POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria informatica: Grafica e Multimedia

Tesi di Laurea Magistrale

"Paesaggi - Landascapes": il progetto Egitto Immersivo nella sua prima realizzazione alle Gallerie d'Italia

Relatore Prof. Riccardo Antonino

Candidato Gioele Maggi

MAT: s303360

Anno Accademico 2023/2024

INDICE

ELENCO DELLE FIGURE

INTRODUZIONE

Negli ultimi anni, l'uso della grafica computerizzata e delle tecnologie digitali ha assunto un ruolo di primo piano in numerosi campi, dalla ricerca scientifica alla produzione artistica. Questa tesi si colloca in tale contesto, affrontando lo sviluppo di video fotorealistici non in tempo reale, focalizzati sulla rappresentazione del fiume Nilo, sui paesaggi circostanti e sui siti archeologici egiziani, sfruttando le potenzialità offerte da Unreal Engine e altre tecnologie avanzate. Il progetto ha avuto, infatti, l'obiettivo di realizzare un video per una sala immersiva presso le Gallerie d'Italia, intitolato “Paesaggi – Landscapes”, commissionato dal Museo Egizio di Torino come anticipazione delle future sale immersive del museo stesso.

Infatti, il presente lavoro di tesi si inserisce nel contesto del progetto “Egitto Immersivo” del Museo Egizio di Torino, un'iniziativa ambiziosa che prevede l'apertura di due sale immersive nel 2025. La realizzazione di questo progetto è stata affidata a Robin Studio, uno studio di produzione video con sede a Torino, con il quale ho collaborato nel corso del tirocinio che ha coronato il mio percorso di laurea magistrale al Politecnico di Torino. Il progetto “Egitto Immersivo” nasce, così, con l’obiettivo di creare un'esperienza multimediale unica che esplori vari aspetti della cultura egizia, integrando contenuti visuali avanzati renderizzati in tempo reale tramite Unreal Engine e proiettati sulle pareti delle sale. La prima sala si concentra sull'importanza del processo di ricerca del Museo Egizio sugli oggetti della sua collezione. L'obiettivo è di ricostruire il contesto originale degli oggetti esposti, ponendo un particolare accento sul rapporto tra questi artefatti e il paesaggio da cui provengono. Attraverso la narrazione di un vaso di terracotta con scene di vita sul Nilo, la sala intende rappresentare il “flusso di coscienza” del museo, suddiviso in tre segmenti principali. Il primo segmento esamina l'evoluzione degli strumenti di analisi degli oggetti, il secondo analizza il territorio egiziano e la sua influenza sulle comunità locali, con studi di caso su Hammamija e Gebelein, e il terzo esplora il processo di conoscenza attraverso l'analisi dei dati delle pubblicazioni scientifiche sull'Egitto. La seconda sala, invece, si focalizza sul ciclo delle stagioni nell'antico Egitto, strettamente legato al ciclo delle inondazioni del Nilo. Le vetrine disposte sulle pareti illustreranno le tre stagioni dell'antico Egitto *Akhet* “inondazione”, *Peret* “germinazione” e *Shemu* “raccolto”, mostrando oggetti legati alla vita agricola. Ogni sezione della sala descriverà gli eventi naturali caratteristici di ciascuna stagione e il loro impatto sulla vita quotidiana degli antichi egizi, mettendoli in relazione con gli oggetti esposti.

Tuttavia, sebbene il progetto originale fosse previsto per l'autunno del 2024, vari problemi logistici e organizzativi hanno portato ad un posticipo al 2025, ritardo che ha motivato la creazione di un progetto preliminare, intitolato "Paesaggi – Landscapes", che funge da anteprima delle future sale immersive. Il progetto "Paesaggi – Landscapes" rappresenta, di fatto, un'importante iniziativa culturale che unisce la fotografia e i video artistici con l'archeologia dell'antico Egitto. E questo progetto è stato reso possibile grazie alla collaborazione tra il Museo Egizio di Torino e le Gallerie d'Italia ed è stato realizzato da Robin Studio. L'installazione, aperta al pubblico dal 13 giugno al 12 settembre 2024, mira a offrire un'esperienza immersiva e riflessiva sul tema della ricostruzione storica e paesaggistica e si articola in tre parti principali, ognuna con un focus specifico che contribuisce a creare una narrazione complessa e coinvolgente. La prima parte è caratterizzata da video girati direttamente in Egitto, utilizzando tecniche avanzate come riprese con droni. Questi video, proiettati nelle sale del museo, offrono una rappresentazione visiva contemporanea dei paesaggi egiziani, permettendo ai visitatori di immergersi nella realtà attuale del territorio egiziano. La documentazione visiva moderna diventa un punto di partenza per un viaggio che cerca di connettere il presente con le tracce del passato. La seconda parte dell'installazione è quella realizzata in Computer Graphics (CG), di cui mi sono occupato personalmente. In questa sezione, l'obiettivo è ricreare gli ambienti dell'antico Egitto attraverso simulazioni fotorealistiche. Utilizzando Unreal Engine, sono stati ricostruiti paesaggi e siti archeologici, con particolare attenzione alle variazioni causate dal fiume Nilo nel corso dei secoli, e infatti sono state create simulazioni che evidenziano come il fiume abbia potuto modellare il paesaggio, erodendo le rocce e cambiando la morfologia del territorio. Una delle altre tecniche impiegate è il timelapse giorno/notte di siti archeologici, che mostra l'evoluzione dei luoghi nel tempo. Questa parte non solo arricchisce la comprensione storica, ma offre anche una visualizzazione dinamica e immersiva delle trasformazioni geografiche e ambientali. La terza parte dell'installazione assume un tono più sperimentale e visionario, utilizzando immagini generate dall'intelligenza artificiale per creare un effetto allucinogeno. Questa sezione esplora il concetto secondo il quale, man mano che ci si allontana nel tempo, diventa sempre più difficile ricostruire con precisione la realtà del passato. Le immagini generate dall'IA rappresentano un caleidoscopio di possibilità, suggerendo che ogni tentativo di comprendere completamente il passato si scontra con l'incertezza e l'indeterminatezza. La narrazione culmina in una scena finale in cui tutte le immagini si condensano in un punto, esplodendo poi in un flusso di pensieri che si espande su tutta la sala. Questa esplosione simbolizza il flusso di

coscienza e la complessità del processo di ricostruzione storica, sottolineando la fragilità delle nostre interpretazioni.

Ci si è resi, così, conto – nel corso del lavoro di produzione del progetto per le Gallerie d'Italia – che il Nilo, con la sua lunga storia e il suo impatto sulla civiltà egizia, rappresenta un soggetto ideale per dimostrare come la tecnologia moderna possa rendere visibili e accessibili patrimoni storici e naturali in modi innovativi, ricordando l'importanza della storia e della natura nella nostra cultura. In questo modo, la creazione di rappresentazioni fotorealistiche di ambienti naturali e storici permette non solo di preservare la memoria di questi luoghi, ma anche di offrire al pubblico esperienze immersive e coinvolgenti: il video nella sua interezza non si concentra solo sul Nilo, ma anche su fiori, piante, paesaggi naturali e siti archeologici egizi, offrendo una visione più ampia e dettagliata della bellezza e della storia dell'Egitto, in cui lo spettatore può immergersi grazie a tale progetto.

Per raggiungere tale scopo, Unreal Engine, con il suo potente sistema di rendering e di creazione di effetti visivi, è stato lo strumento principale. La realizzazione delle scene del Nilo ha richiesto, inoltre, l'uso combinato di diverse tecniche avanzate, tra cui il sistema Niagara per gli effetti particellari, blueprint per l'erosione delle rocce, il PostProcess Volume per migliorare l'aspetto visivo, le Material Functions per simulare fenomeni naturali come le onde di Gerstner e i Level Sequence per creare timelapse per scoprire come si è evoluto l'ambiente nei secoli. Ogni dettaglio è stato curato per garantire il massimo livello di realismo, sfruttando al meglio le capacità di Unreal Engine. Un altro aspetto fondamentale del progetto è stata la fotogrammetria, un processo che permette di creare modelli 3D dettagliati a partire da fotografie reali. In questo caso, le fotogrammetrie non riguardavano i paesaggi vicino al Nilo, ma piuttosto città, templi e altri siti archeologici egizi. Per generare modelli 3D realistici di questi luoghi storici sono state utilizzate le numerose immagini scattate in Egitto dei rispettivi siti, successivamente importati e ottimizzati in Unreal Engine, integrandosi perfettamente con il resto delle scene. Il workflow per la fotogrammetria ha richiesto l'uso di molti programmi diversi, garantendo che i modelli finali fossero di altissima qualità e fedeli alle loro controparti reali. Per di più, oltre agli strumenti di Unreal Engine, è stato utilizzato TouchDesigner per la creazione di sistemi particellari complessi e per la generazione di effetti visivi unici. In particolare, TouchDesigner è stato impiegato per realizzare sistemi particellari che prendono la forma di oggetti specifici, ricreati dai gaussian splats ottenuti dalle immagini. Questo ha permesso di aggiungere un ulteriore livello di interattività e dinamismo alle scene, rendendole ancora più coinvolgenti. L'uso combinato di tutte queste tecnologie ha permesso di creare un

workflow efficace e flessibile, capace di adattarsi alle esigenze specifiche del progetto, un progetto in cui ogni fase – dalla raccolta dei dati alla loro elaborazione e rendering finale – è stata attentamente pianificata ed eseguita, garantendo un risultato di alta qualità.

In definitiva, questa tesi – a partire dal progetto "Paesaggi – Landscapes", anticipatore di quello che sarà "Egitto Immersivo", il cui obiettivo finale ha fortemente influenzato lo svolgimento del progetto per le Gallerie d'Italia – ha l'obiettivo non solo di illustrare le tecniche utilizzate per la creazione di video fotorealistici del Nilo e dell'Egitto in generale, ma anche di offrire un esempio concreto di come la tecnologia possa essere utilizzata per valorizzare e preservare il nostro patrimonio culturale e naturale, attraverso un approccio multidisciplinare e interattivo. Infatti, le esperienze immersive create attraverso questi metodi non solo offrono nuove opportunità di apprendimento e intrattenimento, ma contribuiscono anche a sensibilizzare il pubblico sull'importanza della storia e della natura, facendo riflettere sulle nuove frontiere dell'archeologia ed esplorando i limiti delle ricostruzioni storiche, evidenziando inoltre come la combinazione di tecnologie moderne e pratiche artistiche possa offrire nuove prospettive e comprensioni del passato, contribuendo a una divulgazione culturale più ricca e interattiva.

PRIMA PARTE: LAVORI PRELIMINARI

In questo capitolo ci si vuole soffermare sui lavori svolti prima dell'inizio del progetto vero e proprio, concentrandosi in particolare sulle ricerche teoriche necessarie per la realizzazione di scene realistiche e coerenti con la realtà. L'obbiettivo del video finale risulta essere, infatti, quello di rivelare l'ambiente egiziano dal presente al passato, partendo dall'età contemporanea fino a risalire ai tempi dell'antico Egitto. Avendo questo obiettivo, è stato necessario uno studio naturalistico dell'ambiente egizio, in particolare – specificamente riguardo la mia parte di lavoro – a proposito del Nilo e del territorio intorno ad esso, profondamente da lui dipendente. Fondamentale per questo lavoro, oltre alla ricerca teorica da me effettuata, è stato il sostegno di Robin Studio, che ha messo a disposizione fotografie e filmati dell'ambiente egizio ricavati in loco nel mese di gennaio dell'anno 2024.

1.1 Il Nilo: la storia del fiume e il suo ruolo nello sviluppo della civiltà egizia nel corso del tempo

Il fiume Nilo, situato nel Nord Africa, copre circa 3,4 milioni di chilometri quadrati, attraversando ben 11 paesi africani – Burundi, Repubblica Democratica del Congo, Egitto, Eritrea, Etiopia, Kenya, Ruanda, Sudan, Sud Sudan, Tanzania e Uganda –, e si estende per 6650 chilometri. Per questa ragione, il celebre fiume egiziano, denominato *iteru* "grande fiume" in lingua egizia, è stato a lungo considerato il fiume più lungo del mondo – anche se in realtà, al giorno d'oggi, il primato sembra essere detenuto dal Rio delle Amazzoni.

Il Nilo, celebre anche per la sua antichità – studi recenti testimoniano l'antichità del fiume egiziano, il quale risulta avere un'età stimata di circa 31 milioni di anni –, è composto da due principali affluenti: il Nilo Bianco e il Nilo Azzurro. Il Nilo Bianco ha origine dal lago Vittoria, in Uganda, mentre il Nilo Azzurro nasce dal lago Tana, in Etiopia. Questi due fiumi confluiscono nella capitale sudanese, Khartum, da dove il Nilo procede verso nord, attraversando il Sudan e l'Egitto, fino a sfociare in un vasto delta nel Mar Mediterraneo. Tra l'altro, il delta del Nilo si estende per circa 240 chilometri lungo la costa del Mar Mediterraneo e copre un'area di circa 22.000 chilometri quadrati. Il delta è caratterizzato da numerosi rami e canali che distribuiscono l'acqua del fiume attraverso una vasta rete di terreni coltivabili.

Considerando la rilevanza geografica del fiume, non si può non evidenziare come esso sia stato essenziale per la nascita e lo sviluppo della civiltà egizia. D'altronde, la presenza del Nilo facilitò a suo tempo l'insediamento umano lungo le sue rive, sebbene soltanto in una fascia

di grandezza limitata (da 1 a 6 chilometri). Inoltre, l'importanza del Nilo risiedeva anche nella sua navigabilità, in quanto percorribile in entrambe le direzioni, costituendo così una via di comunicazione cruciale che permetteva di collegare luoghi anche molto distanti.

Figura 1.1: Il fiume Nilo

1.2 La morfologia della valle del Nilo: dalle acque alle rocce

Rinomate sono le storiche esondazioni annuali del Nilo, le quali avvenivano durante l'estate a causa delle piogge monsoniche sull'altopiano etiopico – nonostante differenti siano gli equilibri sorti al giorno d'oggi per via della presenza della diga di Assuan. Queste esondazioni, le quali erano prevedibili, costituivano un ciclo vitale per l'agricoltura nell'antico Egitto, un ciclo vitale che era suddiviso in tre fasi principali: *Akhet* "inondazione", *Peret* "germinazione" e *Shemu* "raccolto". La stagione di *Akhet* iniziava a giugno e durava fino a settembre, periodo durante il quale le acque del Nilo aumentavano notevolmente a causa delle abbondanti piogge. Le inondazioni coprivano così le terre circostanti, depositando strati di limo fertile, essenziale per l'agricoltura. Da ottobre a febbraio, durante la stagione di *Peret*, le acque cominciavano poi a ritirarsi, lasciando il terreno ricco di nutrienti, fondamentali per la semina di colture come grano e orzo. Infine, da marzo a maggio, le piante raggiungevano la maturità necessaria per il raccolto, motivo per cui questo periodo era denominato *Shemu*, traducibile per l'appunto con l'italiano "raccolto", elemento fondamentale per l'economia della cultura dell'antico Egitto.

Il limo prodotto dal Nilo e depositato sulle terre circostanti durante il periodo di *Akhet* era, ed è ancora oggi, fondamentale per le colture in Egitto. Si tratta di un sedimento fertile, ricco di nutrienti come azoto, potassio e fosforo, che migliorano significativamente la fertilità

del suolo. Infatti, la deposizione di limo non solo arricchiva i terreni agricoli, ma compensava anche l'erosione e il degrado del suolo, mantenendo la produttività agricola anno dopo anno. Fu proprio questo ciclo di fertilizzazione naturale a contribuire alla stabilità e prosperità delle civiltà egizie lungo il Nilo, permettendo loro di svilupparsi e prosperare.

Oltre agli effetti benefici per l'agricoltura, l'innalzamento e abbassamento annuale del fiume provocò nel corso dei secoli cambiamenti significativi nella morfologia del territorio. In particolare, per quello che riguarda le rocce della valle del Nilo – su cui ho concentrato la mia attenzione a livello teorico, avendo l'obiettivo di realizzare una scena raffigurante gli effetti del Nilo sulle rocce presenti nel fiume –, due sono gli effetti visibili del passaggio del Nilo: l'erosione e il cambiamento del colore della roccia. Infatti, con il flusso del fiume, le rocce – anno dopo anno e da millenni – vengono erose attraverso processi di abrasione e corrosione, in cui i sedimenti trasportati dall'acqua agiscono come abrasivi, levigando e scolpendo la superficie delle rocce. Inoltre, l'innalzamento e l'abbassamento del livello delle acque del Nilo ha portato alla formazione di linee di demarcazione nette sulle rocce esposte. Per di più, il cambiamento del colore delle rocce risulta essere un altro effetto significativo del passaggio del Nilo: le parti delle rocce costantemente sommerse nell'acqua tendono, col passare del tempo, a scurirsi, in particolare a causa della deposizione di sedimenti e della crescita di microorganismi su di esse; al contrario, le parti esposte all'aria e agli agenti atmosferici subiscono via via processi di ossidazione, conferendo alle rocce una colorazione più chiara. Questo contrasto crea linee di demarcazione ben visibili, caratterizzando le rocce da una parte superiore più chiara e ossidata e una parte inferiore più scura e ridotta, caratterizzazione che non solo testimonia l'azione millenaria del fiume sul territorio circostante, ma – per di più – contribuisce a comprendere meglio le dinamiche geomorfologiche e climatiche dell'area.

Figura 1.2: Le rocce del Nilo

1.3 *Al-Sadd al-ʿĀlī*, ovvero la diga di Assuan

La diga di Assuan, una delle strutture ingegneristiche più significative mai costruite in Egitto, ha avuto un impatto enorme sul fiume Nilo e sul territorio circostante, da lui costantemente influenzato. Infatti, completata nel 1970, questa diga, lunga 3.830 metri e alta 111 metri, ha cambiato radicalmente la gestione delle esondazioni annuali e l'ecologia del fiume. D'altronde, la diga di Assuan fu costruita con l'obiettivo di controllare le esondazioni del Nilo, affrontando così le devastazioni causate dalle inondazioni, le quali, sebbene prevedibili ed essenziali per l'agricoltura, provocavano danni significativi alle infrastrutture e alle abitazioni.

A partire da questo obiettivo, numerosi sono stati e sono tuttora i benefici derivanti dalla diga di Assuan. Uno dei principali benefici della diga è il controllo delle piene del Nilo, controllo che permette una regolazione precisa del flusso d'acqua, prevenendo inondazioni distruttive e garantendo una fornitura costante di acqua per l'irrigazione agricola durante tutto l'anno. Inoltre, la diga di Assuan genera energia idroelettrica, con una capacità installata di 2,1 gigawatt, che ha significativamente contribuito allo sviluppo urbano e industriale dell'Egitto, riducendo la dipendenza del paese dalle risorse energetiche non rinnovabili.

Tuttavia, la costruzione della diga ha portato con sé anche alcuni effetti negativi. Infatti, la regolazione del flusso del Nilo ha portato all'interruzione del naturale ciclo di deposizione del limo, riducendo la fertilità del suolo nelle regioni agricole lungo il fiume. Con la diga in funzione, gli agricoltori hanno dovuto così ricorrere all'uso di fertilizzanti artificiali per mantenere la produttività dei campi, aumentando così i costi di produzione e contribuendo all'inquinamento del suolo e delle acque. Inoltre, la diga ha alterato significativamente gli habitat naturali lungo il Nilo, influenzando negativamente la biodiversità: la riduzione delle inondazioni stagionali ha colpito gli ecosistemi fluviali, portando alla diminuzione di alcune specie di pesci e uccelli che dipendevano dai cicli naturali del fiume; per di più, la riduzione delle aree umide, un tempo vitali per molte specie animali e vegetali, non ha potuto far altro che portare ad una perdita di biodiversità. Un altro impatto significativo della diga di Assuan è stato sul patrimonio culturale dell'Egitto: la formazione del lago Nasser, il grande bacino artificiale creato dalla diga, ha sommerso numerosi siti archeologici di inestimabile valore; per salvare questi siti, furono intraprese importanti campagne di salvataggio, tra cui lo smontaggio e la ricostruzione del tempio di Abu Simbel in una posizione più elevata. Tutti lati negativi, questi, che testimoniano come la diga di Assuan rappresenti un esempio emblematico di come

le grandi opere ingegneristiche possano portare benefici significativi, ma anche sfide ambientali e culturali complesse.

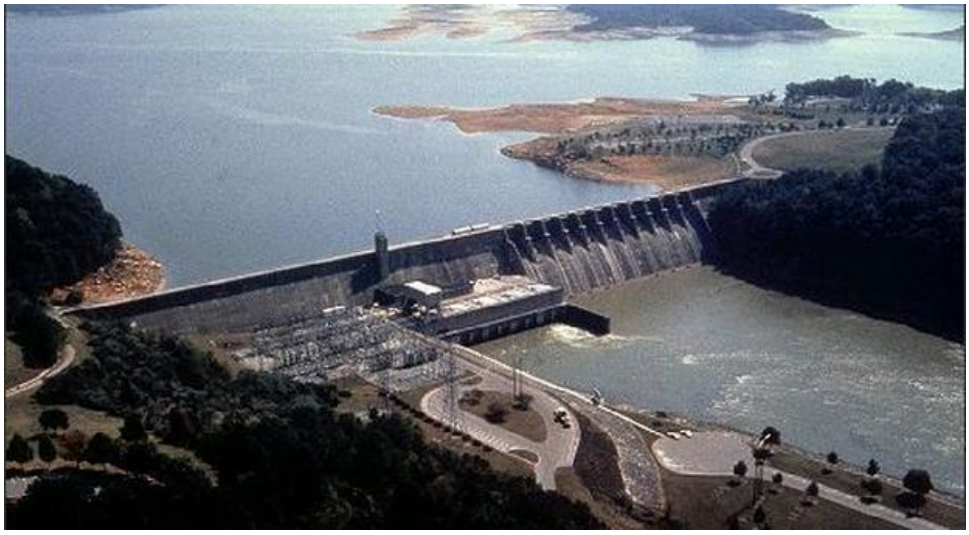

Figura 1.3: La diga di Assuan

1.4 Il sito archeologico di Medina

Deir el-Medina, situato nel cuore del deserto occidentale nei pressi di Tebe, è un sito archeologico egiziano scoperto dall'italiano Ernesto Schiaparelli – che iniziò i primi scavi tra il 1905 e il 1909 – al giorno d'oggi straordinariamente ben conservato. Questa città, originariamente chiamata *Pa Demi* "la cittadina", era caratterizzata da una struttura di forma rettangolare, misurava 163 metri di lunghezza e 50 metri di larghezza, ed era circondata da un muro con un'unica porta d'accesso; questa configurazione limitava i contatti con l'esterno per prevenire il rischio di furti nelle tombe. Il villaggio presentava un'organizzazione ben definita, con abitazioni disposte lungo una strada principale che divideva l'insediamento in due quartieri, est e ovest, richiamando la struttura di una nave. Le case, disposte in file ordinate simili alle moderne case a schiera, erano ciascuna composta da circa 86 metri quadrati suddivisi in atrio, sala, camera da letto, cucina, cantina e un tetto piatto utilizzato come terrazzo.

La città di *Deir el-Medina* ospitava gli artigiani responsabili della costruzione e decorazione delle tombe dei faraoni del Nuovo Regno (1539-1069 a.C.). Le condizioni di vita degli abitanti erano relativamente buone per l'epoca: gli artigiani godevano della sicurezza di una casa, nonché della possibilità di costruire una tomba personale; inoltre, essi godevano di un approvvigionamento costante di cibo e beni di prima necessità. Il villaggio dipendeva interamente dagli approvvigionamenti esterni, che giungevano mensilmente da Tebe tramite una squadra specializzata che portava acqua, legna, combustibile e strumenti di lavoro. Inoltre,

un gruppo di domestiche aiutava le famiglie con le faccende quotidiane. Il lavoro a *Deir el-Medina* era altamente organizzato: gli operai che lavoravano sulla costruzione delle tombe reali erano divisi in squadre, ciascuna guidata da un capo operaio; gli scribi avevano il compito di registrare i pagamenti e monitorare le presenze, utilizzando ostraka[1] e papiri per annotare il lavoro giornaliero. Per di più, ogni ventottesimo giorno del mese, gli operai ricevevano un salario in natura, composto principalmente da grano e orzo, sufficiente a sostenere una famiglia. Infine, durante le festività o in occasione di visite importanti, venivano distribuite razioni extra di cibo e bevande.

Tuttavia, nonostante le condizioni di vita privilegiate, gli abitanti di *Deir el-Medina* affrontarono diverse crisi. Una delle più significative si verificò nel 1156 a.C., durante il regno di Ramses III (Tebe, 1218/1217 a.C. circa – Tebe, 1155 a.C.), quando i lavoratori misero in atto il primo sciopero documentato della storia, sciopero scoppiato a causa dei ritardi nei pagamenti. La protesta, documentata nel *papiro giuridico di Torino*[2], durò diverse stagioni e segnò l'inizio del declino del villaggio. Alla fine del Nuovo Regno, la crisi generale dell'Egitto colpì anche *Deir el-Medina*. I ritardi nei pagamenti portarono alcuni operai a rubare dalle tombe reali, e la corruzione tra i funzionari imperiali aggravò la situazione. Con il tempo, gli approvvigionamenti divennero sempre più scarsi e la manodopera qualificata iniziò ad abbandonare il villaggio, che alla fine fu lasciato deserto. Solo nel IV secolo d.C., alcuni monaci copti[3] tornarono ad abitare questo luogo, ribattezzandolo *Deir el-Medina*, "Monastero della città".

[1] Un *ostrakon* (o *ostracon*, dal greco antico *òstrakon*, plurale *òstraka*) è un frammento di ceramica o pietra, solitamente proveniente da vasi o altri recipienti di terracotta, utilizzato per scopi scrittori. Questi significativi reperti epigrafici possono presentare parole iscritte, incise o altre forme di scrittura. Particolarmente famosi sono gli ostraka utilizzati come schede elettorali nelle procedure di ostracismo, una pratica giuridica della democrazia ateniese volta a esiliare temporaneamente individui che potevano costituire una minaccia per la città.

[2] Il *papiro giuridico di Torino* (noto anche come *papiro legale di Torino* o *papiro della congiura dell'harem*) è un antico documento egizio conservato al Museo Egizio di Torino. Esso tratta dei processi intentati contro i cospiratori che tentarono di assassinare Ramses III, episodio noto come la "congiura dell'harem".

[3] I copti rappresentano un gruppo etnoreligioso cristiano originario dell'Egitto, dove il cristianesimo fu la religione predominante fino al IX secolo. Successivamente, la comunità copta si ridusse a una significativa minoranza.

Figura 1.4: Il sito archeologico di Medina

1.5 Il sacro tempio di Iside a *Philae*

I templi di File, noti anche come *Philae*, costituiscono un complesso templare egizio situato originariamente sull'isola di File, nel Nilo. L'isola di File, il cui nome in egizio significa "l'isola del tempo", ha una storia molto antica, tra l'altro menzionata da autori classici come Strabone[4], Diodoro Siculo[5], Tolomeo[6], Seneca[7] e Plinio il Vecchio[8]. Infatti, File era un'importante frontiera meridionale del regno egizio, utilizzata come guarnigione militare e nodo commerciale tra l'Egitto e la Nubia. La sua posizione strategica rendeva l'isola un punto cruciale per il trasbordo delle merci che non potevano attraversare le cateratte[9] del Nilo. Inoltre, File era anche un sito religioso di grande rilevanza, ritenuto uno dei luoghi di sepoltura di Osiride[10]. Il primo edificio sacro risale al faraone nubiano Taharqa[11]. Nel corso dei secoli, sorsero sull'isola numerosi

[4] Strabone è stato un geografo, storico e filosofo greco vissuto tra il 60 a.C. ed il 20 d.C. circa.
[5] Diodoro Siculo è stato uno storico greco antico vissuto tra il 90 a.C. circa ed il 27 a.C. circa.
[6] Tolomeo è stato un astronomo, astrologo e geografo greco antico con cittadinanza romana, nato e morto in Egitto e vissuto tra il 100 d. C. ed il 170 d.C. circa.
[7] Lucio Anneo Seneca è stato un filosofo, drammaturgo e politico romano vissuto tra il 4 a. C. – anno in cui nacque a Cordova – ed il 65 d. C. – anno della morte, avvenuta a Roma.
[8] Gaio Plinio Secondo, conosciuto come Plinio il Vecchio, è stato uno scrittore, naturalista, filosofo naturalista, comandante militare e governatore provinciale romano vissuto tra il 23 ed il 79 d.C. nella penisola italica.
[9] Le cateratte sono tratte del Nilo situate tra le città di Assuan e Khartum, in cui l'acqua non ha profondità sufficiente per permetterne la navigazione.
[10] Osiride (conosciuto anche come Usiride, Osiris o Osiri, versioni ellenizzate dell'egizio Asar o Asir) è una divinità egizia della religione dell'antico Egitto, membro dell'Enneade e considerato un mitico faraone. Venerato come l'inventore dell'agricoltura e della religione, si credeva che avesse regnato come civilizzatore e benefattore dell'umanità. In una delle numerose varianti del suo mito, Osiride morì annegato nel Nilo, vittima di un complotto ordito dal fratello minore Seth. Nonostante lo smembramento del suo corpo, sarebbe tornato in vita grazie alle pratiche magiche delle sorelle Iside e Nefti.
[11] Re di Kush e faraone della XXV dinastia egiziana (690-664 a.C.), Taharqa governò in un periodo di grande prosperità, evidente anche nell'arte e nell'architettura del tempo. Nei primi anni del suo regno, Taharqa fece costruire templi nelle principali città di Kush, tra cui Napata, Kawa, Sanam e Qasr Ibrim.

templi dedicati a Iside[12], Horus[13] e Hathor[14], trasformando File in uno dei più importanti luoghi di pellegrinaggio dell'antico Egitto. I templi furono chiusi nel VI secolo d.C. dall'imperatore bizantino Giustiniano I (11 maggio 482, Macedonia del Nord – 14 novembre 565, Costantinopoli); nonostante ciò, alcune strutture furono riadattate per il culto cristiano fino all'invasione araba del VII secolo.

