano ricevere un evento di collisione.

- *Rain Splash Small*: viene usato per rappresentare un piccolo schizzo della goccia d'acqua quando collide col terreno. Per ottenere una resa realistica è stata corretta la dimensione dello *sprite* moltiplicando il vettore bidimensionale per uno scalare randomico compreso tra un range di valori. Inoltre, è stato corretto il tempo di vita ed il colore delle particelle, e infine aggiunto il modulo di *Sub UV Animation* per generare l'animazione tipica di uno schizzo avente come sorgente le fasi riportate nello *sprite*. Sono stati altresì modificati i parametri dinamici del materiale per adattarli alle nuove caratteristiche delle particelle prodotte. Tramite l'*Event Hadler* che riceve l'informazione di collisione delle particelle del *Main Rain Drops*, si è fatto in modo che queste possano essere generate solamente nel momento di ricezione dell'evento e di una quantità casuale per frame.
- *Rain Splash Big*: come il precedente *Emitter*, viene impiegato per generare gli spruzzi di pioggia nella collisione con la superficie in una variante con una forma più estesa. In linea di massima, mantiene le stesse caratteristiche

dei moduli e delle fasi di *Rain Spash Small* tranne per l'assenza del *Sub UV Animation* e di alcuni valori ritoccati per restituire un look visivo più attendibile alla realtà.

- *Rain Splash Variation:* una variante dell'emettitore *Rain Spash Small* che utilizza un materiale con uno *sprite* differente. Non ha particolari differenze con gli altri emettitori che generano spruzzi d'acqua se non nella forma dello schizzo prodotto.
- *Rain Ripples*: aggiunge le increspature generate dalle gocce d'acqua quando toccano il suolo. Si è partiti da una copia dell'emettitore *Rain Splash Big* con un nuovo *Material Instance* e sono state modificate le dimensioni ed il colore delle particelle. Essendo degli *sprite*, questi vengono visualizzati di default di fronte alla camera indipendentemente dalla direzione dell'inquadratura. Pertanto, tramite il modulo *Sprite Facing and Alignment*, è stata sovrascritta manualmente la direzione degli *sprite* rivolgendola sempre verso l'alto. Infine, è stata modificata la curva di scala durante il ciclo di vita delle particelle in modo che queste aumentino di dimensione in maniera lineare con l'aumento dei fotogrammi.

Uno dei problemi che si sono presentati durante la realizzazione dell'effetto riguardava la direzione delle particelle. Negli emettitori impiegati per simulare gli spruzzi d'acqua e l'increspatura, la posizione di generazione dei particellari non era del tutto accurata quando le gocce cadevano in punti inclinati della superficie. La soluzione iniziale è stata quella di realizzare un modulo personalizzato per gli emettitori e fare in modo che, quando una particella colpiva un pendio, dall'evento di collisione non si generavano schizzi o increspature.

Per filtrare tutte le particelle che potessero soddisfare il concetto descritto si è partiti dall'analizzare i dati che *Main Rain Drops* manda agli altri emettitori dopo essersi verificata una collisione. Ognuno di questi riceve, all'interno del modulo *Receive Collision Event*, due vettori: uno che indica la posizione nello spazio tridimensionale della collisione ed un altro che ne determina la normale alla superficie.

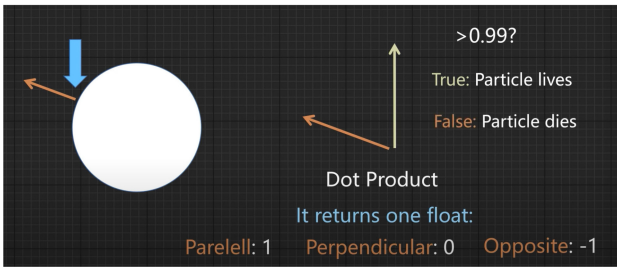

Figura 1.14. Grafico della particella durante la collisione su un piano.

Quando una goccia di pioggia colpisce un pendio, dai dati raccolti è possibile conoscere la normale alla superficie del punto di collisione. Dal grafico mostrato in Figura 1.14, facendo il prodotto scalare del vettore normale con l'*up vector*[33] si ottiene un numero in virgola mobile che: se pari a 1 indica che i due vettori sono paralleli, se vale 0 i vettori sono perpendicolari, mentre se uguale a -1 sono opposti. Di conseguenza, si può pensare di fare la seguente disamina: se il valore restituito dal prodotto scalare è maggiore di 0,99 allora la particella può iniziare il suo ciclo di vita, altrimenti viene eliminata. Questa verifica è stata la base per la realizzazione del modulo *Check Collision Normal* tramite il *Niagara Script Editor* e di cui è riportato il grafico a nodi in Figura 1.15.

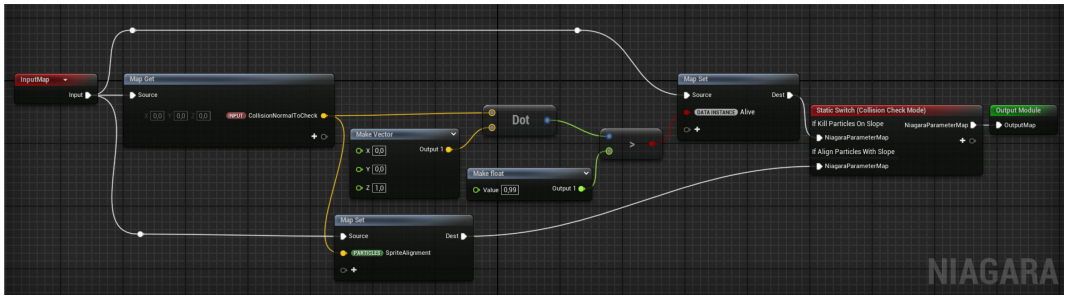

Figura 1.15. Node Graph del modulo personalizzato Check Collision Normal.

Per ricevere la normale alla superficie è stato settato, nel *Receive Collision Event*, il vettore *Collision Normal* come valore di output. Il risultato dell'operazione di controllo è stato poi passato al parametro booleano *Alive* il quale fornisce, al rispettivo emettitore, l'informazione sulla vita della particella.

[33] Con *up vector* si indica, in genere, la componente Z del vettore posizionale locale convertito nello spazio globale e normalizzato ad 1. Allo stesso modo *forward vector* e *right vector* rappresentano rispettivamente la componente X e Y del vettore di posizione 3D.

Nell'implementazione finale, è stata aggiunga la possibilità di allineare le particelle, al posto di eliminarle, qualora la collisione avvenisse in piani inclinati. Questa opzione è stata inserita tramite un *enum*[34] ed uno switch in cui è possibile scegliere il tipo di azione da adottare in presenza di pendii. Per settare l'allineamento dello *sprite* delle particelle è stata usata la normale alla superficie recuperata precedentemente ed impostato il valore *Alignment* del modulo *Sprite Renderer* come *Custom Alignment*. Una dimostrazione dell'effetto di pioggia risultante in una scena di prova è riportata in Figura 1.16.

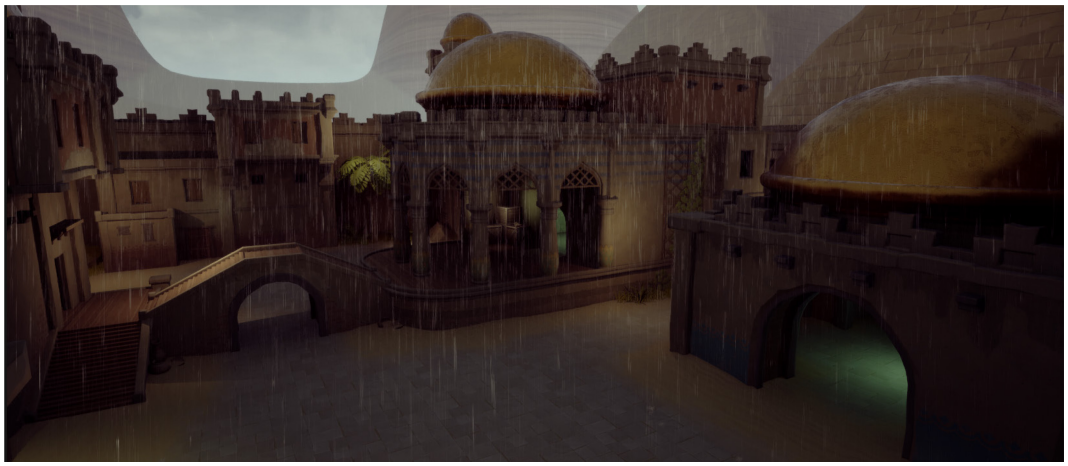

Figura 1.16. Esempio di una scena con un effetto di pioggia intensa.

1.4. Wireframe shader

Un lavoro su Unreal Engine che inizialmente era stato inserito nella lista di implementazioni utili per il progetto *Paesaggi - Landscapes* consisteva nella realizzazione di uno *shader* globale di tipo *wireframe*[35]. In sostanza si volevano mostrare tutti gli *asset* appartenenti allo stesso scenario tramite una "rappresentazione scheletrica" ovvero mettendo in risalto i bordi degli oggetti

[34] Un *enum* (in italiano enumerazione) è un tipo di dato che permette di definire un insieme di valori simbolici costanti. Viene usato principalmente per rappresentare una scelta, uno stato o un'opzione tra un gruppo definito di valori.
[35] I modelli *wireframe* 3D sono costituiti da punti, linee, archi, cerchi e altre curve che definiscono i bordi o le linee centrali degli oggetti.

rispetto alla loro texture. Di seguito verranno brevemente descritti i passaggi che ne consentono una realizzazione.

Il punto di partenza è assicurarsi di avere nella scena su cui si lavora un Actor di tipo *PostProcessVolume* e spuntare la casella di controllo riguardante l'estensione infinita in modo che l'Actor lavori su tutta la mappa e non solo su un'area limitata. Successivamente occorre realizzare il materiale che sarà il *wireframe shader*.

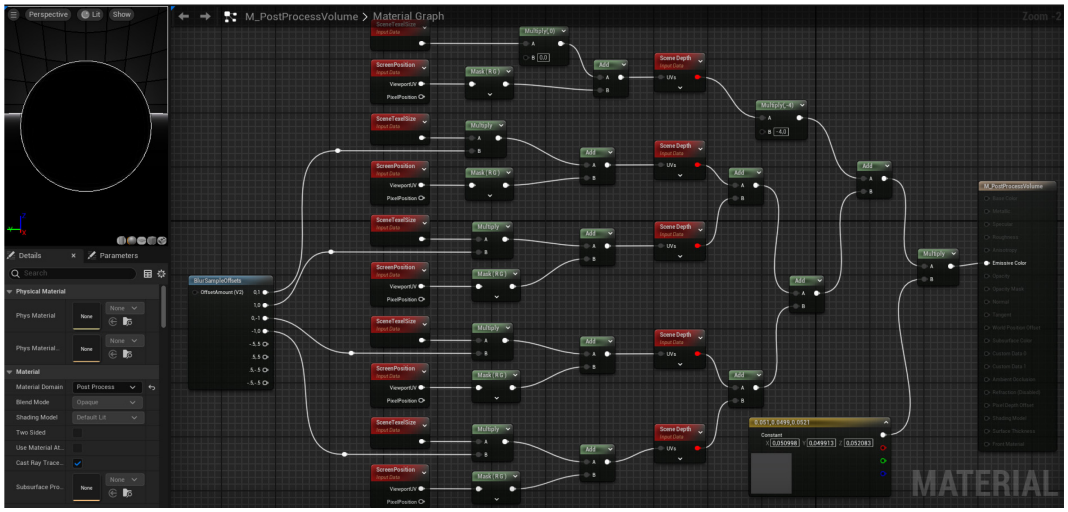

Figura 1.17. Material Graph del *wireframe* shader globale.

Il materiale, come da Figura 1.17, ha un dominio di tipo *Post Process*. In questo modo si ha solo un input di tipo *Emissive Color*, sufficiente per gli scopi preposti. Nel *Material Graph* occorre utilizzare a cascata tre tipologie di nodi:

- *Screen Position*: restituisce la posizione sullo schermo del pixel attualmente in fase di rendering.
- *Scene Depth*: restituisce la profondità della scena esistente e la può campionare in qualsiasi posizione. È possibile usarlo solo per materiali traslucidi.
- *Scene TexelSize*: consente di eseguire l'offset in base alle dimensioni dei *texel*[36], utile per il rilevamento dei bordi.

[36] Unità minima della texture, che può essere applicata su una scala più ampia, occupando quindi uno o più pixel.

Si inizia da una configurazione semplice dove un nodo *Scene TexelSize* viene moltiplicato ad un vettore 2D e sommato all'output di un nodo di tipo *Screen Position* a cui viene applicata una maschera sui canali R e G alla *viewport UV*. Il risultato di questa somma viene passato come input ad un nodo *Scene Depth*. Questa configurazione viene ripetuta quattro volte e infine sommata, ma con la differenza che la moltiplicazione riguardante ogni nodo *Scene TexelSize* avviene prendendo dei valori diversi da un *preset* di vettori 2D. Un'altra di queste configurazioni ha il nodo *Scene TexelSize* moltiplicato per zero ed alla fine della *Scene Depth* si ha una moltiplicazione per un valore -4[37]. Il risultato viene poi sommato alle precedenti e collegato all'*Emissive Color*. Qualora si volesse un colore differente per l'*outline* degli *asset* è possibile moltiplicare un dato valore RGB prima di collegare il risultato all'*Emissive Color*. In questo modo si ottiene il materiale che funge da *wireframe shader*. Infine, affinché questo venga applicato alla scena occorre inserirlo nella lista di *Post Process Material*.

In Figura 1.18 si ha una dimostrazione della resa finale che si raggiunge in una scena di default con e senza l'applicazione dello *shader*.

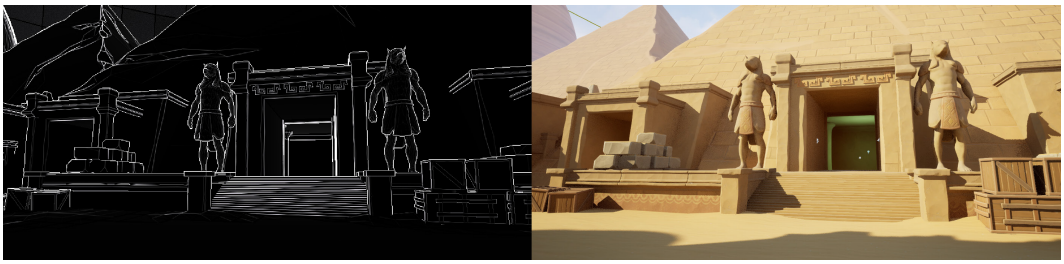

Figura 1.18. Esempio di una scena con e senza *wireframe shader*.

Un'altra rappresentazione *wireframe* poteva essere locale, ossia applicata singolarmente agli *asset* in modo da renderne evidenti le caratteristiche geometriche. Lo stesso procedimento si applica alla realizzazione di uno *shader*.

Il materiale in questione ha una modalità di fusione di tipo traslucente in modo da poter garantire delle gradazioni intermedie della visibilità dell'*outline* sulla mesh.

[37] Minore è il valore moltiplicativo e meno saranno i dettagli dei contorni visibili in scena.

Vengono, inoltre, spuntati il flag *Two Side* per mostrare anche il retro della mesh ed il flag *Wireframe* per visualizzare solo i lati delle facce di cui è costituita la mesh.

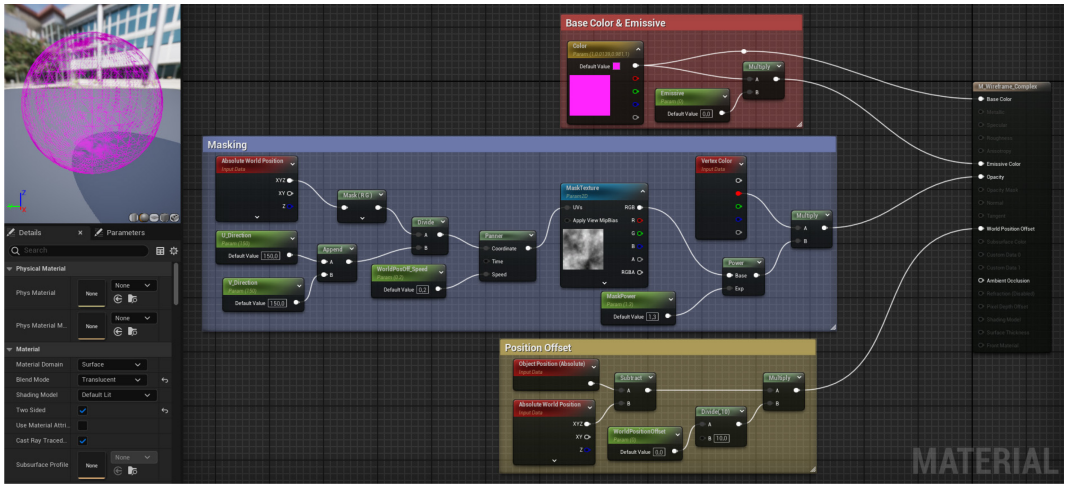

Figura 1.19. Material Graph del *wireframe shader* locale.

A questo punto, come mostrato in Figura 1.19, è possibile organizzare i nodi di cui il materiale è composto in tre parti distinte:

- *Base Color & Emissive*: costituiscono il colore e il valore di emissività del materiale.
- *Masking*: imposta la maschera entro cui lo *shader* ha effetto e la fa muovere con un certo range di velocità e con una determinata potenza.
- *Position Offset*: utile per ingrandire e ridimensionare lo schema *wireframe* sulla mesh di riferimento.

In questi nodi, sono stati parametrizzati alcuni valori[38] in modo da poterne personalizzare le proprietà alle singole istanze. Il passo successivo è quello di duplicare la mesh, applicare a quest'ultima lo *shader* e sovrapporla all'originale. Qualora si volesse applicare il materiale solo a determinate parti della mesh, usando la modalità *Mesh Paint* di Unreal Engine è possibile andare a selezionare le parti in cui lo *shader* ha effetto mascherando dei vertici tramite un pennello. In Figura 1.20, degli esempi di *shader wireframe* applicati a dei vasi di prova.

[38] Nello specifico i dati parametrizzati sono: colore, emissività, maschera della texture, potenza della maschera, velocità di movimento, direzioni della mappa UV, offset globale della posizione.

Figura 1.20. Esempi di shader *wireframe* applicati a vasi.

1.5. Lookup table

Uno dei temi emersi in fase di preproduzione riguardava la resa cromatica delle sequenze realizzate in computer grafica. Infatti, essendo l'esperienza immersiva un insieme di riprese dal vivo e filmati realizzati digitalmente, occorreva fare in modo che lo stacco tra i due tipi di contenuti fosse impercettibile da parte del visitatore. Occorreva pertanto realizzare delle LUT[39] che avessero le stesse sfumature di colore delle riprese fatte in Egitto in modo che, applicandole su Unreal Engine, si potessero esportare dei contenuti direttamente con una correzione colore quanto più fedele alla realtà.

Per fare ciò è stato utilizzato il software Affinity Photo® e delle foto di riferimento scattate in Egitto da una Canon EOS R5 C. A partire da una foto, il software consente di esportane la LUT. Una volta generato il corrispettivo file LOOK, lo si è applicato alla Color Neutral LUT messa a disposizione da Unreal Engine nella documentazione ufficiale ottenendo una LUT personalizzata. Una comparazione tra la Color Neutral LUT con la LUT custom la si può osservare in Figura 1.21.

[39] Nella grafica, una *Look-Up Table* (LUT) è una tabella o un *preset* che applica una correzione cromatica ai colori di un'immagine o di un filmato, modificandoli secondo i valori definiti nello schema della LUT. Unreal Engine usa delle LUT 3D di16×16×16 trasformate in texture 256×16. Esse sono le più precise in quanto riescono a descrivere la trasformazione del colore in maniera più dettagliata grazie ad uno schema cromatico esteso.

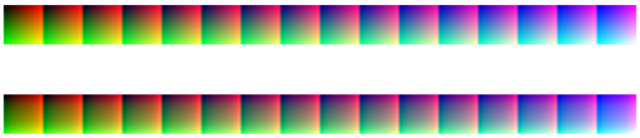

Figura 1.21. Color Neutral LUT (sopra) e LUT custom (sotto).

Questo passaggio è stato reso necessario per passare come input ad Unreal Engine una struttura file che il software fosse in grado di processare. A questo punto, nel progetto di Unreal Engine di cui si vuole fare una correzione colore, si aggiunge un Actor di tipo *PostProcessVolume* e nella voce "Color Gradient LUT" si inserisce la Color Neutral LUT precedentemente modificata. Sempre con stesso Actor è possibile modificare alcuni parametri per conferire un ulteriore tocco di realisticità e naturalezza alla scena.

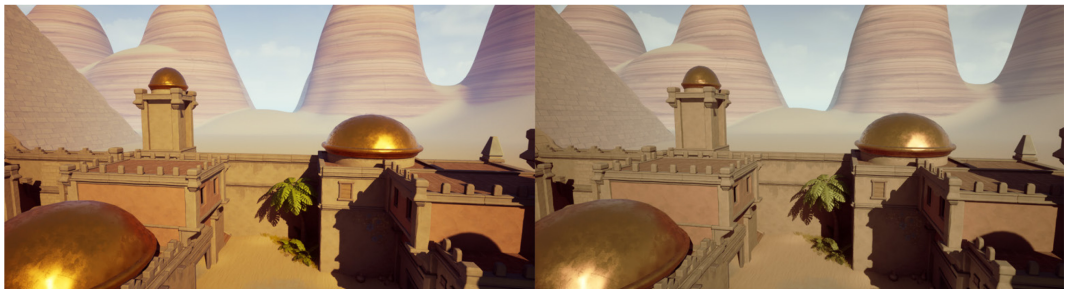

Figura 1.22. Esempio di applicazione di una LUT.

In Figura 1.22, partendo da uno scenario di prova (a sinistra), tramite l'applicazione di una LUT si riesce ad ottenere un paesaggio (a destra) che descrive meglio la palette cromatica che è possibile ritrovare nel territorio egiziano.

2. LA FOTOGRAMMETRIA

Dopo aver trattato i lavori preliminari, svolti in preparazione di *Egitto Immersivo*, a partire dal presente capitolo verranno illustrate le soluzioni implementative adottate per realizzare alcuni *shot* della mostra immersiva *Paesaggi – Landscapes*. Durante la fase iniziale del progetto, come descritto precedentemente, parte della troupe si è recata in Egitto per raccogliere materiale e concepire delle idee per la sceneggiatura dell'installazione. Nel corso di un periodo durato all'incirca dieci giorni, sono state effettuate riprese a 360° e scansioni zenitali utilizzando droni e attrezzatura fotografica di vario genere in alcuni siti archeologici collegati al Museo Egizio quali El-Hammamiya, Gebelein, Aswan, Deir el-Medina. Il girato è stato poi sfruttato per realizzare le fotogrammetrie delle aree storiche, utilizzate successivamente nei *time-lapse* dell'esperienza immersiva.

Nei paragrafi successivi verrà presentata una panoramica introduttiva dei principi basilari della fotogrammetria. Seguirà una descrizione del *workflow* adottato per la realizzazione dei modelli fotogrammetrici e dei software utilizzati per tale scopo. Successivamente, verrà affrontato il tema della rimozione delle ombre presenti nelle texture dei modelli generati. Infine, sarà proposta una breve analisi sulla tecnica di ricostruzione nota come *3D Gaussian Splatting*, con particolare attenzione alle potenzialità applicative nell'ambito del presente progetto.

2.1. Nozioni e principi

La fotogrammetria è una tecnica di rilievo che consente di determinare la posizione, la forma e la dimensione di un oggetto o ambiente reale scansionato tramite una fotocamera. Catturando immagini con prospettive diverse, gli algoritmi di fotogrammetria riescono ad elaborare le acquisizioni prodotte per ricreare l'oggetto digitalmente. Quando viene scattata una fotografia, il dispositivo fotografico cattura le informazioni della luce che entra nell'obiettivo. Queste informazioni contengono dati sulla luminosità e sono ciò che forma l'immagine digitale. Questa non è altro

che una visualizzazione appiattita di un'oggetto che nella realtà è tridimensionale. Anche se l'immagine risulta piatta, al suo interno sono contenute informazioni prospettiche utilizzabili acquisite al momento dello scatto.

I software di fotogrammetria, come RealityCapture o Metashape, calcolano la posizione relativa della fotocamera e l'angolo di visuale dalle informazioni prospettiche contenute all'interno dell'immagine assegnandole una coordinata di posizione nello spazio. Quando due immagini vengono posizionate nello spazio digitale in base alle loro coordinate, l'angolo di visuale viene utilizzato per determinare il loro punto di intersezione. In questo modo, utilizzando un set di immagini dello stesso oggetto scattate da angolazioni diverse, l'algoritmo abbina i pixel correlati in base alla loro somiglianza per ricostruire la geometria dell'oggetto originale.

Attualmente ci sono molteplici settori che fanno un largo uso della fotogrammetria come, ad esempio, l'industria dell'intrattenimento per via della facilità nella generazione di mesh 3D somiglianti ad oggetti reali. Oppure il settore del materiale di archivio culturale per la conservazione digitale e documentazione di beni o, ancora, per la mappatura topologica di paesaggi e siti reali.

La scansione può essere eseguita con apparecchiature progettate per acquisire immagini da ogni possibile angolazione, come droni, fotocamere digitali o, in alcuni casi, anche tramite smartphone se configurati per una risoluzione naturale delle immagini. I tre principi chiave della fotogrammetria che occorre considerare per effettuare una buona scansione sono:

- Qualità dell'immagine: le immagini devono essere nitide, messe bene a fuoco, con una profondità di campo elevata e senza sfocature o rumore digitale. Una buona qualità dell'immagine garantisce che il software possa essere in grado di leggere i dati dei pixel in modo corretto ed accurato.
- Copertura del soggetto: affinché l'algoritmo di ricostruzione funzioni correttamente, risulta essenziale che l'intero oggetto venga catturato all'interno del set di immagini. Se ci sono lacune o informazioni mancanti nelle immagini, il software non può colmare quella lacuna. Per tale ragione,

vengono scattate molte immagini del soggetto da angolazioni diverse in modo da garantire una copertura completa dell'oggetto.

- Sovrapposizione delle informazioni: per mettere in relazione due immagini, l'algoritmo confronta le informazioni sul colore di un'immagine con l'altra per determinare come queste si uniscono. L'algoritmo continua a "piegare" le immagini nel set finché la forma del soggetto non è completamente digitalizzata. Per fornire informazioni sufficienti, le immagini scansionate devono avere circa il 70% di sovrapposizione.

Quando le immagini risultano sfocate o distorte, presentano lacune di copertura o contengono sovrapposizioni insufficienti, il software interpreta questi dati di scansione scadenti come parte dell'oggetto e include la distorsione anche nel modello 3D ricostruito. Per ottenere dati di scansione della miglior qualità è quindi necessario prestare attenzione alla sovrapposizione delle immagini, all'attrezzatura fotografica da utilizzare e all'illuminazione dell'oggetto che si intende digitalizzare.