Il tempio di Iside, edificato durante il periodo tolemaico e ampliato dai Romani, è uno degli esempi meglio conservati di architettura templare egizia. Questo complesso templare combina stili architettonici egizi, greci e romani, con decorazioni elaborate e colori vivaci, i quali furono visibili fino al XIX secolo. Nel VI secolo d.C. il tempio fu trasformato in una chiesa cristiana, motivo per il quale alcune sculture originali furono coperte o addirittura distrutte. Le decorazioni del complesso rappresentano scene mitologiche e rituali, in particolare legate ai miti di Iside e Osiride.

Nel XIX secolo l'isola di File divenne una meta di grande interesse per gli studiosi europei: l'archeologo italiano Giovanni Battista Belzoni visitò il tempio nel 1817 e, l'anno seguente, su ordine del console inglese, asportò l'obelisco, che fu portato in Inghilterra e utilizzato insieme alla stele di Rosetta per decifrare i geroglifici egizi. Tuttavia, la costruzione della diga di Assuan, avvenuta nel 1902, e i successivi innalzamenti della sua struttura nel 1907-1912 e 1929-1933, da cui derivarono importanti cambiamenti del territorio circostante, causarono la sommersione dell'isola di File. Negli anni '60 l'UNESCO avviò una campagna di salvataggio dei monumenti a rischio, per cui una diga temporanea fu costruita intorno al sito per far riemergere i templi dalle acque. Questi furono poi smontati e trasferiti sull'isola di Agilkia, a circa 550 metri di distanza, in un progetto completato nel 1980 grazie all'impresa italiana Condotte-Mazzi Estero. Questa ricostruzione del complesso ha rispettato scrupolosamente la struttura originaria, mantenendo l'armonia con il paesaggio circostante. Al giorno d'oggi *Philae* è uno dei siti archeologici più visitati in Egitto, famoso anche per i suoi spettacoli serali di luci e suoni, che ne esaltano la bellezza e la storia. D'altronde, i templi di

[12] Iside, conosciuta anche come Isi o Aset in lingua egizia, è una dea dell'antico Egitto. Divinità della vita, della guarigione, della fertilità e della magia, ha le sue origini a *Behbeit el-Hagar*, nel Delta del Nilo.
[13] Horus è una delle divinità più antiche e rilevanti della religione egizia. Il suo culto, presente nella Valle del Nilo dalla tarda Preistoria fino all'epoca tolemaica e alla dominazione romana, iniziò a Ieracompoli (*Nekhen*), nell'Alto Egitto. Considerato la prima divinità nazionale, Horus era strettamente associato al faraone, visto come la sua incarnazione vivente, e a Osiride, con cui era identificato dopo la morte.
[14] Hathor, anche conosciuta come Ator, è una delle dee più importanti e adorate nella storia dell'antico Egitto. Dea della gioia, dell'amore, della maternità e della bellezza, il suo culto fu centrale per tutto il periodo egizio.

Philae continuano a rappresentare un esempio straordinario di arte e architettura antica, testimoniando l'importanza religiosa e culturale dell'isola nel corso dei secoli.

Figura 1.5: Un tempio di Philae

SECONDA PARTE: UNREAL ENGINE 5.3 PER IL PROGETTO PAESAGGI – LANDASCAPES

Oggetto di questa parte della mia tesi – cuore pulsante dello scritto come anche del mio lavoro di tirocinio e del progetto a cui ho lavorato – è il software e motore di gioco Unreal Engine, scelto come strumento principale per lo sviluppo del progetto Egitto Immersivo, in particolare per via della sua capacità di realizzare un rendering ad alta risoluzione in tempo reale. Unreal Engine è stato, inoltre, – per una questione di coerenza e di continuità nel lavoro – utilizzato per la realizzazione del progetto Paesaggi-Landscapes. D'altronde, ci sono diversi aspetti per cui Unreal Engine è eccellente anche per la produzione di video offline. Innanzitutto, per via della sua qualità grafica superiore: Unreal Engine è rinomato per la sua capacità di produrre immagini altamente realistiche grazie al suo avanzato sistema di rendering; esso supporta tecnologie come il ray tracing in tempo reale, permettendo di ottenere effetti di luce, ombra e riflessi estremamente realistici. In secondo luogo, per via dei suoi potenti strumenti di rendering: con Unreal Engine, è possibile utilizzare il motore di rendering di alta qualità per creare video con alti livelli di dettagli; il motore supporta tecniche di rendering avanzate come il Lumen per l'illuminazione globale dinamica e il Nanite per la geometria virtualizzata, consentendo un dettaglio senza precedenti nelle scene. Inoltre, per via della sua ampia libreria di risorse: Unreal Engine offre accesso a una vasta libreria di asset e risorse, tramite la libreria Quixel, incluse texture, modelli 3D, materiali e strumenti di animazione; queste risorse possono accelerare notevolmente il processo di sviluppo del video. Per di più, Unreal Engine è eccellente per via della comunità ad esso collegata e del supporto da essa derivante: Unreal Engine ha infatti una vasta comunità di sviluppatori e artisti che condividono risorse, tutorial e supporto; questo facilita l'apprendimento e la risoluzione di problemi, offrendo un ambiente collaborativo per migliorare le proprie competenze. In aggiunta, di grande importanza risulta la questione del costo e della licenza: Unreal Engine è gratuito per uso personale e commerciale, con una licenza basata su royalty solo se il prodotto genera entrate superiori a un milione di dollari. Infine, Unreal Engine permette di creare script che automatizzano processi o creano comportamenti complessi; questo consente di ottimizzare il flusso di lavoro e di realizzare animazioni e sequenze di azioni in modo più efficiente, rendendo il motore ancora più versatile per una vasta gamma di applicazioni.

2.1 Unreal Engine 5.3: un potente strumento di lavoro

2.1.1 Il software: una panoramica sulla sua storia e le sue funzionalità

Unreal Engine è uno dei motori di gioco più celebri e utilizzati nello sviluppo di videogiochi e applicazioni interattive. Creato da Epic Games, Unreal Engine vanta una lunga e affascinante storia che inizia negli anni '90 e continua a evolversi fino ai giorni nostri. Questa sezione fornisce una panoramica storica dettagliata della nascita e dell'evoluzione di Unreal Engine, mettendo in risalto le sue principali versioni e l'impatto sull'industria.

Il progetto Unreal Engine ebbe origine dalla visione di Tim Sweeney, fondatore di Epic Games. Originariamente, il motore fu sviluppato per supportare il gioco “Unreal”, un first-person shooter (FPS) lanciato nel 1998. Sweeney aspirava a creare un motore grafico potente e flessibile, capace di gestire ambienti tridimensionali complessi e di offrire effetti visivi avanzati. Il debutto di Unreal Engine con il gioco “Unreal” fu un grande successo sia critico che commerciale, evidenziando non solo le avanzate capacità grafiche del motore, ma anche strumenti di sviluppo integrati come UnrealEd, che permettevano ai designer di creare e modificare livelli di gioco in maniera intuitiva. Dopo il successo iniziale, Epic Games continuò a migliorare Unreal Engine, rilasciando versioni aggiornate con nuove funzionalità e avanzamenti tecnologici significativi. Nel 2002, Unreal Engine 2 portò grandi miglioramenti alle capacità grafiche e fisiche del motore. Questa versione venne adottata in numerosi giochi di successo, tra cui “Unreal Tournament 2003” e “Tom Clancy's Splinter Cell”, e introdusse il supporto per middleware di terze parti, rendendo il motore più flessibile e potente. Il lancio di Unreal Engine 3 nel 2006 rappresentò un significativo progresso nella tecnologia dei motori di gioco. Con il supporto per le console di nuova generazione, come Xbox 360 e PlayStation 3, e per DirectX 10, Unreal Engine 3 offriva grafica ad alta definizione, shader avanzati e fisica realistica. Titoli come “Gears of War” e “BioShock” dimostrarono le capacità del motore e ne favorirono la diffusione nel settore. Nel 2014, Unreal Engine 4 introdusse ulteriori miglioramenti, tra cui il supporto per la grafica fotorealistica e strumenti di sviluppo ancora più avanzati. Una delle innovazioni più rivoluzionarie di questa versione fu l'introduzione del linguaggio di scripting visivo Blueprints, che permetteva agli sviluppatori di creare gameplay complessi senza dover scrivere codice. Unreal Engine 4 trovò applicazione non solo nei giochi, ma anche in architettura, cinema e realtà virtuale. Annunciato nel 2020 e lanciato ufficialmente nel 2021, Unreal Engine 5 rappresenta l'ultima iterazione del motore. Questa versione introduce tecnologie innovative come Nanite, che consente di gestire geometrie estremamente dettagliate,

e Lumen, un sistema di illuminazione globale dinamica. Queste funzionalità mirano a superare le limitazioni tecniche e a offrire strumenti ancora più potenti per la creazione di mondi virtuali immersivi. Nel corso degli anni, Unreal Engine è diventato uno standard industriale, utilizzato da studi di sviluppo di tutte le dimensioni per creare giochi, simulazioni e applicazioni interattive. La sua flessibilità e potenza lo hanno reso adatto a una vasta gamma di settori, dall'intrattenimento ai campi industriali. Unreal Engine continua a essere una piattaforma di riferimento grazie alla sua capacità di adattarsi alle esigenze in continua evoluzione degli sviluppatori e alla sua attiva comunità che contribuisce costantemente con risorse e supporto.

2.1.2 L'interfaccia grafica di Unreal Engine

In questo paragrafo verrà mostrata l'interfaccia di base di Unreal Engine 5, seguendo le informazioni contenute nella documentazione ufficiale[15]. La numerazione segue quella dell'immagine ufficiale:

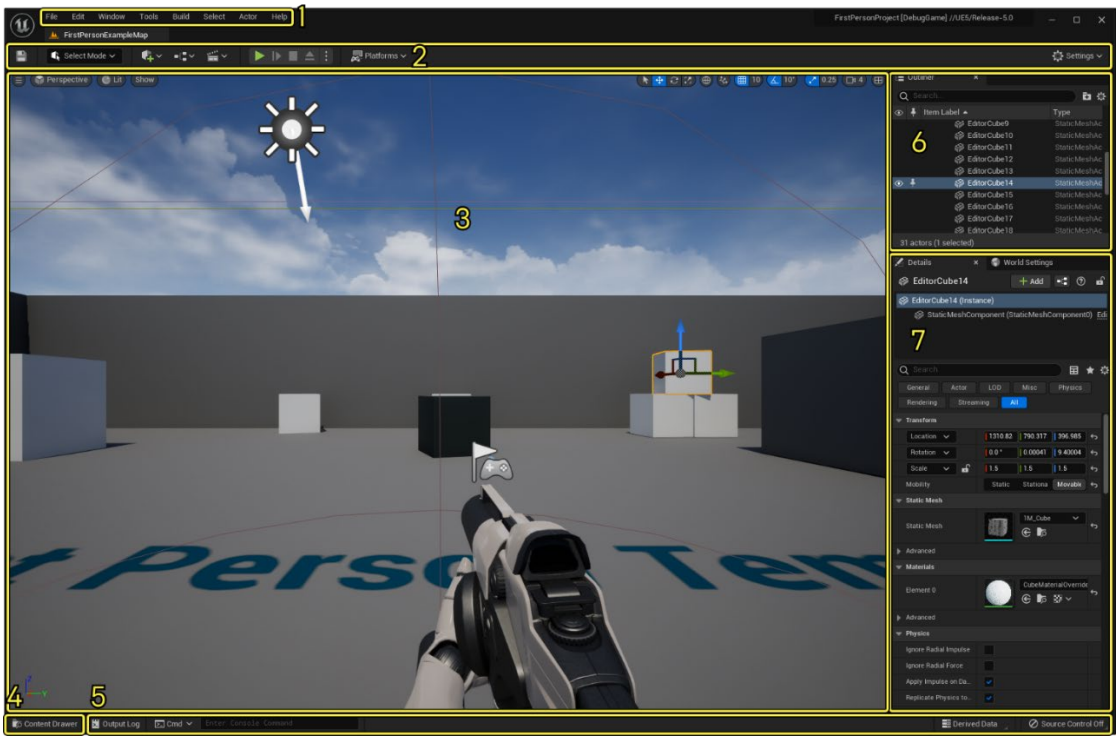

Figura 2.1: Interfaccia di Unreal Engine

1. Barra dei Menu: classica barra dei menu, che ti permette di accedere a funzionalità specifiche nell'editor, quali nella sezione Window per gestire quali finestre aprire o

[15] Informazioni e immagini reperite dalla documentazione ufficiale di Unreal Engine al sito *https://dev.epicgames.com/documentation/it-it/unreal-engine/unreal-engine-5-3-documentation?application_version=5.3*

chiudere, oppure la sezione Actor[16] per eseguire azioni specifiche sull'oggetto selezionato.

2. Barra degli Strumenti Principale: Include collegamenti rapidi per gli strumenti e gli editor più utilizzati in Unreal Engine, facilitando il cambio della modalità di lavoro e l'aggiunta di elementi nella scena. Le modalità di lavoro disponibili includono: Selection Mode[17], Landscape Mode[18], Foliage[19], Mesh Paint[20], Modeling[21], Fracture[22], Brush Editing[23] e Animation[24]. Gli elementi che si possono aggiungere in scena sono molteplici, dalle semplici mesh[25] cubiche ai modelli complessi importati dall'Unreal Marketplace[26] o dalla libreria Quixel. È possibile aggiungere anche altri elementi come il Level Sequencer per le animazioni o i Blueprints[27] per la creazione di script. Inoltre, la barra offre scorciatoie per accedere alla modalità Play (eseguire il proprio gioco all'interno di Unreal Editor) e per distribuire il progetto su diverse piattaforme.
3. Level Viewport: Visualizza i contenuti del livello (ovvero della scena), inclusi telecamere, attori, mesh statiche e altri oggetti. È uno strumento fondamentale per interagire direttamente con gli elementi presenti nella scena, consentendo operazioni di spostamento, rotazione e scalatura. Il Level Viewport permette la navigazione libera nella scena, utilizzando mouse e tasti di movimento, e offre diverse modalità di

[16] Un Actor è qualsiasi oggetto che può essere posizionato in un livello, come personaggi, luci, suoni, mesh statiche, particelle, telecamere e molto altro. Ogni Actor può contenere componenti che definiscono il suo aspetto, comportamento e funzionalità.
[17] La Selection Mode in Unreal Engine permette di selezionare e manipolare gli oggetti nella scena. Usata per spostare, ruotare e scalare gli Actor, è essenziale per posizionare e organizzare gli elementi del livello in modo preciso.
[18] La Landscape Mode consente di creare e modificare terreni dettagliati. Utilizzando strumenti di modellazione, pittura e scultura, permette di generare montagne, vallate e altri elementi naturali per creare ambienti realistici.
[19] Il Foliage Tool permette di dipingere vegetazione e altri elementi naturali direttamente sul terreno. Utilizzato per aggiungere alberi, erba e cespugli, facilita la creazione di paesaggi naturali e dettagliati.
[20] Mesh Paint consente di dipingere colori e texture direttamente su mesh 3D. Utilizzato per aggiungere dettagli come usura, ruggine o personalizzazioni estetiche, permette di migliorare l'aspetto visivo delle superfici in modo intuitivo.
[21] Il Modeling Mode offre strumenti per creare e modificare geometrie 3D direttamente in Unreal Engine. Utilizzato per progettare oggetti complessi, consente di modellare, scolpire e ottimizzare mesh senza dover ricorrere a software esterni.
[22] Il Fracture Tool consente di simulare la distruzione e la frammentazione degli oggetti. Utilizzato per creare effetti realistici di rottura e demolizione, è essenziale per scene di azione e dinamiche di gioco coinvolgenti.
[23] Brush Editing è uno strumento per creare e modificare forme geometriche di base nel livello. Utilizzato per progettare architetture semplici e definire spazi, permette di costruire rapidamente strutture e layout del livello.
[24] Il Animation Mode permette di creare e modificare animazioni per personaggi e oggetti. Utilizzato per animare movimenti, pose e sequenze, offre strumenti per gestire skeleton, keyframe e curve di animazione, rendendo le scene dinamiche e interattive.
[25] Una mesh è una struttura 3D composta da vertici, spigoli e facce che definiscono la forma di un oggetto. Utilizzata per rappresentare modelli 3D in giochi e applicazioni, la mesh è fondamentale per creare la geometria visibile di personaggi, ambienti e oggetti.
[26] Il Marketplace di UE offre asset, plugin e strumenti per migliorare e accelerare lo sviluppo di giochi.
[27] Quixel, Level Sequencer e Blueprints sono argomenti importanti che verranno trattati più avanti nella tesi.

visualizzazione, come Wireframe, Unlit, Lit[28] e modalità speciali per visualizzare le collisioni e i livelli di dettaglio (LODs)[29]. È possibile eseguire il gioco direttamente nella finestra, fornendo feedback immediato sulle modifiche apportate. Il Level Viewport supporta snap tools per l'allineamento preciso degli oggetti e "Show Flags" per gestire la visibilità di specifici elementi nella scena. Inoltre, è personalizzabile, consentendo la configurazione di layout multipli e salvando le preferenze di visualizzazione per ottimizzare il flusso di lavoro, nonché l'impostazione di diverse telecamere e angolazioni per ottenere prospettive ottimali durante lo sviluppo del livello.

4. Content Drawer o Browser: Apre il Content Drawer, che consente di accedere a tutti gli asset del progetto. Questo strumento permette di organizzare, cercare e gestire facilmente modelli 3D, texture, materiali, suoni e altri contenuti essenziali per lo sviluppo del gioco.
5. Barra degli Strumenti Inferiore: Contiene scorciatoie per la Command Console, l'Output Log e le funzionalità dei dati derivati. Visualizza inoltre lo stato del controllo del codice sorgente, permettendo di monitorare e gestire facilmente le modifiche al progetto e le risorse utilizzate.
6. Outliner: Visualizza una vista ad albero gerarchica di tutti i contenuti presenti nel tuo livello, permettendo di organizzare, selezionare e gestire facilmente attori, oggetti e componenti della scena.
7. Pannello Dettagli: Appare quando selezioni un Actor. Visualizza diverse proprietà per quell'Actor, come la sua Trasformazione (posizione nel livello), Mesh Statica, Materiale e impostazioni fisiche. Questo pannello mostra impostazioni diverse a seconda dell'oggetto selezionato nel Level Viewport, permettendo di modificare e personalizzare in dettaglio le caratteristiche specifiche di ciascun elemento, facilitando il controllo e l'ottimizzazione degli aspetti visivi e funzionali della scena.

2.1.3 Blueprints: la programmazione in Unreal

Il sistema di scripting visuale Blueprint in Unreal Engine è un potente linguaggio di programmazione visuale che consente di creare elementi di gioco utilizzando un'interfaccia a

[28] Le modalità di visualizzazione in Unreal Engine includono Wireframe, Unlit e Lit. Wireframe mostra solo le linee delle mesh, utile per analizzare la geometria. Unlit visualizza le texture senza effetti di luce, ideale per controllare le texture. Lit applica ombre e luci per una visione realistica della scena.
[29] I livelli di dettaglio (LOD) ottimizzano le prestazioni del gioco riducendo la complessità delle mesh in base alla distanza dalla camera. Oggetti lontani usano versioni meno dettagliate per migliorare le performance.

nodi all'interno dell'Unreal Editor. Questo sistema è estremamente flessibile, offrendo agli utenti una gamma completa di concetti e strumenti di scripting che sono tipicamente disponibili solo per i programmatori. Con Blueprint, gli utenti possono definire classi e oggetti orientati agli oggetti (OO), permettendo una maggiore interazione e personalizzazione del gameplay. Il flusso di lavoro basato sui nodi di Blueprint fornisce un'interfaccia intuitiva e accessibile, facilitando la creazione di script complessi senza la necessità di una profonda conoscenza della programmazione. Questo sistema non solo amplia le capacità dei designer, ma permette anche ai programmatori di creare strutture fondamentali utilizzando il linguaggio di programmazione C++ comune. Sfruttando le caratteristiche specifiche di C++, i programmatori possono esportare il codice in nodi semplici nel Blueprint, che i designer possono poi estendere. Questo garantisce un alto livello di collaborazione tra programmatori e designer. Il sistema di programmazione è vantaggioso anche per i programmatori esperti, che possono utilizzare Blueprint per gestire la logica di base e generale in modo semplice ed efficiente, riservando il C++ per scripting più complessi o funzioni personalizzate.

Grazie alla sua flessibilità, Blueprint è diventato uno strumento essenziale nello sviluppo di giochi, consentendo una rapida prototipazione e iterazione delle idee di gameplay. La possibilità di utilizzare concetti di programmazione orientata agli oggetti rende Blueprint particolarmente potente, permettendo di creare strutture di gioco complesse e dinamiche. Questo sistema integrato nell'Unreal Engine rappresenta un ponte tra la programmazione tradizionale e il design visivo, democratizzando l'accesso alla creazione di giochi e migliorando significativamente il flusso di lavoro dello sviluppo.

2.1.3.1 L’interfaccia del Blueprint Editor

In questo paragrafo verrà mostrata l’interfaccia del Blueprint Editor e le sue principali caratteristiche.

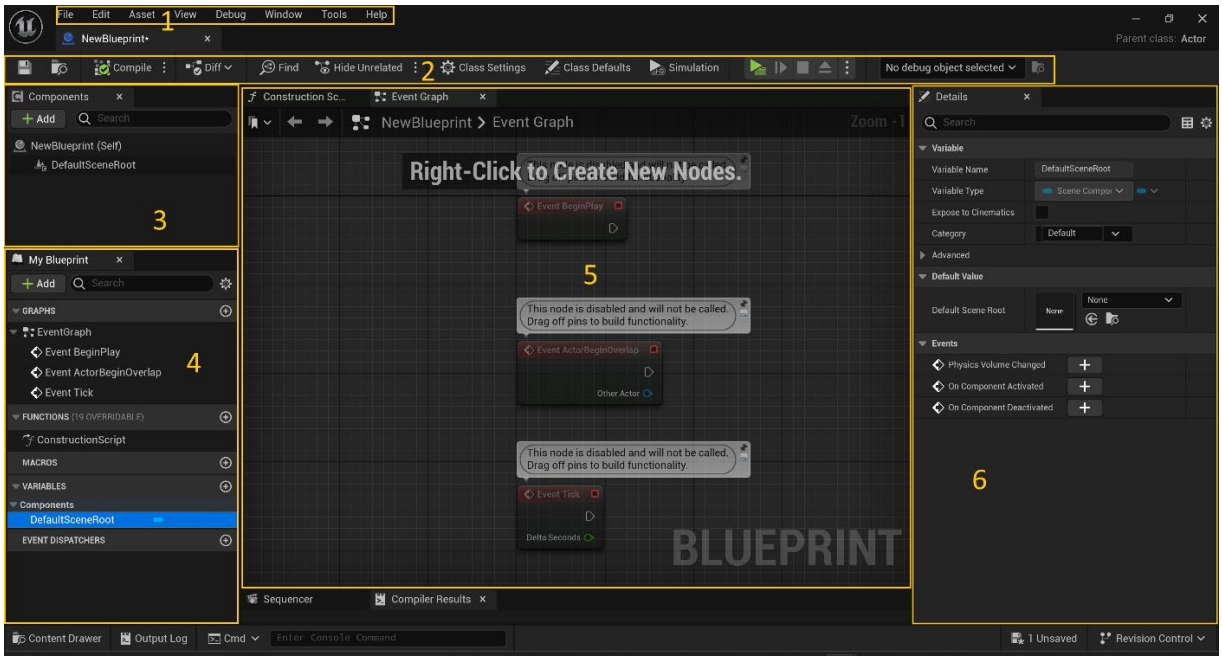

Figura 2.2: Interfaccia del Blueprint Editor

1. Il menu: Questo è il classico menu che permette di eseguire numerose azioni riguardanti principalmente la gestione dei file e le impostazioni dell'editor.

2. Toolbar: In questa barra si trovano i comandi principali per la gestione del blueprint durante la fase di editing. Qui sono presenti i comandi per compilare il blueprint, il tasto "play" per eseguirlo, e i comandi utili in fase di debug come i breakpoint e altri strumenti essenziali.

3. Components: Un componente è un elemento che può essere aggiunto per aumentare le funzionalità del blueprint. In questa sezione, i componenti possono essere aggiunti, modificati tramite il pannello Details, o eliminati. Possono anche essere organizzati gerarchicamente per una migliore gestione.

4. My Blueprint: Questa sezione contiene tutti gli elementi utilizzabili nel blueprint. Include sia elementi di base, come il construction script, sia elementi creati dall'utente, come funzioni o variabili aggiuntive.

5. Graph Editor: Il pannello Graph Editor rappresenta il cuore del sistema Blueprint. In questa sezione si progettano le reti di nodi e fili che definiscono il comportamento dello script. I nodi possono essere selezionati rapidamente cliccandoci sopra, riposizionati trascinandoli e aggiunti con un click destro scegliendo dall'elenco. Il Graph Editor offre una visione chiara e organizzata delle interazioni tra i vari componenti, permettendo di creare logiche complesse in modo intuitivo. Questa interfaccia consente di monitorare e modificare in tempo reale le connessioni e i flussi di dati, ottimizzando lo sviluppo delle funzionalità del gioco.

6. Details panel: Il pannello Details è un'area contestuale che fornisce accesso alle proprietà degli elementi selezionati all'interno del Blueprint Editor, permettendo di modificarle. Questo pannello è composto da una barra di ricerca per trovare rapidamente le proprietà specifiche e include una o più categorie a scomparsa per organizzare le proprietà. Nel pannello Details si svolgono molte delle operazioni di modifica dei Blueprint, come l'editing delle variabili, con la possibilità di cambiarne nome, tipo e stabilire se la variabile è un array. Inoltre, è possibile implementare interfacce Blueprint cliccando sul pulsante Blueprint Props, aggiungere input e output per le funzioni Blueprint e aggiungere eventi per i componenti selezionati. Questo rende il pannello Details uno strumento essenziale per la gestione e la personalizzazione dei Blueprint all'interno dell'editor.

2.1.3.2 Funzioni e variabili

Le funzioni nel sistema Blueprint di Unreal Engine rappresentano blocchi di codice riutilizzabili che facilitano l'organizzazione e la semplificazione dei grafici di script. Sono strumenti fondamentali per migliorare la leggibilità e la gestione del codice, permettendo la creazione di comportamenti complessi in modo modulare. Le funzioni possono essere create direttamente all'interno del Blueprint utilizzando il pannello “My Blueprint” e cliccando su “Add Function”. Una volta creata, la logica della funzione viene definita nel Graph Editor. Le funzioni possono accettare parametri di input e restituire valori di output, che vengono aggiunti tramite il pannello dei dettagli della funzione. Questo consente di passare variabili e valori tra la funzione e il grafico principale. Una volta definita, una funzione può essere chiamata ovunque nel Blueprint aggiungendo un nodo di chiamata della funzione nel Graph Editor. Questo nodo eseguirà il codice definito nella funzione ogni volta che viene attivato. Le funzioni offrono numerosi vantaggi. Innanzitutto, migliorano la modularità del codice, suddividendo il codice in parti più piccole e gestibili, facilitando la manutenzione e la modifica. Inoltre, permettono la riutilizzabilità dei blocchi di codice, evitando la riscrittura dello stesso codice in diverse parti del Blueprint. L'uso delle funzioni rende i grafici Blueprint più ordinati e leggibili, riducendo il disordine visivo e facilitando la comprensione del flusso logico. Infine, le funzioni semplificano il debugging, isolando parti specifiche del codice e rendendo più facile l'identificazione e la correzione degli errori.

Il “Construction Script” nel Blueprint di Unreal Engine è una funzione speciale eseguita automaticamente ogni volta che un attore viene creato o modificato nell'editor e ogni volta che il gioco viene avviato. Serve a inizializzare o aggiornare l'attore e i suoi componenti prima

dell'inizio del gioco. Questo script viene utilizzato per impostare proprietà, creare componenti dinamici e configurare elementi visivi. Grazie al Construction Script, è possibile garantire che gli attori siano correttamente configurati e pronti per l'uso, migliorando l'efficienza e l'organizzazione del progetto.