Per acquisire fotografie chiare e nitide, con buona esposizione e col minimo rumore possibile, è importante avere familiarità con il cosiddetto "triangolo di esposizione"[40]. Dallo studio di questo triangolo segue che una profondità di campo elevata, una velocità dell'otturatore sufficientemente rapida ed un basso valore ISO, rappresentano la configurazione ideale per la fotogrammetria. Da notare, tuttavia, come tutte e tre queste impostazioni riducano l'esposizione alla luce. Per tale motivo, un bilanciamento tra queste tre impostazioni combinate a buone tecniche di illuminazione risulta necessario per garantire un'esposizione corretta.

Gli oggetti di notevole dimensione è consigliabile che vengano scansionati tramite due livelli di acquisizione: un *over-all pass* con ampie inquadrature dove l'oggetto risulta visibile per intero e un *detail pass* per catturare i dettagli più vicini. Ciò risulta particolarmente utile quando è necessario che la fotocamera sia troppo vicina per catturare i dettagli dell'oggetto ma questo è più grande dell'inquadratura.

[40] Il "triangolo dell'esposizione" consiste in tre impostazioni base della fotocamera che costituiscono metaforicamente i vertici di tale triangolo. Queste impostazioni che devono essere bilanciate per ottenere una corretta esposizione sono: apertura, velocità dell'otturatore e ISO.

La maggior parte delle piante pone troppi problemi per essere catturata correttamente con la fotogrammetria. La loro forma sottile e intricata, la miriade di elementi sovrapposti e il movimento del vento rendono estremamente complessa la generazione di mesh. Pertanto, risulta necessario utilizzare un approccio diverso alla scansione tradizionale. Un metodo efficace per ricostruire correttamente la vegetazione in digitale è quello di separare le singole parti della pianta da scansionare in 2D con una fotocamera per poi riassemblarle in ambiente digitale da un artista del settore.

2.2. La sequenza di immagini

Tornando al progetto *Paesaggi – Landscapes*, l'effettivo avvio delle attività ha potuto concretizzarsi una volta acquisito il materiale audiovisivo prodotto durante la permanenza in Egitto. La piena visione del girato ha consentito di valutare in maniera più consapevole la natura delle riprese disponibili, permettendo così di delineare con maggiore accuratezza la struttura narrativa dell'installazione immersiva e tutti gli aspetti creativi in base alle fotogrammetrie ricavabili.

Una volta ottenuto il girato originale dei droni in risoluzione 6016×3200 con profondità colore a 24 bit, sono stati creati i relativi file *proxy*[41] per ogni singola clip al fine di rendere il processo di elaborazione e gestione dei file più fluido ed efficiente. In seguito, come parte del lavoro preparatorio alle fotogrammetrie è stata effettuata un'attenta analisi delle clip *proxy*, identificando quelle più rilevanti ed in grado di offrire buoni risultati fotogrammetrici. A tal fine, sono state privilegiate le riprese in cui i droni e le camere ispezionavano interamente l'area, in modo da ottenere un elevato numero di informazioni utili per la fase di realizzazione della nuvola di punti tridimensionale.

[41] I *proxy* video sono copie a bassa risoluzione dei file originali. Ad esempio, nel caso in cui il girato originale sia in 4K, i *proxy* potrebbero essere copie a risoluzione 1080p di ciascuna clip. L'impiego di questi file durante la fase di editing consente di ottenere una maggiore reattività, riducendo i tempi di caricamento, il lag e altri eventuali problemi che potrebbero presentarsi qualora si lavorasse con video ad alta risoluzione su hardware con prestazioni limitate.

Dopo aver selezionato i file *proxy*, la fase successiva ha riguardato la generazione delle sequenze di immagini sostituendo ai *proxy* il girato originale. Per tale scopo è stato adoperato il software di montaggio DaVinci Resolve, attraverso il quale sono state esportate le sequenze delle immagini in formato JPEG. Si è scelto di mantenere ogni immagine con la risoluzione nativa o almeno in 4K garantendo il massimo livello di dettaglio possibile per l'elaborazione fotogrammetrica, pur adottando un formato immagine con compressione *lossy*[42].

Per mantenere la resa cromatica naturale dei dispositivi di acquisizione, prima dell'esportazione è stata applicata una LUT specifica al modello di drone o fotocamera utilizzato. In particolare, per quanto concerne i droni, essendo prodotti dalla azienda DJI, essi impiegano un profilo proprietario, il D-Log[43]. Per tale ragione, è stata applicata una LUT di tipo "D-Log to Rec.709" che ha permesso la conversione dello spazio colore in uno standard. Allo stesso modo, per le riprese effettuate con fotocamere e smartphone, sono state applicate le LUT corrispondenti al raggiungimento dello spazio colore Rec.709.

Nella Figura 2.1 vengono mostrati due fotogrammi relativi ad una veduta sulle rive del Nilo: a sinistra l'immagine originale acquisita con profilo Apple Log ha colori poco saturi e a basso contrasto, confrontata con la stessa immagine a destra che, a seguito dell'applicazione della relativa LUT di conversione, presenta una resa cromatica più simile alla realtà.

[42] La compressione con algoritmi di tipo *lossy* (come nel caso del JPEG) comporta la rimozione permanente di quei dati che l'algoritmo non ritiene necessari mantenere all'interno dell'immagine in modo che questa possa avere delle dimensioni ridotte. In alternativa, la compressione di tipo *lossless* (come nel formato PNG) permette di ridurre le dimensioni del file senza sacrificare alcun dato, consentendo di ricostruire perfettamente l'immagine originale a partire dalla versione compressa.

[43] Il formato D-Log rappresenta una specifica curva logaritmica sviluppata da DJI per catturare un'ampia gamma e profondità cromatica durante le riprese. Questo formato permette di conservare dettagli sia nelle zone di luce che nelle ombre di un'immagine. DJI fornisce delle LUT specifiche per ciascun modello di drone, che facilitano la conversione del materiale registrato in uno standard colore, come ad esempio Rec.709. In assenza dell'applicazione delle LUT, le immagini catturate in D-Log appaiono con un colore poco saturo e con basso contrasto a causa della gamma dinamica estesa registrata.

Figura 2.1. Comparazione di un frame in Apple Log con applicata una LUT.

Nei software di fotogrammetria utilizzati, l'inserimento di 30 o 60 fotogrammi al secondo risulta spesso ridondante, in quanto le variazioni tra un fotogramma e il successivo sono generalmente minime e non apportano un contributo significativo all'elaborazione tridimensionale. Di conseguenza, per ottimizzare il processo e ridurre i tempi di computazione, le clip sono state opportunamente accelerate tramite DaVinci Resolve, così da ridurre il numero complessivo di fotogrammi pur mantenendone una quantità sufficiente a garantire la presenza di informazioni significative in termini di sovrapposizione. La Figura 2.2 mostra una schermata di DaVinci Resolve in cui è possibile osservare come la sequenza selezionata sia stata accelerata con un fattore di velocità pari a 14. Un'ulteriore operazione effettuata mediante lo stesso software ha riguardato l'editing marginale dell'inizio e della fine delle clip, ritagliando le sequenze video in modo da selezionare l'intervallo temporale più opportuno da includere nell'elaborazione della nuvola di punti.

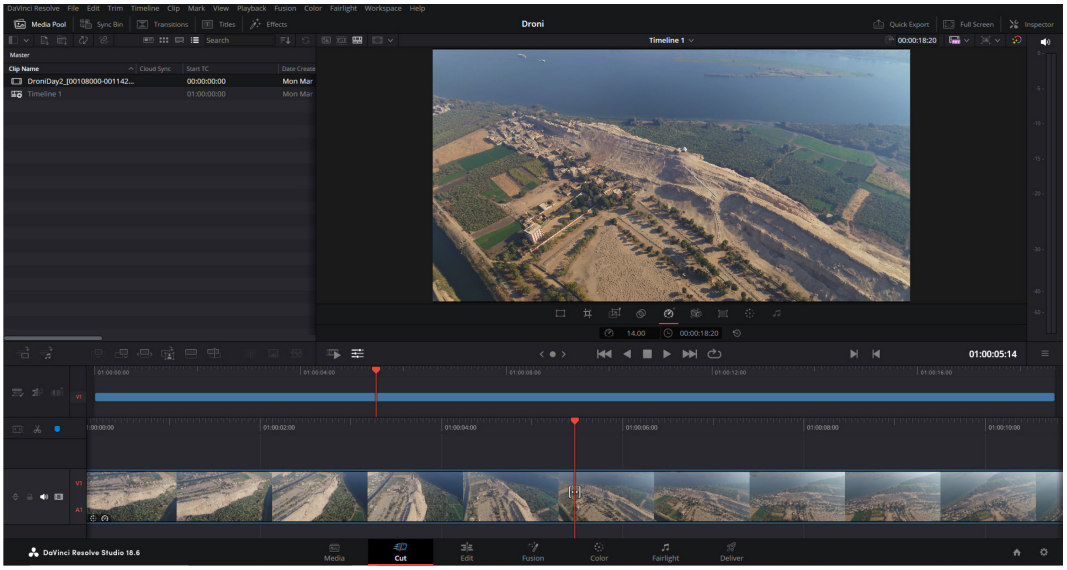

Figura 2.2. Schermata di DaVinci Resolve sull'editing della clip.

2.3. Lo sviluppo dei modelli

Una volta completata la raccolta delle varie sequenze di immagini, il passo successivo ha riguardato la generazione dei modelli tridimensionali con le relative texture e nuvole di punti. Trattandosi di operazioni ad elevata intensità computazionale, eventuali reiterazioni avrebbero rallentato significativamente l'intera pipeline di produzione. Per questo motivo, sono stati utilizzati due software particolarmente apprezzati per l'elevata qualità dei risultati e per la loro semplicità di utilizzo.

In particolare, tramite Reality Capture® sono stati realizzati modelli di grandi dimensioni ricostruendo gli interi paesaggi scansionati dall'alto con i droni. Metashape®, invece, si è rivelato particolarmente efficace nel ricostruire modelli dettagliati basati sulle immagini ravvicinate catturate con le fotocamere digitali.

Nei paragrafi seguenti verranno inizialmente descritti i due software di fotogrammetria, Reality Capture e Metashape, descrivendo il *workflow* adottato ai fini della mostra *Paesaggi – Landscapes*. In un secondo momento si farà menzione di Blender, un software di modellazione tridimensionale utilizzato per la pulizia degli *asset* generati.

2.3.1. RealityCapture

RealityCapture è una delle piattaforme principali utilizzate dalle aziende nel contesto di scansioni laser e fotogrammetrie. In particolare, i campi più comuni del suo utilizzo riguardano il patrimonio culturale, la topografia, l'industria videoludica, gli effetti visivi e la realtà virtuale. Sviluppato dalla società slovacca Capturing Reality, a partire dal 2021 il software è stato associato alla suite di Unreal Engine dal momento in cui l'azienda è diventata proprietà di Epic Games. Nel 2024, Epic Games ha modificato i prezzi del software consentendo agli sviluppatori che guadagnano meno di 1 milione di dollari di entrate lorde di utilizzarlo gratuitamente.

Particolarmente noto per la sua elevata velocità di elaborazione superiore ai suoi competitor, RealityCapture utilizza meccanismi di *parallel computing*[44] e sfrutta algoritmi avanzati di fotogrammetria per realizzare repliche digitali di paesaggi e oggetti altamente fotorealistici. Tra le sue caratteristiche, importante è la sua piena compatibilità con diversi software CAD e di modellazione 3D nell'export dei modelli, rendendolo altamente versatile ed integrabile con altre piattaforme di lavoro. Tra le funzionalità offerte da RealityCapture vi è l'allineamento delle immagini, la calibrazione automatica, il calcolo di mesh poligonali, la texturizzazione, la georeferenziazione, la conversione del sistema di coordinate e il ridimensionamento. RealityCapture consente inoltre di ottimizzare le mesh generate, fornendo degli strumenti di editing per correggere le geometrie, semplificare la nuvola di punti e migliorare complessivamente la resa.

Prendendo come riferimento la Figura 2.3, verrà di seguito descritta la procedura operativa per la generazione di un modello tridimensionale.

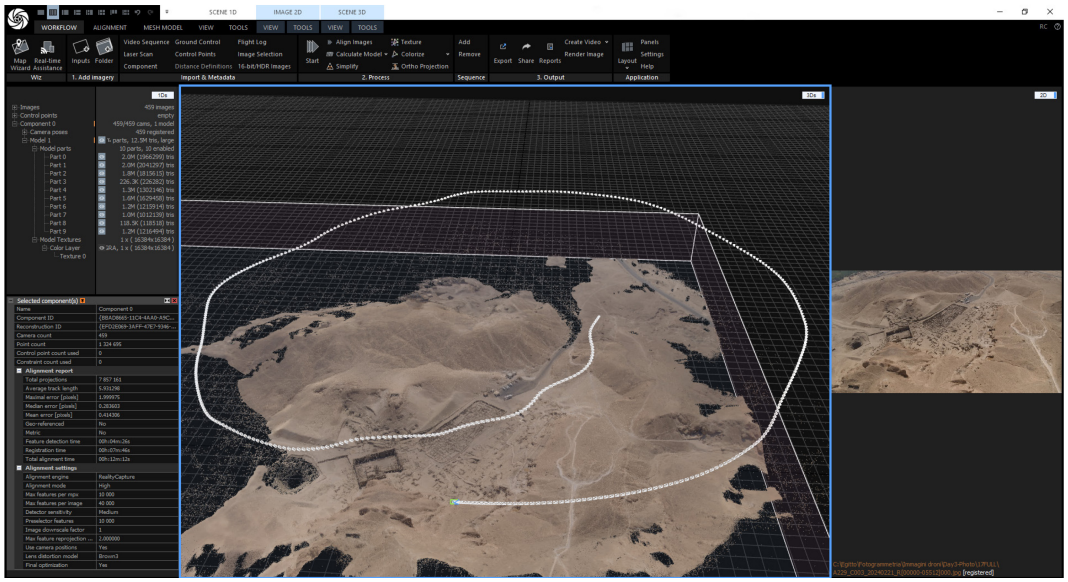

Figura 2.3. Schermata di RealityCapture con una nuvola di punti risultante.

[44] Il *parallel compting* (in italiano, calcolo parallelo) è un meccanismo che permette l'esecuzione simultanea del codice sorgente di uno o più software su più processori (o *core*) allo scopo di suddividere un'operazione complessa in vari problemi minori incrementando le prestazioni del sistema di elaborazione.

La fase iniziale del processo prevede l'importazione dei dati di acquisizione, organizzati in gruppi di file immagine. Nel caso specifico riportato in Figura 2.3 relativo alla mappatura di un'intera area del sito archeologico di Deir el-Medina sono state utilizzate solo 459 immagini rispetto alle migliaia acquisite dal drone. È opportuno sottolineare che un numero più elevato di fotogrammi in input comporta un maggiore consumo di memoria *cache* da parte della piattaforma e un conseguente aumento dei tempi di elaborazione. Tuttavia, ciò non garantisce un aumento significativo della qualità del modello generato, poiché un'eccessiva sovrapposizione delle immagini tende a produrre risultati simili a quelli ottenuti dando in input frame più distanziati, per via dell'efficacia degli algoritmi di elaborazione adoperati dal software.

Una volta completata l'importazione, il passaggio successivo riguarda l'allineamento delle immagini, fase in cui viene generata la nuvola di punti che rappresenta l'ambiente scansionato. In questo stadio, il software elabora anche la posizione spaziale presunta di ciascuna immagine mostrandola nella scena 3D. Nella Figura 2.4 è possibile osservare il reticolo spaziale delle posizioni assunte dal drone durante le acquisizioni delle riprese del villaggio di Deir el-Medina.

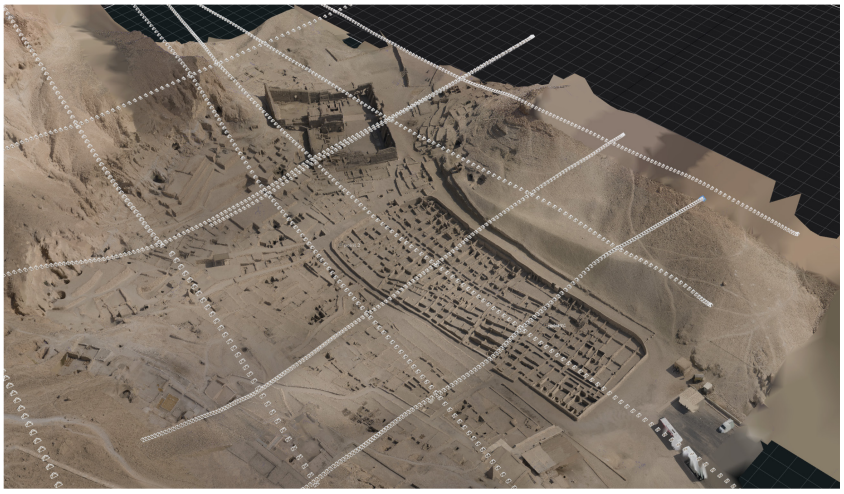

Figura 2.4. Particolare sul reticolo per la fotogrammetria di Deir el-Medina.

In questa fase del processo, è buona prassi procedere all'allineamento della nuvola di punti e all'impostazione di due elementi fondamentali: il *Ground Plane* e la *Recostruction Region*. Definire il *Ground Plane* consente di evitare la generazione di artefatti al di sotto del piano di terra, oltre a rappresentare il riferimento principale

per l'orientamento spaziale del modello. La *Recostruction Region*, invece, permette di delineare un volume entro cui il software eseguirà le operazioni di ricostruzione escludendo i dati in eccesso generati dalla nuvola di punti.

Una volta allineate le immagini è possibile procedere alla fase di ricostruzione del modello tridimensionale. Tale fase ha inizio con il calcolo della mappa di profondità (*depth-map*) per ciascuna immagine. Questa operazione utilizza principalmente la GPU ed è determinante per la qualità del modello finale. Come mostrato in Figura 2.5 è possibile configurare vari parametri di ricostruzione, tra cui *Image downscale*[45] che gestisce la qualità delle mappe di profondità. Dopo il calcolo delle *depth-map*, viene generata la mesh finale.

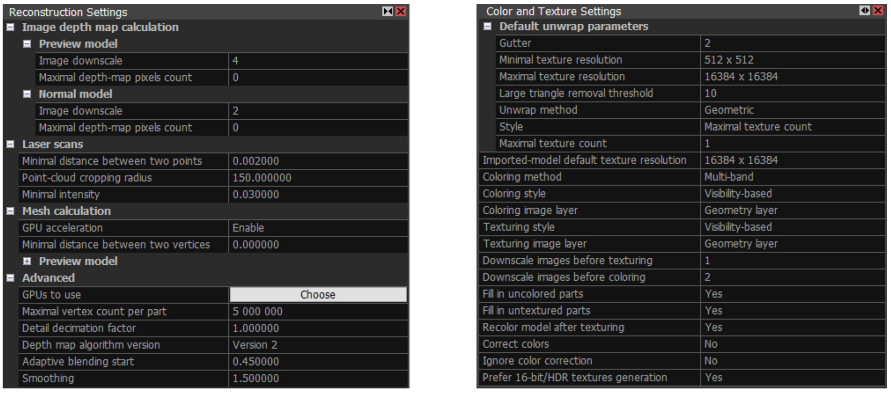

Figura 2.5. Impostazioni sulla ricostruzione e texturizzazione del modello

La fase successiva riguarda la texturizzazione del modello, nella quale vengono create delle piccole immagini per ogni faccia della mesh poligonale. In particolare, vengono generate una o più texture associate alle superfici della mesh mediante una mappatura UV[46] (chiamata anche *unwrap*). La texturizzazione e la sua qualità dipendono fortemente da come vengono generate le mappe UV.

[45] Invece di utilizzare le immagini a risoluzione originale, è possibile impostare un fattore con cui ridimensionare le immagini per calcolare la mappa di profondità. Più alto è il numero e più veloce sarà il calcolo ma minore sarà il livello di dettaglio. Utilizzare il numero 1 significa non applicare alcuna modifica nella scala (risoluzione del 100%), con il numero 2 ogni lato di un'immagine sarà due volte più piccolo (25% della risoluzione originale dell'immagine).

[46] La mappatura UV è una tecnica di *texture mapping* che consente di associare adeguatamente le texture su un modello 3D. Con tale tecnica, la mesh viene distesa su un piano ed ogni vertice dell'oggetto 3D è associato a un set di coordinate 2D che corrispondono all'immagine. Il termine "UV" rappresenta l'equivalente delle coordinate X e Y nel sistema 2D.

RealityCapture consente di definire la dimensione desiderata del *texel* in relazione al sistema di coordinate del modello, oppure di specificare la risoluzione massima della texture e il numero totale di texture utilizzabili per il processo di texturizzazione. Come si può osservare in Figura 2.5, è inoltre possibile intervenire direttamente sui parametri di *unwrap*, qualora il modello non sia stato precedentemente mappato. Da notare come nella risoluzione massima della texture sia stata impostata la dimensione 16384×16384 che corrisponde al più alto valore supportato da RealityCapture. Questa scelta è stata fatta per conferire il maggior livello di dettaglio al modello per una rappresentazione quanto più fotorealistica possibile.

Dopo aver ricostruito il modello, è buona norma ispezionare il risultato finale per verificarne la qualità di generazione. Spesso il modello presenta alcuni errori o lacune causate da una copertura insufficiente delle immagini scansionate. Le aree imperfette e indesiderate vengono quindi filtrate o ripulite utilizzando appositi strumenti della piattaforma.

Completata la pulizia del modello, come ultima cosa si può procedere a semplificarlo. Si cerca di ridurre il numero dei poligoni generati al fine di rendere la mesh più gestibile, sia all'interno del programma stesso che per l'esportazione. Infatti, importare un modello con milioni di vertici all'interno di motori grafici come Unreal Engine o nei software di modellazione 3D potrebbe gravare molto sulle prestazioni del computer.

2.3.2. Metashape

Metashape® è un software sviluppato da Agisoft dedicato all'elaborazione fotogrammetrica di immagini digitali generando dati spaziali tridimensionali da poter utilizzare in vari contesti: applicazioni per il sistema informativo geografico (GIS), *asset* per la computer grafica, documentazione del patrimonio culturale e produzione di effetti visivi, nonché per misurazioni indirette di oggetti di varie scale. Il software consente di elaborare immagini RGB o termiche ottenendo informazioni spaziali sotto forma di nuvole di punti e modelli poligonali strutturati.

L'elaborazione fotogrammetrica, supportata con metodi di computer vision, viene implementata in modo ottimale da permettere di raggiungere un'elevata precisione nei risultati ottenuti. Ciò si traduce in un'interfaccia intuitiva e in un *workflow* di elaborazione semplice che può essere gestito da non esperti nel campo della fotogrammetria. Allo stesso tempo, ha molto da offrire ad un professionista del settore che può beneficiare di funzionalità avanzate come la modalità stereoscopica e avere il controllo completo sulla precisione dei risultati, con un report dettagliato generato alla fine dell'elaborazione.

In Metashape, le fotografie adatte alla ricostruzione nello spazio tridimensionale possono essere scattate da qualsiasi macchina fotografica digitale purché rispettino certi criteri di acquisizione[47]. Nel caso di fotografia aerea, per risultati ottimali si richiede una sovrapposizione laterale di circa il 60% e frontale dell'80%. Tali valori possono essere incrementati in caso di rilievi su foreste o aree ad alta densità di vegetazione per via del movimento causato dal vento che potrebbe ostacolare l'algoritmo del software nel trovare punti in comune tra le immagini.

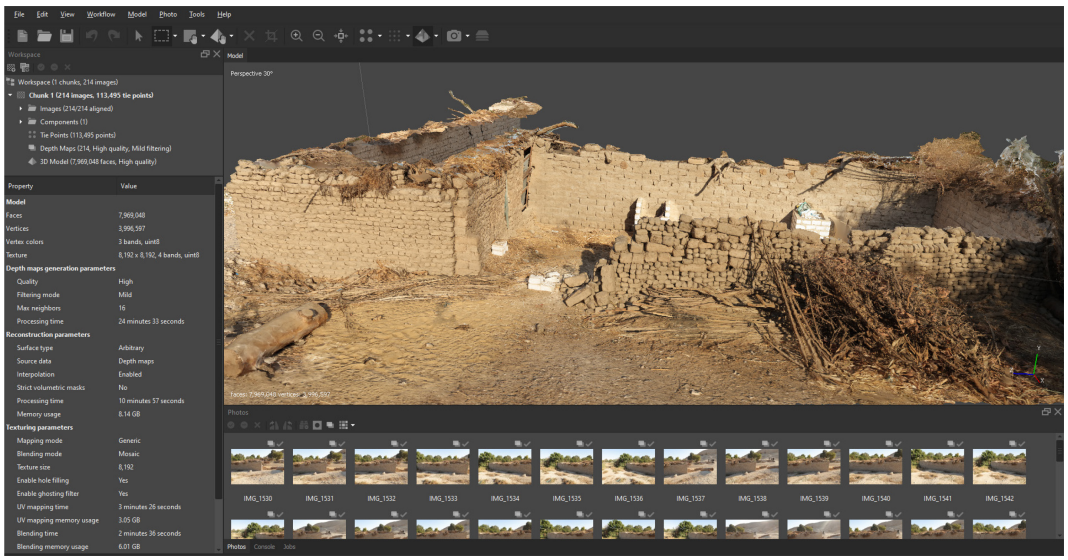

Figura 2.6. Schermata di Metashape con un modello di una rovina.

[47] In particolare, sono da preferire immagini fotografiche scattate alla più alta risoluzione che la fotocamera è in grado di supportare, utilizzando obiettivi fissi con una lunghezza focale compresa tra 20 e 80 mm. Inoltre, è opportuno impostare un valore ISO basso per evitare elementi di rumore digitale nelle immagini, un'apertura sufficientemente ridotta da assicurare una profondità focale che possa garantire foto nitide ed una velocità dell'otturatore non troppo bassa.

La procedura standard per la realizzazione di un modello 3D texturizzato in Metashape, come illustrato in Figura 2.6, si articola in due fasi principali:

- La prima fase è denominata allineamento. Dopo il caricamento di un insieme di immagini, Metashape individua i *key points*, ovvero punti chiave caratterizzati da particolari proprietà, e li abbina tra le diverse immagini. Il risultato di tale operazione è l'individuazione dei cosiddetti *tie points*, punti di collegamento che costituiscono la base per le elaborazioni successive. Il software determina inoltre la posizione della fotocamera per ciascuna immagine e stima i parametri di calibrazione della stessa. Al termine di questa fase, i risultati vengono rappresentati sotto forma di una nuvola sparsa di *tie points* e un set di posizioni delle fotocamere. Questa nuvola è fondamentale per la successiva generazione delle mappe di profondità e può essere esportata per l'utilizzo in altri software.
- Il secondo passaggio riguarda la generazione della mesh 3D. Il modello poligonale così ottenuto può essere texturizzato per conferirgli un aspetto digitale fotorealistico ed esportato in diversi formati compatibili sia con software CAD che con programmi di modellazione 3D. Inoltre, è possibile generare una nuvola densa di punti a partire dalle posizioni stimate della fotocamera e dalle immagini acquisite.

Tra le funzionalità aggiuntive, il software è in grado di realizzare un'immagine panoramica unendo una serie di fotografie scattate dalla stessa posizione. In ogni parte del *workflow* è possibile impostare vari parametri che consentono una ricostruzione più o meno dettagliata della scena, in funzione del tempo e delle risorse computazionali a disposizione dell'utente.

Nell'ambito della mostra *Paesaggi – Landscapes*, i modelli generati tramite Metashape hanno rivestito un ruolo essenziale nelle scene dei *time-lapse* realizzate con Unreal Engine, in cui le fotogrammetrie costituivano l'elemento principale delle inquadrature. Per garantire un'esperienza altamente immersiva, si è optato per impostazioni qualitative comprese tra il livello medio e alto, con texture aventi una risoluzione di 8192×8192, così da assicurare un elevato grado di dettaglio visivo.

2.3.3. Blender

Blender® è un software *open-source* multipiattaforma che permette la modellazione, l'animazione, il *rigging*, la composizione, il montaggio video, il rendering ed il texturing di immagini tridimensionali e bidimensionali. Dispone anche di strumenti per le simulazione fisiche e per lo scripting in Python.

In origine Blender era stato sviluppato principalmente da Ton Roosendaal come applicazione interna della società olandese NeoGeo. Lo sviluppatore fondò poi Not a Number Technologies (NaN) con l'obiettivo di distribuire Blender ad uso commerciale. La bancarotta di questa sua società, ha predisposto Blender verso una pubblicazione come software libero a seguito di una campagna di raccolta fondi. Da quel momento in poi, il suo sviluppo è stato portato avanti dalla Blender Foundation, un'organizzazione *no-profit* finanziata tramite donazioni. La sua lunga serie di aggiornamenti periodici ne hanno ampliato progressivamente le funzionalità ed ottimizzato la gestione della RAM video e l'interfaccia utente. Nel corso del tempo sono stati introdotti strumenti avanzati come i Geometry Nodes per la modellazione procedurale basata su un approccio a nodi, il Grease Pencil per realizzare animazioni 2D tradizionali o *motion graphics* ed il supporto al motore di rendering Cycles per un *path-tracing* fisicamente accurato.

Blender è diventato uno dei software più diffusi ed apprezzati nel settore della creazione di contenuti digitali, affermandosi come un valido strumento di riferimento per i professionisti. Inoltre, la sua natura di distribuzione *open-source* ne incrementa notevolmente l'accessibilità, permettendo a chiunque di intraprendere un percorso di apprendimento nel campo della modellazione ed animazione digitale senza dover affrontare ingenti spese.

Uno degli impieghi di Blender riguarda il *clean-up* di modelli ottenuti mediante fotogrammetria. Infatti, i modelli generati con i software citati nelle precedenti sezioni presentano spesso un numero eccessivo di poligoni, con una conseguente richiesta computazionale elevata nei motori grafici e possibili problemi di gestione della memoria. La semplificazione dei modelli consente di ridurre la loro complessità geometrica, preservando i dettagli essenziali e ottimizzando il consumo

di VRAM della GPU. Inoltre, le scansioni fotogrammetriche possono produrre artefatti e rumore all'interno della geometria della mesh. In tali casi, Blender consente di eliminare frammenti di geometria che sono collegati alla mesh principale, correggere eventuali lacune e risolvere problemi legati a un UV mapping errato della texture. È inoltre possibile eseguire operazioni di *retopology*, ricostruendo la mesh con una struttura topologica ottimizzata a complessità ridotta.

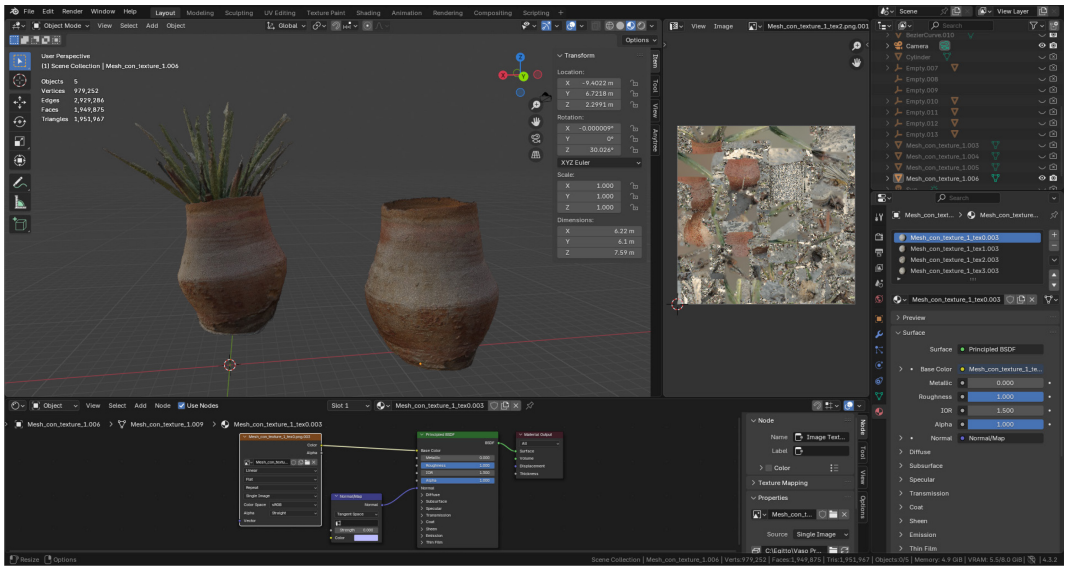

Figura 2.7. Schermata di Blender per il *clean-up* del vaso predinastico.

La Figura 2.7 mostra il risultato della pulizia della mesh di un vaso predinastico eseguita in Blender e successivamente utilizzata in alcune scene all'interno di Unreal Engine. L'operazione è stata effettuata in "Edit Mode", intervenendo a livello di vertici e rimuovendo le sezioni non utilizzabili a causa degli artefatti presenti.

2.4. La rimozione delle ombre

Una volta completata la generazione dei modelli attraverso le fotogrammetrie, la fase successiva ha riguardato la rimozione delle ombre dalle texture associate alle mesh. I software di fotogrammetria impiegati per la creazione delle texture utilizzano, infatti, le informazioni cromatiche provenienti dal set di immagini fornito per la fase di elaborazione. Ne consegue che anche la disposizione delle luci

e delle ombre presenti nelle immagini originali viene incorporata nel processo di texturizzazione del modello.

Nel caso in cui la mesh texturizzata debba essere integrata all'interno di un ambiente digitale dotato di un sistema di illuminazione diverso rispetto a quello delle immagini sorgenti, si possono generare incongruenze visive tra la texture e l'illuminazione della scena. Per evitare tale effetto, risulta necessario intervenire direttamente sulle texture, rimuovendo artificialmente le zone d'ombra, al fine di ottenere un'illuminazione piatta. In questo modo, le uniche ombre e luci visibili sul modello saranno quelle generate dall'ambiente digitale in cui il modello verrà utilizzato.

2.4.1. Texture De-Lighter

Texture De-Lighter® è uno strumento per la post-produzione di modelli 3D progettato per correggere le ombre e l'occlusione ambientale presenti nelle texture generate dai processi di fotogrammetria. Sviluppato da Agisoft, azienda già conosciuta per Metashape, il software è stato introdotto in maniera gratuita per rispondere all'esigenza degli utenti di rimuovere le ombre proiettate durante le acquisizioni fotografiche in contesti di luce non controllabile, garantendo delle texture più coerenti dal punto di vista visivo per i modelli fotogrammetrici.

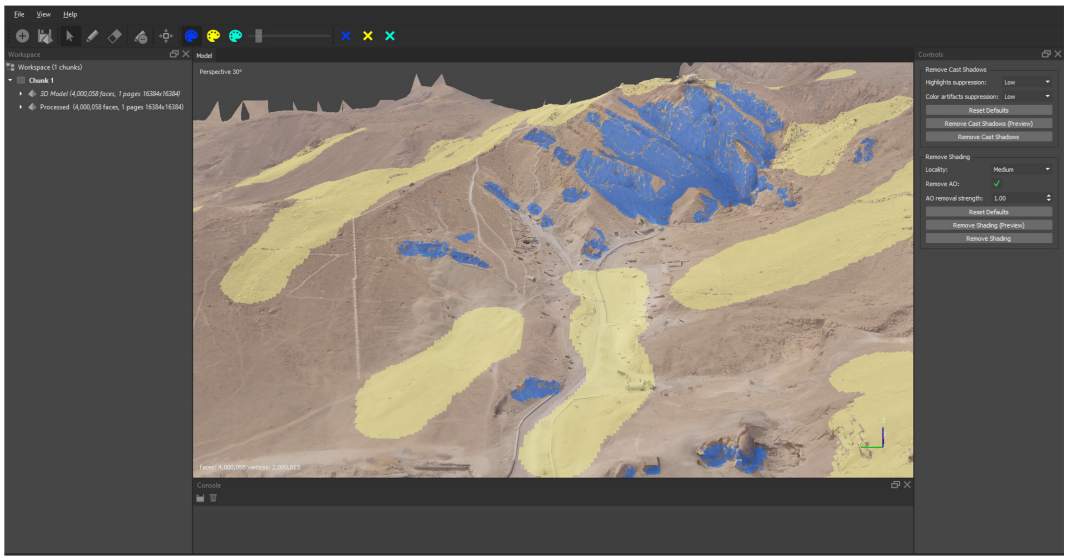

Figura 2.8. Schermata di Texture De-Lighter sul sito di Valle delle Regine.

Una volta importato il modello tridimensionale, come mostrato in Figura 2.8, l'utente è chiamato a fornire indicazioni approssimative sulle zone illuminate (in giallo) e in ombra (in blu) tramite pennellate virtuali. Tali informazioni possono essere fornite direttamente all'applicazione senza la necessità di caricare ulteriori dati, come ad esempio mappe di occlusione ambientale. L'algoritmo di Texture De-Lighter è ottimizzato per texture JPEG a 8 bit compresse, non richiedendo pertanto l'utilizzo di immagini ad alta profondità cromatica.

Il software genera una mappa di illuminazione proiettata sulla superficie del modello, calcolando la differenza tra luce diretta e luce indiretta, e applica un filtro correttivo che consente di attenuare o rimuovere le zone d'ombra, restituendo una texture complessivamente più uniforme. Per la definizione della mappa di illuminazione, l'utente può utilizzare lo strumento "Brush" per tracciare pennellate sulle aree in luce e in ombra, mediante le opzioni *lit color* e *shadow color*. Avviando la funzione "Remove Cast Shadow", il software esegue l'algoritmo di correzione e genera un nuovo livello di texture privo delle ombre proiettate sulla mesh.

In alcune circostanze, possono manifestarsi artefatti dovuti a forti variazioni di luminosità. Per affrontare tali problematiche, lo strumento mette a disposizione due parametri regolabili, "Highlights Suppression" e "Color Artifacts Suppression".

Inoltre, qualora una regione illuminata presenti una texture scura, potrebbe essere erroneamente interpretata come ombreggiata e pertanto rimossa. Per evitare questo inconveniente, è possibile definire delle aree da escludere dal processo di elaborazione, aggiungendo una nuova "Shadow Scale Map" e contrassegnando le aree mediante un pennello con l'opzione *extra color*.

Nel caso in cui non siano presenti ombre proiettate nella texture, la luminosità della superficie dipende principalmente dall'orientamento delle normali della mesh. Questo principio può essere sfruttato per separare in maniera automatica l'ombreggiatura dal colore, utilizzando l'opzione "Remove Shading", disponibile nel pannello dei controlli. Tale operazione produce come risultato un'illuminazione più uniforme sulla superficie del modello.

Figura 2.9. Processo di rimozione delle ombre su una fotogrammetria di aloe.

La Figura 2.9 illustra i risultati ottenuti su un modello fotogrammetrico generato con il software Metashape, in seguito all'applicazione della procedura di uniformazione dell'illuminazione in modo da ottenere una base neutra, seguita dalla rimozione delle ombre più evidenti e localizzate.

2.5. 3D Gaussian Splatting

Introdotta alla fine degli anni '90 da Lee Westover come tecnica di rendering volumetrico destinata a scopi scientifici, la metodologia del 3D Gaussian Splatting, grazie agli sviluppi della computer grafica, consente oggi di rappresentare ambienti tridimensionali attraverso l'utilizzo di distribuzioni gaussiane anisotrope, ciascuna dotata di parametri quali posizione, scala, orientamento, colore ed opacità. Ogni gaussiana contribuisce alla visualizzazione dell'ambiente tramite un processo di rendering chiamato *splatting*[48] che permette di proiettare ogni *splat* gaussiano sul piano immagine, ovvero sul piano visivo percepito dall'utente. In tal modo, la scena tridimensionale viene tradotta in una rappresentazione bidimensionale coerente con il punto di vista dell'osservatore.

L'utilizzo di questi elementi puntiformi, in sostituzione alle tradizionali geometrie poligonali e ai volumi basati su mesh, permette di accorciare significativamente i

[48] Tecnica di rendering che consiste nel rappresentare una scena come una collezione di punti, chiamati *splat*, con distribuzione gaussiana. Avendo una distribuzione a campana, ogni *splat* permette di definire delle aree aventi una certa larghezza e densità di probabilità. Inoltre, sempre per via del tipo di distribuzione, le superfici generate appaiono più morbide (vengono sfocate con l'aumentare della distanza dal punto centrale) rispetto a quelle create con metodi basati su mesh, rendendo questa tecnica ideale per costruire nuvole di punti.

tempi di ottimizzazione rispetto ai metodi convenzionali garantendo una qualità elevata nel rendering in tempo reale ed una maggiore efficienza computazionale.

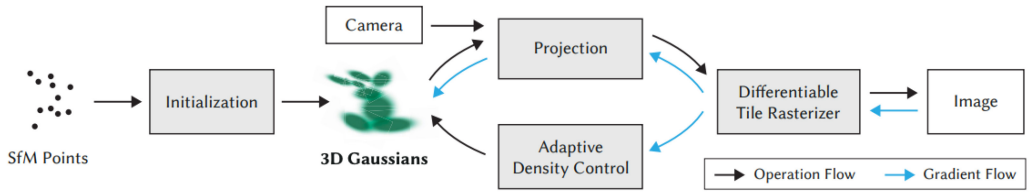

Figura 2.10. Diagramma dell'algoritmo di 3D Gaussian Splatting.

Cercando di riassume il meccanismo dei 3D Gaussian Splatting, schematizzato in Figura 2.10, il punto di partenza dell'algoritmo è dato da un insieme di immagini in input che consentono di generare una nuvola di punti, stimando i parametri della fotocamera e facendo corrispondere i pixel tra le varie immagini. Con l'ausilio di strumenti di *machine learning*[49], per ciascuno *splat* gaussiano inizializzato vengono ottimizzati i parametri di posizione (le coordinate del punto nello spazio), di covarianza (l'estensione e l'orientamento), di colore (espresso attraverso lo schema RGB) e di trasparenza. Questo processo di ottimizzazione è la chiave di tutto l'algoritmo ed è simile a quanto avviene nell'addestramento delle reti neurali. Infine, tramite lo *splatting*, gli *splat* gaussiani vengono proiettati sul piano immagine 2D riproducendo la scena tridimensionale con elevata qualità visiva.

La tecnica del 3D Gaussian Splatting è una soluzione valida utilizzata in molteplici campi, in particolare nel settore dell'intrattenimento. In contesti applicativi che fanno largo uso di ambienti virtuali permette di offrire un elevato livello di immersività. Nel mondo dei videogiochi rende possibile l'inclusione di elementi iperrealistici senza dover impattare sulle prestazioni dell'hardware.

In linea generale il 3D Gaussian Splatting è stato adattato ed esteso a vari settori della computer vision e della grafica in applicazioni che spaziano dal rendering di scene dinamiche alla creazione di contenuti 4D e alle simulazioni di guida

[49] Il *machine learning* è un approccio all'analisi dei dati che automatizza la costruzione di modelli predittivi. Si fonda sull'idea che i sistemi possono apprendere dai dati, identificare pattern in modo autonomo e prendere decisioni con un intervento umano minimo.

autonoma. In particolare, l'estensione di questa tecnica nelle scene dinamiche ha portato allo sviluppo del 3D Temporal Gaussian Splatting, chiamato anche 4D Gaussian Splatting, che incorpora anche una componente temporale consentendo il rendering in tempo reale di scene movimentate con risoluzioni elevate.

2.5.1. Luma AI e SuperSplat

Luma Labs è una società fondata in California nel 2021 e specializzata nello sviluppo di tecnologie 3D avanzate che sfruttano l'intelligenza artificiale. Tra le principali innovazioni in cui si è affermata vi è il Neural Radiance Field (NeRF), una tecnologia capace di creare modelli 3D realistici a partire da fotografie, e più di recente il 3D Gaussan Splatting. Il successo di queste tecniche ha posizionato Luma Labs come leader in settori come la computer vision, il cinema e la realtà virtuale, dove è fondamentale disporre di contenuti 3D realistici e con alta qualità. Luma Labs ha introdotto una sua piattaforma *user-friendly* online, Luma AI® che, tramite un'infrastruttura cloud, permette di gestire applicazioni sul campo della ricostruzione fotorealistica con il supporto dell'intelligenza artificiale.

Di particolare rilievo è Luma Interactive Scenes, che combina l'efficienza del rendering in tempo reale ottenuto attraverso il 3D Gaussian Splatting con la robustezza del rendering basato su nuvole di punti, largamente adottato in applicazioni commerciali. Mediante l'upload di file video o di riprese acquisite tramite l'applicazione mobile, la piattaforma consente la generazione automatica di una rappresentazione 3D Gaussian Splatting della scena, bypassando la necessità di gravose elaborazioni locali sui dispositivi degli utenti. Inoltre, attraverso l'integrazione del plugin Luma AI per Unreal Engine, è possibile importare direttamente i render NeRF o le rappresentazioni 3D Gaussian Splatting all'interno del motore grafico, usufruendo di un'integrazione parziale delle funzionalità offerte della piattaforma Luma AI.

Nell'ambito del progetto *Paesaggi – Landscapes*, sono stati condotti diversi esperimenti di generazione di ambienti tridimensionali fotorealistici mediante 3D Gaussian Splatting, finalizzati alla loro possibile integrazione nella mostra immersiva.

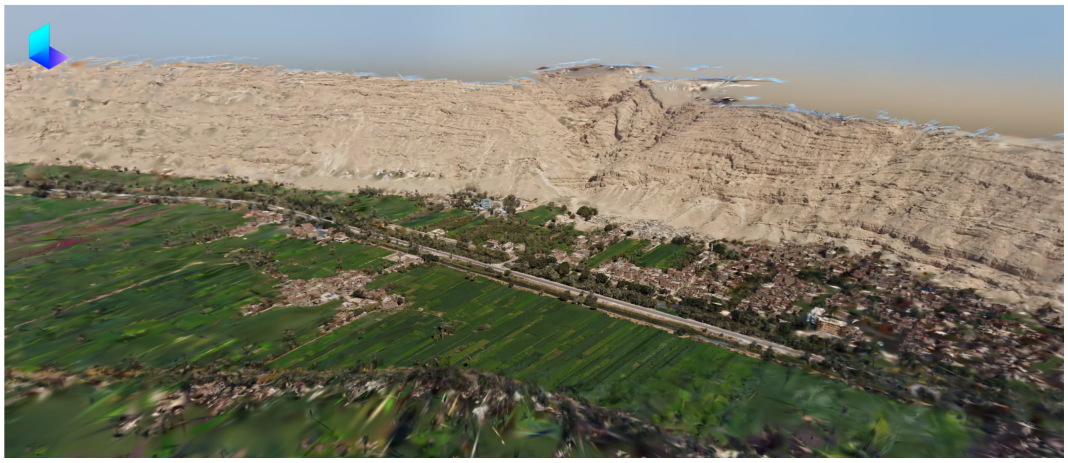

Figura 2.11. 3D Gaussian Splatting di El-Hammamiya generato con Luma AI.

Come illustrato in Figura 2.11, i dati raccolti attraverso voli col drone presso il sito archeologico di El-Hammamiya sono stati utilizzati per produrre rappresentazioni ad alta fedeltà da poter inserire eventualmente all'interno di scene sviluppate su Unreal Engine. Ulteriori sperimentazioni hanno riguardato la rappresentazione della Diga di Aswan, del Tempio di Philae, nonché della vegetazione e delle formazioni rocciose lungo il fiume Nilo, al fine di arricchire alcuni shot del progetto con *asset* realistici.

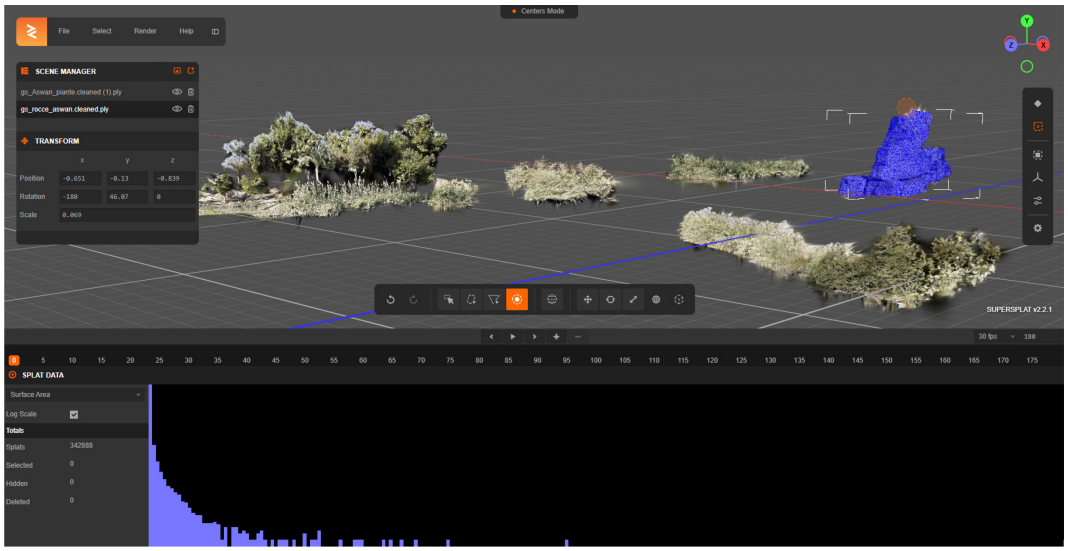

Figura 2.12. Operazioni di editing su vegetazione e rocce mediante SuperSplat.

Per le fasi successive di ottimizzazione e *clean-up* dei 3D Gaussian Splatting generati, è stato impiegato il tool open-source SuperSplat®, un editor *browser-based* intuitivo, che consente di eseguire interventi localizzati sugli *splat* gaussiani.

Tali operazioni si rendono spesso necessarie in quanto le rappresentazioni prodotte presentano, in molti casi, difetti geometrici, lacune o rumore dovuti a errori di elaborazione o a copertura parziale. Come mostrato in Figura 2.12, mediante l'utilizzo dello strumento "Brush" messo a disposizione da SuperSplat, è possibile intervenire direttamente sui singoli *splat*, modificandoli selettivamente all'interno dei 3D Gaussian Splatting importati in formato PLY.

Tuttavia, una criticità fondamentale che ha infine limitato l'integrazione dei 3D Gaussian Splatting nell'esperienza immersiva *Paesaggi – Landscapes* è insita nella natura della rappresentazione stessa. I file generati da tale tecnologia non costituiscono veri e propri modelli tridimensionali convenzionali, bensì proiezioni bidimensionali orientate, che risultano visibili unicamente da specifiche angolazioni e contengono già al loro interno informazioni relative a luci ed ombre. Questo aspetto, sebbene permetta una resa visiva estremamente fotorealistica, limita l'adattabilità degli *asset* all'interno dei motori grafici, dove i modelli ottenuti tramite fotogrammetria tradizionale risultano ancora predominanti per flessibilità, interazione e compatibilità con ambienti di rendering complessi.

3. I TIME-LAPSE

La mostra *Paesaggi – Landscapes*, articolata secondo una struttura narrativa in tre stadi, utilizzando proiezioni a schermo intero e in *split screen*, prevede che all'interno della fase di analisi venga introdotto il concetto del passaggio del tempo.

A partire dalle riprese dell'attuale paesaggio rurale egiziano, l'attenzione viene progressivamente guidata in un viaggio a ritroso nel tempo, al fine di ripercorrere e rielaborare l'eredità culturale tramandata dai paesaggi e dai siti archeologici egiziani. In questa transizione narrativa, con le immagini che raffigurano i territori naturali, entra in gioco la dimensione temporale, illustrando, attraverso dei *time-lapse*, l'evoluzione della luce diurna in diversi scenari e siti storici ricostruiti digitalmente. In questo percorso, dall'Egitto contemporaneo fino alle sue "schegge di memoria", si cerca di stimolare un dibattito interpretativo interiore nello spettatore, che costituisce il fulcro dell'intera esperienza immersiva.

Nel presente capitolo verranno pertanto analizzati i *time-lapse* realizzati in computer grafica mediante l'utilizzo di Unreal Engine, utilizzando alcuni modelli di piante tipiche dell'ecosistema egiziano e delle mesh ricavate tramite le fotogrammetrie trattate nella sezione precedente. Verranno introdotti inizialmente gli strumenti di Unreal Engine adottati, a cui seguirà la descrizione dettagliata del *workflow* relativo alle scene presenti all'interno della mostra.

3.1. Strumenti di Unreal Engine

Unreal Engine offre una vasta gamma di strumenti e funzionalità che permettono agli sviluppatori di sfruttarne al meglio le potenzialità e versatilità di utilizzo. Inoltre, Epic Games distribuisce mensilmente nel proprio Marketplace vari plugin e contenuti gratuiti che possono essere impiegati all'interno del motore grafico in vari scenari applicativi e casistiche d'impiego. In buona sostanza, avere competenze su Unreal Engine implica non solo la padronanza sugli aspetti tecnici fondamentali

ma anche conoscere le funzioni avanzate che l'ecosistema permette di utilizzare appieno. In questa prospettiva, nei paragrafi seguenti verranno presentati gli strumenti che sono stati impiegati per le scene in computer grafica di *Paesaggi – Landscapes* al fine di comprenderne meglio l'impiego di cui si tratterà nel dettaglio nel proseguimento di questo capitolo.

3.1.1. Quixel Bridge

Quixel Bridge è un plugin per Unreal Engine che offre un accesso diretto e illimitato ad una libreria di risorse fotorealistiche gratuite[50]. Acquistato da parte di Epic Games nel 2019, si è contraddistinto per la sua vasta collezione di Megascans ossia un archivio di modelli tridimensionali, texture e materiali realizzati con il supporto di tecniche di fotogrammetria. Quixel Bridge è stato integrato nell'installazione di Unreal Engine 5. Ciò ha segnato un punto di svolta per lo sviluppo, semplificando il *workflow* per importare i contenuti all'interno del motore grafico e favorendo l'adozione dei suoi *asset* in progetti indipendenti.