Le variabili nel sistema Blueprint di Unreal Engine sono fondamentali per memorizzare e manipolare dati durante l'esecuzione del gioco. Si suddividono in diverse categorie e rappresentano informazioni come valori numerici, testi, riferimenti a oggetti, vettori e altri tipi di dati. Le variabili primitive includono gli interi (Integer), utilizzati per memorizzare numeri interi; i float, che memorizzano numeri decimali; i booleani (Boolean), per valori vero/falso; e le stringhe (String), per il testo. Le variabili strutturate comprendono i vettori (Vector), per le coordinate 3D; i rotatori (Rotator), per le rotazioni 3D; e le transform, per la posizione, la rotazione e la scala di un oggetto. Infine, le variabili di oggetto includono i riferimenti ad attori (Actor Reference) e i riferimenti a componenti (Component Reference), usati rispettivamente per riferirsi ad altri attori nella scena e a componenti specifici di un attore. La creazione e l'utilizzo delle variabili inizia nel pannello “My Blueprint”, dove possono essere definite e assegnate a un tipo di dato e a un valore predefinito tramite il pannello “Details”. Le variabili possono essere lette e modificate attraverso i nodi di “Get” e “Set”: il nodo “Get” recupera il valore della variabile, mentre il nodo “Set” lo modifica, e sono di colore differente in base al tipo della variabile. Le variabili possono avere uno scope[30] a livello di Blueprint, rendendole visibili e utilizzabili in tutto il Blueprint, o a livello di funzione, limitandole alla funzione in cui sono definite. Le variabili offrono numerosi vantaggi, infatti permettono di memorizzare dati temporanei e permanenti necessari per il gameplay e la logica del gioco, offrendo flessibilità e controllo nella creazione di logiche complesse e dinamiche. Inoltre, facilitano l'organizzazione e la leggibilità del codice, rendendo più semplice il debugging e la manutenzione.

2.1.3.3 I nodi

Nel sistema Blueprint di Unreal Engine, i nodi sono gli elementi fondamentali utilizzati per costruire logiche e comportamenti del gioco tramite un'interfaccia visiva. Ogni nodo rappresenta una specifica azione, funzione o operazione, e può essere collegato ad altri nodi per creare flussi di esecuzione e catene di operazioni.

[30] Nella programmazione, la visibilità o ambito (in inglese scope) è l'esistenza e la possibilità di richiamare un identificatore, come una variabile o una funzione, in un determinato punto del programma piuttosto che in un altro.

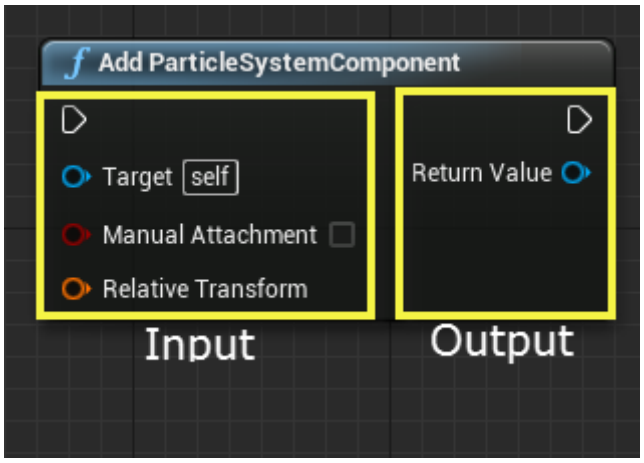

Figura 2.3: Un nodo del Blueprint

I nodi possono avere pin su entrambi i lati. I pin a sinistra sono input, mentre quelli a destra sono output. Ci sono due principali tipi di pin nei Blueprints: pin di esecuzione e pin di dati.

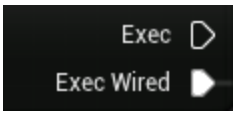

Figura 2.4: Un pin di esecuzione scollegato e uno collegato

I pin di esecuzione vengono utilizzati per collegare i nodi e creare un flusso di esecuzione. Quando un pin di input di esecuzione viene attivato, il nodo viene eseguito. Una volta completata l'esecuzione, il nodo attiva un pin di output di esecuzione per continuare il flusso. I pin di esecuzione sono visualizzati come contorni quando non sono collegati e pieni quando sono collegati ad altri pin di esecuzione. I nodi di chiamata di funzione hanno sempre un solo pin di input di esecuzione e un solo pin di output di esecuzione, poiché le funzioni hanno un solo punto di ingresso e di uscita. Altri tipi di nodi possono avere più pin di input e output di esecuzione, permettendo comportamenti diversi a seconda del pin attivato.

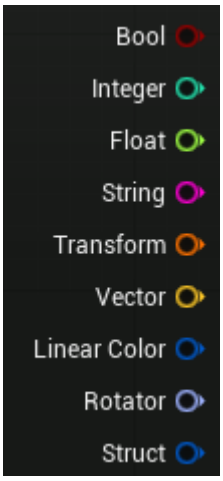

Figura 2.5: I tipi di pin di dati

I pin di dati vengono utilizzati per ricevere o inviare dati da e verso un nodo. Sono specifici per tipo e possono essere collegati a variabili dello stesso tipo o a un pin di dati dello stesso tipo su un altro nodo. Come i pin di esecuzione, i pin di dati sono visualizzati come contorni quando non sono collegati e pieni quando lo sono. I nodi possono avere un numero qualsiasi di pin di input o output di dati. I pin di dati di un nodo di chiamata di funzione corrispondono ai parametri e al valore di ritorno della funzione corrispondente.

Esistono diversi tipi di nodi nel sistema Blueprint di Unreal Engine, ciascuno con uno scopo e un colore specifico che ne indicano la funzione. I nodi rossi rappresentano eventi e servono a definire punti di ingresso del flusso di esecuzione nel Blueprint. Questi nodi si attivano quando si verifica un determinato evento nel gioco. Esempi di nodi rossi includono “Event Begin Play” (attivato all'inizio del gioco), “Event Tick” (attivato ad ogni frame) e “Event Actor Begin Overlap” (attivato quando un altro attore entra in contatto con l'attore corrente). I nodi verdi sono nodi di funzioni pure[31], generalmente privi di pin di esecuzione in ingresso e in uscita. Questi nodi servono principalmente a preparare e ottenere dati necessari per altre operazioni. Al contrario, i nodi blu rappresentano funzioni impure[32], che eseguono azioni specifiche e possiedono pin di esecuzione in input e output. I nodi grigi sono nodi di flusso, utilizzati per controllare il flusso di esecuzione nel Blueprint. Esempi comuni di nodi grigi includono il nodo “Branch” (equivalente all'istruzione if nel linguaggio C) e il nodo “Foreach”, che itera su una serie di elementi.

[31] Nell'ambito della programmazione funzionale le funzioni pure sono quelle che seguono un comportamento matematico: dato un determinato input, restituiscono sempre lo stesso output e non producono effetti collaterali.
[32] Nell'ambito della programmazione funzionale le funzioni non pure sono quelle che possono generare effetti collaterali.

In sintesi, i diversi colori dei nodi nel sistema Blueprint aiutano a distinguere rapidamente le loro funzioni: i nodi rossi per gli eventi, i nodi verdi per le funzioni pure, i nodi blu per le funzioni impure e i nodi grigi per il controllo del flusso. Questa distinzione cromatica facilita la comprensione e la gestione della logica nel Blueprint, migliorando l'efficienza del flusso di lavoro.

2.1.4 I materiali in Unreal Engine

I materiali in Unreal Engine definiscono le proprietà superficiali degli oggetti nella tua scena, fungendo da "vernice" applicata a una mesh per determinare il suo aspetto visivo. Questi materiali indicano al motore di gioco come una superficie deve interagire con la luce, specificando parametri come colore, riflettività, rugosità e trasparenza. Nella pipeline di rendering, gli shader sono essenziali per definire come ogni vertice o pixel viene renderizzato; in Unreal Engine, gli shader sono scritti in High Level Shading Language[33] (HLSL), un linguaggio di programmazione che consente di descrivere con precisione l'interazione tra luce e superfici. Il codice HLSL viene poi convertito in istruzioni in Assembly Language che la GPU può eseguire, determinando così i colori finali dei pixel visualizzati sullo schermo. Tuttavia, in Unreal Engine, non è necessario scrivere manualmente il codice HLSL per creare shader, infatti grazie all'Editor di Materiali, è possibile progettare e personalizzare i materiali attraverso un'interfaccia visiva intuitiva. Questo strumento facilita la creazione di effetti visivi avanzati senza richiedere competenze di programmazione, rendendo il processo di sviluppo più accessibile e efficiente.

2.1.4.1 L'interfaccia grafica del Material Editor

Come per le altre parti di Unreal Engine, in questa sezione verrà analizzata l'interfaccia del Material Editor, editor che serve a creare o modificare un materiale.

[33] HLSL, o High Level Shading Language, è un linguaggio utilizzato per creare shader per le GPU. Simile a linguaggi come C, permette agli sviluppatori di controllare dettagliatamente il comportamento della luce e delle superfici nel rendering 3D.

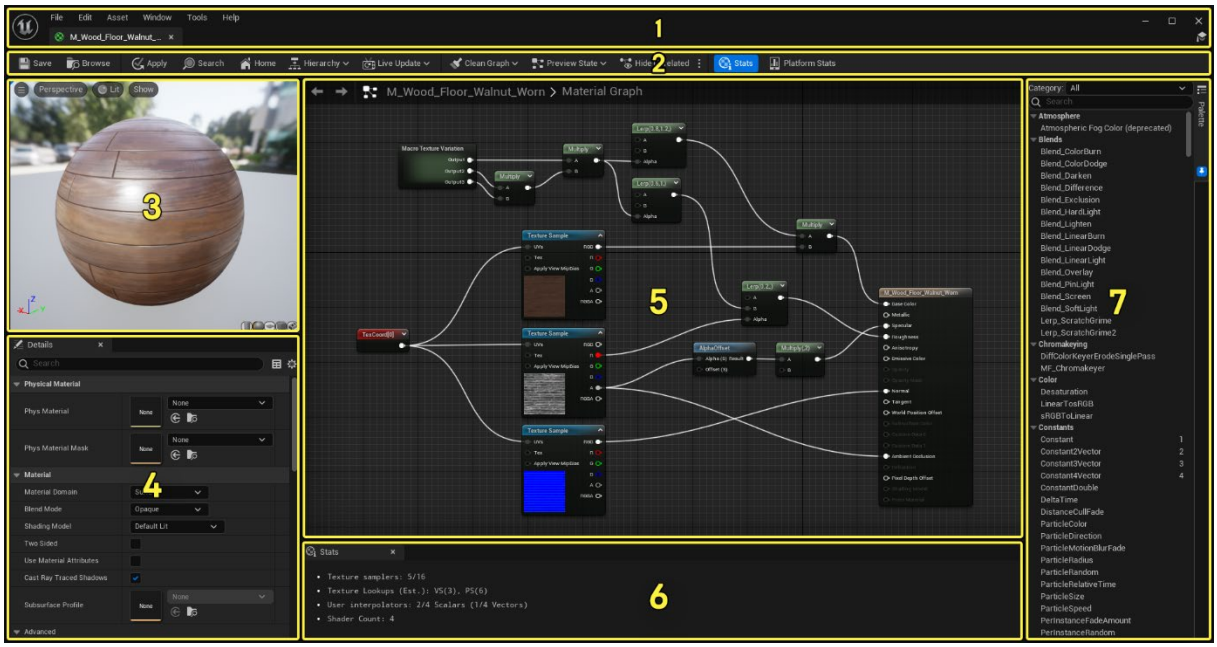

Figura 2.6: L'interfaccia del Material Editor

1. Menu Bar: La classica barra dei menu, che offre le principali utilità per la gestione del file e della finestra.
2. Toolbar: Questa barra contiene gli strumenti principali per la gestione del materiale. Di particolare rilevanza è il pulsante “Apply”, che applica le modifiche apportate al materiale a tutti gli oggetti in scena che utilizzano quel materiale.
3. Viewport Panel: Offre una vista in tempo reale del materiale applicato su una superficie di prova, permettendo di visualizzare immediatamente le modifiche effettuate.
4. Details Panel: Questo pannello presenta una finestra delle proprietà che mostra le caratteristiche dei nodi di espressione e funzione del Materiale selezionati. Se non ci sono nodi selezionati, vengono visualizzate le proprietà di base del Materiale.
5. Material Graph Panel: Contiene tutti i nodi che compongono il materiale e ne definiscono tutte le proprietà. Il funzionamento dei nodi è simile a quello dei blueprint, con nodi che hanno input e output, ma in questo caso non ci sono pin di esecuzione. L’insieme di tutti i nodi e i collegamenti tra essi è detto shader graph.
6. Stats Panel: In questo pannello vengono mostrate le statistiche del materiale, inclusi eventuali errori di compilazione.
7. Palette Panel: Contiene l’elenco categorizzato di tutti i nodi che possono essere inseriti nel Material Graph Panel. I nodi possono essere aggiunti anche premendo il tasto destro all’interno del Material Graph Panel.

2.1.4.2 Nodi e funzionamento base

Come per i Blueprints, esistono vari tipi di nodi nei materiali di Unreal Engine, tra cui nodi matematici (verdi), nodi di input (rossi), nodi di texture (blu) e molti altri. In questa sezione non verranno specificati e dettagliati tutti i tipi di nodi, ma verrà illustrato il loro comportamento generale. I nodi matematici vengono utilizzati per operazioni aritmetiche e logiche, i nodi di input per ricevere dati dall'esterno e i nodi di texture per applicare immagini alle superfici, permettendo una vasta gamma di personalizzazioni visive.

Un materiale è composto da un insieme di nodi che formano lo shader graph, e che confluiscono nel Main Material Node, il nodo finale. La combinazione dei valori applicati a questo nodo determina l'aspetto finale del materiale quando viene compilato e utilizzato nella scena. Non tutti i pin devono essere necessariamente collegati. Come mostrato nell'immagine, alcuni pin possono essere disabilitati, poiché non tutti i pin possono essere utilizzati contemporaneamente. La possibilità di utilizzare determinati pin dipende dal comportamento generale del materiale, come la sua traslucidità o opacità, che può essere modificata dal pannello Details quando non è selezionato nessun nodo. Questa flessibilità consente di creare materiali altamente personalizzati e ottimizzati per diverse esigenze visive e prestazionali.

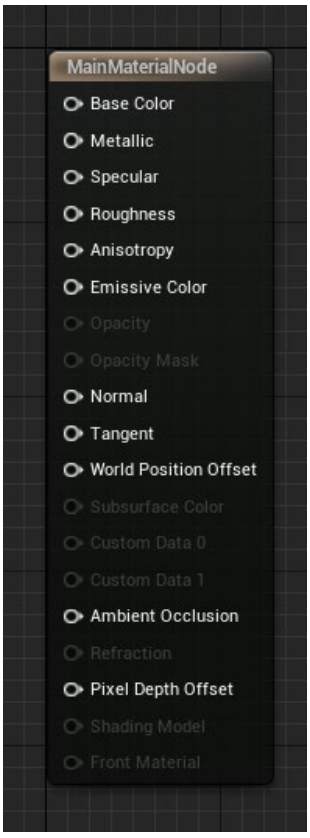

Figura 2.7: Il Main Material Node

Per i materiali più complessi, vengono spesso utilizzati nodi che eseguono funzioni avanzate, composte a loro volta da un material graph. Queste funzioni avanzate sono file separati dal materiale, rendendole riutilizzabili in diversi materiali; sono chiamate Material Function e sono ampiamente utilizzate per ottenere comportamenti specifici applicabili a più materiali. Le Material Function permettono di creare librerie di effetti visivi e proprietà condivisibili tra diversi progetti, migliorando l'efficienza e la coerenza del design.

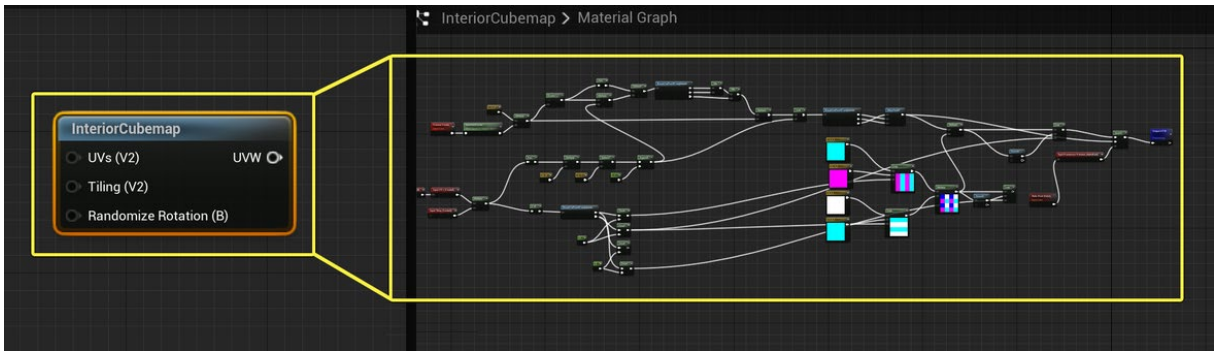

Figura 2.8: Material Function

Un concetto fondamentale riguardante i materiali in Unreal Engine è quello dei Material Instance. I Material Instance vengono usati per cambiare l'aspetto di un materiale senza modificare il Material di base e quindi senza necessità di ricompilazione. Questo permette di avere più oggetti con lo stesso materiale ma con leggere differenze. Se il Material è stato creato con parametri (Scalar o Vector Parameter) ai quali si può assegnare un nome, quando viene generato il Material Instance, in esso vengono mostrati tutti i parametri modificabili senza alterare il Material padre. Infatti, gli oggetti in scena utilizzano sempre un Material Instance piuttosto che il Material di base. Questa funzionalità è particolarmente utile per variare le proprietà di superficie in tempo reale, come il colore o la riflettività, adattandole a diverse condizioni di luce e ambientali.

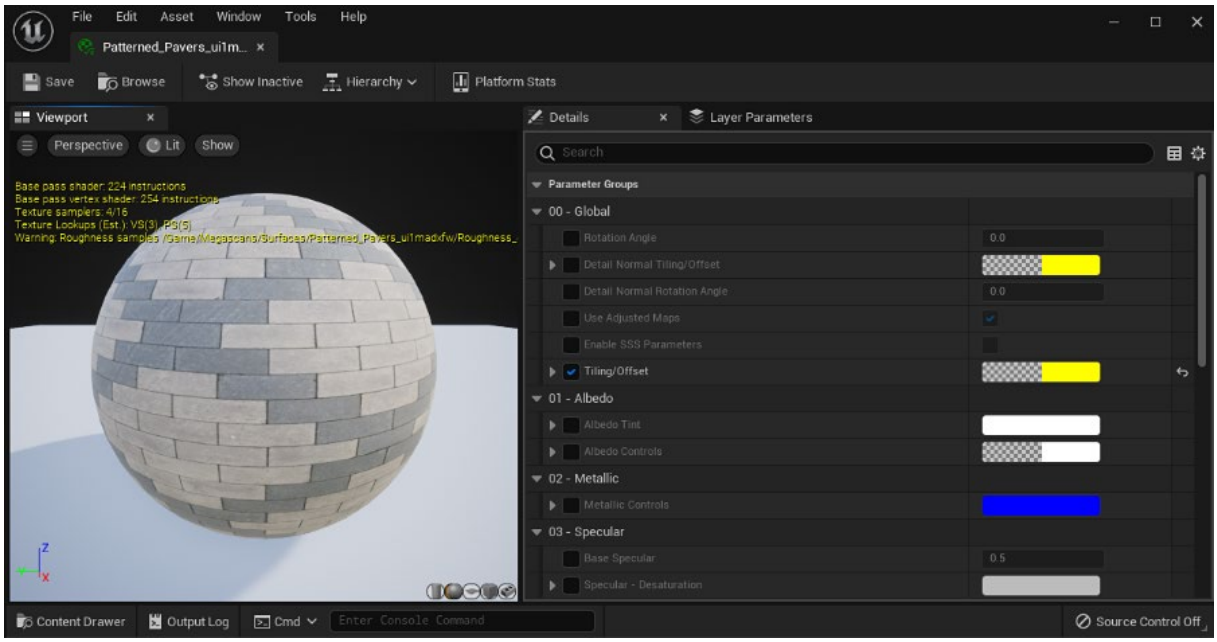

Figura 2.9: Un material instance

Un altro concetto utile sono i Material Parameter Collection (MPC). Questi file raggruppano valori scalari o vettoriali che possono essere utilizzati come input nei Material.

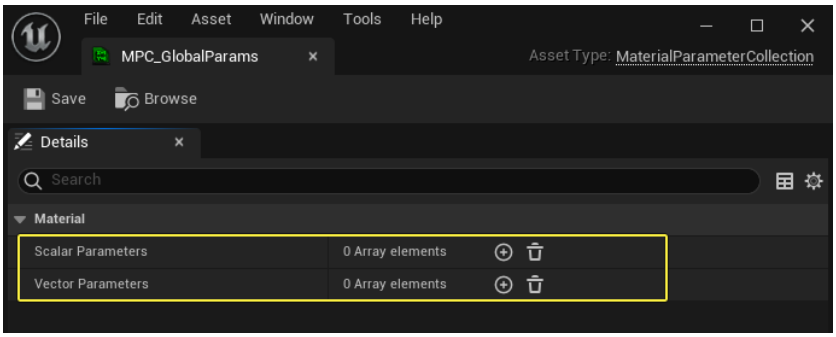

Figura 2.10: Un MPC

Ad esempio, può essere aggiunto un parametro chiamato “Emissive Power”, che può essere usato come un valore di input in un Material, con un aspetto simile a un classico nodo con un pin di output. Gli MPC sono estremamente versatili, permettendo di centralizzare e gestire i parametri che possono influenzare simultaneamente più materiali, migliorando il controllo e la coerenza visiva del progetto.

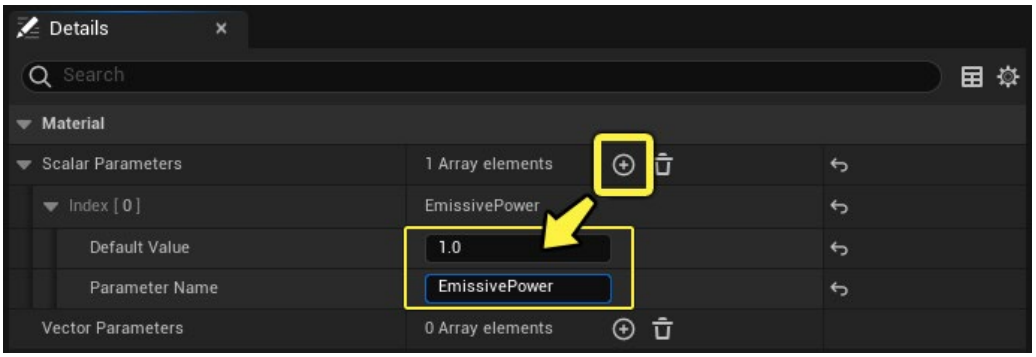

Figura 2.11: Creazione di un parametro in un MPC

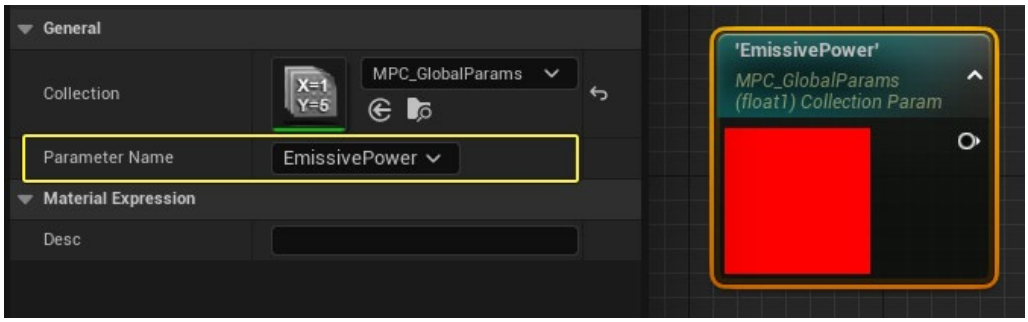

Figura 2.12: Il parametro dell'MPC usato in un Material

Gli MPC possono essere utilizzati anche nei Blueprints, dove possono fungere sia da input che da output, permettendo la modifica dei loro valori. Inoltre, i valori all'interno dei MPC possono essere animati nella timeline del Level Sequencer, argomento che verrà trattato in un capitolo successivo. Questa capacità di animazione consente di creare effetti dinamici e reattivi che possono essere sincronizzati con eventi di gioco o scenografie, arricchendo l'esperienza visiva e interattiva.

2.1.5 Niagara: un sistema per gli effetti visivi particellari

Il sistema Niagara in Unreal Engine è un avanzato sistema di effetti particellari progettato per creare, simulare e rendere complessi effetti visivi in tempo reale. Introdotto per la prima volta con l'aggiornamento 4.20 di Unreal Engine 4 nel 2018, Niagara è stato sviluppato per superare i limiti del precedente sistema di particelle Cascade, offrendo un controllo più granulare e una maggiore capacità di personalizzazione. Con il continuo sviluppo e miglioramento, è diventato uno strumento di riferimento per la creazione di effetti visivi avanzati nei giochi e nelle applicazioni interattive. Niagara è stato progettato per fornire agli sviluppatori un sistema modulare e scalabile per la creazione di effetti particellari, utilizzato in una vasta gamma di applicazioni. Tra queste, vi sono effetti visivi in tempo reale come fumo, fuoco, esplosioni, scintille, neve, pioggia e nebbia, così come animazioni di particelle che interagiscono con

l'ambiente e con altri oggetti di scena. Inoltre, Niagara è ideale per la gestione di effetti speciali per magie, abilità e poteri nei giochi, oltre a supportare simulazioni fisiche avanzate che richiedono particelle con comportamenti complessi e interattivi. Il suo funzionamento si basa su un'architettura modulare che consente agli utenti di costruire effetti particellari complessi combinando vari componenti. Un sistema Niagara è una raccolta di emitter che lavorano insieme per creare un effetto complessivo. Gli emitter sono i componenti di base che generano e gestiscono le particelle, ognuno dei quali può avere le proprie proprietà e comportamenti specifici. Niagara utilizza moduli per definire il comportamento delle particelle, controllando proprietà come velocità, colore, dimensione e durata. Questi moduli possono essere predefiniti o personalizzati dagli utenti, offrendo una grande flessibilità nella creazione degli effetti. Un aspetto fondamentale del sistema Niagara è l'utilizzo di un linguaggio di scripting visivo per definire comportamenti e logiche complesse. Gli utenti possono creare script personalizzati per controllare l'emissione, la dinamica e le interazioni delle particelle, permettendo di definire come le particelle reagiscono a eventi, forze e altri fattori ambientali. Questo linguaggio di scripting visivo consente di costruire effetti particellari estremamente sofisticati senza la necessità di scrivere codice manualmente.

2.1.5.1 L'interfaccia grafica del Niagara Editor

Seguendo come sempre la documentazione ufficiale, in questo paragrafo si tratterà dell'interfaccia del Niagara Editor.