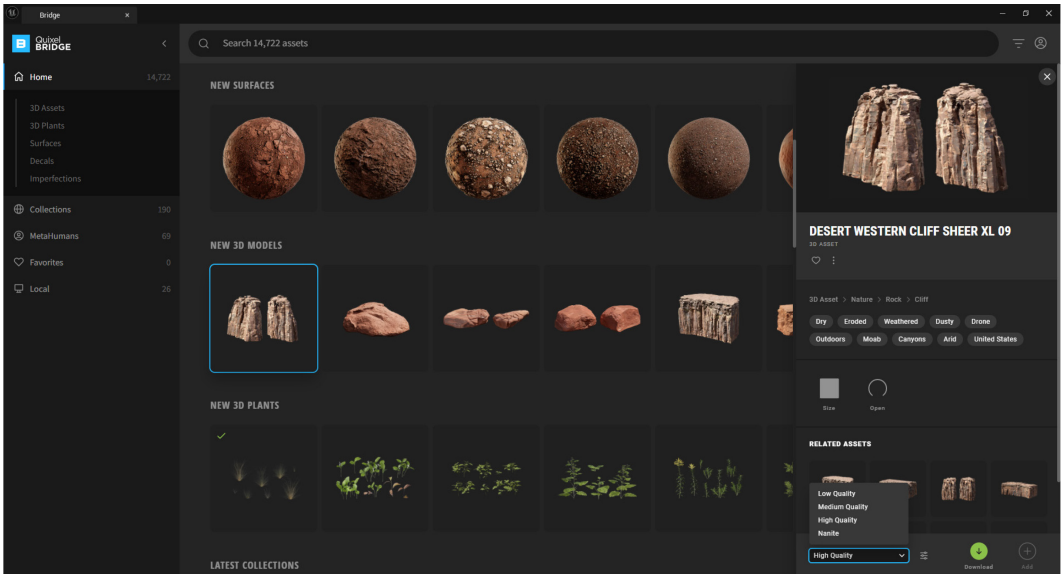

Figura 3.1. Schermata di Quixel Bridge per importare le risorse.

[50] A partire dal 2025, la libreria di Quixel è stata integrata all'interno di Fab, il nuovo marketplace di Epic Games che consente di acquistare direttamente le risorse, offrendone gratuitamente solo una piccola parte dell'intero catalogo.

Come mostrato in Figura 3.1, l'interfaccia di Quixel Bridge è progettata per essere semplice e intuitiva. Essa contiene una barra di ricerca utile per trovare una determinata risorsa o una raccolta corrispondente. Inoltre, è presente un pannello di collezioni di contenuti tematici ed una sezione MetaHumans per importare i personaggi realizzati con l'applicazione MetaHuman Creator. Comprende anche una schermata di accesso rapido alle risorse contrassegnate come preferite e una sezione che mostra gli *asset* scaricati sul dispositivo.

Tutti gli *asset* 3D per Unreal Engine vengono scaricati in formato UAsset e sono disponibili partendo da una risoluzione bassa fino ad arrivare ad una qualità di tipo Nanite. Quest'ultima è una tecnologia supportata dall'IA che consente di utilizzare modelli con un elevato numero di poligoni senza preoccuparsi di doverli ridurre tramite tecniche di LOD (*Level of Detail*) per ottimizzare le prestazioni.

Esistono diversi modi per aggiungere le risorse ad un progetto: è possibile selezionarle dal pannello di Quixel Bridge e trascinarle fino alla *viewport* di Unreal Engine oppure scaricarle e inserirle manualmente nella scena. Una volta importati, gli *asset* funzionano immediatamente e non richiedono alcuna configurazione preventiva delle impostazioni del progetto. Inoltre, la sincronizzazione continua tra gli elementi scaricati con Quixel Bridge e Unreal Engine consente al motore grafico di rilevare le nuove versioni delle risorse, che vengono aggiornate automaticamente senza richiedere alcun intervento dell'utente.

3.1.2. Foliage Mode

La Foliage Mode di Unreal Engine è un modulo operativo avanzato che consente agli sviluppatori di gestire la disposizione della vegetazione all'interno di un progetto. Attraverso strumenti intuitivi come un pennello, un lazo o un secchiello, la Foliage Mode consente di popolare ampi scenari con del fogliame in un breve lasso di tempo migliorando l'efficienza del flusso di lavoro.

Nello sviluppo di scenari realistici, questo strumento è considerato particolarmente rilevante sia per la sua semplicità di utilizzo che per l'elevata qualità dei risultati

ottenibili. Arricchendo di dettagli ambientali le scene naturali, è infatti possibile realizzare dei paesaggi immersivi e stilisticamente accattivanti.

All'interno del progetto *Paesaggi – Landscapes*, la Foliage Mode è stata utilizzata nei *time-lapse* e nei giochi di luce sulle piante, popolando gli scenari con *asset* richiamanti la flora egiziana provenienti dalla libreria Megascans di Quixel Bridge. Questo approccio ha permesso di ottenere un realismo visivo coerente con il paesaggio egiziano, migliorando la resa naturalistica e il senso di immersività.

Lo strumento Foliage è costituito da un pannello suddiviso in cinque sezioni:

1. *Tools Palette*: una raccolta di strumenti che vengono utilizzati per dipingere il fogliame direttamente su un Landscape o su un altro Actor abilitato. Tra gli strumenti più usati si annovera "Paint" che consente di dipingere il fogliame selezionato direttamente sulla mesh.
2. *Brush Options*: contiene dei parametri che regolano la dimensione del pennello, la densità del fogliame e la densità delle piante da rimuovere all'interno di un'area.
3. *Filter Options*: consente di controllare la superficie sulla quale influisce lo strumento selezionato e quali tipi di fogliame sono attualmente attivi.
4. *Foliage Palette*: all'interno è possibile aggiungere tipologie di fogliame sulla scena che possono essere istanze di Native Actor oppure Static Mesh[51]. Le prime sono da utilizzare con parsimonia poiché un numero elevato potrebbe causare problemi di prestazioni. Le Static Mesh vanno bene nei casi di fogliame non distruttivo e vengono renderizzate in gruppi.
5. *Foliage Details*: contiene una varietà di proprietà per personalizzare il comportamento delle singole istanze. È possibile avere un controllo dettagliato su ogni aspetto della vegetazione, come i parametri di scala, la densità, la distanza di rendering e la probabilità di distribuzione.

[51] Un Native Actor è un *asset* creato utilizzando codice C++ o i Blueprint e viene spesso utilizzato quando si desidera un controllo preciso sul comportamento dell'elemento. Una Static Mesh è un tipo di *asset* che rappresenta una mesh inanimata, utile quando si vogliono inserire in una scena degli oggetti che non necessitano di animazione.

Figura 3.2. Modalità Foliage per inserire delle piante in una scena.

Nella Figura 3.2 viene illustrato un esempio di applicazione della Foliage Mode per inserire delle sterpaglie e piante all'interno di una scena. Si noti come sono selezionate nello stesso tempo più tipologie di fogliame in modo da "dipingerle" sulla mesh utilizzando la loro probabilità di disposizione ad area e le relative impostazioni di densità.

3.1.3. Sequencer

Unreal Engine dispone di un set avanzato di strumenti per creare sequenze animate e cinematografiche. Attraverso questi strumenti è possibile pilotare le camere per realizzare dei *fly-through*, animare luci e personaggi ed eseguire il rendering delle sequenze finali. Al centro di questi flussi di lavoro si colloca il Sequencer, una potente suite di editing che permette di creare cinematiche in Unreal Engine utilizzando un sistema di *track* e *keyframe* disposti lungo una timeline.

Il funzionamento del Sequencer si basa principalmente su due costrutti: il Level Sequence Asset ed il Level Sequence Actor. Il primo si trova nel Content Browser e contiene tutte le informazioni necessarie del Sequencer, come *keyframe*, tracce, animazioni, telecamere e altri parametri. Il secondo, invece, costituisce l'istanza attiva nella scena che fa riferimento al Level Sequence Asset contenendo i suoi dati.

Tramite il pannello dei dettagli relativi all'Actor è possibile modificare i parametri di riproduzione come l'avvio automatico, la velocità della sequenza, la ripetizione

in loop e altre specifiche di tipo cinematico come, ad esempio, la possibilità di disabilitare l'input di movimento e nascondere l'HUD durante la sequenza.

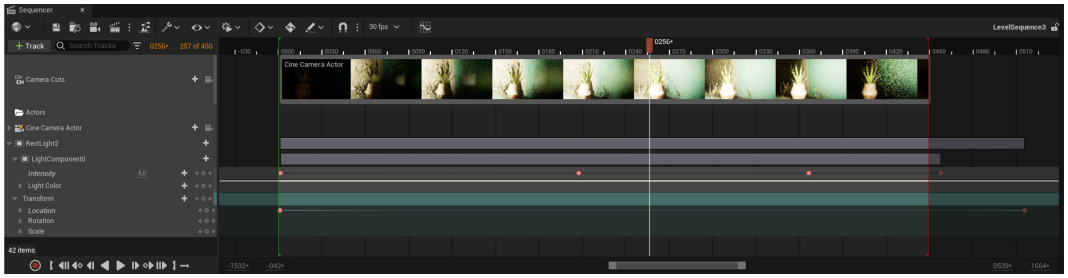

Figura 3.3. Sequencer Editor di una scena con l'aloe a El-Hammamiya.

La finestra del Sequencer contiene, come mostrato in Figura 3.3, il Sequencer Editor, un'interfaccia specializzata alla produzione di contenuti cinematografici. Essa si compone di:

- *Toolbar*: barra degli strumenti che offre opzioni e impostazioni per interagire con il Level Sequence Asset. Tra gli strumenti vi è la possibilità di definire una Cine Camera Actor con una Camera Cut Track che le farà da riferimento, l'opportunità di avviare il rendering tramite una finestra di Movie Render Queue o finalizzare i *keyframe* tramite il Curve Editor.
- *Outliner*: sezione che contiene un elenco di tutte le *track* del Level Sequence Asset, nonché gli strumenti per inserirle, cercarle rapidamente e filtrarle. Le *track* possono rappresentare degli Actor collegati al Level Sequence, come personaggi, effetti, camere e audio.
- *Timeline*: rappresenta il cuore dell'editing non lineare mostrando l'intervallo temporale della sequenza animata attraverso *track* orizzontali, all'interno delle quali possono essere posizionati *asset*, *keyframe* e controlli. L'intervallo di riproduzione del Level Sequence Asset è contenuto tra i marcatori Start ed End mentre la posizione corrente è indicata dal Playhead.
- *Playback Controls*: pannello che include i comandi per gestire la riproduzione della sequenza come play, pausa, registrazione del movimento di un Actor e altri analoghi a quelli comunemente utilizzati nei software di riproduzione multimediale.

3.1.4. Movie Render Queue

Il Movie Render Queue è un plugin per Unreal Engine specializzato nel rendering di filmati. Lo strumento supporta diverse funzionalità per la produzione di rendering di alta qualità, come il sottocampionamento temporale che aiuta a produrre *motion blur*[52] radiali di alta qualità e la possibilità di esportare immagini contenenti valori di pixel traslucidi. Inoltre, consente di generare immagini HDR a 16 bit con dati lineari e di salvare le configurazioni di rendering in *asset* da riutilizzare e condividere. Un ulteriore vantaggio è dato dalla possibilità di gestire più lavori e le relative impostazioni contemporaneamente utilizzando la coda di rendering, che supporta l'esecuzione di lavori di rendering in *batch*[53].

A partire dal Sequencer, è possibile avviare la finestra di rendering selezionando l'opzione di Movie Render Queue, a condizione che il relativo plugin sia correttamente installato e abilitato.

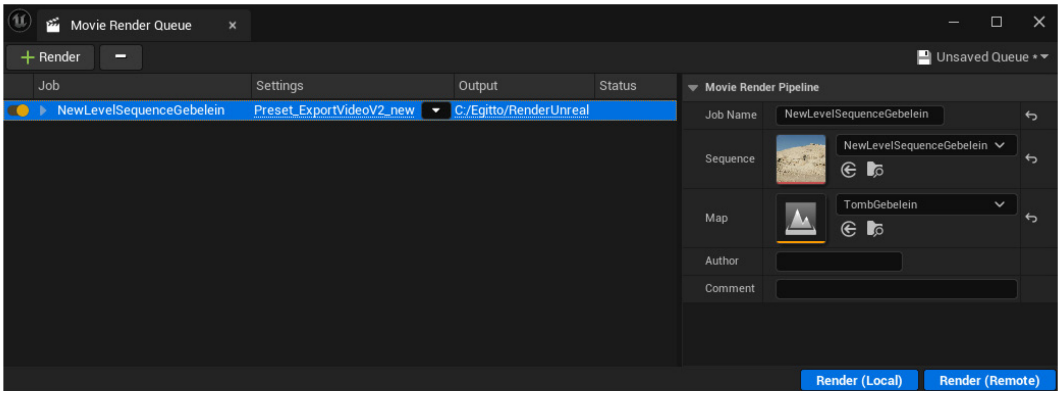

Figura 3.4. Schermata del Movie Render Queue.

L'interfaccia del Movie Render Queue, mostrata in Figura 3.4, si articola in quattro aree principali:

- *Toolbar*: consente di aggiungere o rimuovere le sequenze da sottoporre a rendering, nonché di salvare l'elenco delle sequenze attualmente in coda.

[52] Consiste nell'applicare un effetto di sfocatura quando si hanno oggetti in movimento.
[53] L'esecuzione dei lavori di rendering in *batch* (cioè una lista di lavori) si riferisce al processo di rendering di una serie di scene in modo automatico senza la necessità di un intervento manuale. Questo approccio è utile per ottimizzare i flussi di lavoro nel caso di rendering onerosi e lunghi.

- *Jobs*: mostra le sequenze da renderizzare nell'ordine in cui sono in coda, insieme alle relative impostazioni di rendering e alla directory di output.
- *Job Details*: visualizza i dettagli della sequenza selezionata, tra cui il nome, il Level Sequence Asset, il livello coinvolto nell'operazione di rendering e l'autore del progetto.
- *Start Render*: permette di avviare il rendering in locale sul proprio computer o in remoto attivando un processo separato. Quest'ultima modalità può essere utilizzata per implementare una *render farm* ossia un'infrastruttura di server dedicati all'elaborazione parallela dei render.

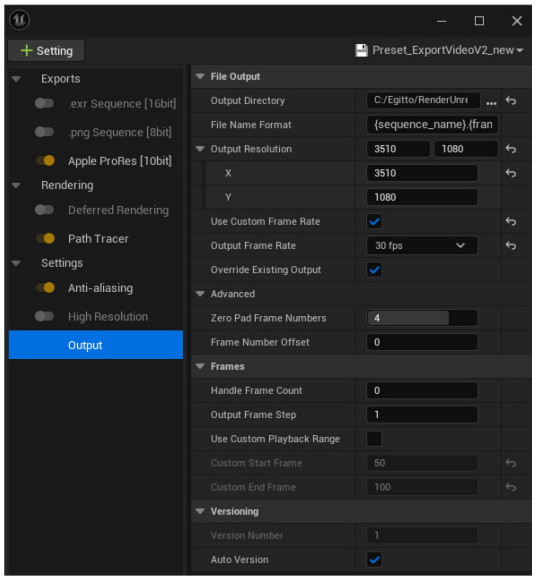

Figura 3.5. Impostazioni del Movie Render Queue.

Una volta inserita una sequenza all'interno del Movie Rende Queue, è possibile regolare alcune delle impostazioni di rendering. Queste opzioni vengono utilizzate per personalizzare il modo in cui vengono renderizzate le sequenze e sono suddivise in tre categorie: *Settings*, *Exports* e *Rendering*. Per la realizzazione delle sequenze del progetto immersivo, si è scelto di regolare quattro impostazioni, come mostrato in Figura 3.5. Queste sono:

- *Output*: definisce la directory del file di output, il nome del file, il frame rate e la risoluzione finale del prodotto.

- *Anti-Aliasing*: regola il numero di campioni utilizzati per produrre il rendering al fine di ridurre gli artefatti di *aliasing*[54]. Il Movie Render Queue consente di combinare più render insieme per produrre un singolo fotogramma. Ciò aumenta significativamente la qualità dell'*anti-aliasing*, del *motion blur* e può ridurre il rumore del *ray-tracing*. Sono possibili due tipi di campionamento che producono il rendering, spaziale e temporale. Il campionamento temporale utilizza le informazioni dei frame precedenti e futuri per calcolare l'*aliasing*. Unreal Engine suddivide l'apertura dell'otturatore della camera in *time slice*, elaborando ognuna di esse mediante interpolazioni basate su campioni temporali. Per ogni campione temporale, ci sono tanti render accumulati in base alla variabile "Spatial Sample Count". Il campionamento spaziale utilizza più campioni di pixel per ogni singolo fotogramma, migliorando la resa visiva della scena. In Unreal Engine ciò si realizza prendendo ogni campione e renderizzarlo più volte, facendo vibrare leggermente la camera ad ogni iterazione. Risulta utile per i render che hanno un *motion blur* limitato e sono comunque necessari più campioni per aumentare l'*anti-aliasing* o ridurre il rumore. Qualsiasi combinazione di campionamento spaziale e temporale produce lo stesso schema di offset utilizzato per l'*anti-aliasing*. Ciò significa che è generalmente più efficiente usare il campionamento temporale anziché spaziale, poiché gli oggetti fermi riceveranno lo stesso *anti-aliasing* e gli oggetti in movimento saranno sfocati, nascondendo l'*aliasing*.
- *Path Tracer*: abilita un sistema di rendering basato sul metodo del *path-tracing*[55], che accumula progressivamente i percorsi della luce per ciascun

[54] Effetti di scalettatura che si verificano nei bordi degli oggetti. L'*aliasing* si verifica quando il numero di pixel utilizzato per rappresentare un'immagine è insufficiente per catturare tutti i dettagli. Risulta essere evidente in oggetti a linee sottili o contorni curvi.

[55] Il *path-tracing* è un algoritmo di rendering che simula il modo in cui la luce interagisce con oggetti, voxel e media coinvolti per generare immagini realistiche. Il processo prevede il tracciamento di raggi di luce che, partendo dalla camera, si propagano nella scena interagendo con le superfici degli oggetti e con i materiali. Rispetto al *ray-tracing*, simula non solo i raggi diretti ma anche i rimbalzi della luce e tutti i suoi possibili percorsi. Ogni raggio, infatti, dopo aver colpito la superficie di un oggetto genera ulteriori raggi, i quali possono rimbalzare, essere assorbiti o rifratti, creando degli effetti di illuminazione molto naturali. Per raggiungere dei risultati realistici esige una notevole potenza computazionale richiedendo numerose iterazioni (*samples*) che consentono di ridurre il rumore che viene generato.

fotogramma che viene renderizzato. Questo comporta del rumore man mano che i pixel dell'immagine vengono riempiti, specialmente se i contenuti del fotogramma della camera cambiano in modo significativo. Per mitigare questo rumore, è possibile aumentare il numero di campioni spaziali nelle impostazioni di *anti-aliasing*.

- *Apple ProRes*: è il formato di compressione video lossy di alta qualità di Apple per la post-produzione e gli effetti visivi, che supporta video fino a 8K. Selezionando questo formato, il rendering produrrà un file video MOV utilizzando uno dei vari codec Apple ProRes, ognuno dei quali fornisce un diverso livello di bitrate ed una profondità di colore variabile tra 10 e 12 bit.

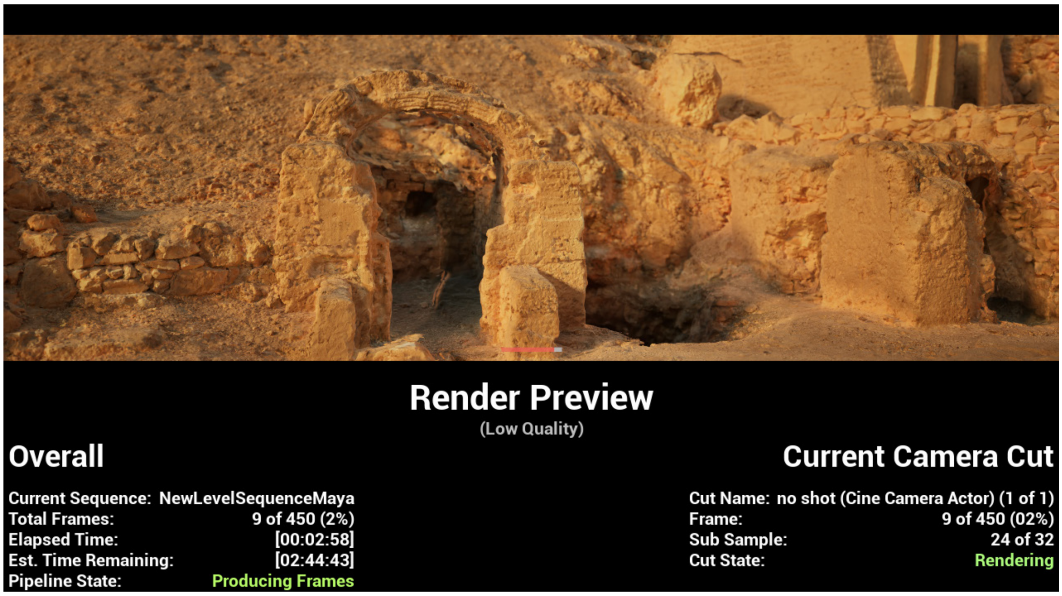

Figura 3.6. Render Preview del Movie Render Queue.

Quando un processo di rendering è in esecuzione, il Movie Render Queue mostra una finestra di anteprima, come in Figura 3.6, che riporta lo stato visivo del rendering con delle informazioni che si possono raggruppare in:

- *Render Preview*: mostra un'anteprima del render corrente basandosi sui *sample* più recenti elaborati dalla GPU.
- *Overall Render Progress*: fornisce informazioni sull'avanzamento generale del rendering, inclusi il tempo trascorso, quello stimato e la sequenza in fase di elaborazione.

- *Current Camera Cut Progress*: riporta indicazioni sulla camera che viene renderizzata e lo stato di avanzamento relativo al suo "Cut State".

Al termine del processo di renderizzazione, è possibile apportare correzioni minori alla clip video esportata utilizzando un software di editing, come ad esempio DaVinci Resolve, oppure impiegarla direttamente nel montaggio finale.

3.1.5. Ultra Dynamic Sky

Ultra Dynamic Sky è un plugin sviluppato per Unreal Engine che consente di creare cieli dinamici e visivamente realistici in un'ambiente digitale. All'interno della directory del plugin sono presenti due Blueprint responsabili, rispettivamente, della gestione della volta celeste e degli eventi meteorologici quali pioggia, nebbia e copertura nuvolosa.

Ultra Dynamic Sky è considerato uno strumento estremamente potente e versatile per via della vasta gamma di parametri consentono la configurazione di condizioni atmosferiche estremamente variegate.

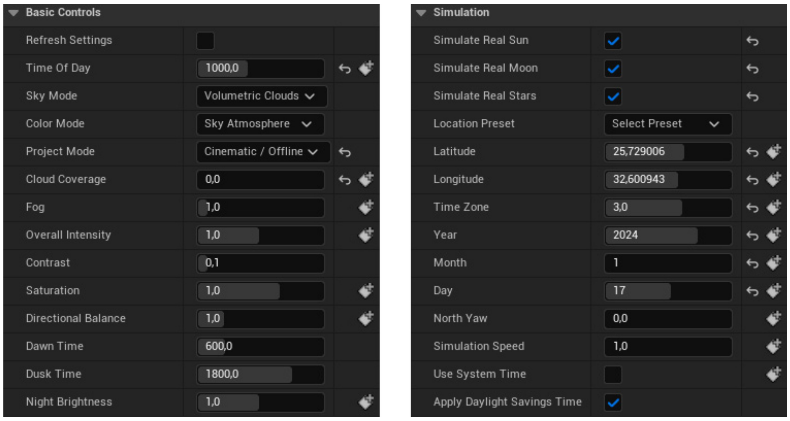

Figura 3.7. Dettagli dell'Actor di Ultra Dynamic Sky.

Attraverso il pannello dei dettagli del Blueprint è possibile accedere a numerose impostazioni, parzialmente riportate in Figura 3.7, per personalizzare il cielo in una scena, gestendo anche la nebbia, le nuvole e gli effetti meteorologici.

Nel pannello è possibile regolare l'aspetto generale del cielo, la posizione e il colore del sole e la densità delle nuvole. Ci sono opzioni anche per regolare l'intensità

della luce ambientale e la visibilità delle stelle. Il plugin permette anche di controllare i tempi e la durata dell'illuminazione diurna. Ciò include parametri per impostare l'ora di inizio e fine del giorno, così come la durata del giorno e della notte, la velocità della progressione temporale e il movimento delle nuvole.

Tra le funzionalità utilizzate per l'installazione immersiva vi è l'opportunità di impostare una determinata ora del giorno che modifica l'illuminazione globale della scena e la possibilità di selezionare una specifica data e localizzazione geografica per simulare in modo realistico la posizione e il percorso del sole e della luna in base alla rotazione della Terra. In questo modo, lo strumento è in grado di ricreare in maniera fedele e coerente il ciclo naturale del giorno e della notte di una determinata area geografica.

3.2. La Tomba di Kha e Merit

Nel contesto del progetto *Paesaggi – Landscapes*, sono state sviluppate diverse scene incentrate su giochi di luce e sequenze di *time-lapse*, aventi come oggetto la vegetazione egiziana e i siti archeologici collegati al Museo Egizio di Torino.

Dopo aver trattato nel capitolo precedente dei fondamenti della fotogrammetria, i paragrafi successivi illustreranno più in dettaglio l'intero *workflow* che conduce, a partire da immagini di un sito archeologico, alla creazione di uno *shot* in computer grafica renderizzato mediante Unreal Engine.

Tra le varie scene presenti all'interno della mostra immersiva, verranno descritte tutte le fasi che hanno condotto alla realizzazione del *time-lapse* della tomba dell'architetto reale Kha e di sua moglie Merit, di cui sono riportate alcune immagini in Figura 3.8. Kha, sovrintendente ai lavori della necropoli di Tebe, fu al servizio di ben tre faraoni: Amenhotep II, Thutmosi IV e Amenhotep III. Egli costruì la cappella della propria tomba, orientata a nord-est e rivolta verso il tempio funerario del suo sovrano Amenhotep III, per onorarlo.

Figura 3.8. Alcune immagini acquisite per la fotogrammetria della tomba.

Scoperta nel 1906 dalla missione archeologica diretta dall'egittologo Ernesto Schiaparelli e identificata con la sigla TT8 (*Theban Tomb 8*), la sepoltura appartiene alle cosiddette Tombe dei Nobili, situate nella necropoli di Tebe sulla riva occidentale del Nilo, dinanzi la città di Luxor. Più precisamente, essa si trova nel villaggio di Deir el-Medina[56], antico insediamento abitato dagli operai e artigiani addetti alla costruzione delle tombe dei faraoni nella Valle dei Re e delle mogli nella Valle delle Regine, abbandonato intorno al XII secolo a.C. La mancata rioccupazione del sito ha permesso la conservazione quasi intatta della struttura originaria del villaggio e i resti delle sepolture degli artigiani.

La tomba di Kha e Merit è considerata ad oggi quella che si trova nel miglior stato di conservazione tra le sepolture egizie non regali, e si distingue per l'eccezionale corredo funerario rinvenuto al suo interno. Esso comprende testi religiosi, gioielli, arredi, oggetti d'uso quotidiano, abiti e generi alimentari, attualmente custoditi in larga parte presso la sala 7 del Museo Egizio di Torino, situata al primo piano del Palazzo dell'Accademia delle Scienze.

Il passo iniziale per la realizzazione della scena digitale ha riguardato la ricostruzione di un modello tridimensionale dell'area sepolcrale ed in particolar modo della cappella relativa alla tomba. Questa operazione è stata svolta tramite tecniche di fotogrammetria, partendo da riprese fotografiche del sito archeologico. Le acquisizioni sono state effettuate impiegando una fotocamera Canon EOS R5 C con obiettivi da 50mm e 25mm, una risoluzione di 8192×5464 pixel, sensibilità ISO impostata a 250 e apertura del diaframma regolata tra f/4 e f/5.

[56] Chiamato dagli antichi egizi *Pa Demi*, letteralmente significa "Convento della città". Questo è dovuto alla presenza di un convento costruito dai cristiani copti intorno al V secolo d.C., ormai andato perduto.

Le immagini ottenute sono state importate nel software Metashape, in modo da avviare la fase di allineamento e generare una nuvola sparsa di *tie-points*. Impostando un livello di accuratezza medio, l'algoritmo ha rilevato 129945 punti. Successivamente, utilizzando come sorgente la mappa di profondità, è stata ricavata la mesh tridimensionale, impostando una modalità di filtraggio "Mild"[57] ed una qualità di ricostruzione media. L'intero processo ha richiesto alcune ore di elaborazione, producendo risultati soddisfacenti nonostante la presenza di lacune dovute ad informazioni mancanti in alcune zone del sito. Dopo aver ottenuto la mesh, è stata generata la relativa texture, impostando una risoluzione di 8192×8192, una mappatura generica[58] ed una modalità di fusione a mosaico[59].

Figura 3.9. Modello digitale ricostruito e versione ottimizzata senza ombre.

Il modello ottenuto, riportato a sinistra nella Figura 3.9, ha evidenziato il problema relativo alla presenza di ombre nelle texture. Tale criticità è dovuta al fatto che le riprese fotografiche sono state realizzate in un ambiente aperto, con condizioni di illuminazione non controllabili. Di conseguenza, in assenza di soluzioni, il modello risulterebbe inutilizzabile per gli scopi del progetto in quanto, sottoponendosi ad

[57] Le modalità di filtraggio controllano la riduzione del rumore nelle mappe di profondità grezze. Se ci sono dettagli piccoli e significativi spazialmente distinti nella scena da ricostruire, è consigliato impostare la modalità di filtraggio della profondità "Mild" (leggera), affinché le caratteristiche importanti non vengano erroneamente scartate come valori anomali.

[58] La modalità di mappatura determina come la texture dell'oggetto sarà inserita nel *texture atlas* (ovvero lo *sprite*). La modalità di mappatura generica consente di parametrizzare il *texture atlas* per geometrie arbitrarie. Non vengono fatte assunzioni riguardo al tipo di scena da elaborare, il programma cerca di creare una texture il più uniforme possibile.

[59] La modalità di fusione gestisce il modo in cui i valori di colore dei pixel provenienti da camere diverse saranno combinati nella texture finale. La tecnica a mosaico implica un approccio a due fasi: esegue la miscelazione della componente a bassa frequenza per le immagini sovrapposte per evitare il problema della linea di giunzione, mentre la componente ad alta frequenza, che è responsabile dei dettagli, viene presa da una singola immagine ossia quella che presenta una buona risoluzione per l'area di interesse, mentre la visuale della telecamera è quasi lungo la normale alla superficie ricostruita in quel punto.

un'illuminazione dinamica di un *time-lapse*, non garantirebbe delle zone d'ombra coerenti dal punto di vista visivo.

Per ovviare a tale problema, è stata sistemata la texture mediante lo strumento Texture De-Lighter. Il software, in base alle indicazioni manuali delle aree illuminate e quelle in ombra, ha applicato un filtro correttivo che ha permesso di attenuare le zone luminose e rimuovere quelle in ombra, restituendo una texture complessivamente più uniforme. Nella Figura 3.9 è illustrato il risultato della procedura di uniformazione dell'illuminazione sulla texture del modello, seguita dalla rimozione delle ombre più evidenti.

Il modello tridimensionale, generato tramite Metashape, presentava inizialmente circa 4 milioni di facce e 2 milioni di vertici. La geometria risultava pertanto eccessiva per un utilizzo ottimale all'interno del motore grafico Unreal Engine. Per tale ragione, prima dell'importazione nel motore di rendering, si è resa necessaria un'operazione di semplificazione della mesh esportata in formato OBJ, svolta tramite Blender.

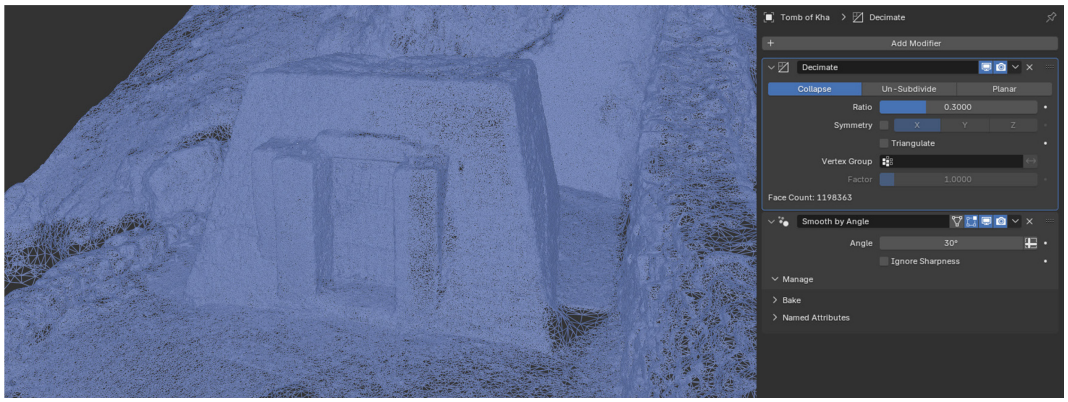

Figura 3.10. Visualizzazione *wireframe* del modello e finestra dei modificatori.

Con l'ausilio del modificatore "Decimate" di Blender, sono stati ridotti i poligoni della mesh mantenendo inalterata la forma complessiva. In particolare, è stata utilizzata la modalità *Collapse* del modificatore con un coefficiente di compressione pari 0.3, che ha permesso di unire progressivamente i vertici secondo la morfologia del modello. Inoltre, è stato applicato il modificatore "Smooth by Angle" impostando un angolo di soglia pari a 30°, al fine di gestire la nitidezza dei

bordi in funzione dell'orientamento delle facce adiacenti. La Figura 3.10 mostra una visualizzazione *wireframe* del modello con i modificatori attivi.

A questo punto, il modello esportato da Blender in FBX è stato importato all'interno di Unreal Engine ed inserito in un *Level* creato appositamente per la realizzazione della scena. All'interno di questo ambiente, sono stati aggiunti altri elementi per realizzare la sequenza in *time-lapse*: la Cine Camera Actor, il Level Sequencer Actor, il Post Process Volume ed il Blueprint del plugin Ultra Dynamic Sky.

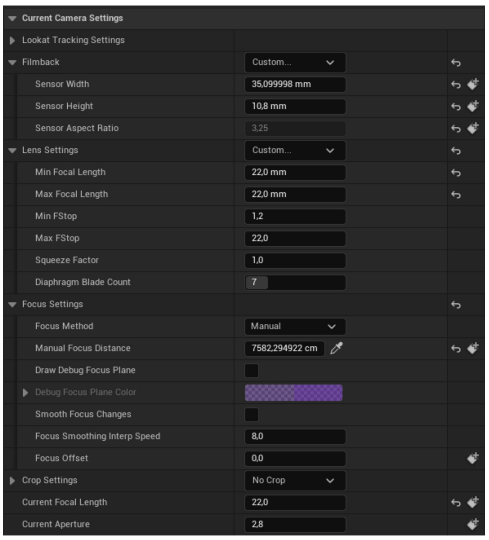

Figura 3.11. Impostazioni della Cine Camera Actor.

La Cine Camera Actor rappresenta una telecamera avanzata di Unreal Engine, dotata di impostazioni che replicano il comportamento reale della cinepresa. Attraverso i parametri delle sezioni *Filmback*, *Lens* e *Focus*, illustrati in Figura 3.11, è possibile realizzare inquadrature realistiche capaci di rispettare gli standard dell'industria cinematografica. La sezione *Filmback* fornisce delle preimpostazioni corrispondenti ai corpi macchina reali, emulando le dimensioni del sensore e le proporzioni del display. Per lo *shot* in oggetto, sono stati scelti i parametri "Sensor Width" e "Sensor Height" in modo da ottenere un Aspect Ratio pari a 3,25[60]. La sezione *Lens* offre delle opzioni per selezionare obiettivi della fotocamera predefiniti, impostare la lunghezza focale, l'intervallo di apertura della camera,

[60] Il valore fa riferimento al rapporto tra larghezza e altezza di una risoluzione pari a 3510×1080.

specificare un fattore di compressione per obiettivi anamorfici e controllare il diaframma. Nello specifico, è stato impostato un intervallo di apertura compreso tra f/1.2 e f/22 con una lunghezza focale di 22 mm. Infine, la sezione *Focus* include strumenti per il controllo manuale della distanza di messa a fuoco, la possibilità di utilizzare un Actor come riferimento per la profondità di campo della telecamera, opzioni visive per il debug della messa a fuoco e ulteriori parametri. Per la scena in esame, è stata adottata una modalità di messa a fuoco manuale, indicando come distanza di riferimento quella che intercorre tra la camera e il centro fisico della cappella della tomba di Kha e Merit.

Nel Blueprint di Ultra Dynamic Sky sono stati configurati vari parametri per definire la resa visiva del cielo, dell'illuminazione solare e della nebbia in modo da riprodurre fedelmente le condizioni ambientali tipiche del panorama egiziano, generalmente privo di nuvole. Nella sezione "Simulation", abilitando le opzioni "Simulate Real Sun" e "Simulate Real Moon", è stata sostituita la logica predefinita per il posizionamento del sole e della luna sulla base delle coordinate del sito archeologico e della data di acquisizione delle immagini fotografiche. Inoltre, attivando "Simulate Real Stars", è stata impiegata una mappa stellare a 360° al posto di una semplice texture tessellata. Nella sezione "Project Mode", che determina il comportamento del sistema in fase di esecuzione, è stata selezionata la modalità "Cinematic/Offline". Pensata per il rendering di filmati o immagini fisse, questa modalità disabilita le ottimizzazioni utili in condizioni di rendering in *real-time*, in modo che tutto ciò che è dinamico nel cielo venga aggiornato completamente a ogni fotogramma. Inoltre, aumenta automaticamente molte impostazioni assicurando la massima qualità del rendering finale.

Il Level Sequence Actor è stato configurato per ricevere in input il Level Sequence Asset che contiene i dati del Sequencer, lo strumento usato da Unreal Engine per gestire le animazioni della scena. All'interno del Sequencer è stato animato, tramite *keyframe*, il parametro "Time Of Day" relativo al Blueprint di Ultra Dynamic Sky, al fine di simulare il rapido scorrere del tempo, tipico di un *time-lapse*. In particolare, nella sequenza composta da 450 frame, l'orario simulato si estende dalle 10:00 del mattino fino a circa le 18:30 del pomeriggio. Per rendere più fluida

la transizione verso l'illuminazione serale, è stato inserito un *keyframe* intermedio nella timeline della variabile "Time Of Day", correggendo la curva di interpolazione in modo che la pendenza possa risultare minore negli ultimi 60 fotogrammi. Questo accorgimento ha attenuato l'effetto di un brusco passaggio al buio, che sarebbe stato raggiunto con una transizione lineare senza modulazione.

Attraverso il Post Process Volume della scena sono state configurate delle specifiche proprietà per il rendering in *path-tracing*, una volta abilitato il supporto all'hardware *ray-tracing* nelle impostazioni del progetto. Nello specifico, il numero massimo di rimbalzi dei raggi di luce è stato impostato a 32, i campioni per pixel a 4096 ed associato un valore pari a 30 all'esposizione massima consentita per il *path tracing*, al fine di limitare la comparsa di artefatti simili a "lucciole". Inoltre, è stato abilitato l'Intel Open Image Denoise con l'obiettivo di ridurre il rumore presente nelle immagini renderizzate.

Il processo di rendering dei fotogrammi è stato effettuato tramite il Movie Render Queue, impostando una risoluzione di output di 3510×1080 a 30fps. Come tecnica di *anti-aliasing* è stato adottato il MultiSample Anti-Aliasing (MSAA) con 8 campioni per pixel, così da ridurre l'*aliasing* della geometria lungo i bordi. Nelle opzioni del Path Tracer è stato abilitato il "Reference Motion Blur" in modo da rendere più accurata la qualità del *motion blur* all'interno della scena. In fase di esportazione, è stato utilizzato il codec Apple ProRes 422 HQ, con profondità colore a 10 bit, che consente una compressione video *lossy* di alta qualità.

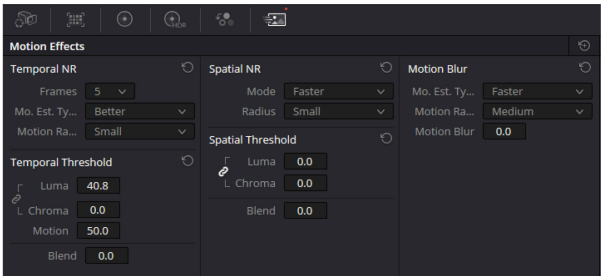

Figura 3.12. Schermata sui Motion Effects di DaVinci Resolve.

Successivamente, la clip video esportata è stata sottoposta ad operazioni di post-produzione su DaVinci Resolve al fine di migliorarne la qualità visiva. In particolare, nonostante si fosse utilizzato un rendering in *path-tracing* con tutte le

accortezze per diminuire il rumore sui pixel, durante la transizione repentina dell'illuminazione da diurna a notturna erano ancora presenti delle zone rumorose. Per tale motivo, è stato utilizzato il pannello di Motion Effects di DaVinci Resolve, riportato in Figura 3.12, che consente di ridurre gli effetti di grana che possono distorcere la nitidezza e l'aspetto generale del contenuto, ottenendo un prodotto finale più pulito e visivamente più accattivante. Il pannello si articola in tre aree di controllo: "Temporal NR" che analizza il video su più fotogrammi per rilevare soggetti e sfondi in movimento, "Spatial NR" che attenua il rumore ad alta frequenza mantenendo i dati con un elevato livello di dettaglio e "Motion Blur" che aggiunge sfocature artificiali ai soggetti in movimento.

Nella clip in esame è stata applicata una riduzione di tipo "Temporal NR" impostando un valore di 5 ai *Frame Count* utilizzati per la separazione dei dettagli dal rumore e una *Motion Estimation Type* "Better" per una rilevazione del movimento con buona precisione. Inoltre, è stato selezionato un *Motion Range* "Small", adatto per soggetti statici con minima sfocatura, consentendo a una parte maggiore dell'immagine di essere interessata dalla riduzione del rumore.

Attraverso i parametri di "Temporal Threshold" è stata modulata l'intensità con cui la riduzione del rumore viene applicata ai livelli di luminanza e crominanza. In questo caso, l'intervento è stato applicato solamente alla componente di luminanza, con un valore di riduzione di circa 40, lasciando la soglia che separa i pixel in movimento da quelli statici al valore predefinito.

Infine, dopo aver apportato le correzioni sul rumore, la clip è stata esportata mantenendo le stesse codifiche video utilizzate in Unreal Engine. In Figura 3.13 sono riportati alcuni fotogrammi rappresentativi del *time-lapse* sulla cappella della tomba di Kha e Merit, in diverse fasi della giornata. È possibile osservare l'elevato impatto visivo della scena, ottenuto grazie al realismo reso possibile dall'ultima versione del motore grafico di Epic Games, capace di gestire in maniera perfettamente accurata l'illuminazione dinamica e le ombre proiettate sul modello fotogrammetrico del sito archeologico.

Figura 3.13. Fotogrammi in varie fasi del giorno della cappella di Kha e Merit.

3.3. Altri scenari

Oltre alla tomba di Kha e Merit, il progetto ha previsto la realizzazione di ulteriori sequenze di *time-lapse* in computer grafica incentrate su differenti siti archeologici, tra cui Gebelein, El-Hammamiya e la Valle delle Regine. Altre inquadrature, invece, sono state dedicate alla riproduzione della flora tipica egiziana. In questi casi, tuttavia, non si tratta di simulazioni *time-lapse*, bensì di scene caratterizzate da giochi di luce che attraversano la vegetazione.

Nei paragrafi seguenti verrà presentata una carrellata di *shot* rappresentativi, sia dei suddetti siti archeologici, sia delle composizioni ambientali focalizzate sulla vegetazione.

3.3.1. Le piante alofite e xerofile

La vegetazione desertica rappresenta, di gran lunga, la componente più importante e caratteristica della flora naturale egiziana. Essa occupa vastissime aree ed è formata principalmente da arbusti e "piante xerofile" come l'Aloe vera e la Leptadenia, in grado di tollerare condizioni di siccità prolungata. Lungo le coste del Nilo e nelle paludi salmastre possono trovarsi le "piante alofite" come le tamerice, in grado di resistere e adattarsi alle condizioni saline. Tutte queste specie crescono in condizioni estreme e alcune di esse sono particolarmente dominanti negli ecosistemi locali.

Figura 3.14. Fotogramma di una sequenza sulle piante desertiche egiziane.

La realizzazione di scenari con piante e flora tipica dell'ambiente egiziano è stata inizialmente concepita per essere inserita all'interno della mostra *Paesaggi – Landscape*, nella fase di analisi del contesto. A tal fine, sono state create delle ambientazioni virtuali recuperando dei Megascan di piante dalla libreria di Quixel Bridge ed inseriti nei paesaggi rurali digitali con la Foliage Mode di Unreal Engine.

Una volta composte le scene in modo da evocare atmosfere suggestive e riconducibili al paesaggio egiziano, sono state aggiunte luci puntiformi per illuminare selettivamente le aree di interesse affiancate da Rect Light[61] di tipo

[61] Una Rect Light emette luce da un piano rettangolare con larghezza e altezza definite. Ogni Rect Light ha due impostazioni chiave, "Source Width" e "Source Height", che determinano la dimensione del suo rettangolo lungo gli assi Y e Z locali. Inoltre, presenta un raggio di attenuazione sferico, simile a quello di una Point Light. Le Rect Light hanno tre impostazioni di mobilità: "Static" (sia l'illuminazione diretta che quella indiretta della luce vengono precostruite nella *lightmap*), "Stationary" (solo l'illuminazione indiretta della luce viene precostruita mentre quella diretta è calcolata dinamicamente), "Movable" (tutto viene calcolato dinamicamente).

"Movable", ovvero luci completamente dinamiche. Queste sono state animate nel Sequencer, tramite dei *keyframe*, nelle componenti di posizione spaziale e intensità luminosa. In questo modo, una Rect Light entrava progressivamente in scena aumentando la quantità di luce emessa, la manteneva per un breve intervallo di tempo e infine si affievoliva gradualmente mentre usciva dall'inquadratura.

In Figura 3.14 e Figura 3.15 sono riportati due fotogrammi di sequenze di scene ottenute utilizzando tamerice, sterpaglie e pietrisco tipici delle aree desertiche.

Figura 3.15. Fotogramma di una sequenza su piante desertiche e pietre.

Tra le piante xerofile maggiormente rappresentative degli usi e costumi dell'antico Egitto figura l'aloe vera. Impiegata nei rituali e nelle offerte religiose in qualità di simbolo di protezione e rinascita, la testimonianza scritta più antica dell'aloe risale al 1500 a.C., nel celebre Papiro Ebers, i cui originali sono conservati presso l'Università di Lipsia. Questi testi egizi evidenziano come le proprietà curative dell'aloe fossero già ampiamente riconosciute da secoli, e il loro utilizzo medicinale fosse diffuso per l'apparente efficacia miracolosa nel trattamento di ustioni, ferite e diverse forme di dolore.

Figura 3.16. Fotogramma di una sequenza su un aloe in primo piano.

Nella mostra *Paesaggi – Lanscapes* la figura dell'aloe è stata integrata in varie forme e contesti. Tra le soluzioni proposte durante la fase di ideazione dei contenuti digitali vi era anche la realizzazione di sequenze ispirate alle scene vegetali descritte nel paragrafo precedente. Come illustrato in Figura 3.16, utilizzando degli *asset* di Quixel Bridge e delle luci dinamiche, è stato realizzato uno *shot* che generava un suggestivo contrasto di luci e ombre con le sue caratteristiche foglie rigide e lanceolate, dotate di apice appuntito e spine ai lati.

Figura 3.17. Fotogramma di una sequenza su un aloe con vaso predinastico.

In una delle scene realizzate, caratterizzata da una forte connotazione simbolica, l'aloe è stata inserita all'interno del modello fotogrammetrico di un vaso predinastico, collocato in un contesto ambientale che richiamava le tombe musulmane del cimitero di El-Hammamiya di cui un render è riportato in Figura 3.17. I vasi predinastici rappresentavano un elemento essenziale del corredo funerario dei defunti e, nella cultura egizia, venivano metaforicamente interpretati come contenitori di vita, protezione e rinascita.

3.3.2. I siti archeologici

Le sequenze di *time-lapse* in computer grafica, realizzate su Unreal Engine, hanno riguardato una moltitudine di aree archeologiche egiziane. Oltre a Deir el-Medina, di cui si è già descritta la tomba di Kha e Merit, sono stati ricostruiti digitalmente ulteriori siti che riguardavano la Valle delle Regine e le necropoli di Gebelein e El-Hammamiya. Le ricostruzioni, basate sulle stesse tecniche di fotogrammetria descritte nei paragrafi precedenti, sono state sviluppate a partire da scansioni aeree e immagini acquisite a distanza ravvicinata.

Figura 3.18. Fotogramma di un *time-lapse* sulla Cappella di Maia.

Un'ulteriore *shot* dedicato al villaggio di Deir el-Medina ha riguardato la cappella di Maia, visibile in Figura 3.18. Conosciuta anche come TT338 (*Theban Tomb 338*), la struttura è stata scoperta nell'ambito della stessa missione archeologica che ha portato alla luce la tomba di Kha e Merit. Pittore della necropoli regale, Maia visse nella seconda metà della XVIII dinastia nel villaggio insieme agli artigiani specializzati nelle decorazioni degli ipogei regali. In questo contesto, egli realizzò la propria cappella funeraria, la cui struttura è costituita da mattoni di fango e paglia, successivamente ricoperti da intonaco dipinto con tempera applicata a secco i cui colori erano stati ricavati da pigmenti minerali e vegetali. Questi dipinti murali rappresentano un importante testimonianza della tecnica pittorica egizia. Le decorazioni originali, asportate "a strappo"[62] dal restauratore Fabrizio Lucarini, sono attualmente esposte all'interno della cappella ricostruita presso il Museo Egizio di Torino.

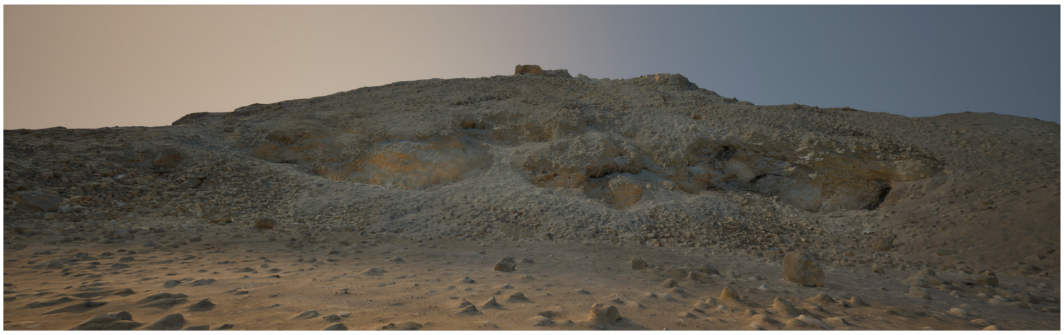

Figura 3.19. Fotogramma di un *time-lapse* su delle tombe a Gebelein.

[62] La tecnica consiste nell'applicare tele sulla superficie dipinta per mantenerla integra durante il processo di rimozione. Successivamente, si utilizzano solventi per separare le tele dalla pittura. Questo processo permette di non sezionare l'intonaco e di preservare al massimo i dipinti.

Una sequenza di *time-lapse*, mostrata in Figura 3.19, è stata riservata ad una delle aree cimiteriali di Gebelein. Il sito archeologico, situato a circa 40 km a sud di Tebe, è caratterizzato da due alture desertiche, da cui deriva il suo nome. Una delle colline conserva i resti della città egizia di *Per Hathor*, la casa di Hator e l'altra ospita le vestigia della città di *Inty-shuy*, nota in epoca greca con il nome di Crocodilopoli. Gebelein costituì, già dalle epoche faraoniche, un importante centro abitato di cui si ha testimonianza fino al periodo tolemaico. Le indagini archeologiche condotte nell'area hanno portato alla luce numerose tombe, alcune delle quali ancora intatte con i loro corredi funerari, oltre a una vasta gamma di reperti databili tra il Periodo Predinastico e il Medio Regno.

Figura 3.20. Fotogramma di un *time-lapse* su delle tombe a El-Hammamiya.

Un'altra scena, raffigurata in Figura 3.20, documenta alcune tombe situate nel cimitero islamico del sito archeologico di El-Hammamiya, situato lungo la riva orientale del Nilo, a nord di Qaw el-Kebir. Questo cimitero contemporaneo si sviluppa sul margine dell'area archeologica, al di fuori delle tombe rupestri e delle necropoli predinastiche. Le tombe sono generalmente di tipo familiare, costruite con materiali locali e presentano recinti murari rettangolari con funzione di protezione delle sepolture. Un elemento distintivo è l'uso di colori vivaci, come il verde chiaro e il turchese, simbolicamente associati alla spiritualità islamica. Le tombe sono inoltre caratterizzate dalla presenza di targhe commemorative con iscrizioni in arabo, spesso riportanti il nome del defunto, versetti coranici e date di morte. Un ulteriore dettaglio architettonico ricorrente è la dentellatura sulla sommità dei muri, tipico dei cimiteri islamici rurali dell'Alto Egitto.