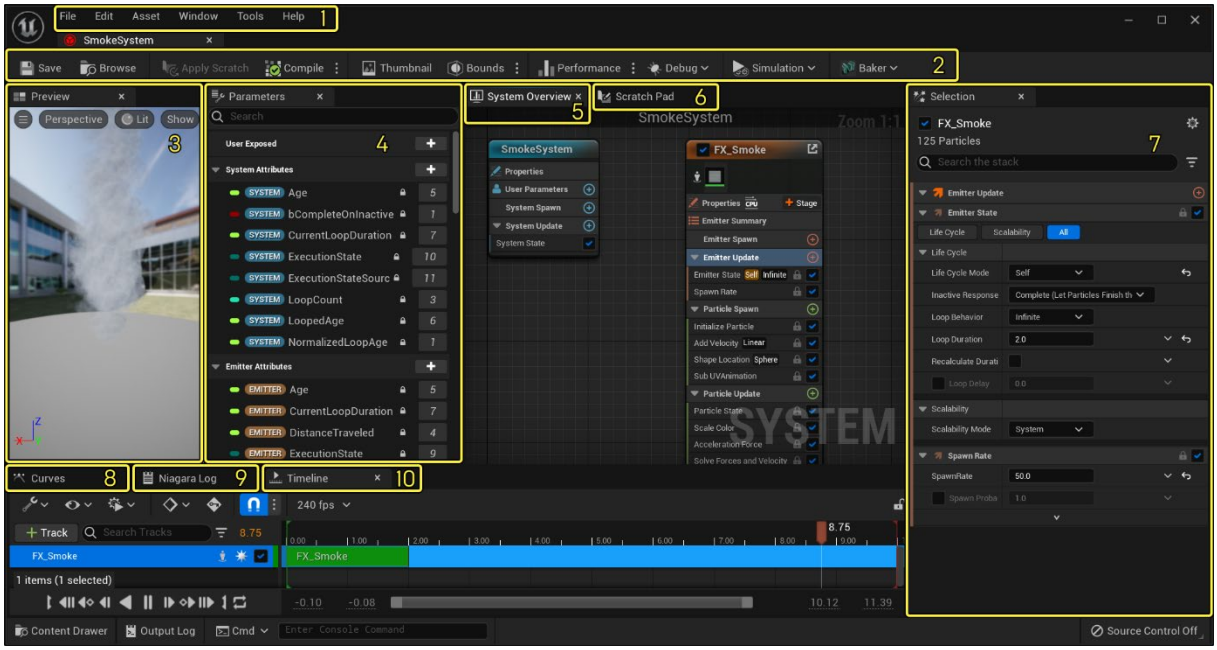

Figura 2.13: L'interfaccia del Niagara Editor

1. Menu Bar: La tradizionale barra dei menu, che offre le principali utilità per la gestione dei file e della finestra dell'editor.
2. Tool Bar: Barra delle funzioni utili per il sistema Niagara, con pulsanti per le operazioni di Compile, Debug e Simulation.
3. Preview Panel: Pannello che mostra l'anteprima del sistema Niagara attualmente in uso.
4. Parameters Panel: Questo pannello elenca i parametri utilizzati dagli emitter o dai sistemi attivi, consentendo di trascinarli nei nodi appropriati. Mostra quante volte sono referenziati e supporta la condivisione di dati tra emitter.
5. System Overview: Offre una panoramica generale del sistema o dell'emitter in modifica, combinando una vista grafica panoramica e versioni compatte dello stack. Facilita la navigazione tra diverse parti dei dati e fornisce una visione d'insieme quando si apre un emitter o sistema per la prima volta.
6. Scratch Pad Panel: Consente di creare moduli riutilizzabili o input dinamici locali, facilitando la progettazione e visualizzando immediatamente i risultati nell'emitter o sistema.
7. Selection Panel (the Stack): Permette di visualizzare i dettagli del nodo o modulo selezionato e di modificarne i valori.
8. Curves Panel: Offre un Curve Editor che permette agli utenti di modificare i valori che devono variare durante la durata di una particella o lungo l'intera vita di un emitter.
9. Niagara Log Panel: Mostra eventuali errori o avvisi di compilazione.
10. Timeline Panel: Consente di controllare il loop, il numero di ripetizioni, i burst, gli avvii e gli arresti casuali, e la frequenza di spawn. Questi elementi possono essere configurati e interagire direttamente all'interno della Timeline.

2.1.5.2 I nodi e i loro moduli

Il sistema Niagara in Unreal Engine è uno strumento avanzato per la creazione di effetti particellari, che consente di sviluppare, simulare e rendere visibili complessi effetti visivi in tempo reale. Nel pannello dei nodi dell'emitter (System Overview) e nel pannello di selezione (Selection Panel), i vari gruppi sono codificati a colori per facilitare la distinzione tra le diverse funzioni: l'arancione indica i moduli a livello di emitter, il verde rappresenta i moduli a livello di particella, il rosso è utilizzato per gli elementi di rendering e il blu identifica i gruppi relativi al sistema nel System Overview. Un sistema Niagara funge da contenitore per tutto ciò che è necessario per costruire un effetto, combinando vari blocchi costitutivi che si sovrappongono per creare l'effetto complessivo. È possibile modificare alcuni comportamenti a livello di

sistema che verranno poi applicati a tutto l'effetto, tramite il nodo di sistema. Invece, tramite il nodo emitter, si può modificare il comportamento delle particelle alla loro generazione, controllando come nascono le particelle, cosa accade durante la loro vita e come appaiono e si comportano.

L'emitter è organizzato in una pila (stack) suddivisa in vari gruppi. Il gruppo Emitter Spawn, codificato in arancione, definisce cosa succede quando un emitter viene creato per la prima volta sulla CPU, ed è utilizzato per impostazioni iniziali e predefinite. Il gruppo Emitter Update specifica i moduli a livello di emitter che si verificano ogni frame sulla CPU, utilizzato per generare particelle in ogni frame. A livello di particella, il gruppo Particle Spawn, codificato in verde, si attiva una volta per particella alla sua nascita, dove vengono definiti i dettagli di inizializzazione delle particelle come posizione, colore e dimensione. Il gruppo Particle Update viene chiamato per ogni particella in ogni frame e viene utilizzato per definire ciò che deve cambiare frame dopo frame mentre le particelle invecchiano, come il cambiamento del colore nel tempo o l'influenza di forze come gravità, rumore di curl o attrazione puntiforme. Il gruppo Render, codificato in rosso, definisce la visualizzazione delle particelle e configura uno o più renderizzatori per le particelle stesse. Ad esempio, si può utilizzare un renderizzatore di mesh per applicare un modello 3D o un renderizzatore di sprite per definire le particelle come sprite 2D (forma geometrica 2D).

Se si seleziona il nodo del sistema nel pannello di panoramica del sistema, i gruppi relativi al sistema saranno colorati in blu. Questi includono le impostazioni generali con i parametri utente e le proprietà del sistema, il System Spawn che definisce cosa accade quando il sistema viene creato, e il System Update che descrive gli eventi durante la vita del sistema.

Figura 2.14: Il nodo Emitter e di sistema

All'interno dei nodi, vi sono i moduli, che rappresentano i componenti essenziali per la creazione di effetti in Niagara. Questi elementi, quando aggiunti ai gruppi, formano uno stack che viene elaborato in ordine sequenziale, dall'alto verso il basso. Nel contesto degli emitter, un modulo può essere visto come un contenitore per l'esecuzione di calcoli matematici. In pratica, si immettono dei dati nel modulo, dove vengono elaborati, e successivamente si ottengono i dati modificati. I moduli in Niagara sono progettati utilizzando il linguaggio di shading ad alto livello (HLSL), ma possono essere creati anche visivamente tramite un grafico a nodi. Questo approccio consente di creare funzioni, includere input e scrivere su valori o mappe di parametri e inoltre, è possibile inserire codice HLSL direttamente nel grafico utilizzando il nodo CustomHLSL. Gli utenti possono esplorare i calcoli interni di qualsiasi modulo facendo doppio clic su di esso all'interno di un emitter in Niagara. Questo permette non solo di comprendere il flusso dei dati, ma anche di copiare e creare moduli personalizzati.

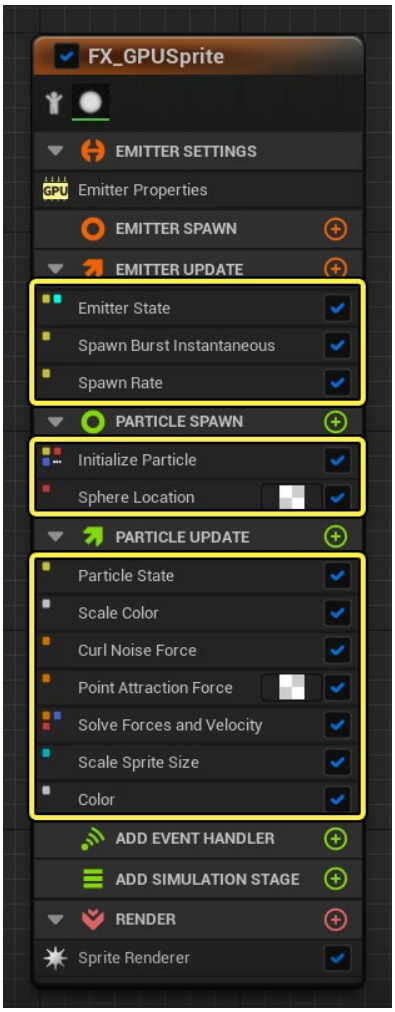

Figura 2.15: Il nodo Emitter con rilevanza sui moduli

2.1.6 Il Level Sequencer

Il Sequencer permette agli utenti di creare scene cinematografiche all'interno del gioco utilizzando un editor multitraccia specializzato. Attraverso la creazione di Level Sequences, l'aggiunta di tracce e la definizione di keyframe, è possibile controllare e animare oggetti, personaggi e telecamere. Il sistema Sequencer in Unreal Engine si basa su due componenti principali: il Level Sequence Asset e il Level Sequence Actor. Il Level Sequence Asset, un file situato nel Content Browser, contiene tutte le informazioni necessarie come tracce, telecamere, keyframe e animazioni.

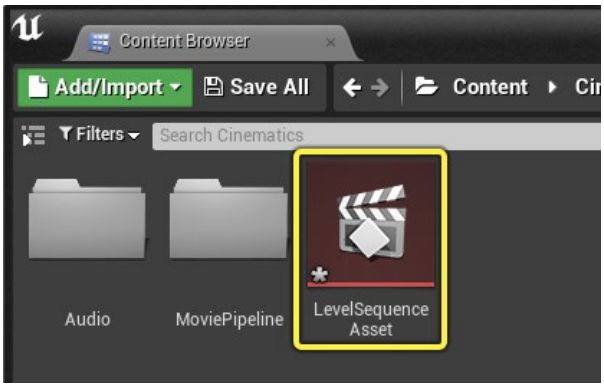

Figura 2.16: Level Sequencer Asset nel Content Browser

Questo asset viene associato a un Level Sequence Actor, che funge da contenitore all'interno del livello per i dati del Sequence Asset. Selezionando il Level Sequence Actor, è possibile visualizzare e modificare i suoi dettagli nel pannello Details, consentendo di regolare opzioni come la ripetizione in loop, l'avvio automatico e altre configurazioni avanzate.

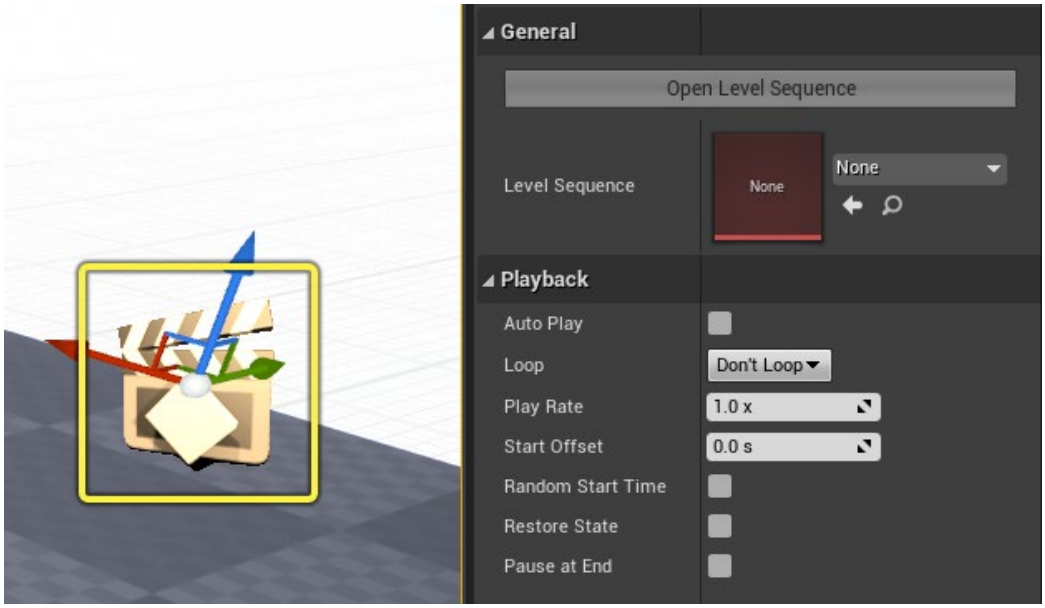

Figura 2.17: Il pannello Details del Level Sequence Actor

2.1.6.1 L'interfaccia grafica del Sequencer Editor

Il Sequencer Editor rappresenta l'interfaccia principale per l'editing dei Level Sequence assets, fondamentale per creare contenuti cinematografici in Unreal Engine. Partendo dalla sua interfaccia, spiegherò brevemente il suo funzionamento.

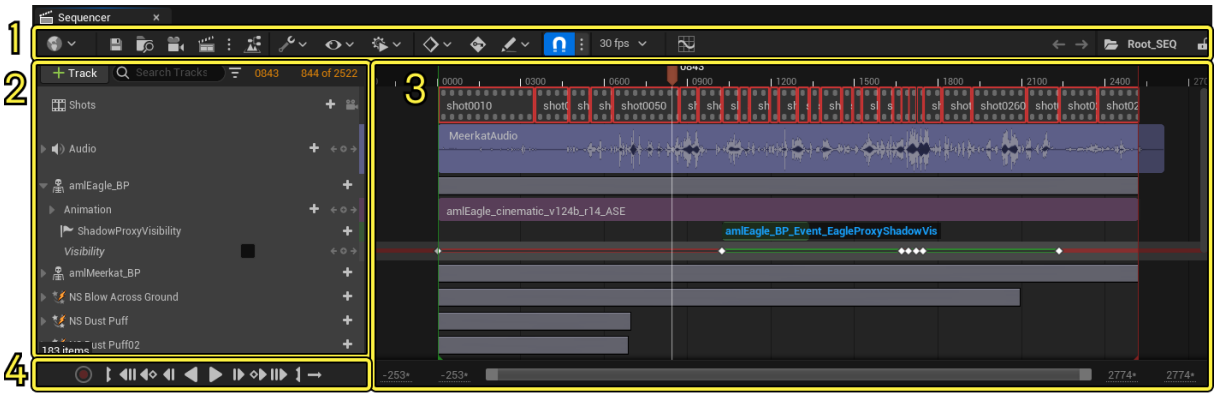

Figura 2.18: L'interfaccia del Level Sequence Editor

1. Toolbar: La Toolbar del Sequencer Editor offre una gamma di strumenti, opzioni e impostazioni per interagire con i Level Sequence assets. Di particolare rilevanza sono i pulsanti “Render”, che apre la finestra di dialogo delle impostazioni di rendering[34] del filmato o la Movie Render Queue se il plugin è abilitato, e “Curve Editor”, che apre l'Editor delle Curve utilizzato per la messa a punto dei keyframe[35] di animazione, delle tangenti e per la scelta del tipo di curva che interpola i vari keyframe.
2. Outliner: L'Outliner del Sequencer Editor contiene un elenco delle tracce “track” del Level Sequence asset, oltre a strumenti per aggiungere, filtrare e cercare tracce. Le tracce possono rappresentare attori associati al tuo Level Sequence, come telecamere, personaggi, audio ed effetti. Questo pannello facilita la gestione e l'organizzazione degli elementi presenti nella sequenza, ed è essenziale inserire il “track” della Camera per poter effettuare un rendering della scena.
3. Timeline: La Timeline del Sequencer è un ambiente di editing non lineare che rappresenta l'intera regione riproducibile del tuo Level Sequence asset. La Timeline include regioni orizzontali per ciascuna traccia e può contenere asset, keyframe e controlli della timeline. La gamma di riproduzione del tuo Level Sequence è delimitata

[34] Il termine “rendering” si riferisce al processo di generazione dell'immagine finale o del video a partire da una scena 3D. In Unreal Engine, quando si effettua il rendering, solo gli elementi visibili dalla camera impostata vengono processati e inclusi nell'output finale. Questo significa che oggetti, luci e dettagli fuori dal campo visivo della camera non vengono renderizzati, ottimizzando così le risorse e il tempo di calcolo. Il pulsante "Render" nel Sequencer apre le impostazioni per configurare il rendering del filmato, permettendo di definire parametri come risoluzione, frame rate e formato di output. Questo processo trasforma i dati della scena, comprese geometrie, texture, luci e ombre, in un'immagine o video completo e visibile, risultando in una rappresentazione visiva di alta qualità della scena vista dalla camera.

[35] I keyframe sono punti specifici nel tempo in cui vengono registrati valori particolari per le proprietà di un oggetto. In animazione, i keyframe definiscono gli stati iniziali e finali di un'animazione. Per esempio, se si vuole animare un personaggio che si muove da un punto A a un punto B, si impostano i keyframe all'inizio e alla fine del movimento. Il software di animazione calcolerà automaticamente i valori intermedi tra i keyframe per creare un movimento fluido. In sostanza, i keyframe permettono di controllare l'andamento delle animazioni definendo posizioni, rotazioni, scale e altre proprietà degli oggetti in momenti specifici del tempo.

dai marcatori di inizio (verde) e fine (rosso), mentre la posizione corrente di riproduzione è indicata dalla testina di riproduzione (Playhead).

4. Playback Controls: I controlli di riproduzione si trovano nell'angolo in basso a sinistra del Sequencer e funzionano in modo simile alle applicazioni standard di riproduzione multimediale. In questa sezione, si trovano i pulsanti per attivare, mettere in pausa e gestire altre funzioni relative alla riproduzione.

2.1.7 Altri strumenti in Unreal Engine

In questo capitolo si tratterà di alcuni altri strumenti che si sono rivelati essenziali per la realizzazione del progetto.

2.1.7.1 Quixel Bridge

La libreria Quixel Bridge rappresenta una risorsa integrata estremamente potente in Unreal Engine, progettata per semplificare l'importazione e l'uso di asset 3D di alta qualità nei progetti. Acquisita da Epic Games nel 2019, Quixel è nota per la sua vasta collezione di Megascans, una raccolta di asset 3D fotorealistici che include texture, modelli e materiali scansionati direttamente dal mondo reale. Quixel Bridge funge da collegamento tra questa libreria e Unreal Engine, consentendo agli sviluppatori di accedere facilmente a questi asset e integrarli nei loro progetti con pochi clic. Questo strumento offre quindi un'interfaccia intuitiva che permette di navigare, cercare e scaricare gli asset dalla libreria, tutti caratterizzati da una qualità fotorealistica eccezionale. La perfetta integrazione di Quixel Bridge con Unreal Engine è un altro dei suoi punti di forza principali. Gli asset scaricati possono essere importati direttamente nel progetto senza la necessità di conversioni manuali o complesse configurazioni, permettendo agli sviluppatori di risparmiare tempo prezioso e concentrarsi sulla parte creativa del loro lavoro. Inoltre, gli asset di Quixel Bridge sono forniti con materiali e shader preconfigurati, ottimizzati per Unreal Engine. Ciò significa che non solo appaiono realistici, ma sono anche progettati per garantire prestazioni ottimali, assicurando che i progetti funzionino senza intoppi anche con asset di alta qualità. Quixel Bridge viene inoltre costantemente aggiornato con nuovi asset, garantendo che gli sviluppatori abbiano sempre accesso alle risorse più recenti. La sincronizzazione continua tra Quixel Bridge e Unreal Engine permette agli asset di essere automaticamente aggiornati nel progetto quando vengono rilasciate nuove versioni.

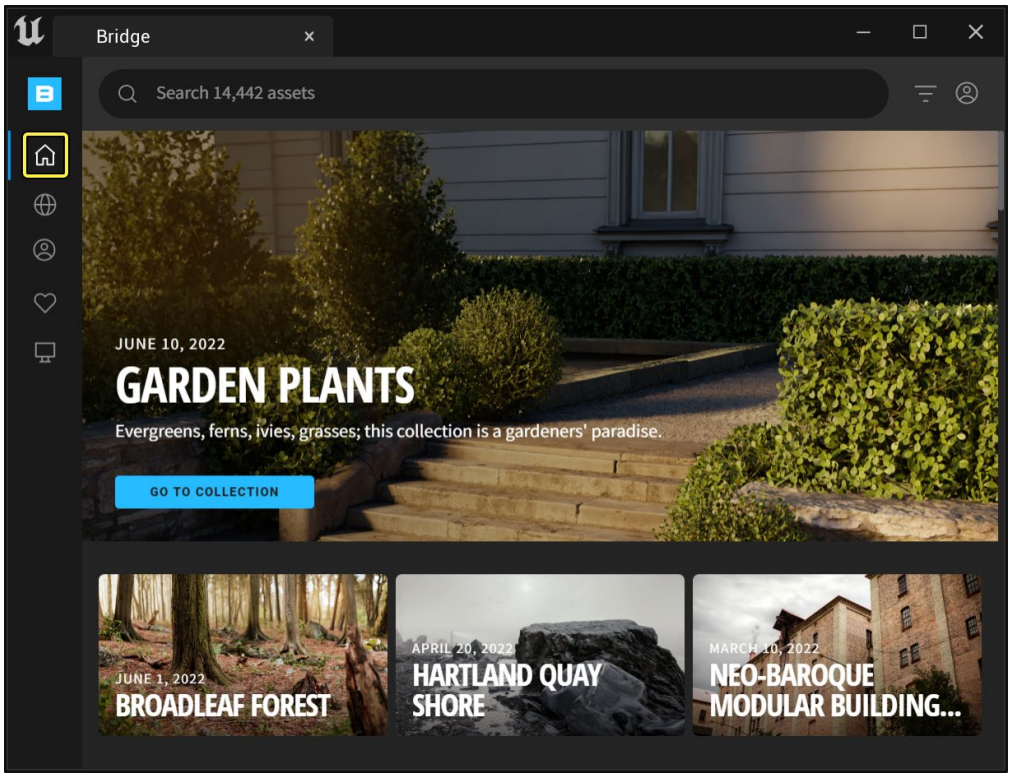

Figura 2.19: Interfaccia all'apertura di Quixel Bridge

2.1.7.2 Foliage Editor

Il sistema di Foliage di Unreal Engine è uno strumento potente che consente di creare ambienti naturali dettagliati e realistici. Il Foliage Editor, un modulo specifico all'interno del motore, permette agli sviluppatori di dipingere, modificare e gestire la vegetazione nei loro progetti. Questo strumento è essenziale per creare paesaggi immersivi e visivamente accattivanti, utilizzati in giochi, simulazioni e applicazioni architettoniche. In questa tesi, verrà spiegato brevemente il funzionamento generale del Foliage Editor, senza entrare nei dettagli tecnici, per illustrare come la vegetazione sia stata integrata nelle scene utilizzando asset della libreria Megascans.

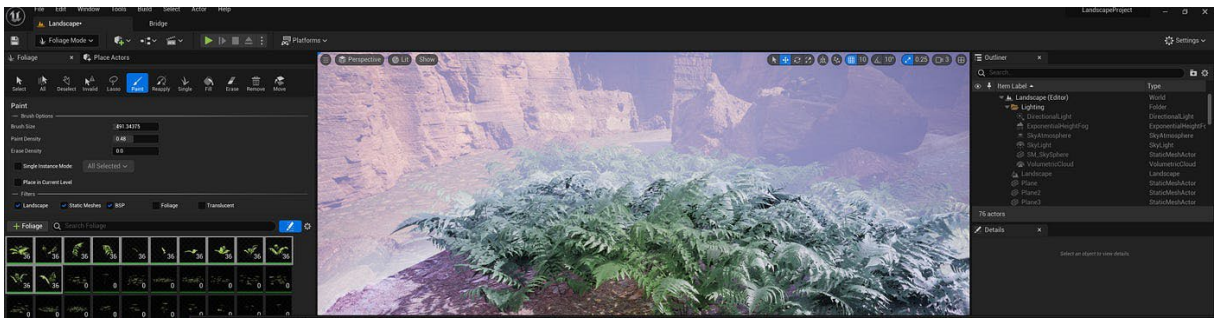

Figura 2.20: L'interfaccia del Foliage Editor

Il Foliage Editor offre la possibilità di dipingere vari tipi di vegetazione direttamente sul terreno, inclusi alberi, cespugli, erba e fiori. Selezionando uno o più asset di foliage, gli sviluppatori possono "dipingerli" sulla superficie del terreno con pochi clic, rendendo il processo di creazione di ambienti naturali rapido ed efficiente. Una delle caratteristiche più potenti di questo strumento è la capacità di distribuire gli oggetti in modo randomizzato. Definendo parametri come densità, scala e rotazione, il sistema distribuisce automaticamente gli oggetti in modo naturale e realistico, evitando che l'ambiente sembri artificiale. Il Foliage Editor include anche strumenti per la gestione delle risorse di vegetazione, permettendo di aggiungere, rimuovere o modificare facilmente gli asset, mantenendo il controllo completo sull'aspetto dell'ambiente. L'uso del Foliage Editor offre numerosi vantaggi: permette la creazione rapida di grandi aree di vegetazione, risultando particolarmente utile per progetti che richiedono ambienti naturali estesi, come giochi open world o simulazioni ambientali; in più ha la capacità di distribuire e personalizzare la vegetazione in modo dettagliato, il che consente di ottenere un alto livello di realismo visivo, migliorando l'immersione del giocatore. Gli strumenti intuitivi e le opzioni di personalizzazione avanzate semplificano il processo di aggiunta di vegetazione, riducendo il tempo necessario per creare paesaggi complessi e migliorando l'efficienza del flusso di lavoro. Gli sviluppatori hanno un controllo dettagliato su ogni aspetto della vegetazione, inclusi i parametri di crescita, la densità e la distribuzione, garantendo che l'ambiente naturale possa essere modellato esattamente secondo le esigenze del progetto.

2.1.7.3 Water System

Per creare l'acqua del Nilo è stato utilizzato il plugin "Water" per Unreal Engine, il plugin ufficiale gratuito che consente di inserire l'acqua nelle scene con numerose possibilità di personalizzazione. Sebbene vi fossero alternative, come altri plugin a pagamento creati da utenti o la creazione manuale del materiale dell'acqua, queste opzioni erano meno convenienti sia per i costi elevati sia per i risultati meno soddisfacenti e i tempi di sviluppo più lunghi.

Il sistema Water di Unreal Engine permette di aggiungere vari tipi di corpi d'acqua, tra cui fiumi, laghi e oceani. Tuttavia, nel contesto di questa tesi, ci concentreremo solo sugli aspetti rilevanti per la creazione del progetto specifico. Dato l'obiettivo di ricreare il fiume Nilo, i WaterBody Lake e WaterBody Ocean sono stati esclusi. La scelta è ricaduta quindi tra il WaterBody River e il WaterBody Custom: il primo è specificamente progettato per i fiumi e consente di modellare il corso d'acqua tramite una curva direttamente nell'editor e questo facilita

la personalizzazione della forma del fiume. Al contrario, il WaterBody Custom è un rettangolo di base, ma offre un'ampia possibilità di personalizzazione, come l'aggiunta di onde.

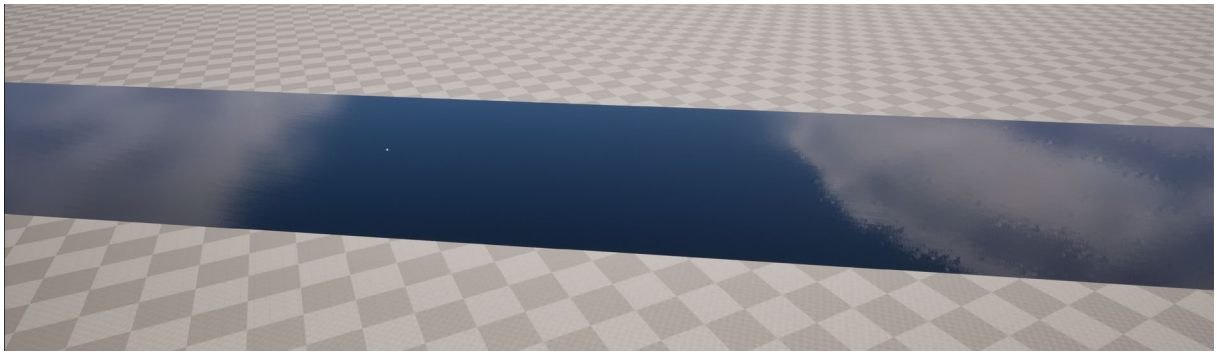

Figura 2.21: Il Water Body Custom

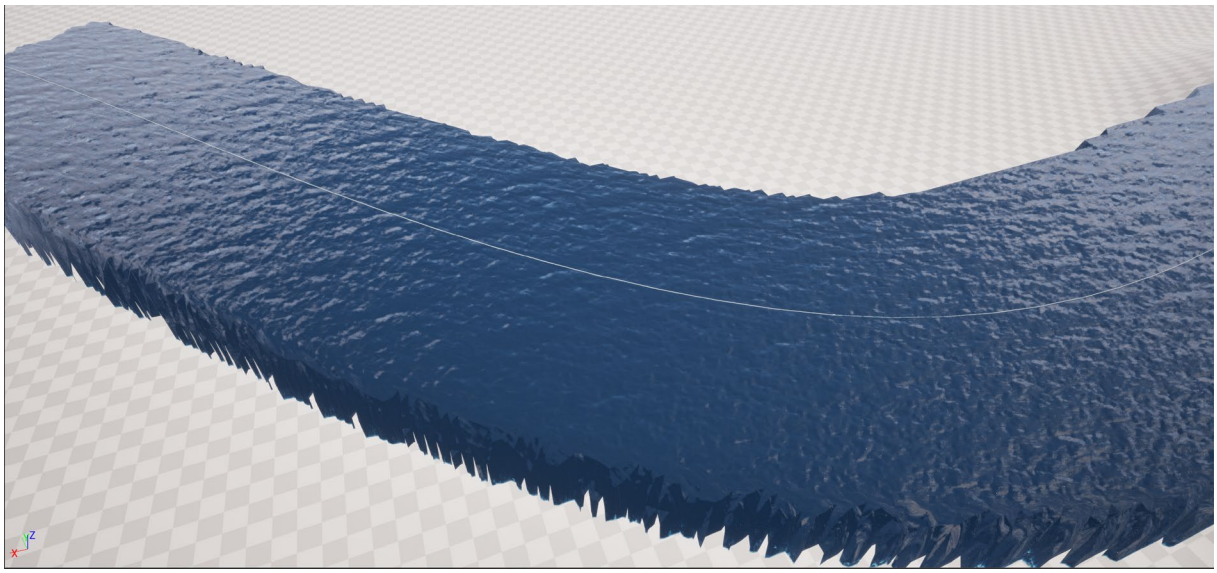

Figura 2.22: Il Water Body River

La decisione finale di utilizzare il WaterBody Custom è stata determinata dalla necessità di animare il movimento dell'acqua per simulare le piene del Nilo, funzione non supportata dal WaterBody River. La possibilità di variare il livello dell'acqua è cruciale per rappresentare accuratamente le caratteristiche del fiume Nilo durante le inondazioni stagionali. Per superare la limitazione del WaterBody Custom, che non permette la creazione di fiumi curvilinei, è stato sviluppato un sistema tramite Blueprint che ha aggiunto questa funzionalità. Anche se ciò ha comportato la perdita della possibilità di utilizzare le onde predefinite, queste sono state ricreate manualmente modificando il materiale dell'acqua. Inoltre, è stata implementata una funzione per gestire i “ripple”, ovvero le increspature che si formano quando l'acqua entra in contatto con una superficie. Molti parametri di base sono stati modificati per ottenere l'aspetto desiderato dell'acqua, rendendola il più simile possibile al fiume Nilo.