Un'ultima sequenza in *time-lapse* ha riguardato il sito archeologico della Valle delle Regine, luogo dove sono state sepolte le madri, le consorti e i figli dei faraoni. L'area, visibile in Figura 3.21, è costituita da un'estesa necropoli caratterizzata da tombe scavate nella roccia, situata alla convergenza di una serie di antichi letti di torrenti ormai prosciugati, che incidono profondamente le alture calcaree del deserto occidentale, presso Luxor. L'attuale denominazione, "Valle delle Regine", è attribuibile ai primi esploratori europei che, nel corso delle loro spedizioni, si avventurarono in questa porzione della necropoli tebana meridionale.

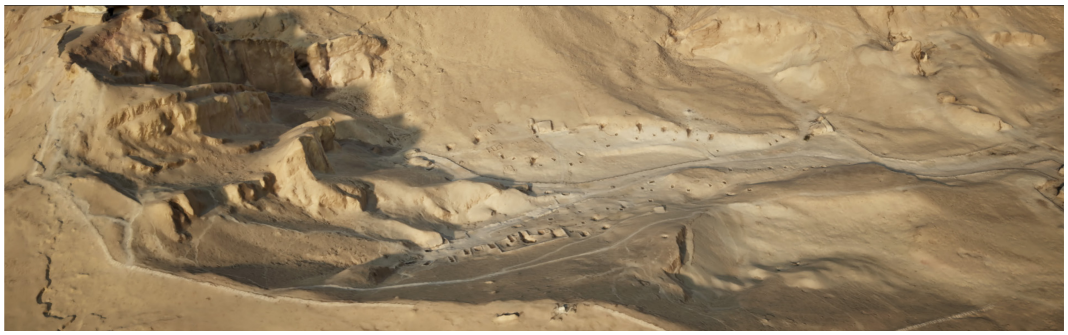

Figura 3.21. Fotogramma di un *time-lapse* sulla Valle delle Regine.

4. LE PIANTE

All'interno dell'esperienza immersiva, in alcune delle sequenze della fase di analisi, si voleva offrire un ruolo di primo piano alla vegetazione che contraddistingue l'Egitto mostrando alcune fasi dello sviluppo delle piante più caratteristiche. Nel panorama egiziano la vegetazione si sviluppa maggiormente nelle zone umide della valle del Nilo. In questi luoghi trovano il loro ambiente ideale, il papiro, il giunco ed il loto. In contrapposizione, nelle zone aride del deserto ed in prossimità delle oasi si trovano le piante xerofile che hanno la peculiarità di resistere alla siccità. Tra queste si citano l'acacia, l'aloe vera e la palma da dattero. Infine, nelle regioni con elevata concentrazione salina, si riscontrano piante alofite, come la tamerice e la salicornia.

In particolare, in questo capitolo verrà fatta una digressione su alcune metodologie per creare in maniera efficace delle animazioni di crescita per le piante utilizzando i Geometry Nodes di Blender. Successivamente verrà introdotto il software Houdini approfondendo in particolar modo i passaggi necessari alla realizzazione dell'animazione dell'aloe che viene mostrata all'interno dell'esperienza immersiva *Paesaggi – Landscapes*.

4.1. Le animazioni di crescita

Nelle prime fasi del progetto, si prevedeva la realizzazione di varie animazioni che riguardavano la crescita di piante ed in particolar modo dell'acacia. Attraverso degli stacchi di inquadratura si era pensato di dividere l'animazione in due parti: la prima che riguardasse il germoglio della pianta appena nata con uno stelo e qualche foglia; la seconda che mostrasse delle ramificazioni che sviluppavano delle infiorescenze.

Una delle sfide è stata quella di trovare una tecnica efficace che riuscisse a simulare in maniera realistica l'animazione di crescita di una pianta generica e successivamente adattarla a seconda delle necessità. Tra le varie metodologie di

animazione, quella che alla fine è stata presa in considerazione riguardava l'utilizzo di tecniche procedurali che facevano dei Geometry Nodes di Blender la loro componente principale.

4.1.1. I Geometry Nodes

Introdotti dalla versione di Blender® 2.92, i Geometry Nodes sono un sistema procedurale per realizzare e manipolare geometrie attraverso delle operazioni basate su nodi offrendo flessibilità ed elevate capacità di controllo rispetto agli approcci tradizionali. Un Geometry Nodes viene applicato ad una mesh come un normale modificatore di Blender. Un albero iniziale di un Geometry Nodes è costituito dal nodo *Group Input* che riceve lo stato della geometria della mesh prima dell'applicazione del modificatore e, attraverso delle specifiche operazioni, può operare sulla mesh fino ad arrivare al nodo *Group Output* dove il controllo della geometria viene passato al modificatore successivo. Lo scopo di ogni combinazione di nodi è trasformare una geometria iniziale in una geometria finale diversa, comportando una variazione, più o meno significativa, nel numero di facce e vertici. È anche possibile che la geometria finale possa risultare una combinazione di differenti mesh, sia primitive che precedentemente modellate.

Nel caso della pianta in esame, tramite i Geometry Nodes è possibile creare delle animazioni procedurali senza la necessità di operare manualmente a livello di *keyframe*. L'idea alla base è quella di utilizzare delle curve di Bézier[63] per delineare la crescita della pianta (in particolar modo del suo stelo) animandone la lunghezza e la crescita delle foglie al variare del tempo.

Verrà di seguito descritto il procedimento implementativo per la realizzazione della pianta prendendo come riferimento la Figura 4.1.

[63] Le curve di Bézier rappresentano il tipo più rilevante di curve polinomiali. Consistono in curve chiuse definite da un insieme di punti di controllo che ne influenzano la forma. Nella computer grafica sono ampiamente utilizzate per la modellazione di curve smussate. Le curve di Bézier rimangono completamente contenute nell'insieme convesso dei loro punti di controllo, permettendo così di manipolarle mediante l'applicazione di trasformazioni geometriche ai punti stessi.

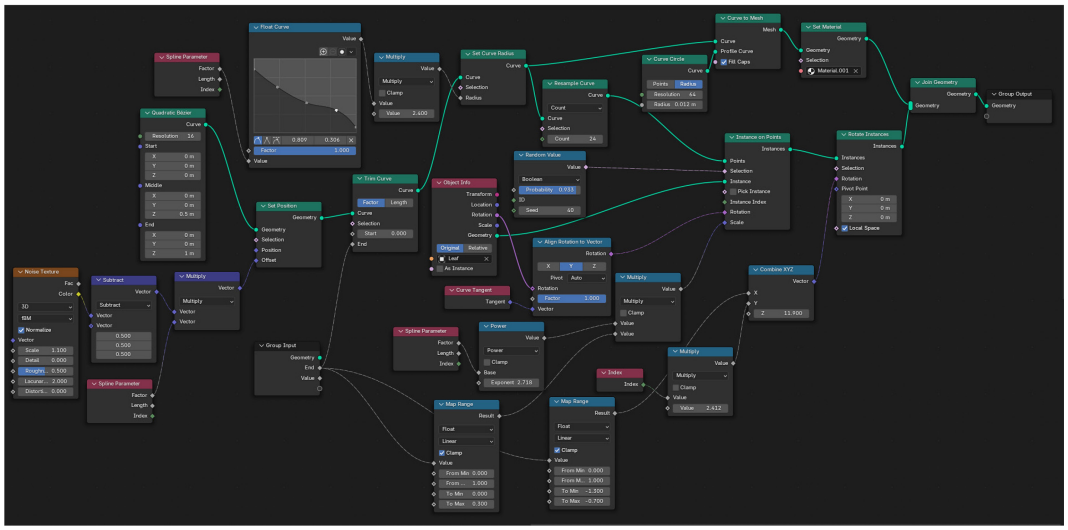

Figura 4.1. Geometry Nodes per la realizzazione della pianta.

Si inizia dallo stelo del germoglio costituito da una curva di Bézier parallela all'asse Z, il cui spessore è definito tramite i nodi "Curve to Mesh" e "Curve Circle". Per rendere lo stelo incurvato viene utilizzato un nodo di "Noise Texture" che consente di generare valori casuali che possano rendere più morbida la curvatura a seconda della scala della texture. Questo nodo viene applicato all'offset della geometria della curva subendo dapprima una sottrazione e quindi una moltiplicazione che riporta l'origine dello stelo nella posizione iniziale[64].

Per fare in modo che l'estremità dello stelo risulti più sottile rispetto alla base, il suo raggio (impostato col nodo "Set Curve Radius") viene parametrizzato con un nodo di "Float Curve" che prende come valore in ingresso il fattore dello "Spline Parameter"[65].

La crescita dello stelo viene realizzata tramite *keyframe* variando il valore *end* del nodo "Trim Curve". Questo nodo permette di controllare la lunghezza della curva

[64] Questa operazione è necessaria in quanto una texture di rumore contiene valori compresi tra 0 ed 1 e quindi ogni punto avrà un valore medio di 0,5. Di conseguenza, lo stelo partirà spostato su tutti gli assi di un valore di 0,5. Per fissarlo nell'origine occorre sottrarre tale valore su tutti gli assi e moltiplicare il vettore risultante per il fattore del nodo "Spline Parameter" (che restituisce un valore in un intervallo tra 0 e 1 in base alla posizione della curva).

[65] Come spiegato nella nota precedente, il fattore del nodo "Spline Parameter" corrisponde ad un valore compreso tra 0 ed 1 in relazione alla posizione della curva rispetto al suo inizio e fine.

in base al tempo. Sullo stelo viene infine applicato un Material costruito *ad-hoc* che possiede proprietà cromatiche similari a quelle di una pianta.

L'aggiunta delle foglie animate lungo lo stelo è la parte più articolata dell'implementazione tramite Geometry Nodes. Si parte realizzando sullo stesso progetto una mesh generica di una foglia a cui viene applicata una texture specifica al tipo di pianta da riprodurre. Utilizzando il nodo "Instance on Points" è possibile andare ad inserire istanze della foglia nei punti dello stelo impostati tramite il nodo "Resample Curve". Nello stesso nodo di istanza è possibile randomizzare il modo con cui vengono generate le foglie nello stelo e modificarne la dimensione relazionandola alla posizione in cui si trovano rispetto all'estremità. Con il nodo "Align Rotation to Vector" si allinea un'asse del vettore rotazionale delle foglie rispetto ad uno specifico vettore. Nel caso in esame, il vettore scelto è tangente alla curva nel punto in cui le foglie vengono istanziate e l'asse è Y.

Alla fine, per fare in modo che ogni foglia si trovi ruotata di una quantità pseudo random viene utilizzato il nodo "Rotate Instances" in cui è possibile impostare una rotazione nello spazio locale di ciascuna foglia. In natura le foglie crescono impiegando l'angolo aureo, di conseguenza è possibile settare la rotazione sull'asse Y moltiplicando l'indice dell'i-esimo elemento istanziato per il valore di 2,412[66]. Per la rotazione sull'asse X viene impiegato un numero collegato al valore *end*, (rimappato in un range) visto precedentemente, in modo da simulare l'effetto di apertura delle foglie man mano che la pianta cresce. Per l'asse Z viene scelto un valore arbitrario che nel contesto produce l'effetto desiderato migliore.

Come ultima cosa, per conferire del realismo nell'animazione delle singole foglie, tramite il modificatore "Simple Deform" di Blender è possibile applicare ed animare degli angoli di deformazione sugli assi X e Z della foglia per conferirne il tipico effetto di arricciamento. In Figura 4.2 sono raffigurati in sequenza dei render prodotti in varie fasi dell'animazione risultante.

[66] Blender usa come unità di misura i radianti per cui, essendo 1 radiante pari a cica 57°, per ottenere l'angolo aureo di 137,5° occorre moltiplicare l'indice numerico per il valore 137,5°/57° ovvero 2,412.

Figura 4.2. Render di varie fasi dell'animazione della pianta.

4.1.2. Gli alberi procedurali

Un *add-on* particolarmente efficace per realizzare arbusti in maniera semplice e veloce su Blender è AnyTree sviluppato da Märten Rääli[67]. Questo plugin mette a disposizione una vasta gamma di parametri personalizzabili che consentono di generare in maniera procedurale un albero in forma realistica o stilizzata senza la necessità di una modellazione manuale. Il plugin si basa sui Geometry Nodes di Blender, ma grazie alla sua interfaccia *user-friendly* rappresenta un'alternativa per gli artisti che desiderano creare rapidamente alberi ed animarne la crescita, accettando però un compromesso in termini di controllo rispetto all'utilizzo diretto di questi ultimi. A titolo esemplificativo, per introdurre le peculiarità di AnyTree viene di seguito illustrata la creazione di un albero di acacia.

Una volta installato il plugin, utilizzando il comando "Add new Shape Tree" su di una mesh poligonale (nel caso d'esempio, un cubo) verrà generata l'intera struttura gerarchica per realizzare l'albero. Questo sarà contenuto all'interno della mesh poligonale che fungerà proprio da *bounding box* per l'albero. Pertanto, modificando a piacimento le dimensioni della mesh poligonale, verrà automaticamente modificata la struttura dell'arbusto in modo da adattarsi a questa. L'oggetto *Tree System* generato conterrà, nella sezione dei modificatori, i parametri che serviranno nella creazione dell'albero in forma procedurale. In Figura 4.3 viene riportato il

[67] La licenza ad uso personale o commerciale del plugin AnyTree la si può acquistare sul sito web di Blender Market.

pannello con i parametri ed i relativi valori che sono stati utilizzati per la costruzione dell'albero di acacia.

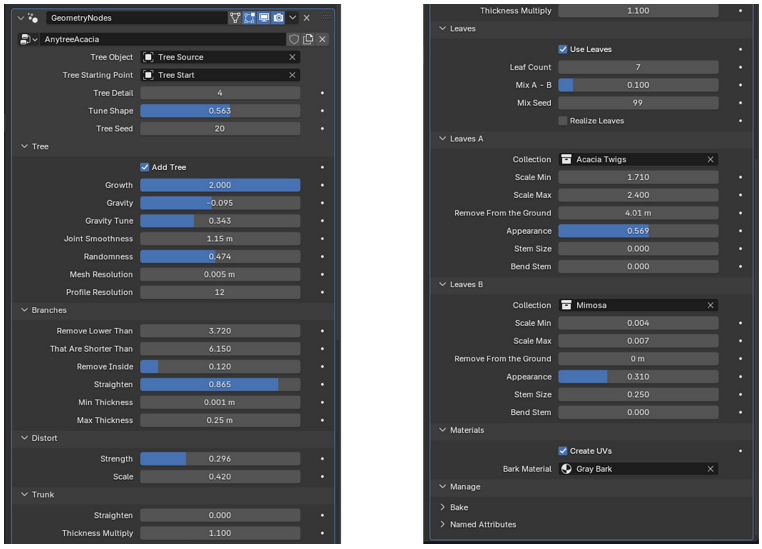

Figura 4.3. Pannello dei parametri a disposizione con AnyTree.

Partendo dai parametri mostrati in Figura 4.3, attraverso l'uso degli strumenti *Tree Detail* e *Tune Shape* è possibile modificare il livello di dettaglio e la quantità di connessioni presenti nell'albero. Nella sezione "Tree" di particolare importanza è il parametro *Growth* che, tramite *keyframe*, permette di realizzare un'animazione di crescita dell'albero. Altri parametri, come *Gravity* e *Joint Smoothness*, consentono di regolare la forza di gravità che agisce sui rami e la levigatezza delle giunzioni. Nella sezione "Branches" è possibile rimuovere i rami che si trovano al di sotto di una certa altezza e che presentano una determinata lunghezza, settare un fattore di ondulazione e regolare lo spessore dei rami stessi. Nelle sezioni "Distort" e "Trunk" è possibile stabilire un fattore di distorsione in grado di rendere la topologia dell'albero più frastagliata e raddrizzare il tronco principale per far apparire l'albero più diritto.

Il fogliame e le infiorescenze possono essere aggiunti importando nelle sezioni "Leaves" delle collezioni di mesh personalizzate e texturizzate. Di default, AnyTree mette già a disposizione delle raccolte di foglie per le famiglie di alberi più comuni. È possibile regolare la quantità delle foglie, applicare un range di scala nonché rimuoverle al di sotto una certa altezza. Inoltre, tramite appositi parametri è

possibile modificarne l'aspetto, la dimensione dello stelo e piegarle in modo che puntino verso il basso.

Nella sezione "Material", infine, è possibile selezionare e definire la texture che meglio si adatta a rappresentare la corteccia dell'albero. In Figura 4.4 è mostrato un esempio di render di un albero di *Vachellia seyal*, conosciuto anche come acacia egiziana, realizzata con il plugin AnyTree.

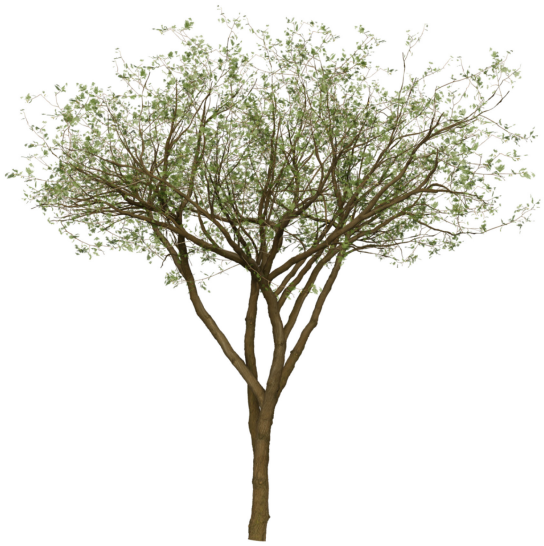

Figura 4.4. Esempio di albero di acacia egiziana realizzato con AnyTree.

4.2. L'aloe vera

Nella cultura e simbologia egizia così come nell'uso comune, l'aloe vera rivestiva un ruolo di particolare rilevanza per via delle sue proprietà terapeutiche e spirituali. Conosciuta anche come "pianta dell'immortalità", l'aloe veniva adoperata nei riti funebri per conservare meglio i corpi grazie le sue proprietà antibatteriche. Associata come simbolo legato alla rinascita, era usanza coltivarla in prossimità delle tombe come segno di protezione e purificazione per i defunti, rafforzando la componente spiritale e religiosa di connessione alle divinità.

Considerata uno dei simboli predinastici già prima dell'avvento dell'età dei faraoni, non poteva di certo mancare all'interno della mostra *Paesaggi – Landscapes*. L'aloe compare, infatti, sia in alcuni *time-lapse* che raffigurano scenari tombali sia

all'interno di animazioni in *split-screen* e nella fase di sintesi dell'esperienza immersiva. I paragrafi seguenti approfondiranno la realizzazione dell'animazione dell'aloe proposta in primo piano all'interno della fase di analisi del progetto a partire dal software utilizzato per la sua realizzazione, Houdini® nella versione 19.5.435.

4.2.1. Houdini

Houdini è un software avanzato per modellazione procedurale, animazione, effetti visivi, simulazione, rendering e compositing. La sua caratteristica principale è il *workflow* procedurale basato su nodi, che permette di realizzare dei contenuti con ampio controllo ed altamente personalizzabili incoraggiando la prototipazione rapida. Poiché Houdini si basa sulla generazione procedurale di elementi, dispone di numerosi strumenti in grado di gestire scene estremamente grandi e complesse.

Ampiamente impiegato per la realizzazione di animazioni ed effetti visivi, grazie ai suoi operatori particolarmente efficaci, il software consente di gestire simulazioni avanzate ed animazioni basate su espressioni matematiche e parametri modificabili in *real-time*. Inoltre, tramite i plug-in offerti da Houdini Engine si ha la possibilità di portare ad un approccio a nodi applicazioni di modellazione 3D come Autodesk Maya o motori grafici come Unreal Engine e Unity, ampliando ulteriormente il suo campo di applicazione e rendendolo uno strumento estremamente potente nel settore multimediale.

L'interfaccia di Houdini, mostrata in Figura 4.5, è progettata per offrire un *workflow* modulare ed è organizzata in diverse aree principali:

- *Main Menus*: consentono un accesso rapido alle funzionalità comuni, come l'importazione di file e la gestione dei rendering.
- *Scene View*: mostra una rappresentazione 3D della scena corrente. È possibile cambiare lo stile di ombreggiatura e aumentare o diminuire la qualità della visualizzazione.
- *Pane Tabs*: contiene delle *shortcut* sui pannelli principali del software.

- *Shelf*: contiene una vasta gamma di strumenti utili per lavorare in modo interattivo nella *Scene View*.
- *Help and Takes*: contiene una scorciatoia per la guida online e la possibilità di creare catene di modifiche sovrapposte.
- *Toolbar*: mostra i controlli e parametri più importanti quando si utilizza uno strumento dello *Shelf* o si seleziona un nodo nel *Network Editor*.
- *Toolbox*: una raccolta di strumenti di uso comune, come la selezione, lo spostamento, la rotazione ed il ridimensionamento, nonché di opzioni come la modalità ispezione e lo *snapping*.
- *Display Options*: permette un accesso rapido alle opzioni di visualizzazione comunemente utilizzate.
- *Parameter Editor*: consente di modificare i parametri del nodo attualmente selezionato per cambiarne il funzionamento.
- *Network Editor*: mostra i nodi della rete corrente. È possibile creare i nodi, selezionarli per vedere i loro parametri e collegarli insieme.
- *Playbar*: contiene i controlli di riproduzione e la *timeline* per scorrere i fotogrammi dell'animazione e mostrare quelli che contengono *keyframe*.
- *Status and Cooking*: visualizza aggiornamenti sullo stato e controlli che permettono di disattivare l'aggiornamento automatico della *Scene View* quando questa è estremamente complessa e lenta da aggiornare.

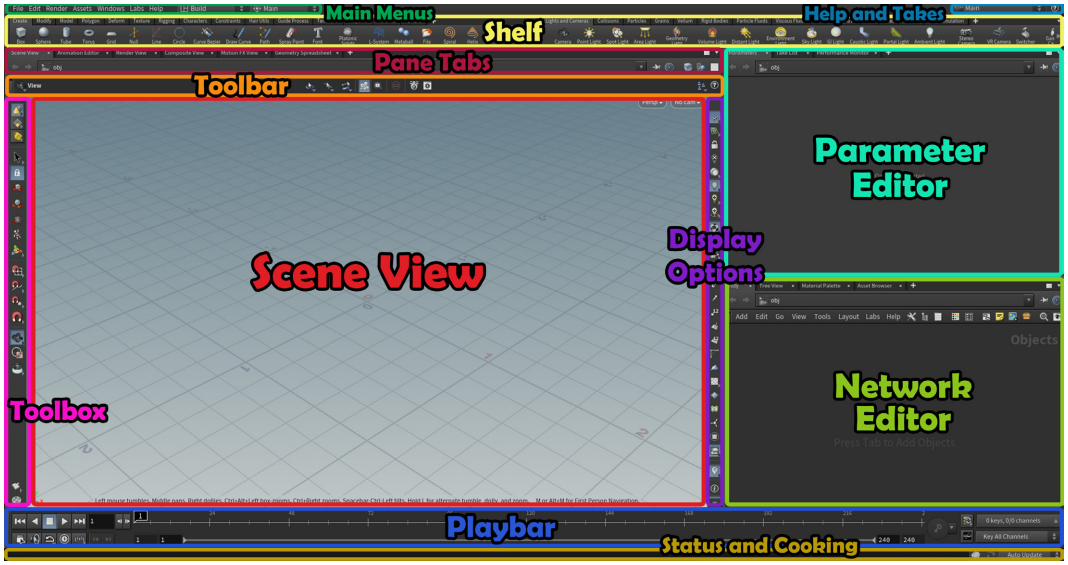

Figura 4.5. User Interface di Houdini.

Houdini permette di lavorare nella vista 3D utilizzando gli strumenti dello *Shelf* senza la necessità di maneggiare i nodi e le reti. Ad ogni modo, poterne comprendere il funzionamento è essenziale per sfruttare al massimo le potenzialità che vengono offerte dal software.

Le reti di Houdini possono essere considerate come il *file system* di un computer, dove le cartelle ed i file corrispondono rispettivamente alle reti e ai nodi. In Houdini vi è un'organizzazione gerarchica e modulare. Infatti, all'interno di una rete è possibile trovare delle sottoreti che contengono ulteriori nodi. La rete *root* di Houdini include alcune reti predefinite come la rete di livello oggetto o "scena" (*/obj*) e la rete di nodi di rendering (*/out*). Ogni rete ha un "tipo" che determina le categorie di nodi che possono essere inserite al suo interno. La rete a livello di oggetto contiene elementi come personaggi, oggetti di scena, luci e telecamere. A questo livello, è possibile includere sottoreti che racchiudono nodi specifici. A titolo di esempio, una sottorete geometrica racchiude nodi che permettono di definire la forma di un oggetto, mentre una sottorete dinamica comprende nodi che specificano il comportamento di una simulazione.

In Houdini, i nodi dispongono di input e output. Si possono creare delle relazioni tra i nodi collegando l'output di uno all'input di un altro. Nella maggior parte dei casi, questa connessione indica il passaggio di dati da un nodo a quello successivo. In altri, come le reti di livello oggetto, i collegamenti tra nodi rappresentano delle relazioni gerarchiche tra oggetti.

Quando si seleziona un nodo, compare una finestra dedicata alla gestione di impostazioni chiamati "parametri". È possibile modificare questi parametri nel riquadro del *Parameter Editor*, permettendo di cambiare la funzione del nodo in base alle esigenze. I principali tipi di nodi sono:

- CHOP (*Channel Operator*): utile per la manipolazione di dati basati sul tempo, come curve di animazione o audio.
- COP (*Compositing Operator*): viene utilizzato per il compositing di immagini ed in generale per la manipolazione di dati basati su pixel.

- DOP (*Dynamic Operator*): impiegato per realizzare simulazioni fisiche realistiche e complesse.
- ROP (*Render Operator*): responsabile della gestione del rendering e l'esportazione delle scene.
- SOP (*Surface Operator*): adoperato per la modellazione di geometrie 3D e nella generazione di ambienti e oggetti procedurali.
- VOP (*Vector Operator*): permette di realizzare shader, effetti avanzati con operazioni matematiche complesse e manipolazioni geometriche impiegando una logica a nodi basata sul linguaggio di scripting VEX.

Houdini supporta una vasta gamma di geometrie, come poligoni, curve di Bézier e NURBS, primitive geometriche come cerchi, sfere e *metaball*. Il software utilizza il termine "primitiva" per riferirsi ad un qualsiasi frammento immutabile di una geometria. Tuttavia, alcune primitive sono costituite da componenti modificabili; ad esempio, una faccia di un poligono è formata da punti e vertici, i quali possono essere modificati singolarmente.

Gran parte delle informazioni che compongono una scena sono memorizzate sotto forma di "attributi", ossia dati nascosti associati a modelli, primitive, punti e vertici. Un esempio è dato dall'attributo di posizione *P* che determina la collocazione nello spazio di un punto della scena. Houdini fornisce la possibilità di utilizzare il *Geometry Spreadsheet*, uno strumento che permette di visualizzare i valori degli attributi geometrici dei punti presenti in scena.