2.1.7.4 Plugin Unreal: UltraDynamicSky

Quest'ultimo elemento non è necessario per creare fotogrammetrie, ma è stato fondamentale per la realizzazione della scena su Unreal, quindi è importante menzionarlo. Per ottenere un cielo realistico e una transizione giorno/notte con colori accurati, è stato scaricato e utilizzato il plugin UltraDynamicSky. Dopo il download, si ottiene una cartella da inserire nel proprio progetto, all'interno della quale, nella sottocartella Blueprint, si trovano due blueprint: uno per il cielo e l'altro per le precipitazioni meteorologiche, che possono essere inseriti nella scena. UltraDynamicSky è un plugin estremamente potente per la realizzazione di cieli diurni e notturni, e per la creazione dell'atmosfera in generale, poiché permette anche di gestire elementi come nebbia e precipitazioni. Tramite il pannello Details, esso offre numerose impostazioni per una completa personalizzazione, tra cui la possibilità di impostare il cielo inserendo le coordinate globali e la data, permettendo così di avere le durate giorno/notte e l'orientamento del cielo perfettamente allineati con la realtà. Grazie a queste funzionalità avanzate, il plugin contribuisce in modo significativo alla resa fotorealistica della scena su Unreal, rendendo l'ambiente virtuale più immersivo e credibile.

Tutti questi aspetti aggiuntivi verranno trattati nel dettaglio nel prossimo capitolo, dove si discuterà la creazione vera e propria delle scene. Questo permetterà di comprendere meglio come le personalizzazioni e le funzionalità aggiunte abbiano contribuito a ricreare un ambiente realistico e immersivo, fondamentale per il successo del progetto.

2.2 Le acque e le rocce del Nilo su Unreal Engine

In questo capitolo verrà descritto in maniera dettagliata il processo di realizzazione delle scene riguardanti il Nilo. In particolare, sono state create due scene principali, ciascuna con due inquadrature diverse, al fine di offrire una maggiore varietà di scelta per il video finale. Entrambe le scene si concentrano sulle piene del Nilo, rappresentate come una sorta di timelapse per evidenziare gli effetti del fiume sulle rocce nel corso dei secoli. Inoltre, durante l'animazione, le rocce subiscono un'erosione visibile e un cambiamento di colore, diventando progressivamente più scure.

Le differenze principali tra le due scene risiedono nella posizione della camera. Nella prima scena, la camera è posizionata a livello dell'acqua, permettendo una visione ravvicinata delle rocce lungo il corso del Nilo. Questa inquadratura consente di osservare con precisione il processo di erosione durante le piene, quando l'acqua sommerge le rocce e la camera, offrendo un dettaglio visivo intenso e ravvicinato. Nella seconda scena, invece, la camera è posizionata

in alto, molto sopra il livello dell'acqua, offrendo una panoramica più ampia del paesaggio del Nilo visto dall'alto. Questa inquadratura permette di apprezzare la vastità del fiume e del suo ambiente circostante, pur continuando a mostrare l'erosione delle rocce e il loro cambiamento di colore, anche se in secondo piano rispetto alla vista complessiva del paesaggio.

2.2.1 Spline Blueprint per la forma dell'acqua

Come anticipato nel capitolo precedente, il principale limite del WaterBody Custom è la sua forma rettangolare. Per ovviare a questo problema, è stato sviluppato un Blueprint che crea un WaterBody Custom deformato tramite una curva Spline. Una spline è una curva matematica definita da una serie di punti di controllo: la curva passa attraverso o vicino a questi punti, e la loro posizione determina la forma complessiva della spline. È possibile aggiungere o rimuovere punti di controllo e ciascuno di essi può essere traslato, ruotato o scalato indipendentemente dagli altri, per ottenere la forma desiderata.

Nella seguente immagine sono messe a confronto due versioni dell'acqua: a sinistra, “My_Water_Blueprint”, creata utilizzando la spline con tre punti di controllo, e a destra, la versione base del WaterBody Custom, di forma rettangolare.

Figura 2.23: Il "My_Water_Blueprint" e il WaterBody Custom

Il primo passo è stato creare un nuovo blueprint, denominato “My_Water_Blueprint”, e aggiungere tra i componenti del blueprint il componente “Spline”. La modifica principale si concentra nel Construction Script, poiché le operazioni di creazione della spline e quindi dell'acqua devono essere eseguite immediatamente, durante la fase di creazione dell'asset. All'interno del construction graph è stata inserita tutta la logica necessaria per ottenere il risultato voluto. Il concetto fondamentale di questo blueprint è quello di aggiungere correttamente una mesh, in questo caso un piano, per ogni coppia consecutiva di punti della

spline, in modo che la curva non sia solo una linea, ma venga rappresentata come un oggetto visibile. Lo script inizia con il nodo di base, il “Construction Script”, che ha solo il pin di esecuzione di output. Questo si collega a un nodo “For Loop”, che permette di iterare sull'intera spline.

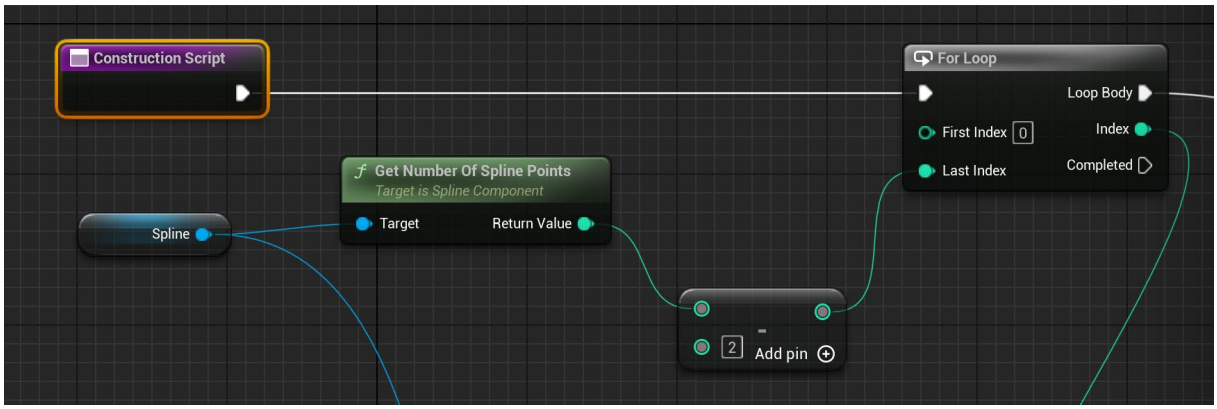

Figura 2.24: Script Blueprint

Il nodo “Spline” rappresenta il riferimento al componente spline aggiunto nel blueprint; esso serve come input per vari nodi, ma in questa fase viene utilizzato per determinare il numero di iterazioni del loop, ovvero quanti punti di controllo ha la spline. Come si può notare dall'immagine del nodo, il “For Loop” richiede come input l'indice iniziale e finale su cui iterare. L'indice iniziale è stato impostato a 0, come consuetudine in informatica per indicare le posizioni nei vettori. L'indice finale è il numero di punti della spline (ottenuto dal nodo “Get Number of Spline Points”) meno due. Questa riduzione di due è dovuta a due fattori: primo, l'indicizzazione che parte da zero, per cui se una curva ha tre punti di controllo, il nodo restituirà 3; tuttavia, poiché il ciclo parte da zero, impostare l'indice finale a 3 farebbe iterare quattro volte (indici 0-1-2-3), causando un errore. Secondo, la logica successiva del blueprint lavora partendo da un indice e quello successivo, inserendo in mezzo la mesh desiderata; perciò, l'ultimo punto di controllo non deve essere iterato, altrimenti si tenterebbe di aggiungere un elemento tra l'ultimo punto di controllo e uno successivo inesistente.

Dopo il nodo di loop, un elemento cruciale è rappresentato dal nodo “Add Spline Mesh Component”, il quale aggiunge una mesh alla spline. Questo nodo non solo inserisce la mesh e il materiale sulla spline, ma restituisce anche un riferimento alla mesh, permettendone la modifica in termini di posizione, rotazione e altre proprietà. Successivamente vi è semplicemente un nodo che imposta la direzione con cui la mesh viene applicata, in questo caso l’asse x.

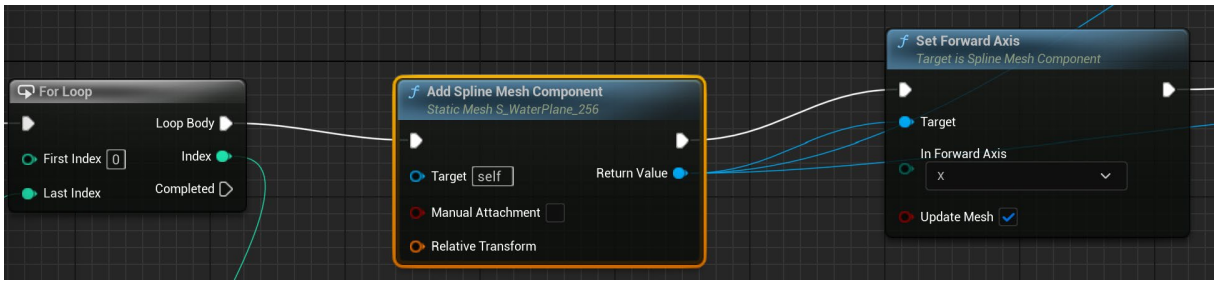

Figura 2.25: Script Blueprint

Invece, cliccando sul nodo di aggiunta della mesh e accedendo al pannello Details, è possibile impostare varie proprietà; quelle di maggiore rilevanza per il nostro scopo sono “Static Mesh” e “Element 0” nella sezione Materials.

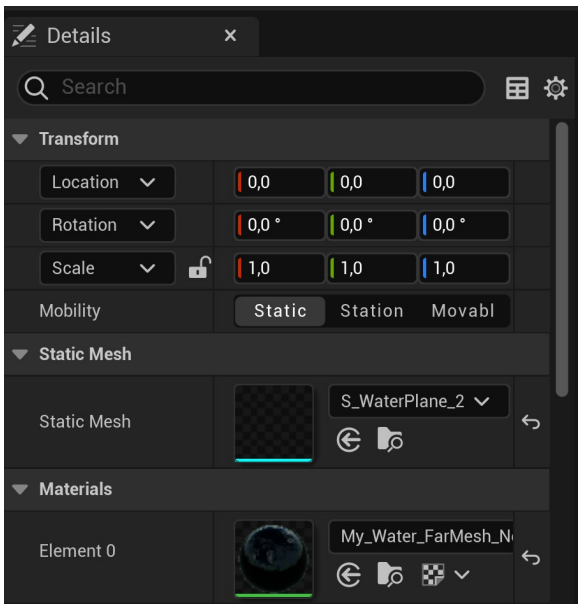

Figura 2.26: Pannello Details del nodo “Add Spline Mesh Component”

Per quanto riguarda la proprietà “Static Mesh”, è necessario scegliere quale mesh aggiungere alla spline. In questo caso, è stata selezionata la mesh “S_WaterPlane_256”, che è la stessa utilizzata dal WaterBody Custom. A prima vista può sembrare un piano normale, ma in realtà è composto da circa 4000 vertici e questo è un aspetto fondamentale: se fosse composto da soli 4 vertici, non potrebbe essere deformato dalle curve della spline, poiché le rette tra due vertici non sono deformabili. Anche questo piano segue lo stesso principio, ma grazie all'alto numero di vertici, la curvatura visibile è formata da molte rette orientate diversamente da vertice a vertice, simulando una curva quando vista da lontano. Per quanto riguarda la proprietà “Element 0”, si sceglie il materiale che la mesh deve avere. In questo caso, è stato selezionato il Material Instance “My_Water_FarMesh_NoRiver” derivato dal Water Material “My_Water_Material_noRiver”, appositamente modificato per lo scopo e che sarà argomento

del prossimo capitolo di questa tesi. Tuttavia, è possibile utilizzare qualsiasi materiale, incluso il materiale dell'acqua fornito dal plugin.

Successivamente, è necessario impostare le caratteristiche spaziali della mesh appena aggiunta, in particolare la posizione (data dalle coordinate x, y, z), la rotazione e lo scalamento. Queste dipendono dai punti di controllo della spline, poiché la mesh deve iniziare dal primo punto di controllo e terminare in quello successivo, seguendo lo stesso principio per rotazione e scalamento. Prima verranno analizzati brevemente i nodi che impostano queste proprietà, seguiti dalla procedura per ottenere i valori di input per questi nodi. In primo luogo, le proprietà riguardanti la posizione e la rotazione iniziali e finali vengono impostate tramite il nodo “Set Start and End”, che riceve in input la posizione e la tangente (che indica la rotazione) iniziali e finali. Successivamente, la proprietà di scalamento iniziale e finale viene impostata attraverso i nodi “Set Start Scale” e “Set End Scale”, che ricevono rispettivamente lo scalamento iniziale e finale. Infine, il nodo “Set Relative Rotation” viene utilizzato per evitare possibili errori e fissare a zero le rotazioni relative della mesh. Tutti questi nodi ricevono in input il riferimento alla mesh corrente (in azzurro), fornito in output dal nodo precedente “Add Spline Mesh Component”.

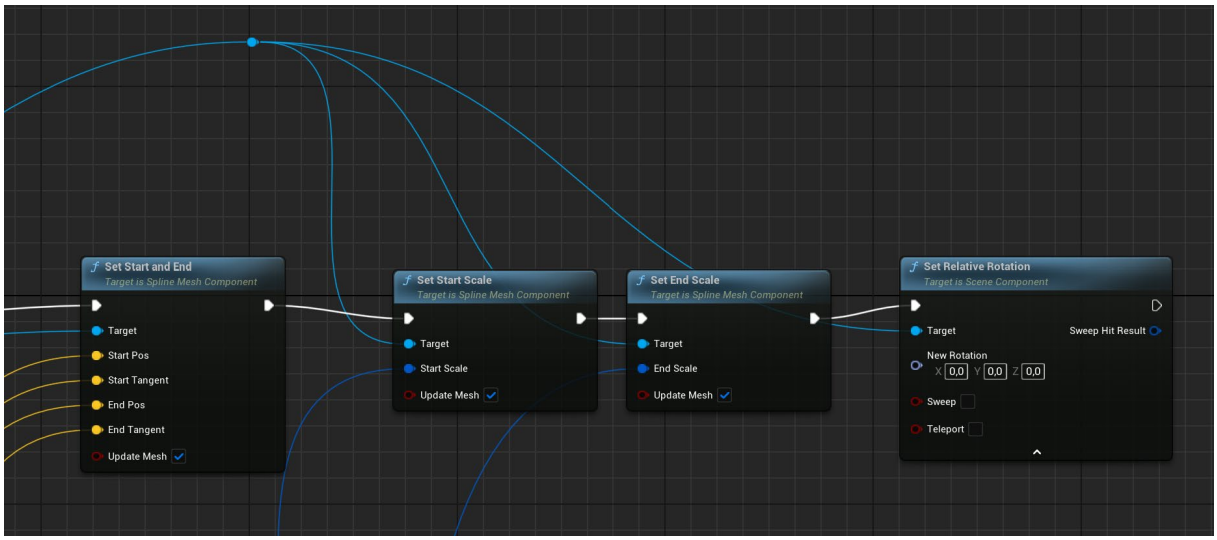

Figura 2.27: Script Blueprint

Per ottenere tutti gli input necessari ai nodi precedenti, è stato necessario estrarre le informazioni dai vari punti di controllo della spline. Ad ogni ciclo del loop verrà analizzata una coppia diversa di punti di controllo, e tramite nodi specifici verranno ricavate le informazioni di posizione (“Location”), rotazione (“Tangent”) e scalamento (“Scale”). Questi nodi richiedono come input la spline di interesse e l’indice del punto di controllo da cui ricavare le informazioni, che in questo caso corrisponde all’indice dato dal loop per il punto di controllo

attuale e all'indice aumentato di uno per quello successivo. È qui che diventa evidente il motivo per cui all'inizio si è scelto di diminuire di due l'indice finale nel ciclo loop. I nodi utilizzati per ottenere queste informazioni sono "Get Location at Spline Point", "Get Tangent at Spline Point" e "Get Scale at Spline Point", ripetuti due volte per gli indici iniziale e finale. Un aspetto particolare nell'utilizzo di questi nodi riguarda l'operazione eseguita con l'output di "Get Scale at Spline Point": poiché i nodi "Set Start/End Scale" richiedono uno scalamento composto da due valori (x e y) e non da tre (x, y e z), è stato necessario rimappare lo scalamento 3D dei punti di controllo in un vettore 2D, utilizzando solo i valori y e z. Questo perché lo scalamento della mesh piana deriva solo dalle componenti y e z dei punti di controllo, ignorando la componente x.

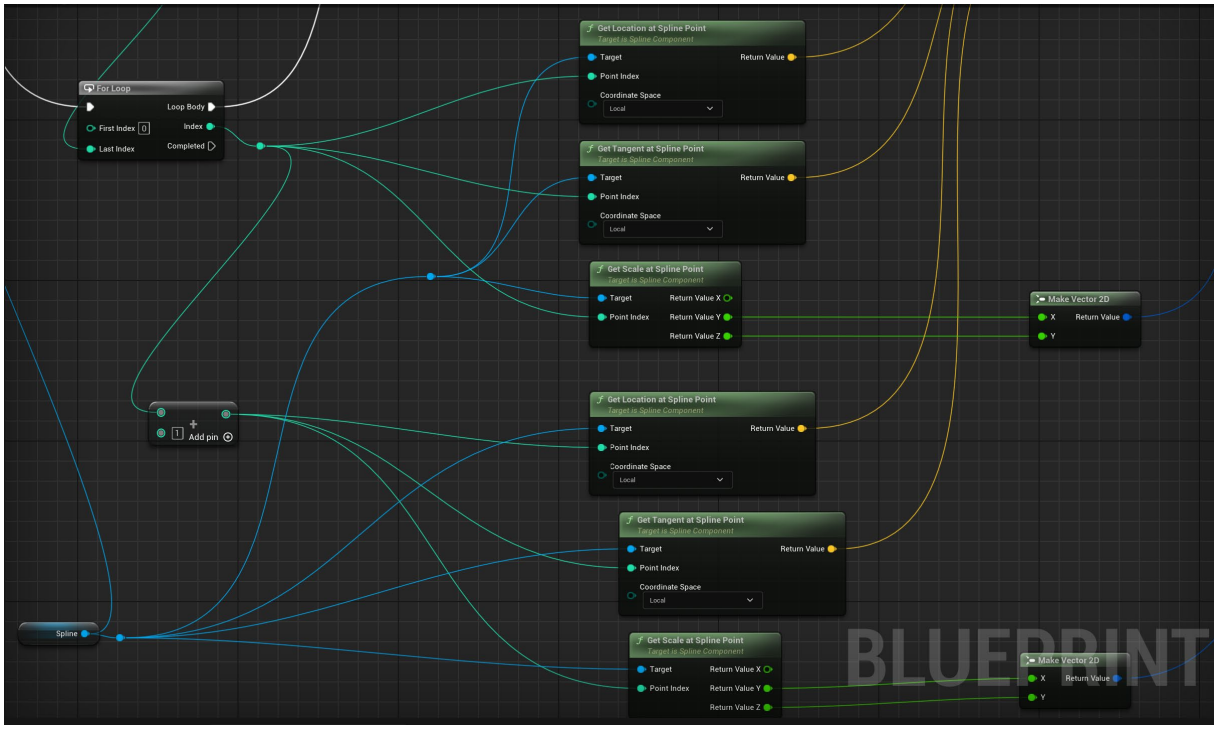

Figura 2.28: Script Blueprint

Il Blueprint è ora completo e pronto per l'uso: inserendolo nella scena, sarà possibile visualizzare l'acqua e aggiungere o modificare i punti della spline per creare il corso d'acqua desiderato.

2.2.2 Water Material

In questo sottocapitolo si analizzeranno le modifiche apportate al materiale dell'acqua, con l'obiettivo di ottenere i risultati desiderati. Il "Water Material" di Unreal Engine è estremamente complesso e articolato, rendendo impraticabile un'analisi completa. All'apertura, il numero di

nodi potrebbe non sembrare eccessivo, ma molti di essi contengono funzioni che a loro volta includono altri grafi di nodi, creando una struttura altamente complessa.

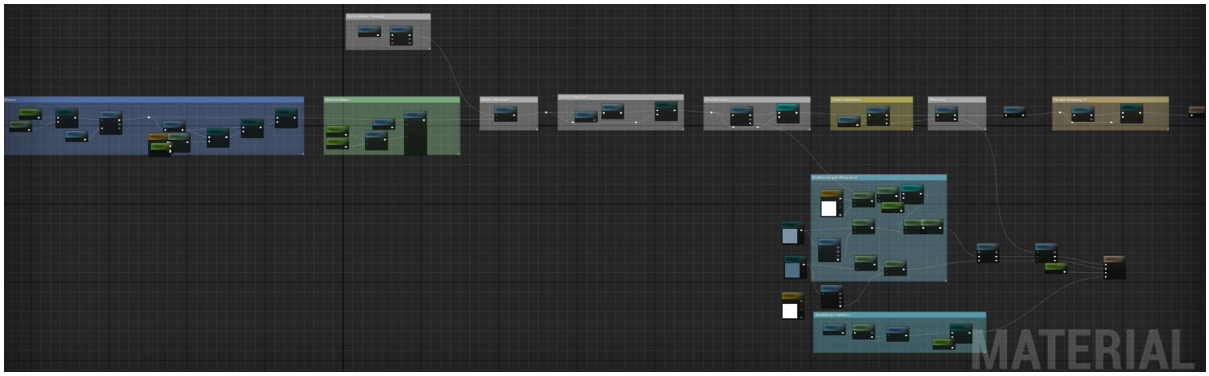

Figura 2.29: Il Water Material

Due sono le principali ragioni che hanno reso necessarie queste modifiche al materiale dell'acqua. La prima riguarda la reintroduzione delle onde, che avevano smesso di funzionare a causa del sistema di blueprint implementato e questo ha richiesto un intervento manuale per ripristinare l'effetto delle onde. La seconda motivazione è stata la volontà di rendere la scena più realistica attraverso l'introduzione dei "ripples", cioè le increspature che si formano quando l'acqua entra in contatto con una superficie.

2.2.2.1 Waves

In questa sezione verrà trattata in dettaglio l'implementazione delle onde per simulare l'acqua in Unreal Engine. Il processo sarà diviso in due parti principali: inizialmente verranno analizzate le Gerstner Waves dal punto di vista teorico-matematico, approfondendo il loro funzionamento e le equazioni che le descrivono; successivamente, si esaminerà l'implementazione pratica di queste onde in Unreal Engine, illustrando come una funzione specifica sia stata utilizzata per creare onde realistiche e come questa sia stata integrata nel materiale dell'acqua per ottenere un risultato visivamente accurato.

Le onde di Gerstner sono un modello matematico avanzato utilizzato per simulare il movimento realistico delle onde sulla superficie dell'acqua. Questo modello è particolarmente utile nella grafica computerizzata e nelle simulazioni fisiche, poiché rappresenta il movimento circolare delle particelle d'acqua, offrendo una rappresentazione più accurata rispetto alle semplici onde sinusoidali.

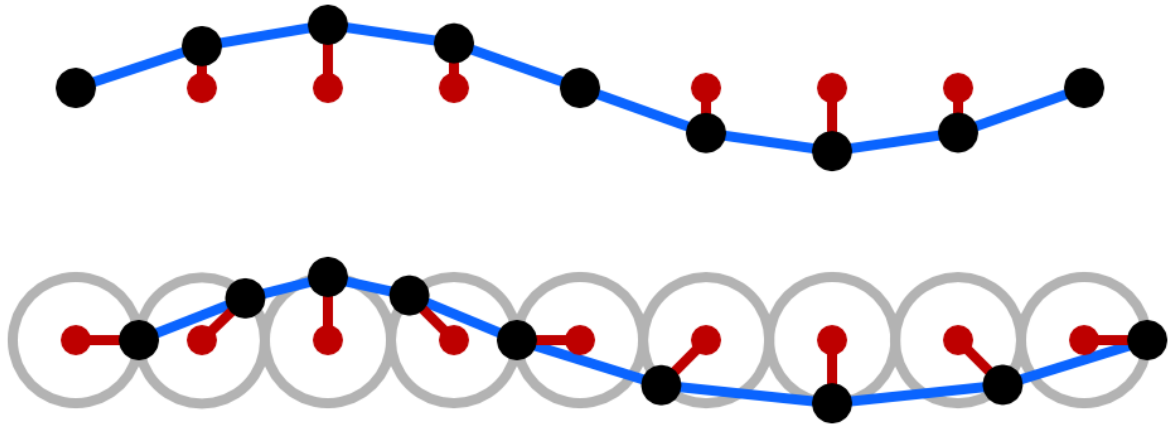

Figura 2.30: Onda sinusoidale e onda di Gerstner

Le onde di Gerstner sono basate su un modello di onde trocoidali, in cui le particelle d'acqua si muovono lungo percorsi circolari o ellittici. Questo movimento è descritto dalle seguenti equazioni parametriche per le coordinate orizzontali (x), verticali (y) e la posizione verticale (z) della superficie dell'acqua:

$$x = x_0 + A * \cos(k * x_0 - \omega * t) \qquad z = z_0 + A * \sin(k * x_0 - \omega * t)$$

dove:

- x_0 è la posizione orizzontale iniziale della particella.
- z_0 è la posizione verticale iniziale della particella (livello medio dell'acqua).
- A è l'ampiezza dell'onda.
- k è il numero d'onda (legato alla lunghezza d'onda λ da $k = \frac{2\pi}{\lambda}$).
- ω è la frequenza angolare (legata al periodo dell'onda T da $\omega = \frac{2\pi}{T}$).
- t è il tempo.

Le particelle d'acqua si muovono in percorsi circolari con un raggio proporzionale all'ampiezza dell'onda. La velocità di propagazione dell'onda è data dalla relazione tra la lunghezza d'onda e il periodo:

$$c = \frac{\lambda}{T}$$

Per simulare superfici d'acqua più complesse e realistiche, è possibile combinare più onde di Gerstner con diverse ampiezze, lunghezze d'onda e direzioni di propagazione. L'altezza finale della superficie dell'acqua in un punto specifico è la somma delle altezze di tutte le onde in quel punto:

$$z_{tot}(x,t) = \sum_{i=1}^{N} A_i * \sin(k_i * x - \omega_i * t)$$

dove *N* è il numero totale di onde combinate, e ogni onda *i* ha una propria ampiezza A_i , numero d'onda k_i, e frequenza ω_i.

Quando si modifica la superficie dell'acqua utilizzando onde di Gerstner, è necessario aggiornare anche i vettori normali della superficie per riflettere le nuove pendenze. I vettori tangenti devono essere calcolati per determinare la normale corretta in ogni punto della superficie. Il vettore tangente T viene calcolato come:

$$T = [1 - k * A \sin(k * x_0 - \omega * t)]$$

Il vettore normale *N* è quindi ottenuto come:

$$N = (-T_y, T_x, 0)$$

Questo assicura che la normale della superficie sia aggiornata in base alla nuova geometria creata dalle onde di Gerstner. Un problema che può sorgere nell'implementazione delle onde di Gerstner è la formazione di loop o sovrapposizioni nella superficie dell'acqua quando l'ampiezza delle onde è troppo grande rispetto alla lunghezza d'onda. Questo si verifica quando l'ampiezza è talmente elevata da far sì che le particelle d'acqua superino il livello medio dell'acqua, creando anomalie visive. Per prevenire queste anomalie, è necessario assicurarsi che il rapporto tra l'ampiezza e la lunghezza d'onda sia appropriato. Questo può essere controllato matematicamente attraverso l'uso di parametri di ripidità normalizzati:

$$a = \frac{steepness}{k * A}$$

dove a rappresenta la ripidità normalizzata dell'onda.