Houdini rappresenta uno strumento che fa della potenza nelle simulazioni e della versatilità il suo punto di forza. Si presta particolarmente bene ad essere impiegato in settori che spaziano da dagli studi di VFX alla realtà virtuale. Tra le tante cose, è da sottolineare il suo importante ruolo nell'industria dell'intrattenimento dove viene tutt'oggi considerato il software di punta per progetti di grande impatto visivo e ad alto budget. La stragrande maggioranza delle aziende cinematografiche lo utilizza per la produzione di effetti visivi in quei prodotti che fanno dell'accuratezza e del realismo delle scene una loro prerogativa. Infine, la sua integrazione con altri software, come Unreal Engine, e la compatibilità con vari formati di file gli permette di essere utilizzato in vari ambiti della pipeline produttiva.

4.2.2. L'animazione dell'aloe

L'animazione procedurale dell'aloe può essere introdotta partendo dalla tecnica adottata per la riproduzione della sua geometria all'interno di Houdini. Come illustrato nella Figura 4.6, il processo si basa sulla generazione di linee disposte in una configurazione a spirale mediante una rotazione. Ciascuna linea è composta da un numero definito di punti distanti tra loro con un determinato offset, ai quali vengono ancorate le foglie della pianta. L'animazione viene realizzata guidando tali punti lungo i percorsi delineati dalle curve. Di seguito verranno descritti i nodi utilizzati per la costruzione della geometria e animazione della pianta.

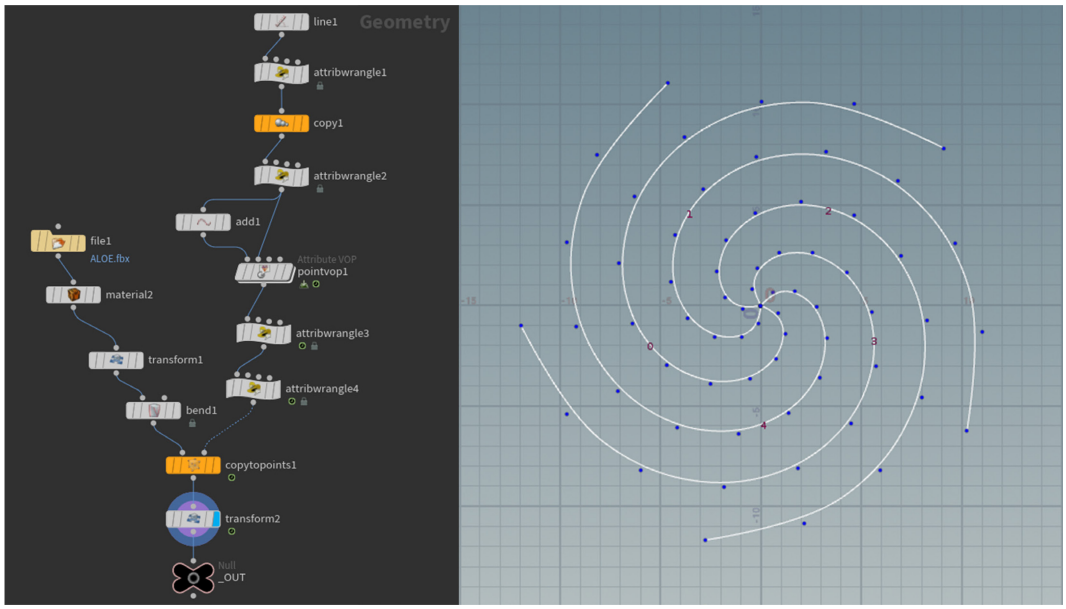

Figura 4.6. Albero dei nodi e geometria iniziale per la creazione dell'aloe.

Si parte con la creazione di una linea lungo l'asse X, definita come una primitiva di tipo NURBS, alla quale vengono assegnate una determinata lunghezza e un numero iniziale di punti. Successivamente, mediante un nodo *Attribute Wrangle*, viene utilizzato lo scripting VEX per definire l'attributo di offset dei punti, calcolato come il rapporto tra l'indice del punto corrente e il numero totale di punti della linea. Quindi, la primitiva viene duplicata cinque volte e le copie vengono ruotate in modo da formare una struttura radiale. Attraverso un ulteriore nodo *Attribute Wrangle*, viene implementata una funzione che consente di ruotare i punti in modo da generare una configurazione a spirale, come mostrato in Figura 4.6.

Il passo successivo consiste nella creazione di un nodo Point di tipo *Attribute VOP* il quale consente di guidare i punti lungo il percorso definito dalle primitive. Come illustrato in Figura 4.7, all'interno di questo nodo il movimento dei punti viene ottenuto attraverso l'applicazione dell'operatore modulo alla somma tra il valore "primuv", che rappresenta la posizione del punto lungo il percorso, e la variabile temporale opportunamente scalata[68]. All'interno del nodo viene definito il valore dinamico "u" corrispondente alla posizione del punto lungo il percorso aggiornata a ogni fotogramma.

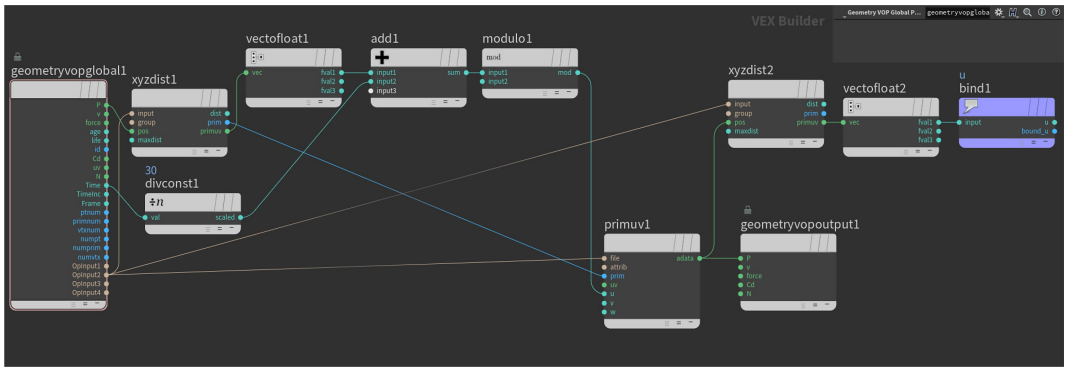

Figura 4.7. Contenuto dell'operatore vettoriale Point.

Successivamente, viene impiegato un nodo di tipo *Attribute Wrangle* per implementare uno script in grado di orientare i versori dei punti nella direzione desiderata. Questo passaggio risulta fondamentale per assicurare il corretto allineamento delle foglie una volta istanziate sulla geometria. L'ancoraggio delle foglie ai punti avviene attraverso un nodo di tipo *Copy to Points*, previa applicazione di opportune trasformazioni geometriche e deformazioni. Per incrementare il realismo, sia il modello che la texture delle foglie sono stati ottenuti da una fotogrammetria di una pianta di aloe vera. Inoltre, un ulteriore intervento volto a migliorare l'accuratezza della rappresentazione della pianta consiste nell'impostare, sempre all'interno di un nodo di tipo *Attribute Wrangle*, l'attributo di scala delle foglie e il loro indice di flessione lungo l'asse Z, correlando tali

[68] L'operazione di *scaling* risulta necessaria in quanto, se si utilizzasse un valore temporale standard, la velocità di movimento dei punti sarebbe talmente eccessiva da generare l'illusione di un movimento dall'esterno verso l'interno anziché il contrario.

parametri al valore dinamico "u" precedentemente calcolato. Il risultato sarà un modello animato di aloe parametrizzato in base a dei valori matematici.

4.2.3. Rendering ed esportazione

Una volta completata la modellazione procedurale dell'aloe vera, si procede con l'allestimento della scena 3D, inserendo una camera e opportune fonti luminose per valorizzare al meglio la pianta.

Per quanto concerne l'illuminazione, nel progetto vengono utilizzate tre luci di tipologia ad area, ciascuna caratterizzata da una diversa intensità ed una sfumatura cromatica tendente al verde. In questo modo si ottiene un buon livello di profondità e realismo della composizione. L'inquadratura finale utilizza una camera con lunghezza focale di 18 mm, un'apertura pari a 8,3 mm ed una risoluzione di 3510×1080 pixel.

Houdini integra nativamente il motore di rendering Mantra e implementa Karma, un renderer basato su Hydra e USD, ottimizzato per una maggiore efficienza. Inoltre, il software consente, previa licenza, l'utilizzo di motori di rendering di terze parti, come RenderMan, Redshift, Arnold e V-Ray, permettendo di ottenere una grande flessibilità nella produzione dei contenuti. Per il presente progetto è stato scelto Karma, in quanto offre un sistema di *path-tracing* avanzato in grado di generare risultati fotorealistici con un'efficienza superiore rispetto al nativo Mantra.

Karma supporta sia il rendering su CPU che un approccio ibrido XPU, che sfrutta anche le capacità di calcolo della GPU attraverso i CUDA core, garantendo una maggiore velocità di elaborazione. Inoltre, il motore di rendering include strumenti per l'ottimizzazione dell'immagine, tra cui *denoiser* avanzati come OptiX di NVIDIA e Intel Open Image Denoise, che consentono una significativa riduzione del rumore senza compromettere i dettagli. Tra le altre cose, permette di esportare passi di rendering multipli per il compositing.

Per il rendering dell'aloe vera è stata scelta la modalità XPU, sfruttando la GPU per accelerare il calcolo, e il *denoiser* OptiX per migliorare la qualità dell'immagine. Il

numero di *samples*[69] è stato impostato a 64, un valore che garantisce un buon equilibrio tra qualità e tempi di rendering. Per l'esportazione del risultato finale è stato adottato il formato EXR a 32 bit, particolarmente indicato per successive operazioni di post-produzione e compositing. In Figura 4.8 è riportato un fotogramma del rendering finale dell'aloe vera.

Figura 4.8. Render finale dell'aloe vera.

Una volta completate le operazioni di rendering di tutti i fotogrammi dell'animazione dell'aloe, che nel caso del progetto in esame ammontano a 300, il passo finale riguardava la fase di *color correction*[70], in modo da enfatizzare maggiormente la palette cromatica della pianta, ed esportare l'animazione come un unico file video. Per l'esecuzione di queste operazioni, è stato impiegato il software DaVinci Resolve. Nella sezione "Color" di questo strumento si ha la possibilità di operare nel dettaglio sulla regolazione di specifici parametri e curve colore che

[69] Nel rendering in *path-tracing*, il numero di *samples* indica quante volte un pixel viene calcolato. Un valore alto permette di migliorare la precisione di calcolo della luce ed in generale la qualità dell'immagine finale. Impostando un valore di *samples* di 64, il motore di rendering eseguirà 64 raggi per direzione per ogni pixel dell'immagine.
[70] La *color correction* è un aspetto fondamentale nella post-produzione di immagini e video, mirato a ottimizzare i colori e a migliorare l'aspetto visivo di un progetto regolando curve e parametri.

permettono di ottenere i risultati di *color grading* desiderati. I valori specifici di correzione colore usati nel progetto sono illustrati in Figura 4.9.

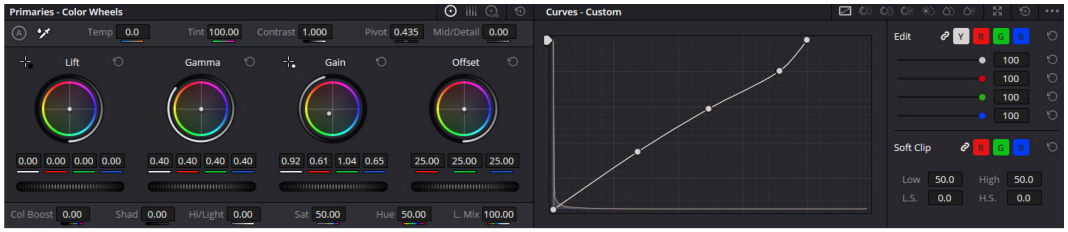

Figura 4.9. Parametri di correzione colore su DaVinci Resolve.

Per l'esportazione finale è stata scelta una codifica DNxHR 422 a 12 bit in modo da garantire un'elevata qualità video per via dei suoi algoritmi di compressione *lossy* ed una gamma dinamica estesa.

5. IL SISTEMA PARTICELLARE

In quest'ultimo capitolo verranno discusse le implementazioni riguardanti il flusso particellare finale del video immersivo. In maniera simbolica ciò che si voleva realizzare era un "flusso di conoscenza" che, partendo da un punto centrale, si diramasse in ogni direzione fino a ricoprire le pareti. In *Paesaggi – Landscapes* è stato utilizzato il *Particle System* di TouchDesigner, nella versione 2023.11600, adattato appositamente per la mostra.

La scelta di usare un software diverso rispetto a ciò che già offre la suite di Unreal Engine è dovuta a una motivazione ben precisa. Il Niagara System, pur essendo uno strumento avanzato, non è pensato per lavorare con un numero molto elevato di particelle in *real-time* risultando pertanto non idoneo alla realizzazione di un flusso particellare estremamente denso. TouchDesigner, al contrario, si dimostra una soluzione più performante per questa tipologia di elaborazione. Inoltre, il suo approccio basato su nodi consente un alto grado di personalizzazione, permettendo di modificare parametri e connessioni in modo dinamico senza la necessità di ricostruire l'intero sistema da zero.

Nei successivi paragrafi, verrà fornita inizialmente una panoramica su TouchDesigner. In seguito, si descriveranno i passaggi necessari alla realizzazione del sistema di particelle utilizzato per *Paesaggi – Landscapes*.

5.1. TouchDesigner

TouchDesigner® è una piattaforma di sviluppo visuale utilizzata per la realizzazione di contenuti multimediali interattivi in tempo reale come installazioni immersive e live performance. Il sistema è caratterizzato da una struttura a nodi simile ad Houdini. Ciò permette di manipolare dati in maniera flessibile combinando tra loro blocchi chiamati operatori. Questi ultimi vengono collegati tra loro per creare effetti procedurali e animazioni.

Ogni operatore è dotato di un set univoco di parametri e flag che ne controllano il funzionamento. Tra i principali operatori si annovera:

- COMP (*Component Operator*): può ospitare intere reti di altri operatori e si distingue in quattro categorie (3D Objects, Panel, Other e Dynamics).
- TOP (*Texture Operator*): fondamentale nei progetti multimediali, serve per manipolare immagini e texture 2D.
- CHOP (*Channel Operator*): viene utilizzato per gestire dati di movimento, segnali audio, controlli su schermo o altri valori di input.
- MAT (*Material Operator*): serve per creare *shader* per le superfici 3D.
- SOP (*Surface Operator*): è il responsabile delle operazioni e modellazioni di geometrie 3D.
- DAT (*Data Operator*): viene usato per contenere dati testuali (XML, JSON), tabelle e script.

L'operazione più rilevante in TouchDesigner consiste nel collegamento degli operatori. Quando più operatori vengono combinati all'interno di una rete, essi possono risolvere problemi di complessità sempre più elevata. In TouchDesigner tutti i dati scorrono da sinistra verso destra: gli input sono sempre posizionati sul lato sinistro, mentre gli output si trovano sul lato destro. Inoltre, tutti gli ingressi e le uscite sono ordinati dalla prima all'ultima partendo dall'alto fino al basso.

Una caratteristica di TouchDesigner è la possibilità di visualizzare in background le modifiche che vengono apportate in *real-time* ai singoli nodi del progetto. Inoltre, integra al suo interno un linguaggio di scripting basato su Python che consente di estendere le funzionalità dei nodi. Ciò risulta utile nei casi in cui per realizzare alcuni processi complessi occorrono delle personalizzazioni che ne rendono più efficiente l'uso e che non trovano riscontro nelle caratteristiche standard dei nodi. Supporta anche vari protocolli come MIDI, OSC, DMX per la comunicazione con altri software o hardware esterni. Molto utilizzati nelle esperienze immersive sono i sensori di movimento, come Kinect, e la rilevazione dell'audio ambientale per realizzare effetti in *real-time* che permettono un forte coinvolgimento del pubblico all'interno dell'installazione.

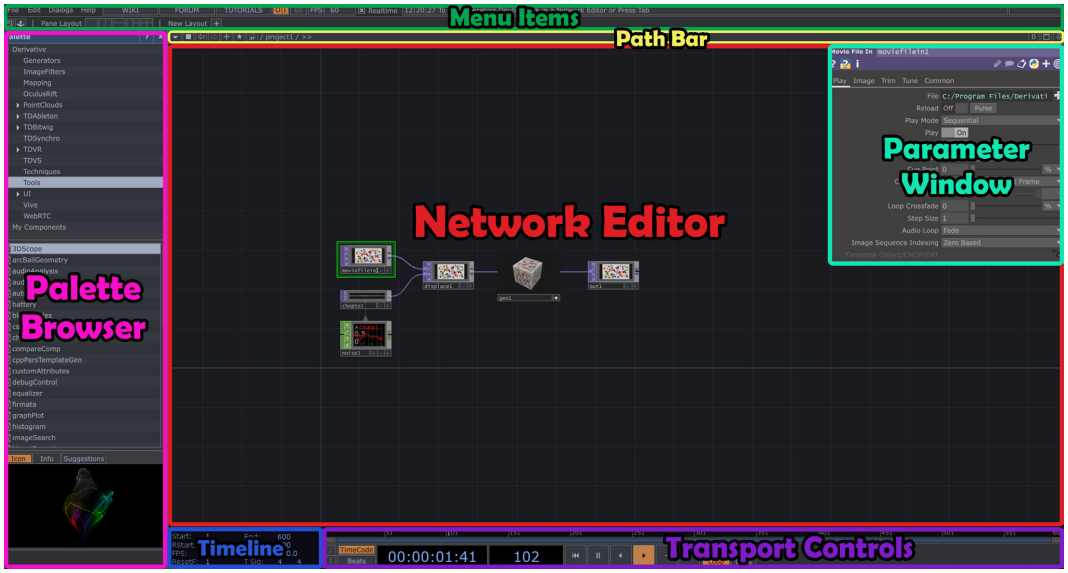

Figura 5.1. User Interface di TouchDesigner.

L'interfaccia utente di TouchDesigner la si può suddividere in varie sezioni. Come è possibile notare dalla Figura 5.1, si distinguono:

- *Network Editor*: è l'area centrale di lavoro in cui si creano, organizzano e connettono gli operatori che plasmano la struttura logica del progetto. È organizzato secondo una gerarchia a livelli in cui ogni rete di nodi può contenere a sua volta delle sottoreti.
- *Timeline*: contiene gli strumenti utili a sincronizzare eventi, animazioni o fare modifiche temporali. Tramite essa è possibile controllare il tempo del progetto e la frequenza a cui il progetto renderizza i fotogrammi (*fps*).
- *Transport Controls*: permettono il controllo della timeline offrendo funzionalità come play, pausa, ripartire dal primo frame, far partire in *reverse* la timeline, impostare il loop dell'animazione.
- *Parameter Window*: consente di modificare le impostazioni del nodo selezionato ed osservarne le proprietà.
- *Palette Browser*: è a tutti gli effetti una libreria facilmente accessibile di componenti (file TOX, *TouchDesigner Component*) ognuno dei quali comprende un singolo operatore COMP che può a sua volta contenere una rete di altri operatori.
- *Menu Items*: fornisce accesso alle funzionalità generali del progetto, a finestre di dialogo utili e alla documentazione ufficiale. Ne fa parte il

contrassegno *realtime* che cambia il comportamento di TouchDesigner in modo significativo. Quando è attivo viene data sempre priorità al mondo reale facendo in modo che l'animazione venga riprodotta rispettando la sua durata effettiva. Ciò potrebbe comportare eventuali perdite di fotogrammi in caso di animazioni onerose da processare in tempo reale. Quando il contrassegno è spento il software impiegherà tutto il tempo necessario a processare e renderizzare ogni frame.

- *Path Bar*: permette di navigare all'interno delle sottoreti del progetto.

Una delle caratteristiche di TouchDesigner è quella di sfruttare intensamente le risorse del computer sia in termini di CPU che di GPU. Il software, infatti, per gestire i suoi compiti richiede molto spesso specifiche di medio-alto livello. Di conseguenza, per evitare di incorrere in possibili colli di bottiglia mentre si lavora in un progetto, risulta essenziale avere la possibilità di monitorare il carico di lavoro che il programma richiede, identificando i punti di rallentamento e ottimizzando le prestazioni. Una delle principali criticità che si può verificare all'interno del software riguarda il *pixel shading*, un processo che coinvolge la GPU e che risulta oneroso quando si lavora con i TOP: il carico computazionale della GPU cresce proporzionalmente all'aumento della risoluzione delle texture. La maggior parte degli operatori, tuttavia, dipende dalla CPU, il che può generare rallentamenti significativi qualora il numero di operatori attivi per ciascun frame risulti eccessivo. Le congestioni a livello di CPU sono generalmente più semplici da individuare, poiché gli operatori che richiedono un tempo di elaborazione elevato sono evidenziati all'interno del *Performance Monitor*, uno strumento utile ad analizzare il tempo di esecuzione di un frame.

TouchDesigner è un software potente, parecchio versatile per gli artisti del settore. Infatti, si presta particolarmente bene in settori che spaziano dall'arte interattiva a performance live *audio-reactive* e visualizzazioni scientifiche. La sua capacità di integrare vari tipi di media, sensori e input, permette di creare esperienze uniche all'insegna dell'interattività. Questa caratteristica lo rende apprezzato nella realizzazione di grandi eventi che fanno dell'effetto scenico di coinvolgimento del pubblico una loro prerogativa.

5.2. Il flusso di conoscenza

Alla fine dell'esperienza immersiva si ha una fase di sintesi che si concentra sulle nozioni acquisite. La conoscenza ultima, inarrivabile, è rappresentata tramite un flusso di particelle che riportano l'osservatore al punto di partenza (l'inizio del video). Tale flusso, metaforicamente, offre una riflessione sulla complessità e sull'inaccessibilità della conoscenza stessa e si dipana da un'origine comune verso tutte le direzioni. Tramite poi degli effetti di *blob tracking*[71], all'interno dei flussi sono state inserite delle parole evocanti l'antico Egitto quasi a simboleggiare i nodi della conoscenza che si espande.

Per la realizzazione del filmato sul flusso di conoscenza sono stati fatti numerosi tentativi che hanno portato a varie versioni del flusso. A titolo esemplificativo, nella Figura 5.2 sono riportati due dei pattern che sono stati considerati inizialmente ma successivamente scartati. A sinistra è rappresentata una struttura generativa che ha un movimento ad onda mentre a destra ha un motivo riconducibile alla formazione del micelio.

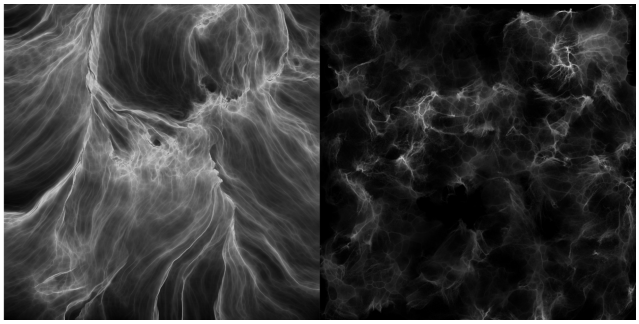

Figura 5.2. Esempi iniziali di pattern per il flusso particellare.

Si descriveranno a seguire i passaggi cruciali che hanno portato alla realizzazione del flusso finale scelto per l'installazione immersiva dividendo il processo di produzione in sei macro-blocchi.

---

[71] Il *blob tracking* è una tecnica che ha come obiettivo quello di rilevare punti o regioni (chiamati *blob*) in un'immagine che differiscono per proprietà come la luminosità o il colore rispetto all'ambiente circostante.

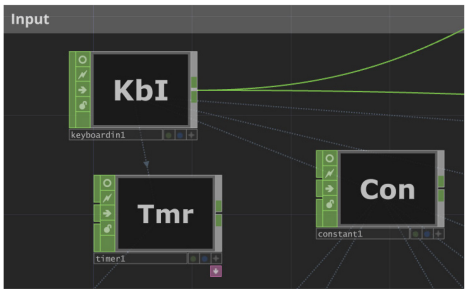

Figura 5.3. Flusso di particelle: blocco Input.

Nel blocco di Input si hanno tre nodi di tipo CHOP. Il primo, *Keyboard In*, reagisce alla pressione di un tasto (nel caso in esame, "1") e la sua funzionalità è quella di far ripartire dall'inizio l'animazione delle particelle associandolo al valore "start" del nodo *Timer* e ai valori di tipo "pulse" dei nodi che verranno visti in seguito. Infine, il nodo *Constant* viene utilizzato come una variabile globale per indicare una frazione del numero di particelle presenti in scena.

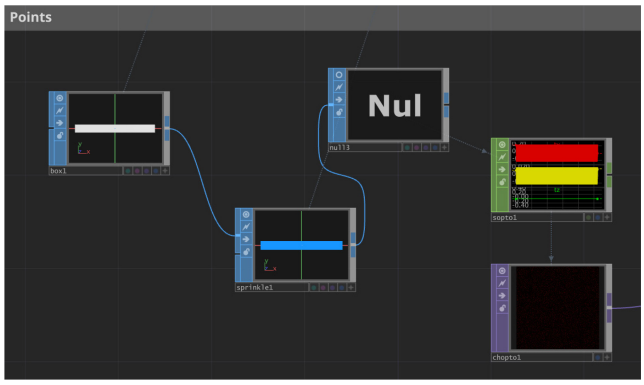

Figura 5.4. Flusso di particelle: blocco Points.

Nel blocco Points si hanno dei nodi SOP e delle conversioni tra operatori (*SOP to CHOP* e *CHOP to TOP*). Il nodo *Box* serve a creare il piano di partenza che verrà successivamente convertito in particelle. Questo è implicitamente collegato al *Timer* visto precedentemente nella sua dimensione sull'asse y in modo che all'aumentare del valore temporale, la dimensione del piano sull'asse orizzontale aumenti. Il nodo *Sprinkle*, invece, aggiunge le particelle dentro al piano precedente (prendendo come riferimento il valore *Constant* del blocco analizzato prima). Tramite un nodo di passaggio, *Null*, il nodo precedente, che contiene delle informazioni geometriche, viene convertito fino ad arrivare ad un'informazione di tipo texture.

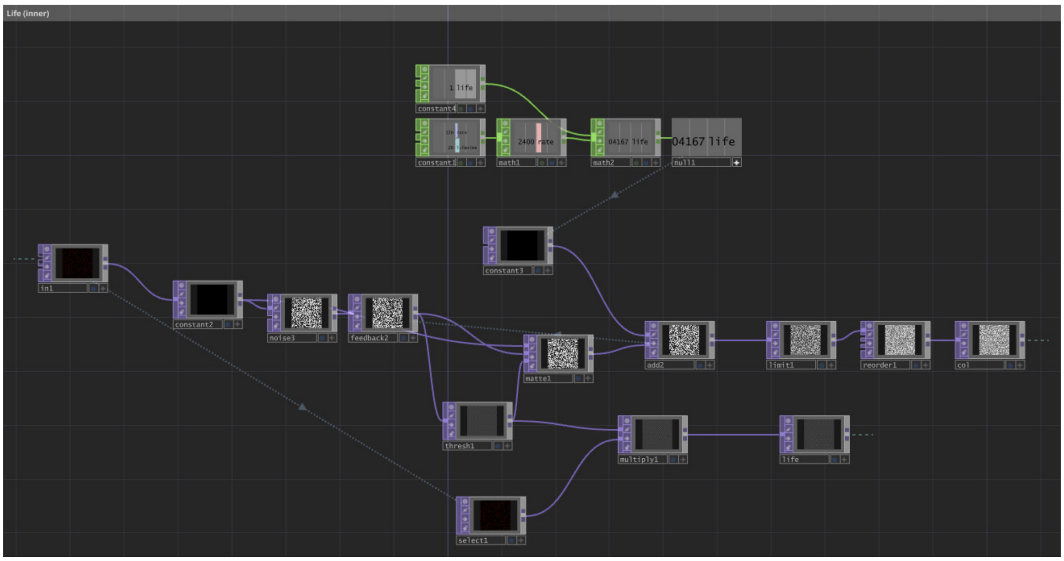

Figura 5.5. Flusso di particelle: blocco Life.