Fatta questa breve analisi matematica, qui sotto è riportato il codice per implementare le onde di Gerstner in Unreal Engine, con commenti numerati che corrispondono all'elenco puntato successivo per facilitare la comprensione:

```
// 1. Inizializzazione delle coordinate per la posizione della superficie
float2 xy = float2(0.0f, 0.0f);
float z = 0.0f;

// 2. Inizializzazione delle coordinate per la normale
float2 nxy = float2(0.0f, 0.0f);
float nz = 0.0f;

// 3. Numeri primi arbitrari per generare direzioni casuali
int randx = 10007;
int randy = 802709;

// 4. Calcolo numero massimo di onde
float waves = min(wavecountlimit, wavecount);


// 5. Ciclo attraverso il numero massimo consentito di onde
for (int i = 0; i < waves; i++)
{
    // 5.1 Calcolo dell'indice normalizzato dell'onda
    const float thisIndexFrac = (float)i / (float)wavecount;

    // 5.2 Generazione di una direzione pseudo-casuale per l'onda
    randx = (randx * 1103515245) + 12345;
    randy = (randy * 1103515245) + 12345;
    float2 direction = float2(cos((float)randx / 801571.f), sin((float)randy / 10223.f));
    direction = normalize(lerp(winddirection, (direction * 2.f) - 1.f, spread));

    // 5.3 Calcolo dei parametri dell'onda in base all'indice normalizzato
    const float thisAlpha = pow(thisIndexFrac, wavedistribution);
    const float wavelength = lerp(maxwavelength, minwavelength, thisAlpha);
    const float steepness = lerp(maxsteepness, minsteepness, thisAlpha);
    const float amplitude = lerp(maxamplitude, minamplitude, thisAlpha);

    // 5.4 Calcolo della dispersione e della velocità dell'onda
    float dispersion = 2.f * PI / wavelength;
    float wavespeed = sqrt(dispersion * 981.f);
    float2 wavevector = direction * dispersion;
    float wavetime = wavespeed * time;

    // 5.5 Calcolo della posizione dell'onda
    float wavepos = dot(float2(position.x, position.y), wavevector) - wavetime;
    float wavesin = sin(wavepos);
    float wavecos = cos(wavepos);

    float wKA = amplitude * dispersion;
    float q = steepness / wKA;

    // 5.6 Calcolo delle normali
    nxy += wavesin * wKA * direction;
    nz += wavecos * steepness * saturate((amplitude * 50.f) / wavelength);

    // 5.7 Calcolo della posizione della superficie
    xy += -q * wavesin * direction * amplitude;
    z += wavecos * amplitude;

}

// 6. Normalizzazione delle normali
Normal = normalize(float3(nxy.x, nxy.y, 1.f - nz));

// 7. Restituzione della posizione finale della superficie
return float3(xy.x, xy.y, z);
```

1. Inizializzazione delle Coordinate per la Posizione della Superficie: le variabili xy e z sono inizializzate a zero; queste variabili rappresentano le coordinate orizzontali e verticali della superficie dell'acqua. La posizione della superficie dell'acqua verrà calcolata iterativamente durante il ciclo delle onde.
2. Inizializzazione delle Coordinate per la Normale: le variabili nxy e nz sono inizializzate a zero e queste variabili rappresentano le coordinate delle normali della superficie dell'acqua. Le normali saranno calcolate in base alla direzione e all'ampiezza delle onde di Gerstner.
3. Generazione di Direzioni Casuali: randx e randy sono numeri primi arbitrari utilizzati per generare direzioni pseudo-casuali per le onde. Questi valori vengono aggiornati iterativamente per garantire che ogni onda abbia una direzione unica.
4. Calcolo del numero massimo di onde: per inizializzare la variabile waves, e capire quanti cicli fare, viene preso il valore minimo tra il numero di onde richiesto e il numero massimo di onde possibili (wavecount o wavecountlimit).
5. Ciclo delle Onde:

   5.1 Calcolo dell'Indice Normalizzato dell'Onda: thisIndexFrac rappresenta l'indice normalizzato dell'onda corrente, calcolato come la frazione dell'indice dell'onda rispetto al numero totale di onde.

   5.2 Generazione di una Direzione Pseudo-Casuale: vengono generate direzioni pseudo-casuali utilizzando randx e randy, che vengono trasformate in angoli trigonometrici e normalizzate.

   5.3 Calcolo dei Parametri dell'Onda: thisAlpha determina i parametri dell'onda in base alla sua distribuzione. Vengono calcolate la lunghezza d'onda (wavelength), la ripidità (steepness) e l'ampiezza (amplitude).

   5.4 Calcolo della Dispersione e della Velocità dell'Onda: la dispersione (dispersion) e la velocità dell'onda (wavespeed) vengono calcolate utilizzando le formule delle onde di Gerstner.

   5.5 Calcolo della Posizione dell'Onda: wavepos, wavesin e wavecos calcolano rispettivamente la posizione dell'onda, il seno e il coseno della posizione dell'onda.

   5.6 Calcolo delle Normali: nxy e nz aggiornano le coordinate delle normali in base alla direzione e all'ampiezza dell'onda.

   5.7 Calcolo della Posizione della Superficie: xy e z aggiornano la posizione della superficie dell'acqua utilizzando le equazioni delle onde di Gerstner.

5.8 Calcolo della Maschera: mz calcola una maschera basata sulle onde, che potrebbe essere utilizzata per effetti visivi aggiuntivi.

6. Normalizzazione delle Normali: il vettore normale della superficie viene calcolato e normalizzato per garantire che abbia una lunghezza unitaria.
7. Restituzione della Posizione Finale della Superficie: viene restituita la posizione finale della superficie dell'acqua, calcolata come somma delle influenze di tutte le onde di Gerstner.

Una volta creato il codice descritto in precedenza, si è proceduto a sviluppare una Material Function denominata “MF_My_GerstnerWaves”. Questa funzione contiene un nodo con il codice per le onde di Gerstner, integrato con altri nodi che gestiscono gli input e output, rendendo la funzione facilmente integrabile in qualsiasi materiale.

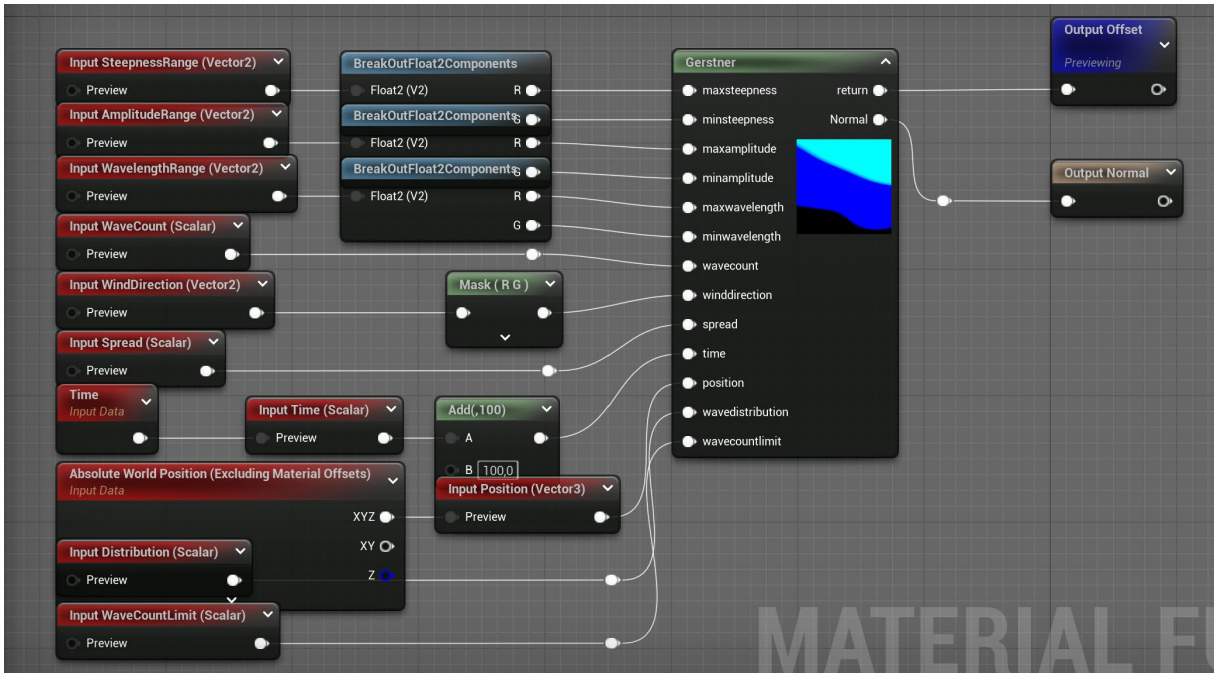

Figura 2.31: Script di “MF_My_GerstnerWaves”

Nel nostro caso, la funzione è stata integrata nel materiale dell’acqua:

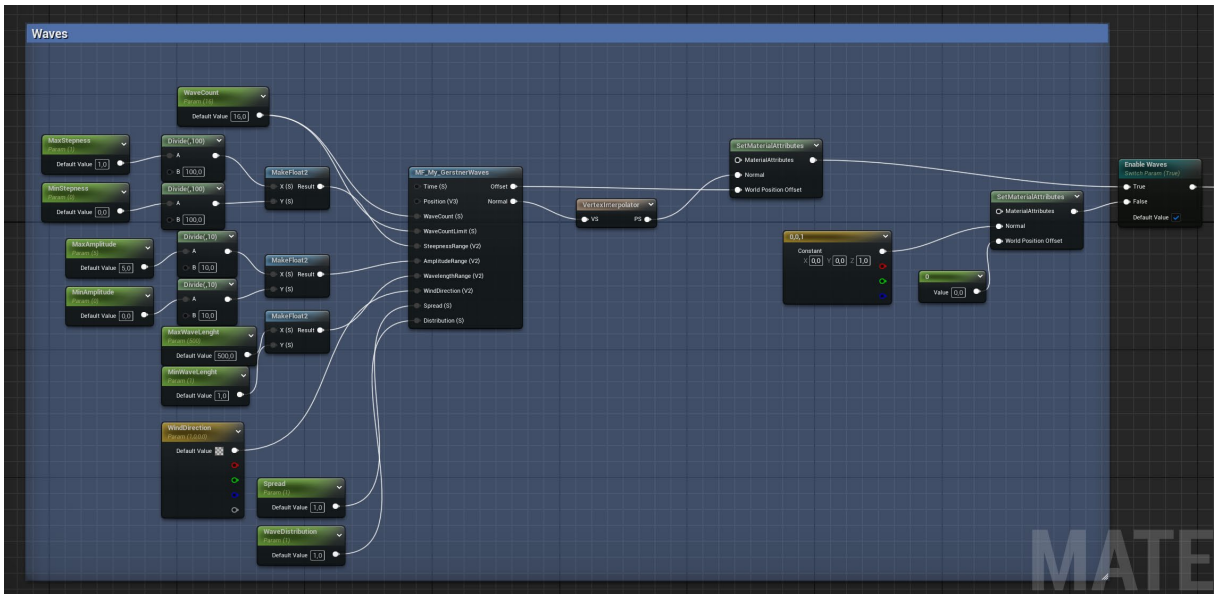

Figura 2.32: Integrazione delle onde nel WaterMaterial

Naturalmente, è stato necessario rimuovere interamente la sezione relativa alle vecchie onde e sostituirla con i nuovi nodi illustrati in figura. Per personalizzare l'aspetto delle onde, sono stati introdotti diversi parametri di input posizionati sulla sinistra del grafico del materiale, che possono essere modificati attraverso il Material Instance del materiale. Questi parametri consentono di regolare aspetti come ampiezza, lunghezza d'onda, direzione e velocità delle onde, offrendo un alto grado di controllo sulla loro simulazione. Un altro parametro modificabile, che non si trova sulla sinistra, è un booleano che attiva o disattiva l'utilizzo delle onde. Questo parametro è gestito dal nodo "Enable Waves" sulla destra del grafico. Se il flag è disattivato, invece di utilizzare i valori della normale e del World Position Offset calcolati dalla funzione di Gerstner, vengono applicati valori di default pari a zero, eliminando così qualsiasi contributo delle onde alla superficie dell'acqua.

2.2.2.2 Ripples

L'ultima modifica al Water Material per aumentarne il realismo ha riguardato l'implementazione delle increspature ("ripples") dell'acqua. Queste increspature appaiono quando l'acqua entra in contatto con una superficie, creando onde concentriche attorno ad essa. Diverse ricerche sono state condotte per capire se fosse possibile ottenere questo effetto in Unreal Engine utilizzando il materiale del WaterBody Custom. Combinando varie tecniche, è stato possibile raggiungere il risultato desiderato, modificando una funzione interna del Water Material chiamata "Water_Attributes". Questa funzione è situata nel primo nodo della sezione "Basic Water Shading" del materiale e si occupa dell'aspetto base dell'acqua, come il colore di base, la

specularità, l'opacità e altri parametri. Tuttavia, l'aspetto che ci interessa modificare è la rifrazione.

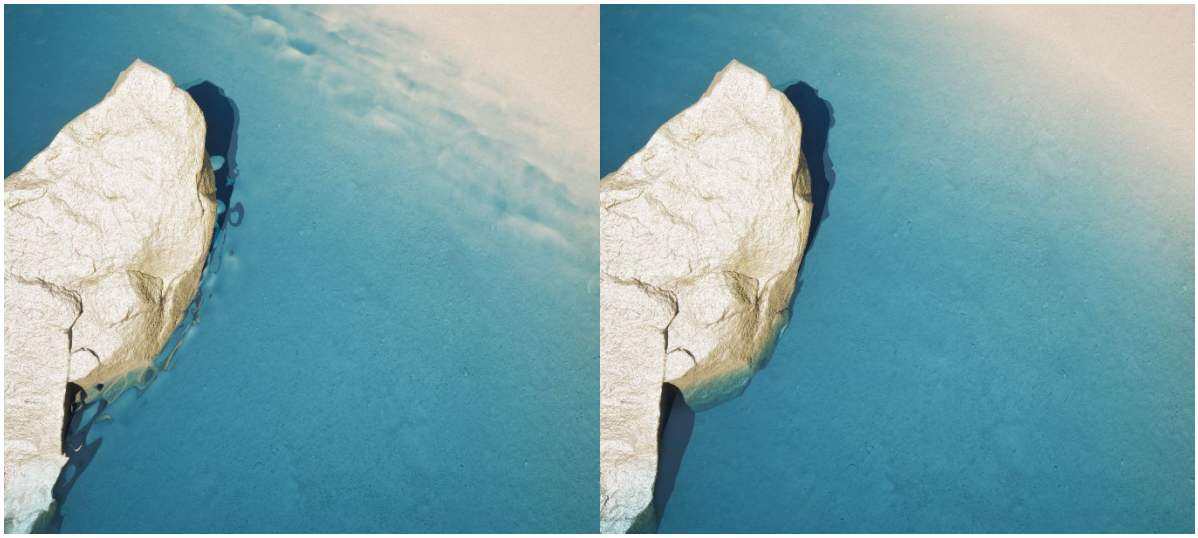

Figura 2.33: Scena con e senza i ripple

Per mantenere la possibilità di attivare o disattivare i ripples, è stato inserito uno switch controllato dal parametro booleano “Enable Ripples”. Questo switch permette di ottenere il valore finale di rifrazione dai nodi originali se il parametro è impostato su false, oppure dai nuovi nodi dei ripples se è impostato su true.

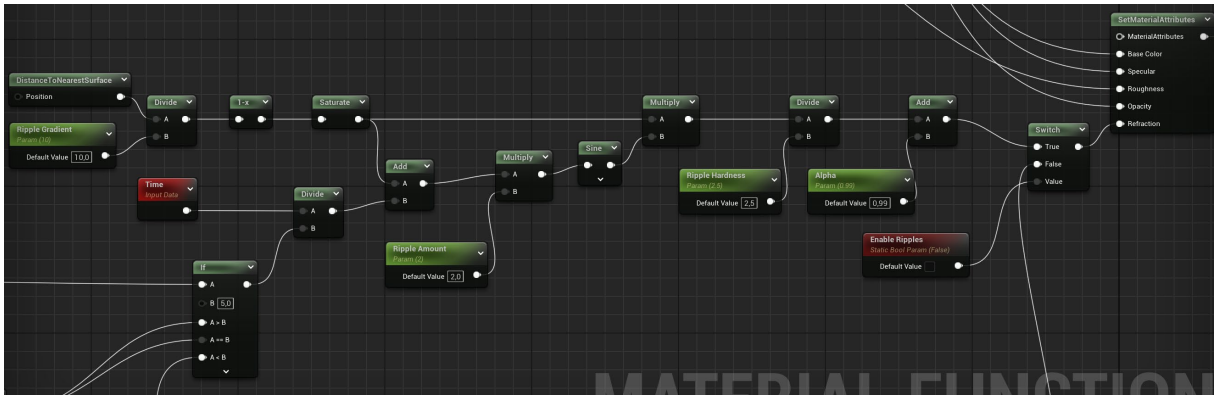

Figura 2.34: Script Ripple nella funzione “WaterAttributes”

L'immagine mostra quasi tutti i nodi coinvolti nella creazione dei ripples, ad eccezione di una logica per l'inserimento lineare dei valori della velocità dei ripples, che sarà trattata successivamente. I principali parametri di input sono “Ripple Gradient” per gestire lo spessore delle ondine, “Ripple Amount” per determinare il numero di ondine, “Ripple Hardness” per definire l'intensità visiva delle ondine, e “Alpha”, un parametro che regola l'effetto finale, rendendo i ripples più o meno trasparenti. Oltre agli input, altri nodi regolano la logica di creazione delle ondine; tre sono i nodi fondamentali che permettono di ottenere il risultato

finale: “DistanceToNearestSurface”, “Time” e “Sine”. Poiché il valore di rifrazione finale è unico, deve essere animato per variare nel tempo; a questo scopo viene utilizzato il nodo “Time”, che restituisce il valore temporale del momento. Questo valore viene poi diviso per la velocità desiderata per ottenere un cambiamento coerente nel tempo. Il nodo “DistanceToNearestSurface” evita che le ondine vengano create in tutta l'acqua, restringendole alle zone vicine a una superficie; il valore di distanza dalla superficie più vicina è diviso per il gradiente per personalizzare la grandezza delle ondine. Infine, il nodo “Sine” assicura che i valori finali seguano un movimento ondulatorio tipico dei ripples.

Infine, è opportuno esaminare brevemente la parte relativa alla gestione dell'input della velocità, ottenuta tramite il parametro “Ripple Speed”:

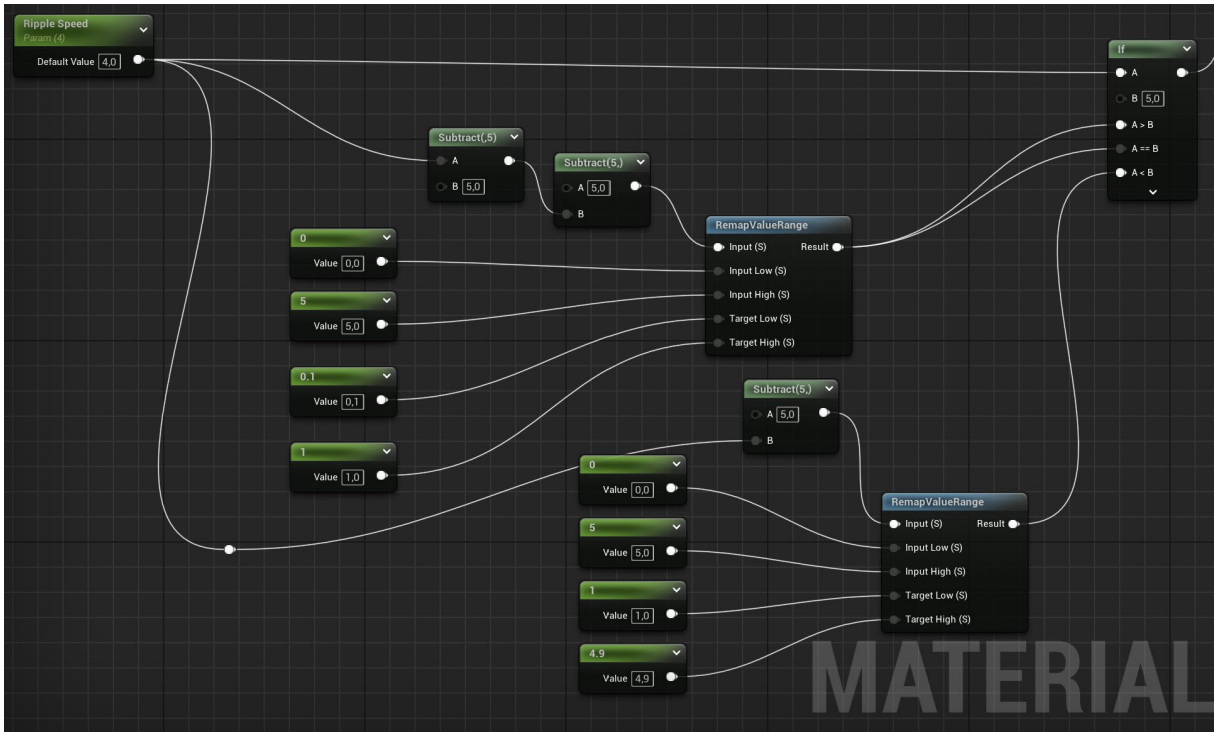

Figura 2.35: Script Ripple nella funzione “WaterAttributes”

Questa logica è stata implementata per rimappare i valori di input, trasformandoli nei valori necessari per ottenere i risultati desiderati. In assenza di questa parte, i valori di input che aumentavano la velocità delle ondine variavano tra 0,1 e 0,9, con 0,1 rappresentante il valore più veloce. I valori più lenti, invece, erano compresi tra 1 e 5, con le ondine che rallentavano progressivamente al crescere del numero. Pertanto, questi nodi sono stati inseriti per rendere i valori di input più intuitivi: valori bassi di input corrispondono a basse velocità delle ondine, mentre valori alti corrispondono a velocità elevate. Il nodo cruciale per questo scopo è il “RemapValueRange”, il quale prende un valore in input, compreso tra “Input Low” e “Input

High", e lo rimappa in un range definito tra "TargetLow" e "TargetHigh". Così, ad esempio, se il valore in input al primo "RemapValueRange" fosse 0, il nodo restituirebbe 0.1; se fosse 5, restituirebbe 1.

2.2.3 La creazione di una funzione per il cambiamento del colore delle rocce

Come evidenziato dallo studio del Nilo e dei suoi effetti sull'ambiente circostante, le rocce immerse nell'acqua, nel corso dei secoli, sono diventate sempre più scure. A causa dell'innalzamento periodico del livello del Nilo, che ha raggiunto approssimativamente la stessa altezza, molte rocce attualmente visibili mostrano una netta divisione di colore, indicante il livello che il fiume raggiungeva durante le sue piene. Per ricreare questo effetto in Unreal Engine, e ripercorrere così gli effetti del Nilo nei secoli, è stata creata una Material Function denominata "Color_Fade", applicabile ai vari materiali delle rocce presenti nella scena. Questa funzione è controllata da parametri riferiti a variabili contenute in un Material Parameter Collection (MPC), appositamente creato per permetterne l'animazione tramite Level Sequence o Blueprint. La funzione si presenta in modo da consentire un controllo dinamico ed efficace della colorazione delle rocce.

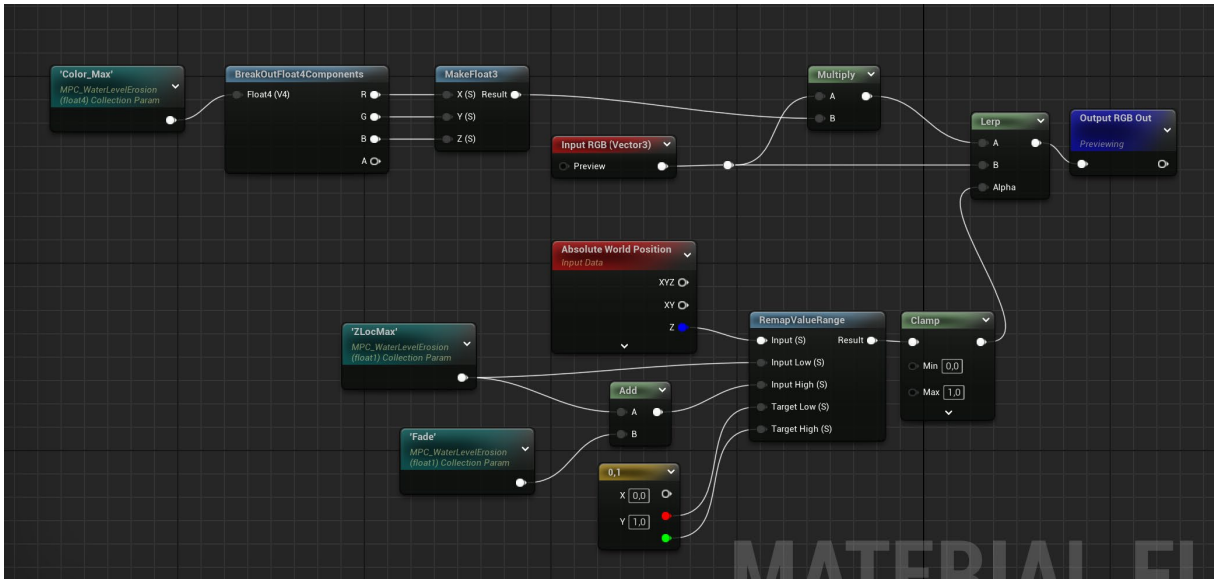

Figura 2.36: Script "Color_Fade"

Gli input principali della funzione, "Color_Max", "ZLocMax" e "Fade", sono valori provenienti dall'MPC "MPC_WaterLevelErosion". Il parametro "Color_Max" è un input di quattro valori (RGBA), corrispondenti al colore con cui si desidera miscelare il colore originale per ottenere il colore finale. In questo caso, il colore finale desiderato è un inscurimento del colore originale, quindi "Color_Max" sarà un colore tendente al nero. Il parametro "ZLocMax"

indica la posizione massima alla quale il pixel deve cambiare colore: tutti i pixel con valori di z superiori a questa soglia manterranno il colore originale, mentre quelli inferiori avranno il colore modificato. Il parametro “Fade” regola la sfumatura del passaggio da un colore all’altro: se il valore di “Fade” è zero, il passaggio tra i due colori sarà netto, posizionato ad altezza z= “ZLocMax”.

Figura 2.37: Roccia con “Fade” =0

Il nodo “Input RGB” della funzione riceve il colore originale, generalmente derivato da una texture, sebbene possa essere anche di altra origine. Un altro nodo fondamentale è il “Absolute World Position”, che determina la posizione del pixel esaminato, essenziale per decidere il colore finale in base alla soglia impostata. Due nodi chiave per ottenere l’effetto desiderato sono il “RemapValueRange” e il “Lerp”. Il nodo “Lerp” prende in input due valori, A e B (nel nostro caso il colore modificato e quello originale), e restituisce un valore compreso tra i due basato sull’input alpha: se alpha è 0, il nodo restituisce A; se alpha è 1, restituisce B; per valori compresi tra 0 e 1, restituisce un valore intermedio tra A e B. L’input alpha è determinato dalla logica che analizza la posizione z dei pixel, considerando anche l’effetto del parametro “Fade”, attraverso l’uso del “RemapValueRange”, nodo di cui si è già spiegato il funzionamento nel capitolo precedente. In breve, se la posizione del pixel è compresa tra “ZLocMax” e “ZLocMax”+”Fade” il nodo restituirà un valore tra 0 e 1; se la posizione del pixel è inferiore o superiore a questo range, restituirà valori inferiori a 0 o superiori a 1, e per questo viene utilizzato il nodo “Clamp” in uscita per convertire i valori fuori range in 0 o 1.

Figura 2.38: Roccia originale e con "Color_Fade"

Questa funzione può, quindi, essere applicata a qualsiasi materiale, a patto che in input riceva il colore originale e che l'output controlli il parametro BaseColor del materiale stesso. Inoltre, deve essere presente il corrispondente MPC, per garantire che la funzione riceva correttamente i valori necessari.

2.2.4 La creazione di un Blueprint per l'erosione delle rocce

Un altro elemento fondamentale per la realizzazione della scena è stato sviluppare una logica in grado di simulare l'erosione delle rocce. L'obiettivo era creare un metodo per modificare le mesh in tempo reale durante l'animazione, riducendone le dimensioni solo al di sotto del livello dell'acqua. Era inoltre necessario che questa logica fosse, almeno in parte, personalizzabile, permettendo la regolazione di parametri come velocità e quantità di erosione. A tal fine, è stato sviluppato un blueprint denominato "RockErosion_ALL". Questo blueprint sfrutta il sistema delle mesh procedurali[36] di Unreal Engine, convertendo le mesh statiche, che non sono modificabili, in mesh procedurali, che possono essere modificate in tempo reale. Attraverso una funzione matematica appositamente creata, il blueprint muove vertice per vertice, riducendo le dimensioni della mesh, ma solo per i vertici situati al di sotto del livello dell'acqua. Per implementare questo processo, il blueprint creato è un blueprint actor senza componenti aggiuntivi oltre al "DefaultSceneRoot". È stato inoltre utilizzato il Material Parameter Collection (MPC) denominato "MPC_WaterLevelErosion" per creare i parametri che rendono l'erosione personalizzabile e dinamica rispetto al livello dell'acqua. In questo MPC sono stati aggiunti i seguenti parametri scalari: "ErosionFactor" (che indica l'intensità dell'erosione),

[36] Le mesh procedurali in Unreal Engine sono oggetti tridimensionali generati dinamicamente tramite algoritmi, senza bisogno di modelli predefiniti. Questa metodologia consente la creazione di geometrie complesse e dettagliate in maniera flessibile ed efficiente, ottimizzando i tempi di sviluppo e migliorando le prestazioni in tempo reale.

"Max Erosion" (che definisce la massima profondità di erosione), "ClockErosion" (che specifica l'intervallo di tempo in secondi tra un'erosione e l'altra) e "ZLocErosion" (che determina la posizione z al di sotto della quale applicare l'erosione, legata al livello dell'acqua). Per assicurare il corretto funzionamento del blueprint, è necessario verificare tre elementi sulle mesh da erodere: il primo consiste nell'aprire il file della mesh statica e abilitare l'opzione "Allow CPU Access"; il secondo richiede di aggiungere nel pannello Details, per ogni actor presente in scena, il componente ProceduralMesh come figlio del componente StaticMeshComponent già presente.

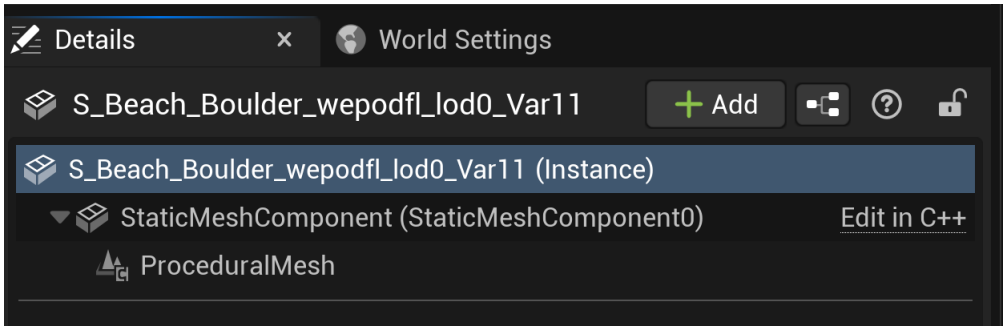

Figura 2.39: Pannello "Details" di una roccia da erodere

Infine, nel pannello Details degli actor da erodere, nella sezione Actor TAG, deve essere aggiunto il tag "Erode" affinché solo questi actor vengano erosi, riducendo così il costo computazionale.

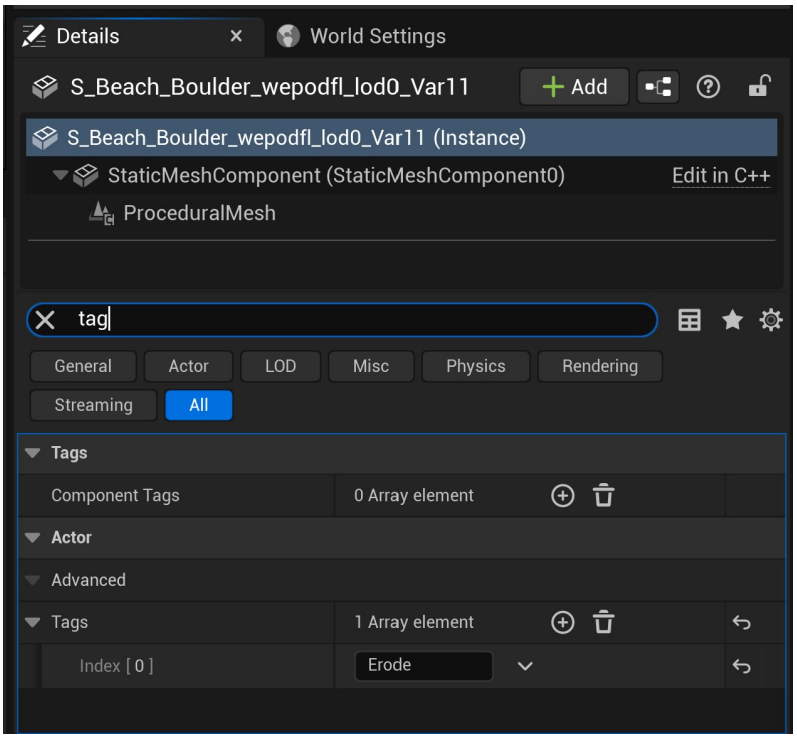

Figura 2.40: Inserimento TAG "Erode" nel pannello "Details"

Con queste premesse, possiamo ora procedere all'analisi dettagliata del funzionamento del blueprint.

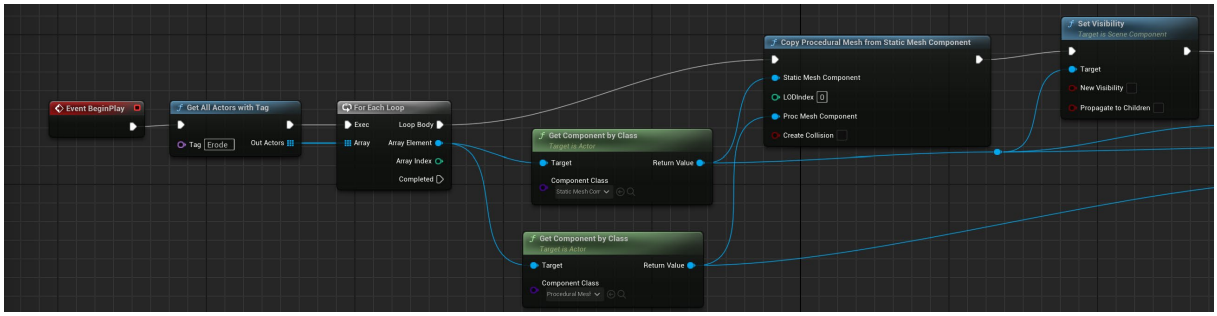

Figura 2.41: Script Blueprint di "RockErosion_ALL"

Tutti i nodi si sviluppano nel "Event Graph", iniziando con il gruppo collegato all'evento "Event BeginPlay", che viene eseguito una sola volta, all'inizio del livello. Questo flusso di operazioni serve a inizializzare i vettori necessari per le operazioni successive; in particolare, l'obiettivo è ottenere tutte le static mesh e tutte le procedural mesh che devono essere erose, organizzandole in due vettori separati, in modo che a parità di indice ci sia la stessa mesh nelle due versioni diverse. In questa prima parte del codice, tramite il nodo "Get All Actors with Tag", vengono individuati tutti gli attori con il tag "Erode". L'array di output viene quindi inserito come input nel nodo "For Each Loop" per iterare su tutti gli elementi del vettore. Per ogni elemento del vettore, ossia per ogni actor che deve essere eroso, vengono ottenuti i riferimenti ai componenti delle mesh statica e procedurale tramite il nodo "Get Component by Class", utilizzato due volte con diverse impostazioni per selezionare il componente desiderato. All'uscita di questo nodo, si ottengono i riferimenti ai due componenti: la prima operazione cruciale è eseguita dal nodo "Copy Procedural Mesh from Static Mesh Component", che copia tutti i dati dalla mesh statica a quella procedurale, rendendole identiche. Questo passaggio è fondamentale perché inizialmente il componente procedurale è vuoto, mentre si desidera che sia identico alla mesh statica posizionata nella scena. Il riferimento alla mesh statica è ancora necessario per impostarne la visibilità su false, in modo che non venga più visualizzata in scena, tramite il nodo "Set Visibility". Questa operazione è indispensabile per evitare che vengano visualizzate entrambe le mesh, statica e procedurale, sovrapposte; in questo modo si rende visibile solo la mesh procedurale che verrà poi erosa.

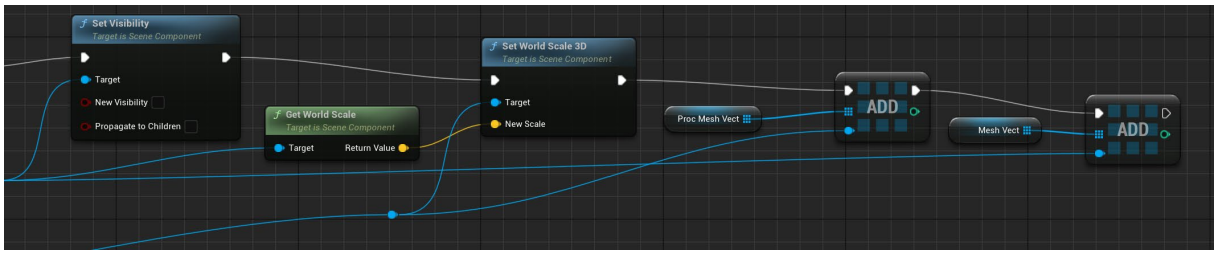

Figura 2.42: Script Blueprint di "RockErosion_ALL"

A questo punto, è necessario trasferire le informazioni di scala dalla mesh statica a quella procedurale, poiché il nodo "Copy Procedural Mesh from Static Mesh Component" non lo fa automaticamente. Per fare ciò, si utilizza il nodo "Get World Scale" per ottenere i valori di scala dalla mesh statica e successivamente il nodo "Set World Scale 3D" per impostare questi valori sulla mesh procedurale. Infine, entrambe le mesh vengono aggiunte ai rispettivi vettori utilizzando i nodi "ADD", inserendo le mesh statiche nel vettore "Mesh Vect" e quelle procedurali nel vettore "Proc Mesh Vect".

Prima di analizzare ciò che verrà eseguito a ogni frame, è utile comprendere rapidamente una funzione che sarà utilizzata a breve: la funzione "Setup Variables".

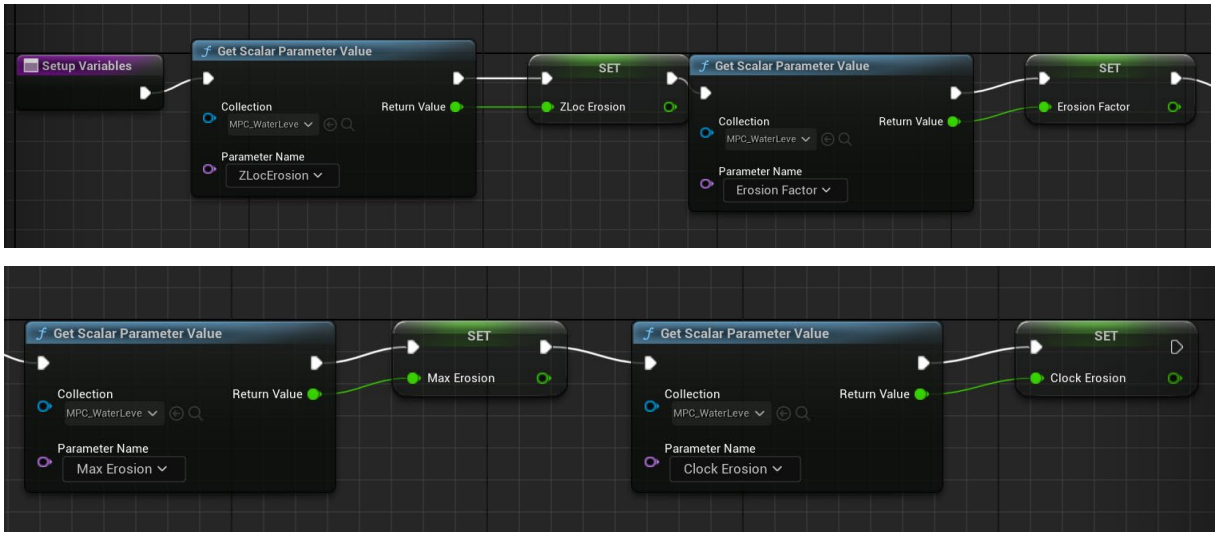

Figura 2.43: Script Blueprint di "RockErosion_ALL", funzione "Setup Variables"

Essa recupera le quattro variabili contenute nell'MPC e le memorizza in altrettante variabili (con lo stesso nome) all'interno del blueprint. Questa operazione non è strettamente obbligatoria, poiché potremmo richiamare "Get Scalar Parameter Value" ogni volta che abbiamo bisogno del valore di una variabile. Tuttavia, questo approccio sarebbe meno conveniente e più oneroso dal punto di vista computazionale. Il motivo per cui la funzione viene chiamata a ogni frame -come vedremo a breve- è che le variabili all'interno dell'MPC possono

essere animate e modificate in tempo reale, rendendo necessario l'aggiornamento costante dei valori.

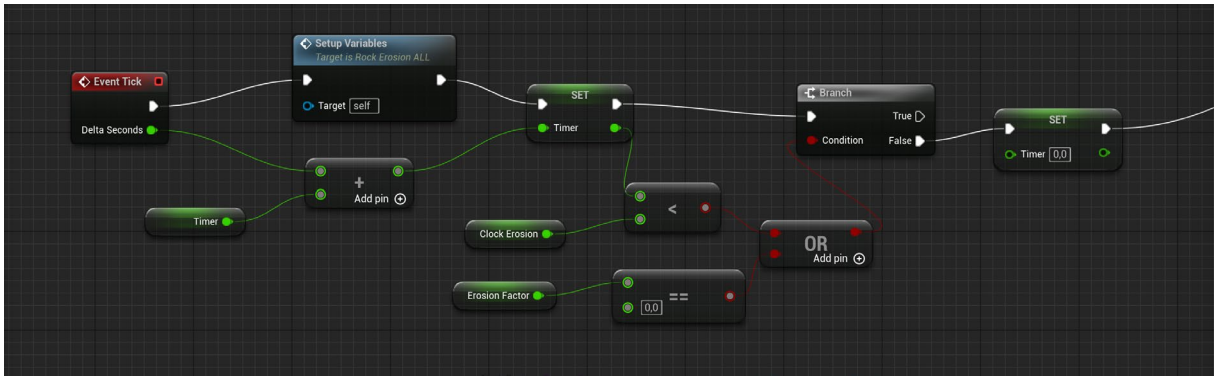

Figura 2.44: Script Blueprint di "RockErosion_ALL"

Ora possiamo analizzare tutto ciò che è connesso all'evento "Event Tick", che viene eseguito ogni frame. Questo ci consente di eseguire l'erosione in tempo reale, poiché tutte le operazioni connesse verranno eseguite a ogni frame. La prima operazione eseguita è la chiamata alla funzione "Setup Variables", già descritta precedentemente. Successivamente, viene gestita la variabile "Timer", che tiene traccia del tempo trascorso in secondi. L'"Event Tick" restituisce una variabile chiamata "Delta Seconds", che rappresenta il tempo trascorso dall'ultima chiamata, ovvero dall'ultimo frame. Questo valore varia in base alla velocità di elaborazione della CPU/GPU e al frame rate del livello. Ad ogni frame, questo valore viene sommato alla variabile "Timer", che scandisce, quindi, il tempo. Per garantire che l'erosione non dipenda dalla velocità della CPU/GPU né dal frame rate, utilizziamo "Timer" e "ClockErosion" per determinare quando eseguire l'erosione. "ClockErosion" contiene il valore, in secondi, dell'intervallo di tempo tra un'erosione e l'altra. Tramite il nodo "<", otteniamo "true" finché "Timer" è minore di "ClockErosion". Grazie al nodo "Branch", si prosegue con le operazioni di erosione solo quando il tempo trascorso è superiore o uguale al tempo specificato. Quando questo avviene, prima di eseguire le operazioni di erosione, la variabile "Timer" viene resettata per consentire un corretto riavvio del conteggio. Inoltre, viene verificata un'altra condizione: il valore di "Erosion Factor" deve essere diverso da zero. Le operazioni di erosione vengono eseguite solo se la condizione finale è "false". A tal fine, viene aggiunta un'operazione booleana finale: ("Timer" < "ClockErosion") OR ("Erosion Factor" == 0); in questo modo, solo se entrambe le condizioni sono false, si prosegue con l'erosione. Questa condizione evita di

eseguire inutilmente le operazioni di erosione quando “Erosion Factor” è nullo, risparmiando così tempo computazionale prezioso: si può dire che sia un’operazione di pruning[37].

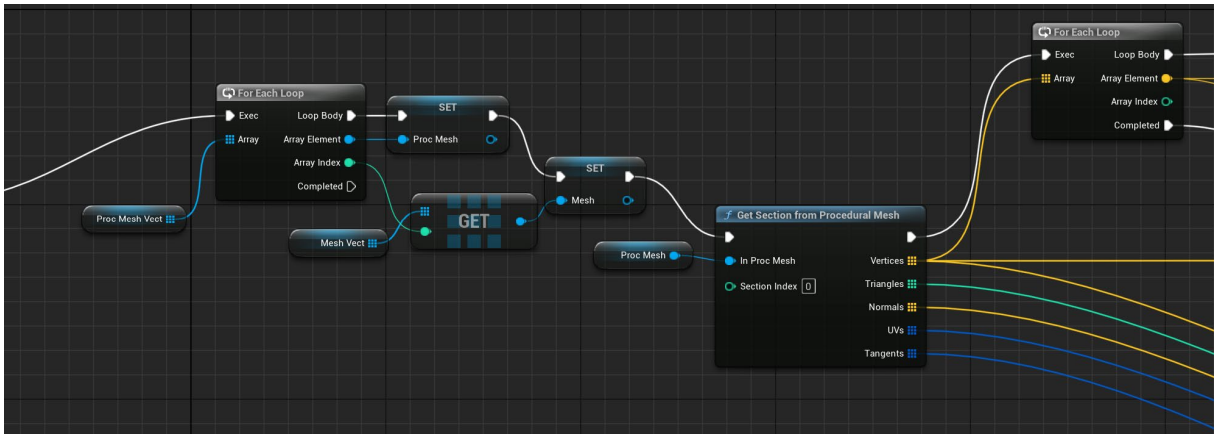

Figura 2.45: Script Blueprint di “RockErosion_ALL”

Dopo questa parte iniziale, si esegue un “For Each Loop” che itera sul vettore delle mesh procedurali. Ad ogni ciclo, la mesh attuale viene salvata in una variabile denominata “Proc Mesh”, mentre la mesh statica corrispondente, con lo stesso indice del ciclo, viene salvata in una variabile chiamata “Mesh”. Questo meccanismo consente al programma di elaborare ogni mesh procedurale contenuta nel vettore “Proc Mesh Vect”, ossia ogni mesh che deve essere erosa. Successivamente, con “Proc Mesh” come input, si esegue il nodo “Get Section from Procedural Mesh”, che fornisce le informazioni strutturali della mesh procedurale, come triangoli, normali, mappe UV, tangenti e, per il nostro scopo, i vertici. Essi sono composti da una terna di valori (x, y, z) che indicano la loro posizione spaziale. Le mesh procedurali possono avere più sezioni interne, come se fossero suddivise in parti distinte, ma nel nostro caso le mesh utilizzate hanno una sola sezione; perciò, l'indice in input è fissato a zero. Da questo nodo partono numerosi output, ma è essenziale seguire il flusso di esecuzione; quindi, si passa a un altro “For Each Loop”, che in questo caso itera su tutti gli elementi del vettore “Vertices”, cioè su tutti i vertici della mesh. Da questo ultimo ciclo for partono due pin di esecuzione: il primo, "Loop Body", rappresenta tutte le operazioni eseguite ad ogni ciclo del for, mentre il secondo, “Completed”, rappresenta il flusso delle operazioni che continua solo quando il ciclo for ha

[37] Il pruning in informatica si riferisce a una serie di tecniche impiegate per ridurre la complessità dei modelli o degli algoritmi, eliminando rami superflui o meno significativi. Questo processo è particolarmente vantaggioso nel contesto delle reti neurali, dove il pruning aiuta a rimuovere nodi e pesi che contribuiscono marginalmente alla precisione del modello, migliorando l'efficienza e riducendo i tempi di elaborazione. Inoltre, il pruning è utilizzato in algoritmi di ricerca, come negli alberi decisionali, per limitare l'esplorazione di percorsi non promettenti, ottimizzando così le prestazioni complessive del sistema.

completato tutte le sue iterazioni. Prima si analizzerà il corpo del ciclo for, e successivamente ciò che accade dopo la sua conclusione.

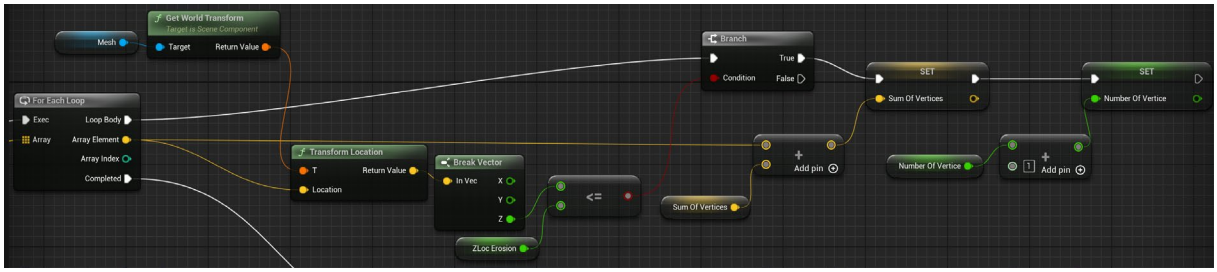

Figura 2.46: Script Blueprint di "RockErosion_ALL"

Sia per ottenere un'erosione più naturale e dipendente dalla forma della mesh sia per sapere in che direzione spostare i vertici erodendoli, è stato deciso di utilizzare nella formula di erosione un valore rappresentativo della media di tutti i vertici da erodere. In particolare, si è calcolato un punto medio, dato dalla somma dei valori di x, y e z di tutti i vertici da erodere, divisa per il numero totale di vertici, e infatti il corpo di questo ciclo for permette di ottenere questo risultato. Tralasciando per un momento i nodi iniziali che verranno spiegati a breve, si ottiene il valore z del vertice corrente e lo si confronta con "ZLoc Erosion". Se il valore è minore o uguale, indica che il vertice deve essere eroso, quindi la condizione restituisce true, e conseguentemente il nodo "Branch" eseguirà i nodi connessi all'uscita true. Il gruppo di nodi successivo incrementa il valore della variabile "Sum Of Vertices" (che, come si vede dal colore, è una variabile vettoriale contenente la somma, per ogni componente - x, y, z - dei valori dei vertici da erodere). Il gruppo di nodi ancora successivo incrementa la variabile "Number of Vertices" di uno per ogni iterazione, che alla fine del ciclo conterrà il numero totale dei vertici da erodere.

Prima di procedere con il resto del blueprint, è necessario ritornare a quei nodi che sono stati omessi nella spiegazione precedente, riguardanti il modo di recuperare il valore z del vertice. Quando si leggono le proprietà di posizione (coordinate x, y e z) dei vertici delle mesh procedurali, queste non riflettono le eventuali trasformazioni successive. Ciò significa che se la mesh procedurale, che riceve le trasformazioni dalla mesh statica posizionata in scena, è stata traslata, ruotata o scalata, la posizione dei vertici viene modificata, ma queste posizioni aggiornate non sono presenti nel vettore dei vertici, il quale contiene solo le posizioni originali. Le trasformazioni vengono applicate da Unreal automaticamente alla fine di tutto, ad ogni frame, utilizzando le trasformazioni ottenute nella prima parte del nostro codice attraverso i nodi "Copy Procedural Mesh from Static Mesh Component" e "Set World Scale 3D". Per stabilire se un vertice deve essere eroso o meno, è fondamentale analizzare il suo valore finale

sull'asse z, non quello originale. Pertanto, è stato necessario trovare un metodo per ottenere manualmente la trasformazione da applicare al vertice per calcolarne il valore finale. A tal fine, si utilizza il nodo "Get World Transform", che tuttavia richiede in input un riferimento a una mesh statica, non procedurale. Per questo motivo, viene utilizzata in input la variabile "Mesh", che contiene la mesh statica corrispondente a quella procedurale attuale. Ottenuta la trasformazione, questa viene applicata al vertice attraverso il nodo "Transform Location", che restituisce in output il vettore del vertice con i valori trasformati. Dato che ci interessa solo il valore sull'asse z, il vettore viene scomposto nei suoi componenti con il nodo "Break Vector" per ottenere infine il valore z, che sarà poi utilizzato nelle successive operazioni di analisi e erosione del vertice.

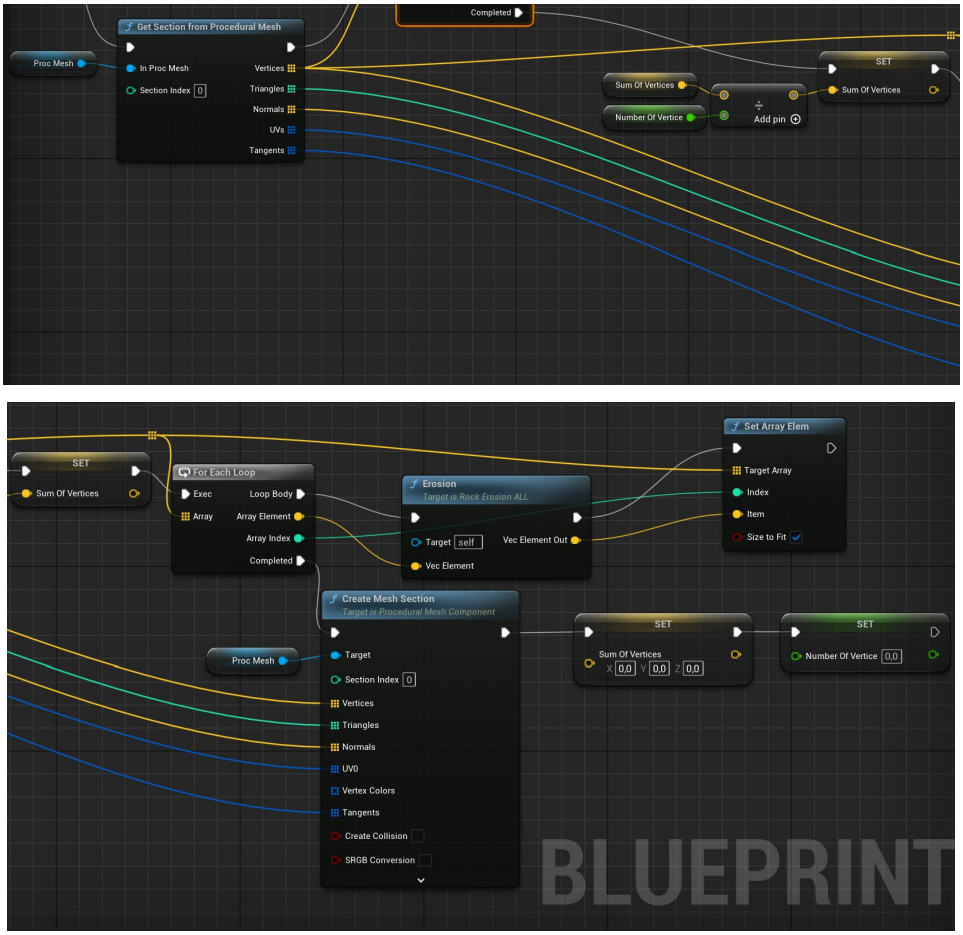

Figura 2.47: Script Blueprint di "RockErosion_ALL"

Ora possiamo analizzare l'intera parte di codice eseguita al completamento del ciclo for precedentemente osservato. La prima operazione eseguita è quella di calcolare la media dei valori dei vertici mediante il semplice calcolo $\frac{Sum\ of\ Vertices}{Number\ of\ Vertices}$, il cui risultato sovrascrive il valore precedente di "Sum of Vertices". Successivamente, viene iterato nuovamente l'intero

vettore dei vertici tramite un altro ciclo “For Each Loop”, durante il quale vengono eseguite operazioni sia nel corpo del ciclo, per ogni singolo vertice, sia al completamento dell'intero vettore. Nel corpo del ciclo, per ogni vertice viene eseguita la funzione “Erosion”, creata appositamente per applicare l’erosione, ovvero uno spostamento specifico del vertice. Successivamente, il nuovo valore del vertice viene salvato nel vettore originale tramite il nodo “Set Array Elem”, che riceve in input il riferimento del vettore, l'indice dell'elemento da modificare (ottenuto dal ciclo for) e il nuovo valore del vertice. Una volta completate tutte le iterazioni, vengono eseguite le operazioni finali del blueprint: in primo luogo è sovrascritta la sezione della mesh procedurale con gli stessi elementi precedenti, che ora includono i valori aggiornati dall’erosione; questa operazione viene eseguita dal nodo “Create Mesh Section”. Successivamente, le variabili utilizzate per ottenere la media dei valori dei vertici vengono resettate a zero, in modo da essere pronte per un nuovo ciclo su una nuova mesh. Ora che abbiamo spiegato tutto il livello superiore del blueprint, rimane solo da analizzare la funzione “Erosion”.