Il blocco Life non è altro che la logica per gestire la vita delle particelle che sta all'interno del nodo omonimo costruito *ad-hoc* da un'idea di Simon Rydén e pubblicata nel suo profilo Patreon. Esso prende come input l'uscita dal precedente nodo *CHOP to TOP* e tramite una serie di operazioni che riguardano *noise*, controlli di *feedback* e soglie, produce due output: "col" che verrà utilizzato per generare un effetto di *fade* delle particelle e "life" che ne gestirà la vita e la posizione casuale.

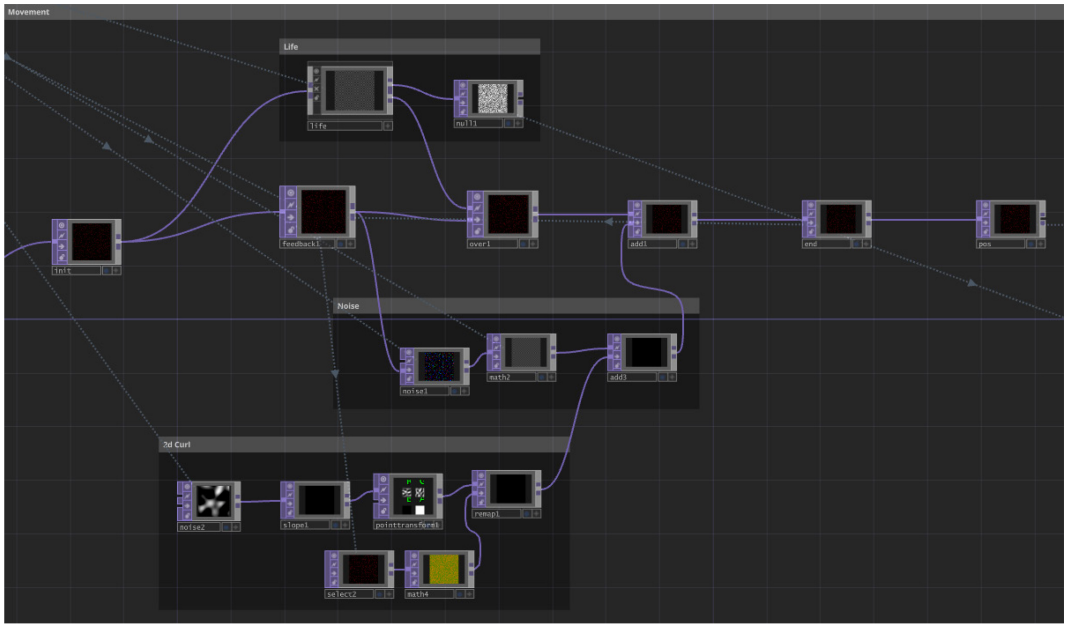

Figura 5.6. Flusso di particelle: blocco Movement.

Il blocco Movement contiene tutto ciò che concerne la logica di movimento delle particelle nella scena. Il nodo di input *Init* non è altro che l'output del precedente nodo *CHOP to TOP* e rappresenta l'insieme delle possibili posizioni delle particelle. Tale nodo trasferisce queste informazioni ai nodi *Life* e *Feedback*. *Life*, analizzato precedentemente, si occupa di indurre un moto alle particelle tramite delle soglie casuali sulle possibili posizioni. Il nodo *Feedback* funge da classico meccanismo di retroazione: recupera le informazioni dei fotogrammi passati (trasmesse dal nodo *End*) e li concatena a quelli presenti per generare degli effetti visivi (come, ad esempio, del *motion blur*). Questi due nodi (*Life* e *Feedback*) confluiscono in *Over* che si occupa di sovrapporre le due informazioni usando il canale alpha dell'output di *Life*. Il sotto-blocco 2D Curl si occupa di creare degli effetti di arricciamento del flusso particellare mentre il sotto-blocco Noise si occupa di definire la direzione (data da una mappa casuale definita nel nodo *Noise*) che devono seguire particelle. Entrambi questi sotto-blocchi vengono sommati e il risultato viene poi composto con *Over*. Infine, i nodi *End* e *Pos* non sono altro che delle riproposizioni dell'uscita *Add*. A questo punto si hanno tutte le informazioni sulla posizione delle particelle.

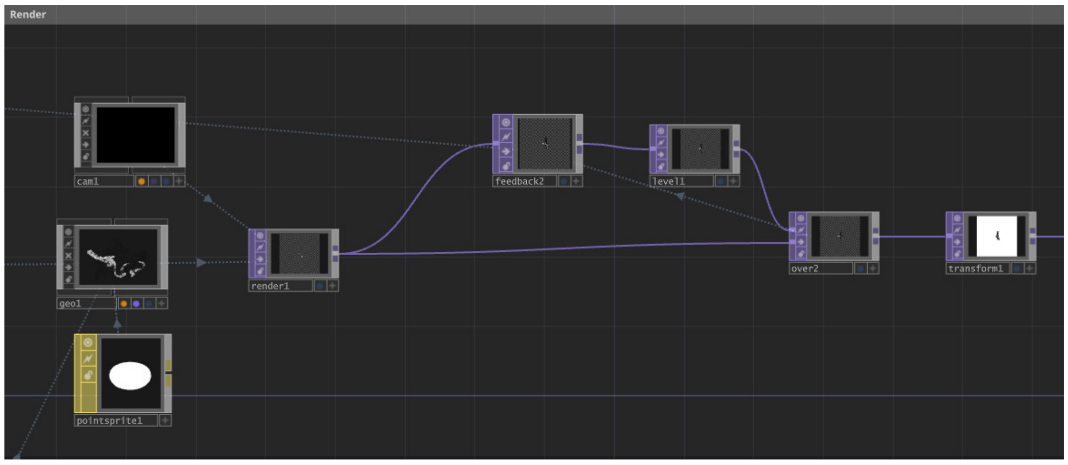

Figura 5.7. Flusso di particelle: blocco Render.

Nel blocco Render viene originato il flusso renderizzato. Il nodo *Geometry* unisce le informazioni precedentemente calcolate e inizializza le particelle. Nella sua proprietà di "instance" il parametro di traslazione viene preso dal nodo *Pos* ed il mapping sui tre assi X, Y, Z viene fatto prendendone le componenti r, g, b. Per quanto concerne il colore delle particelle, questo viene direttamente preso

dall'informazione in uscita del nodo *Life* riguardante il *fading* con una corrispondenza diretta dei valori r, g, b e alpha. Per il materiale, viene utilizzato un *Point Sprite* bianco collegato direttamente alla proprietà "render" di *Geometry*. A questo punto, viene scelta una *Camera* nella posizione desiderata che inquadra in maniera ottimale il flusso e viene utilizzato il nodo *Render* per renderizzare l'animazione.

TouchDesigner rende al meglio quando si lavora con una risoluzione quadrata 1:1, in quanto le particelle non vengono deformate sull'asse X o Y. Data la complessità del progetto e il grande numero di particelle che lo compongono, è stata scelta la massima risoluzione che potesse permettere di avere quanti più dettagli possibili senza mandare in crash il software: con una scheda grafica dedicata da 8 GB di memoria tale limite è 6750×6750. Successivamente, all'animazione prodotta, viene fatta un'operazione di *upscaling* per coprire due pareti della sala senza avere perdita di qualità (le altre pareti contengono l'animazione specchiata).

Al flusso renderizzato, senza sovraccaricare ulteriormente la memoria del sistema, è stato applicato un ulteriore *Feedback* per rendere ancora più marcata la scia di particelle presente in scena. Questo nodo prende le informazioni passate da *Over*. Un nodo *Level* è stato poi aggiunto per regolare i valori di gamma, luminosità e opacità dell'animazione prodotta.

Figura 5.8. Flusso di particelle: blocco Export.

Il blocco finale, Export, viene utilizzato per esportare il video dell'animazione, fare le ultime correzioni sulla posizione e adattare la risoluzione al formato richiesto. Tramite un nodo *Transform* l'immagine viene ruotata di -90° e con un nodo *Fit*

viene adattato alla risoluzione 6750×540. Tale discrepanza tra la componente verticale e quella orizzontale è dovuta al fatto che occorre coprire più pareti lunghe usando lo stesso video. Alla fine, il nodo *Movie File Out* esporta l'animazione di una durata complessiva di 20 secondi. È stata scelta una codifica video Apple ProRes 442 HQ 10 Bit a 120 fps (i frame al secondo sono elevati per accelerare eventualmente la velocità del video senza perdere informazioni). In Figura 5.9, alcuni fotogrammi dell'animazione prodotta.

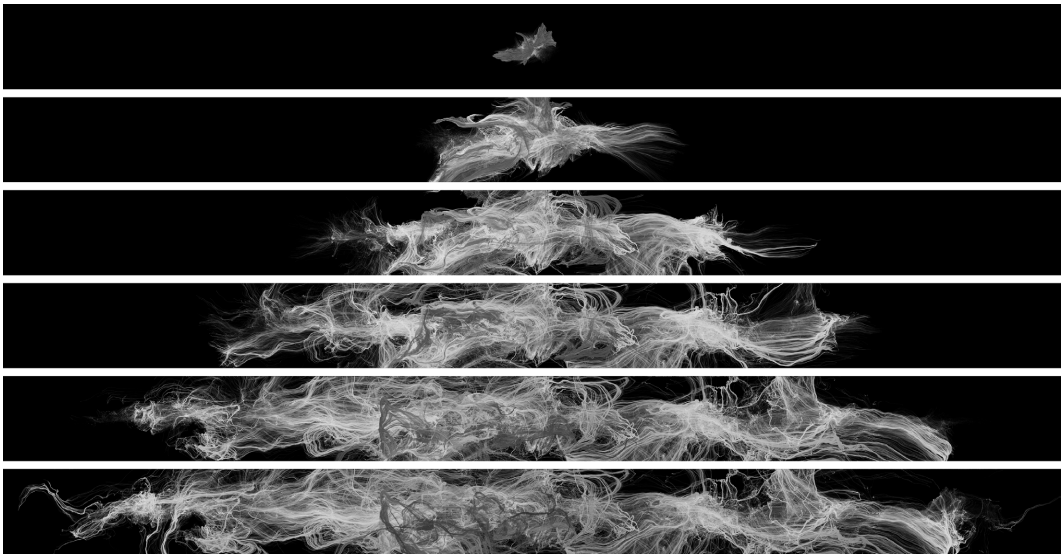

Figura 5.9. Fotogrammi dell'animazione del flusso di conoscenza.

5.2.1. L'upscaling tramite AI

Il video esportato da TouchDesigner è stato trasferito in un software di *upscaling*[72] in modo tale da raddoppiare la sua risoluzione senza perdere in qualità tramite meccanismi di intelligenza artificiale. In questo modo si ottiene un video di output con una risoluzione ottimale per coprire due pareti della sala immersiva, ovvero 13500×1080.

Uno dei software più popolari per *upscaling* video con IA è Topaz Video AI®. Esso utilizza delle reti neurali per migliorare la qualità del video usando tecniche di

[72] L'*upscaling* tramite intelligenza artificiale (IA) è un procedimento che ha come scopo quello di aumentare la risoluzione e la qualità visiva di un video o di un'immagine, preservando o migliorando i dettagli.

upscaling per recuperare dettagli, informazioni e nitidezza senza al contempo introdurre evidenti artefatti visivi. Ciò avviene tramite un approccio basato sulle cosiddette CNN ovvero reti neurali convenzionali[73] e sul *machine learning*. I modelli utilizzati da Topaz Video AI sono addestrati su grandi set di dati (milioni di video e immagini) a varie risoluzioni. Questo consente loro di imparare a identificare i dettagli di un video e a ricostruirli in maniera quanto più realistica possibile. I modelli più comunemente usati sono:

- *Artemis*: è un modello di miglioramento che offre un buon equilibrio tra dettagli, riduzione del rumore e riduzione degli artefatti. Ha delle varianti specifiche in base alla qualità e alla compressione del file in input.
- *Gaia*: esistono due sue varianti, una ottimizzata per rifinire input con una già elevata fedeltà ed un'altra invece pensata per video in computer grafica che gestisce bene i bordi e le linee. Per il progetto è stato scelto questo modello.
- *Theia*: è un modello che consente di controllare in maniera manuale il livello di nitidezza e i dettagli del filmato tramite dei parametri.
- *Proteus*: il modello eccelle nella riduzione del rumore e con filmati di qualità bassa o media e funziona bene in molte situazioni.
- *Nyx*: è un cosiddetto modello di *denoise* ovvero ottimizzato nella riduzione del rumore specialmente nei filmati ad alta risoluzione in condizioni di scarsa illuminazione.
- *Iris*: è un modello generale per la riduzione del rumore e degli artefatti di compressione anche se molto spesso viene utilizzato per il miglioramento del viso e delle caratteristiche facciali in riprese di bassa o media risoluzione.

Le applicazioni di Topaz Video AI variano dal restauro di vecchi filmati catturati a bassa risoluzione ai video professionali e cinematografici per ovviare a problematiche di contenuti provenienti da fonti compresse. Il suo grande vantaggio

[73] Una CNN è un'architettura di rete per il *deep learning* che apprende direttamente dai dati. È utile per trovare dei pattern nelle immagini in modo da riconoscere oggetti, classi e categorie.

consiste nell'aumentare in maniera significativa la qualità finale senza la necessità di avere a disposizione un hardware specializzato recuperando quei dettagli che verrebbero altrimenti persi con tecniche tradizionali.

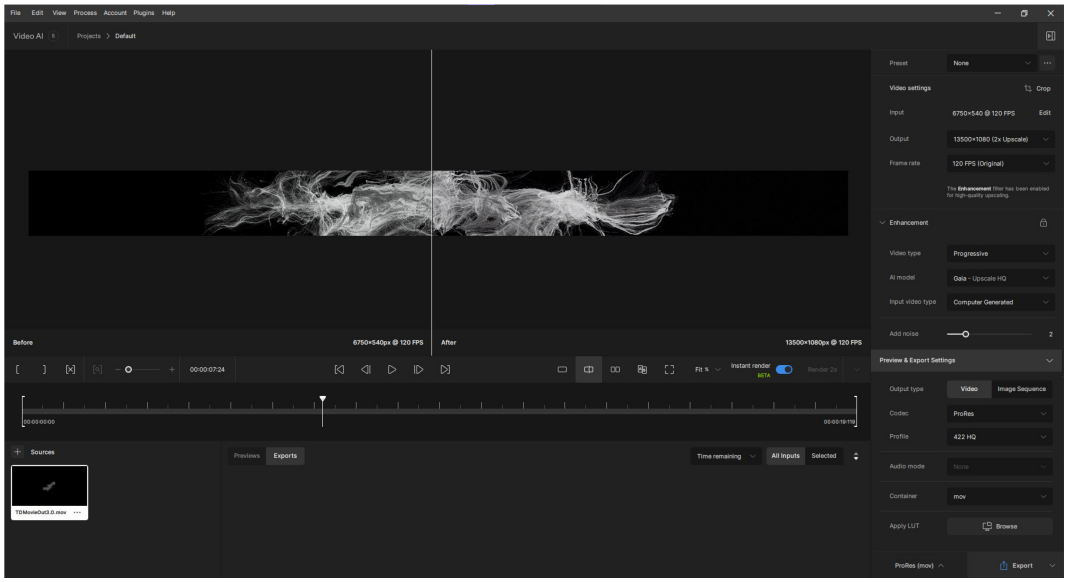

Figura 5.10. Schermata di Topaz Video AI sul flusso di particelle.

Per il video del flusso di particelle finale, come è possibile notare dalla Figura 5.10 è stato scelto un preset di default che consente di fare un *upscale* di tipo 2x sul filmato in input. Il frame rate è stato mantenuto a 120 fps ed il tipo di scansione è stata mantenuta progressiva[74]. Come modello di AI è stato scelto Gaia con input di tipo "Computer Generated" ed è stato aggiunto un livello di rumore utile per ammorbidire i lineamenti delle texture. In output si è scelta una codifica video Apple ProRes 442 HQ in modo da ottenere una bassa compressione e mantenere quanti più dettagli possibili.

[74] La scansione progressiva è un metodo di visualizzazione delle immagini in cui tutte le linee di ciascun fotogramma vengono mostrate in sequenza. Questo processo differisce dalla scansione interlacciata, in cui l'immagine è suddivisa in due semiquadri: uno contenente le linee pari e l'altro quelle dispari.

CONCLUSIONI

Il progetto che ha visto la realizzazione dell'installazione immersiva *Paesaggi – Landscapes* si è ufficialmente concluso nella prima settimana di giugno del 2024, raggiungendo pienamente tutti gli obiettivi prefissati. La mostra, fruibile gratuitamente presso le Gallerie d'Italia, ha offerto un'anteprima di *Egitto Immersivo*, un progetto di ampia scala che sarà ospitato al Museo Egizio di Torino, all'interno di uno spazio di mille metri quadrati realizzato in collaborazione con l'Istituto Italiano di Tecnologia.

Le ricostruzioni fotogrammetriche, le sequenze digitali realistiche, le animazioni sulla flora e gli effetti particellari hanno rappresentato un'opportunità per approfondire le conoscenze sulle tecniche e sugli strumenti di particolare rilievo all'interno del settore dell'intrattenimento. La mostra artistica si è sviluppata attraverso tre fasi narrative, della durata complessiva di circa otto minuti, ed è stata realizzata mediante l'impiego di software quali la suite di Adobe, DaVinci Resolve, Unreal Engine, Blender, Houdini e TouchDesigner.

Oltre agli aspetti tecnici, il progetto ha rappresentato un'importante esperienza formativa, consentendo di comprendere le dinamiche aziendali e rimarcando l'importanza del lavoro di squadra come elemento fondamentale all'interno di una produzione. Doversi occupare individualmente di specifiche sequenze, garantendo al contempo coerenza stilistica del prodotto finale, ha favorito lo sviluppo di un'efficace visione condivisa. Il progetto, tuttavia, ha presentato numerose sfide che hanno messo alla prova le capacità di *problem solving* dei singoli componenti del team. Emblematica, in tal senso, è stata l'animazione finale del "flusso di conoscenza", sottoposta a ripetute revisioni e rielaborazioni per garantire un risultato in linea con le aspettative e la visione artistica iniziale. Nonostante molte delle tecniche adottate hanno rappresentato una prima esperienza per alcuni componenti del gruppo, il progetto è riuscito a raggiungere un elevato livello qualitativo.

L’installazione immersiva ha offerto la possibilità di vivere un'esperienza sensoriale suggestiva, un viaggio nella storia in grado di coniugare la dimensione passata con quella presente del panorama egiziano. La combinazione tra un’estetica ricca di dettagli, un comparto sonoro multidirezionale avvolgente e la narrazione dell’intelligenza artificiale, ha guidato lo spettatore verso un percorso interpretativo interiore, suscitando emozioni che vanno dal semplice stupore alla curiosità per arrivare al desiderio di scoperta.

Le ricostruzioni digitali e gli autentici suoni ambientali hanno trasformato il paesaggio egiziano in una dimensione viva e mutevole, in cui la conoscenza non è mai statica, ma si evolve continuamente attraverso le scoperte e la percezione individuale. Il processo di interpretazione della storia e della cultura egizia è stato adattato metaforicamente con immagini che si formavano e modellavano in continuazione, fino al momento in cui un flusso di particelle, espressione della conoscenza ultima e inafferrabile, riporta l’osservatore al punto di partenza. La mostra cerca di riprodurre la natura stessa della ricerca storica, dove ogni tentativo di comprensione è inevitabilmente destinato a essere riscritto, reinterpretato e messo in discussione. L’installazione invita così il visitatore a confrontarsi con l’impossibilità di raggiungere una conoscenza definitiva della storia, dando modo di immedesimarsi in uno studioso che decifra la conoscenza procedendo per ipotesi e continue riscritture.

Il riscontro ottenuto dalla partecipazione del pubblico e la risonanza mediatica innescata dall’inaugurazione della mostra confermano l’efficacia del progetto nel ridefinire i confini della narrazione convenzionale. Il risultato è un ponte tra reale e onirico, un racconto che unisce la documentazione storica ad una narrazione che procede per immagini. *Paesaggi – Landscapes* propone una modalità innovativa di fruizione della conoscenza, sfruttando la creatività umana, il potenziale offerto dalle installazioni immersive e dalla New Media Art per veicolare concetti in modo dinamico e interattivo. La capacità di creare ambienti coinvolgenti rende queste tecnologie avanzate degli strumenti di divulgazione culturale estremamente potenti.

Nell’attuale momento storico in cui si stanno continuamente ridefinendo le modalità di accesso alla conoscenza, le tecnologie digitali si configurano non

soltanto come un'evoluzione del linguaggio espositivo, ma anche come nuove forme di espressione artistica, capaci di rendere il passato osservabile, percepibile ed alla portata di tutti. L'integrazione di queste nuove tecniche di storytelling con l'arte digitale apre delle prospettive inedite verso scenari in cui l'esperienza sensoriale diventa un mezzo per l'apprendimento e la riflessione, ridefinendo il ruolo del museo come uno spazio di dialogo tra la storia, l'innovazione e la percezione individuale.

INDICE DELLE FIGURE

BIBLIOGRAFIA

Agisoft LLC., *Agisoft Metashape User Manual: Standard Edition*, Agisoft LLC: St. Petersburg, Russia, 2024.

Burley B., Walt Disney Animation Studios, *Physically Based Shading at Disney*, Acm Siggraph (vol. 2012, pp. 1-7), 2012.

Del Vesco P., Moiso B., *Missione Egitto 1903-1920: l'avventura archeologica M.A.I. raccontata*, Franco Cosimo Panini, 2017.

Fissoun D., *The Colorist Guide to DaVinci Resolve 18*, Blackmagic Design Learning Series, 2022.

Kerbl B., Kopanas G., Leimkuehler T., Drettakis G., *3D Gaussian Splatting for Real-Time Radiance Field Rendering*, ACM Transactions on Graphics (vol. 42, no. 4, pp. 1-14), 2023.

Museo Egizio di Torino, Donadoni Roveri A. M., *Civiltà degli Egizi: le arti della celebrazione*. Istituto bancario San Paolo di Torino, 1989.

Museo Egizio di Torino, Intesa Sanpaolo, *Intesa Sanpaolo sostiene il riallestimento della Galleria dei Re del Museo Egizio per il Bicentenario. Il Museo Egizio e le Gallerie d'Italia presentano "Paesaggi – Landscapes"* [Comunicato stampa], Torino, 12 giugno 2024. Documento ufficiale consultabile all'indirizzo: https://drive.google.com/file/d/1XdHgCti1BBw9930eM8E4ISTZ0XGJ2nMR.

Rolle A., *Il villaggio operaio di Deir el-Medina dalla fondazione all'abbandono*, Egittologia.net Magazine (vol. 4, pp. 44-51), 2012.

Schiaparelli E., *Relazione sui lavori della missione archeologica italiana in Egitto (anni 1903-1920). Volume secondo: La tomba intatta dell'architetto Cha nella necropoli di Tebe*, Casa Editrice Giovanni Chiantore, 1927.

Skousen M. B., *Aloe Vera Handbook: The Ancient Egyptian Medicine Plant*, Book Publishing Company, 1992.

Sorkhabi E., *Introduction to TouchDesigner 099*, nVoid, Lean Publishing, 2019.

Yao-An Lee A., *A Guide To Capturing and Preparing Photogrammetry For Unity*, The Emerging Media Lab and Department of Geography at the University of British Columbia, UBC Teaching, Learning Enhancement Fund and BCcampus Open Education, 2017.

Töpfer S., Del Vesco P., Poole F., *Deir el-Medina Through the Kaleidoscope: Proceedings of the International Workshop*, Torino, 2022.

Zahran M., Willis A., *The Vegetation of Egypt*, Springer Netherlands, 2009.

SITOGRAFIA

Blender Foundation, *Blender Manual*
https://docs.blender.org/manual/en/latest/

Blockade Labs, *Blockade Labs API Documentation*
https://api-documentation.blockadelabs.com/api/.

Capturing Reality, *RC Help: Documentation and Support*
https://rchelp.capturingreality.com.

Derivative, *TouchDesigner User Guide*
https://docs.derivative.ca/Main_Page

Epic Games Developers, *Mastering Megascans: A Guide to Photogrammetry and Asset Creation*
https://dev.epicgames.com/community/learning/paths/yzG/unreal-engine-capturing-reality-mastering-megascans-a-guide-to-photogrammetry-and-asset-creation.

Epic Games Developers, *Unreal Engine 5.3 Documentation*
https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-5-3-documentation?application_version=5.3

Gallerie d'Italia, *"Paesaggi – Landscapes" al Museo Egizio*, 13 Giugno 2024
https://gallerieditalia.com/it/torino/mostre-e-iniziative/in-evidenza/2024/06/13/paesaggi-landscapes-museo-egizio

Gunther E., *Ultra Dynamic Sky*
https://www.ultradynamicsky.com.

Kilic A., *SkyBoxGenerator*
https://github.com/ademkilic7/SkyBoxGenerator

Museo Egizio, *Il Museo Egizio e Gallerie d'Italia presentano "Paesaggi – Landscapes"*, 12 Giugno 2024
https://www.museoegizio.it/esplora/notizie/il-museo-egizio-e-gallerie-ditalia-presentano-paesaggi-landscapes

PlayCanvas, *SuperSplat Wiki*
https://github.com/playcanvas/supersplat/wiki

Rääli M., *AnyTree*
https://blendermarket.com/products/anytree

Rydén S., *Particle Flowfields*
https://www.patreon.com/supermarketsallad

SideFX, *Houdini Documentation*
https://www.sidefx.com/docs/houdini/

Tavan G., *Torino: Alle Gallerie d'Italia la video installazione immersiva Paesaggi – Landscapes, progetto culturale ideato dal Museo Egizio in cui la fotografia e la video-arte incontrano l'archeologia e l'antico Egitto*, 13 Giugno 2024
https://archeologiavocidalpassato.com/2024/06/13/torino-alle-gallerie-ditalia-la-video-installazione-immersiva-paesaggi-landscapes-progetto-culturale-ideato-dal-museo-egizio-in-partnership-con-intesa-sanpaolo/

Topaz Labs, *Guide for Topaz Video AI*
https://docs.topazlabs.com/video-ai/quick-start