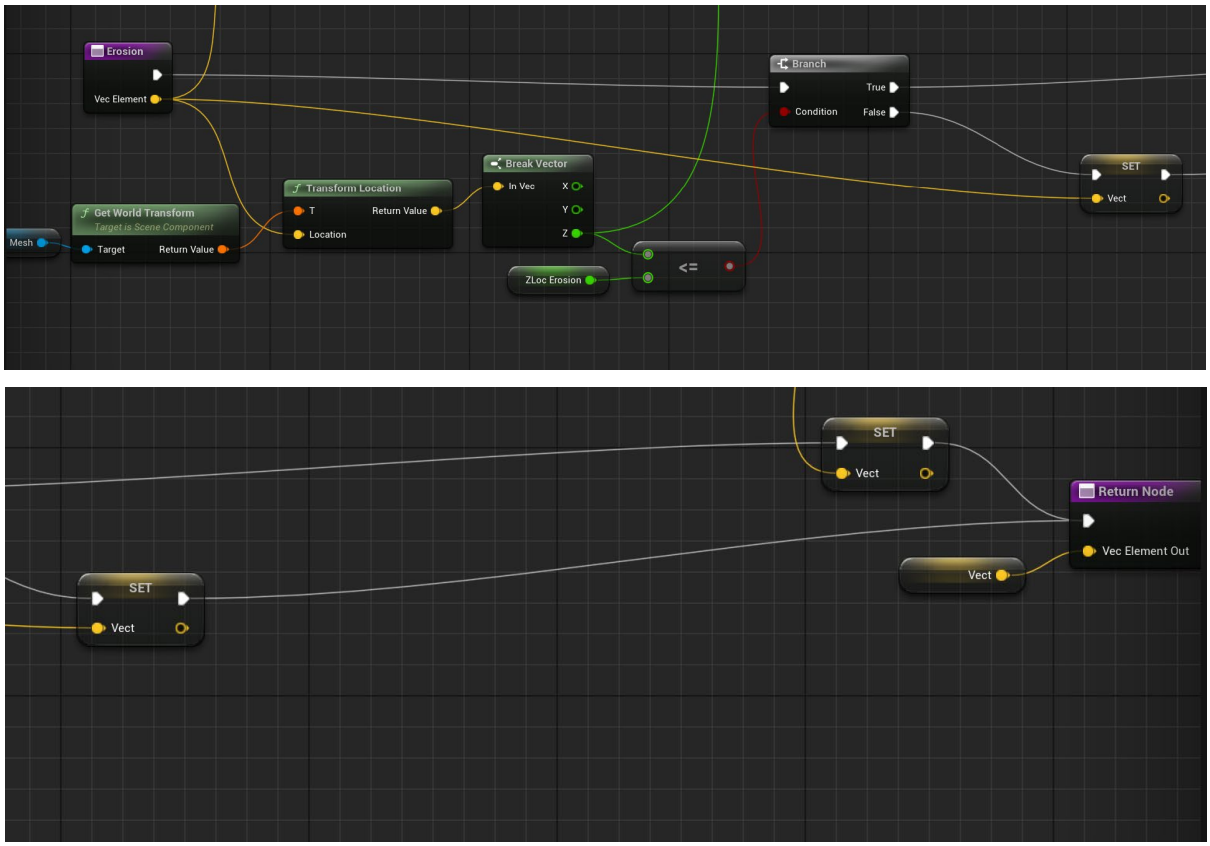

Figura 2.48: Script Blueprint di “RockErosion_ALL”, funzione “Erosion”

In questa prima parte della funzione “Erosion” viene gestita la logica di esecuzione e il calcolo del valore z dopo la trasformazione. La funzione riceve in input la terna di valori che

rappresentano la posizione del vertice. Innanzitutto, si determina se il vertice deve essere eroso o meno, utilizzando i nodi descritti nella parte precedente. In particolare, si calcola la trasformazione applicandola e confrontando il valore z ottenuto con la variabile "ZLoc Erosion": se il vertice non è da erodere, il nodo "Branch" farà proseguire il flusso su false, salvando il valore originale del vertice nella variabile "Vect" e restituendolo in output. Al contrario, se il vertice è da erodere, viene calcolato un valore modificato e salvato nella variabile, per poi restituirlo in output.

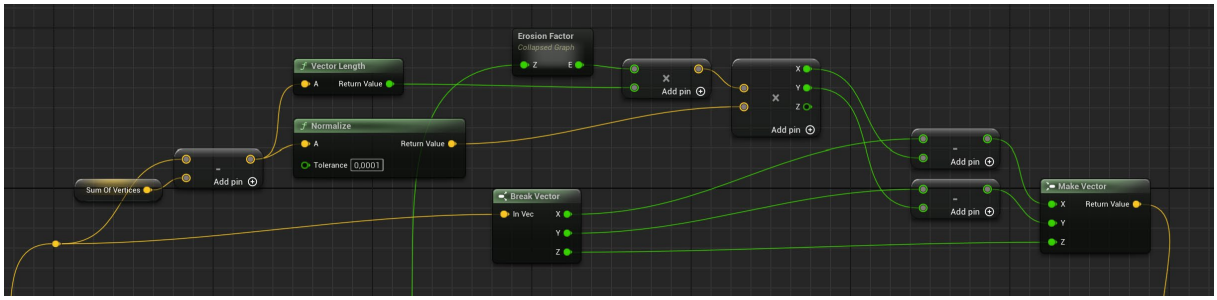

Figura 2.49: Script Blueprint di "RockErosion_ALL", funzione "Erosion"

La logica che modifica il valore del vertice si basa su calcoli vettoriali fondamentali. Il primo passo consiste nel sottrarre la posizione del vertice dal valore medio di tutti i vertici, ottenendo così un vettore che collega il vertice al punto medio. Con questa terna di valori, vengono eseguite due operazioni parallele: la prima è il calcolo della lunghezza del vettore tramite il nodo "Vector Length", che rappresenta la distanza tra il vertice e il punto medio. Questo servirà a far sì che l'erosione sia più forte per i vertici più lontani dal punto medio e più debole per quelli più vicini, ottenendo un comportamento più realistico. La seconda operazione è la normalizzazione della terna di valori con il nodo "Normalize", ottenendo un vettore unitario che indica la direzione di spostamento dei vertici, dalla posizione attuale verso il punto medio.

Una volta completate queste operazioni, con l'input del valore z trasformato del vertice si può calcolare il fattore di erosione "E" tramite la funzione "Erosion Factor". Per determinare gli spostamenti Δx e Δy, si esegue l'espressione: $E * VectorLenght * VectorNormalize$. Successivamente, si sottrae dal valore originale del vertice il corrispondente delta, ottenendo $x_{fin} = x_{in} - \Delta x$ e $y_{fin} = y_{in} - \Delta y$. Si ricostruisce poi il vettore posizione del vertice con i nuovi valori utilizzando il nodo "Make Vector", che incorpora le coordinate x e y aggiornate e la componente z originale. La scelta di mantenere invariata la z è stata fatta per ottenere un'erosione realistica; infatti, se i vertici si muovessero anche lungo l'asse z, si otterrebbe uno spostamento non sensato, con le rocce che inizierebbero a fluttuare. Uno spostamento delle sole

coordinate x e y, insieme all'utilizzo del punto medio, garantisce invece uno spostamento naturale verso il centro della mesh, simulando efficacemente un'erosione.

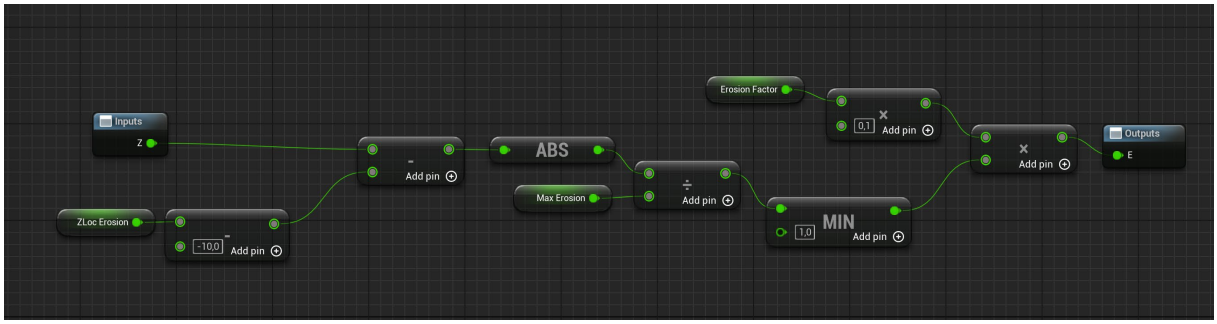

Figura 2.50: Script Blueprint di "RockErosion_ALL", funzione "Erosion Factor" in "Erosion"

L'ultimo passaggio consiste nell'analizzare il calcolo del fattore di erosione "E". La creazione di questo fattore mirava a ottenere un'espressione che incrementasse l'erosione man mano che i vertici si trovassero a una maggiore profondità e che si basassero sui valori di input definiti dall'utente tramite l'MPC. Per una comprensione più agevole, è preferibile osservare l'espressione matematicamente piuttosto che tramite il blueprint.

$$E = "Erosion\ Factor" * MIN\left(\frac{|Z - "\text{ZLoc Erosion}"|}{"Max\ Erosion"}, 1\right)$$

- "Erosion Factor": È un coefficiente che controlla la forza complessiva dell'erosione: un valore più alto implica una maggiore erosione. Nel blueprint, esso è moltiplicato per 0.1 per offrire all'utente un controllo maggiore quando modifica il valore del parametro, evitando che l'utente debba lavorare su valori di erosione troppo piccoli.
- Z: La coordinata Z del vertice in questione.
- "ZLoc Erosion": Il livello di riferimento per l'erosione, che in questo caso è pari al livello dell'acqua. Nel blueprint, ad essa viene sottratto 10 per evitare che il movimento dei vertici appena al di sotto del livello dell'acqua provochi movimenti della mesh anche sopra al livello dell'acqua. Questo valore è stato determinato sperimentalmente e può variare a seconda della scena.
- "Max Erosion": Il valore massimo di profondità, a partire da "ZLoc Erosion", fino al quale il fattore di erosione aumenta. I vertici più distanti (in profondità) non avranno un fattore di erosione crescente oltre questo valore.
- $|Z - "\text{ZLoc Erosion}"|$: Viene usato il valore assoluto perché Z è sempre inferiore a "ZLoc Erosion", visto che l'erosione avviene solo per i vertici al di sotto di questa soglia; l'uso del valore assoluto garantisce un risultato positivo.

- MIN (A,1): L'operazione di MIN prende il valore minimo tra il primo e il secondo operatore. Questa operazione garantisce che, anche se il rapporto fosse maggiore di 1, il valore massimo del fattore di erosione "E" rimanga quello deciso dall'utente ("Erosion Factor"). Questo meccanismo è attivato quando Z < "ZLoc Erosion" - "Max Erosion".

Questa formula matematica crea un'erosione che aumenta linearmente con la distanza dalla soglia, rappresentata da una retta diagonale. In alcune situazioni, si è voluto un comportamento di erosione più verticale e per ottenere questo, basta collegare direttamente l'uscita del calcolo ("ZLoc Erosion" – 10) all'input della divisione con "Max Erosion".

Figura 2.51: Gli effetti dell'erosione diagonale e verticale

Come evidenziato dalle immagini precedenti, esiste una differenza significativa nell'erosione delle due rocce a seconda che sia stata scelta un'erosione diagonale o verticale; entrambi i tipi di erosione sono stati applicati con gli stessi valori, sia nella versione base sia in quella con erosione più verticale, e come si può notare, quella verticale ha un impatto maggiore. Sebbene sia stata applicata un'erosione intenzionalmente esagerata e in un punto non ideale della mesh, creando artefatti visivi particolari, questo era necessario per evidenziare chiaramente il comportamento dell'erosione nelle immagini. Inoltre, nell'erosione diagonale, si può apprezzare l'effetto del "Max Erosion" osservando come il profilo della roccia rimanga diagonale fino a una certa profondità, per poi diventare verticale, indicando che l'erosione non incrementa più il suo effetto oltre quel punto.

Un ultimo passaggio consiste nel collegare il livello dell'acqua alla variabile "ZLocErosion" dell'MPC "MPC_WaterLevelErosion", in modo che il cambiamento della

posizione dell'acqua si riflettesse anche sulla variabile dell'erosione. Per fare ciò, è stato creato un nuovo blueprint, separato da quello dell'erosione per mantenerlo il più generico possibile.

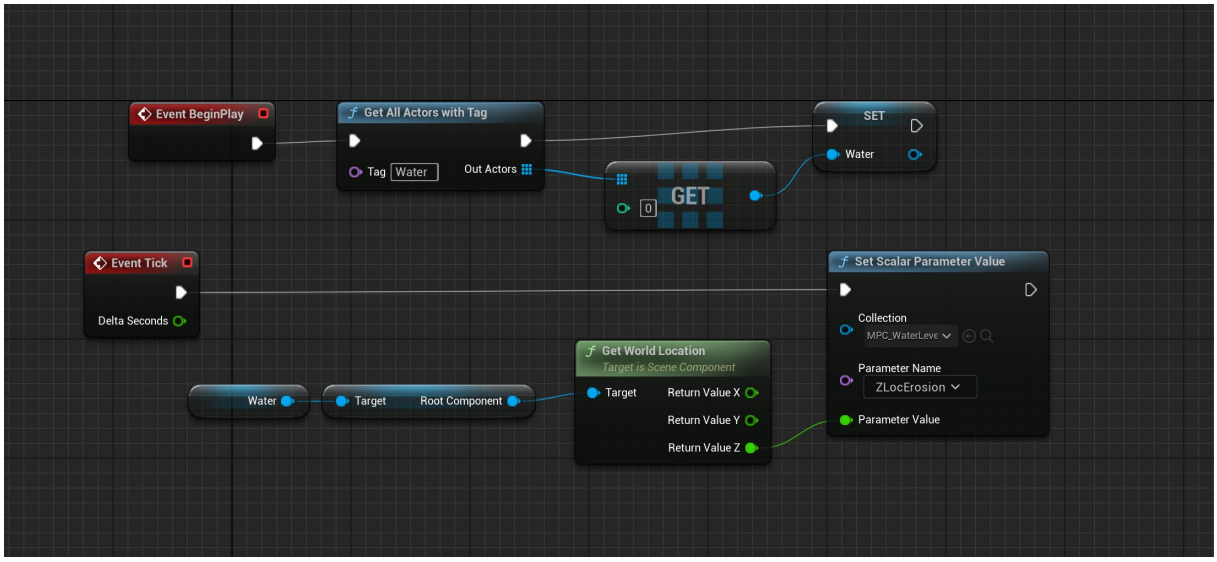

Figura 2.52: Script Blueprint per collegare il livello dell'acqua a "ZLocErosion"

Questo nuovo blueprint è molto breve: nella prima parte, eseguita solo all'avvio tramite il nodo "Event BeginPlay", viene preso il riferimento all'actor dell'acqua presente in scena (a cui è stato assegnato il tag "Water" per un riconoscimento rapido) e viene salvato nella variabile "Water". Nella seconda parte, ad ogni frame, viene acquisito il valore della coordinata z dell'acqua tramite "Get World Location" e assegnato alla variabile nell'MPC tramite "Set Scalar Parameter Value".

In conclusione, il blueprint dell'erosione realizzato per il progetto ha dimostrato l'efficacia delle tecnologie avanzate di Unreal Engine nel simulare processi naturali complessi in tempo reale. Grazie all'uso dei procedural mesh, è stato possibile trasformare mesh statiche in entità dinamiche, capaci di rispondere a variabili ambientali e parametri definiti dall'utente. La combinazione di nodi logici e matematici ha permesso di creare una simulazione realistica dell'erosione, che tiene conto della posizione dei vertici rispetto al livello dell'acqua e della loro distanza dal punto medio della mesh.

2.2.5 Una visione aerea del fiume Nilo

Grazie alle tecniche precedentemente analizzate, sono state realizzate due inquadrature della scena vista dall'alto, permettendo di osservare il Nilo e il suo paesaggio circostante. Durante l'animazione, il livello dell'acqua sale, simulando una delle piene tipiche del fiume e contemporaneamente, sott'acqua, vi è l'erosione e il cambiamento di colore delle rocce, per poi

assistere alla discesa del livello dell'acqua, mostrando così come la morfologia sia stata modificata.

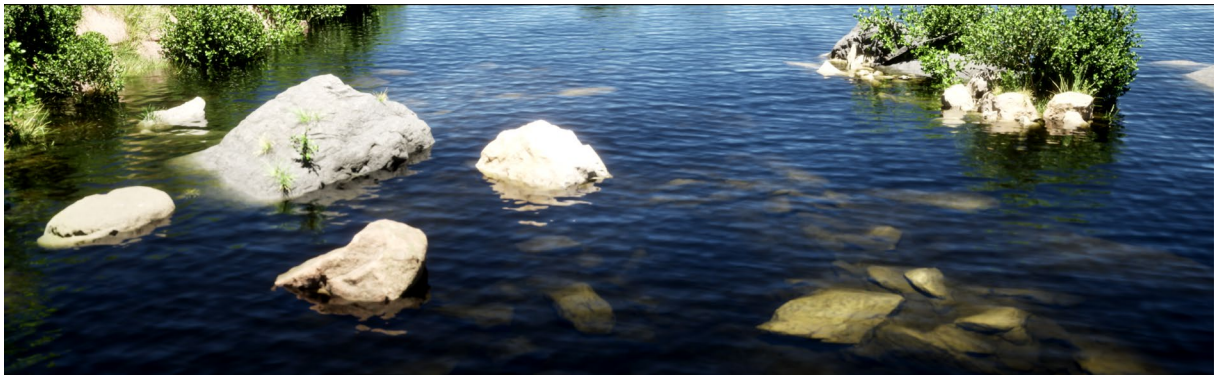

Figura 2.53: Inquadratura 1 della scena dall'alto

Figura 2.54: Inquadratura 2 della scena dall'alto

Per creare questa scena, sono stati importati tutti i modelli da Megascan, compresa la vegetazione, applicata tramite l'editor Foliage per ottenere un effetto più realistico e casuale delle piante. Per quanto riguarda l'animazione, è stato utilizzato un Level Sequence, in cui sono stati aggiunti track a vari elementi: la camera, la Directional Light, il blueprint dell'acqua e l'"MPC_WaterLevelErosion". Per aggiungere proprietà da tracciare che non sono presenti di default, basta premere sul "+" alla destra del nome del componente a cui si vuole aggiungere una proprietà da animare.

- "CineCameraActor": La camera è il primo elemento da inserire nel Level Sequence, poiché è essenziale per poter renderizzare una scena. Essa determina cosa verrà visualizzato e le sue proprietà stabiliscono come sarà vista la scena, similmente a una telecamera reale, inclusi formato e messa a fuoco. Quando si inserisce questo componente, viene aggiunto automaticamente il Camera Cuts, che consente di selezionare, lungo la timeline, quando utilizzare la visuale della camera per il rendering.

Per la camera è stato aggiunto un unico keyframe al primo frame, per impostare la sua posizione e orientamento per tutta la durata dell'animazione.

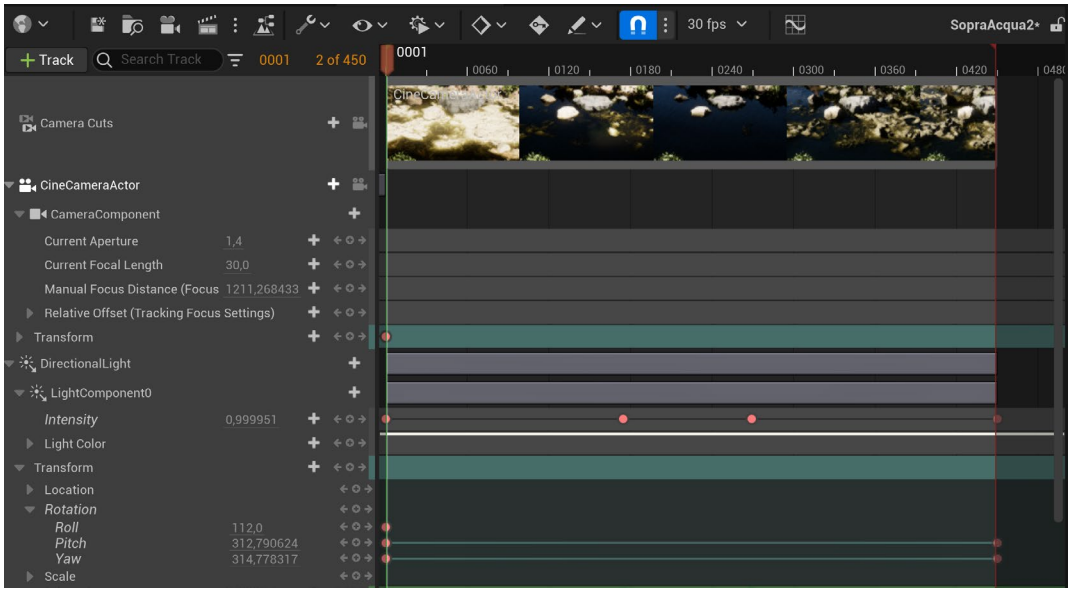

Figura 2.55: Level Sequence della scena vista dall'alto

- "Directional Light": Questa luce, disponibile in Unreal per illuminare il livello, simula il sole essendo una luce direzionale. Nella scena la luce non era di primaria importanza, quindi non sono state utilizzate tecniche di illuminazione avanzate. Nel Level Sequence sono stati aggiunti alcuni keyframe per modificare leggermente la direzione della luce durante la scena (cambiando i valori di Pitch e Yaw, ovvero rotazione lungo gli assi y e z), nonché altri keyframe per animare l'intensità della luce, che durante l'animazione diminuisce, rimane stabile per un po' e poi aumenta leggermente.

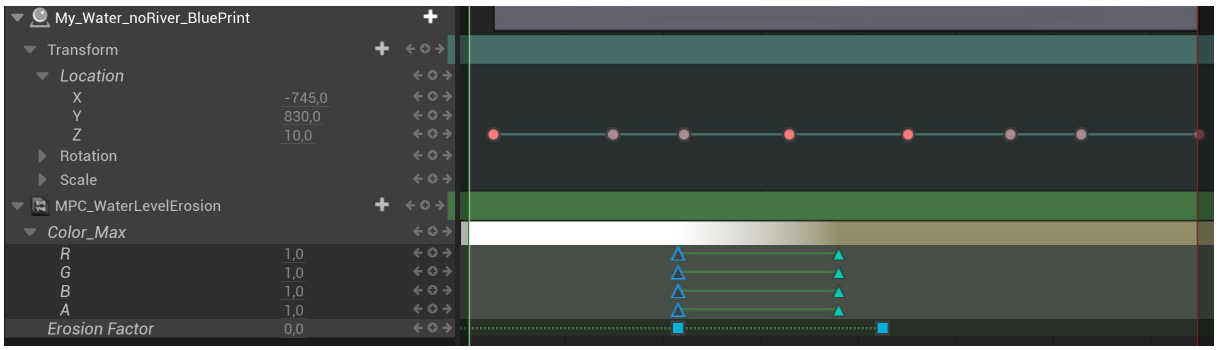

Figura 2.56: Level Sequence della scena vista dall'alto

- "My_Water_noRiver_Blueprint": Questo blueprint dell'acqua è l'elemento centrale dell'animazione. Sono stati aggiunti vari keyframe per la sua posizione sull'asse z durante la timeline, in modo da far muovere l'acqua in alto e in basso. Potenzialmente potevano bastare tre keyframe, due per il livello minimo (all'inizio e alla fine) e uno per

il livello massimo (al centro della timeline). Tuttavia, per simulare una salita e discesa più realistica del fiume, sono stati aggiunti molti keyframe, ottenendo una salita più "a gradini" e un livello massimo dell'acqua che rimane stabile per un po'. All'inizio e alla fine ci sono anche alcuni secondi con l'acqua ferma al livello minimo, permettendo all'utente di osservare il paesaggio prima e dopo l'erosione. Per ottimizzare i keyframe e le transizioni tra di essi è stato utilizzato il Curve Editor.

- "MPC_WaterLevelErosion": Questo MPC contiene i parametri dell'erosione e del cambio del colore. Tutti i parametri sono stati impostati staticamente ad hoc per la scena, tranne due: "Color_Max" e "Erosion Factor", che sono stati aggiunti alla timeline per modificarli durante l'animazione. Il "Color_Max", utilizzato per cambiare il colore delle rocce, è stato animato in modo che all'inizio le rocce mantengano il colore originale (variabile settata al bianco), poi durante il livello massimo dell'acqua il colore diventa marrone scuro per scurire le rocce. Anche l'"Erosion Factor" è stato animato similmente: inizialmente l'erosione è nulla, poi al livello massimo dell'acqua aumenta e infine torna a zero quando l'acqua scende. Per ottenere un cambiamento non lineare ma a gradini di questa variabile, è stato utilizzato il Curve Editor per impostare la funzione di interpolazione dei keyframe come una funzione a gradini.

Figura 2.57: Frame 0 dell'inquadratura 1

Figura 2.58: Frame finale dell'inquadratura 1

Figura 2.59: Frame 0 dell'inquadratura 2

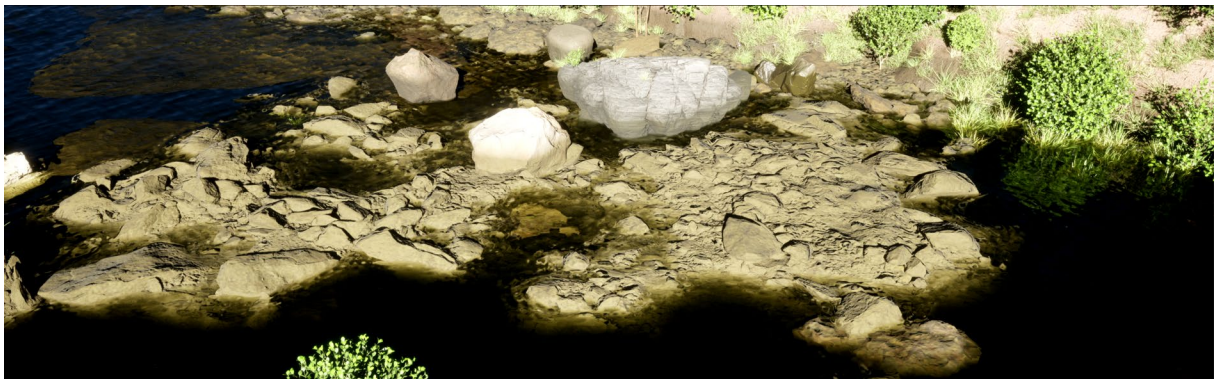

Figura 2.60: Frame finale dell'inquadratura 2

Grazie a queste tecniche, è stato possibile creare una scena dall'alto del Nilo che mostra l'interazione tra l'acqua e il paesaggio circostante in modo dinamico e realistico, evidenziando i cambiamenti morfologici e visivi dovuti all'erosione e alle piene del fiume.

2.2.6 Una nuova visione del Nilo: la scena subacquea

Nella scena dall'alto, l'erosione e il cambio di colore delle rocce sono visibili solo nel loro effetto finale, senza mostrare il processo che porta a tali trasformazioni. Per evidenziare in modo diretto questi cambiamenti, è stata creata una scena subacquea, con due diverse inquadrature, in cui la camera sprofonda sotto la superficie dell'acqua durante la piena del fiume. In questo modo, si può osservare da vicino come le rocce si erodono e cambiano colore. Per la realizzazione di questa scena, è stato necessario l'utilizzo di due elementi aggiuntivi fondamentali: i Niagara System e un PostProcess Volume.

Figura 2.61: Inquadratura 1 della scena subacquea

Figura 2.62: Inquadratura 2 della scena subacquea

2.2.6.1 Niagara System

I Niagara System sono stati utilizzati per simulare l'effetto di sbriciolamento delle rocce e l'effetto "fumo" intorno ad esse, rendendo più evidente il processo di erosione e quindi per ciascuna roccia in scena, sono stati impiegati due sistemi Niagara. Pur avendo una struttura di base simile, i Niagara sono stati adattati alle diverse forme e colori delle rocce tramite la modifica dei parametri. In questa sezione, analizzeremo la struttura generale dei Niagara e solo alcuni dei parametri chiave. Il primo tipo di Niagara, denominato "Erosion", simula lo sbriciolamento delle rocce e si presenta con la seguente configurazione:

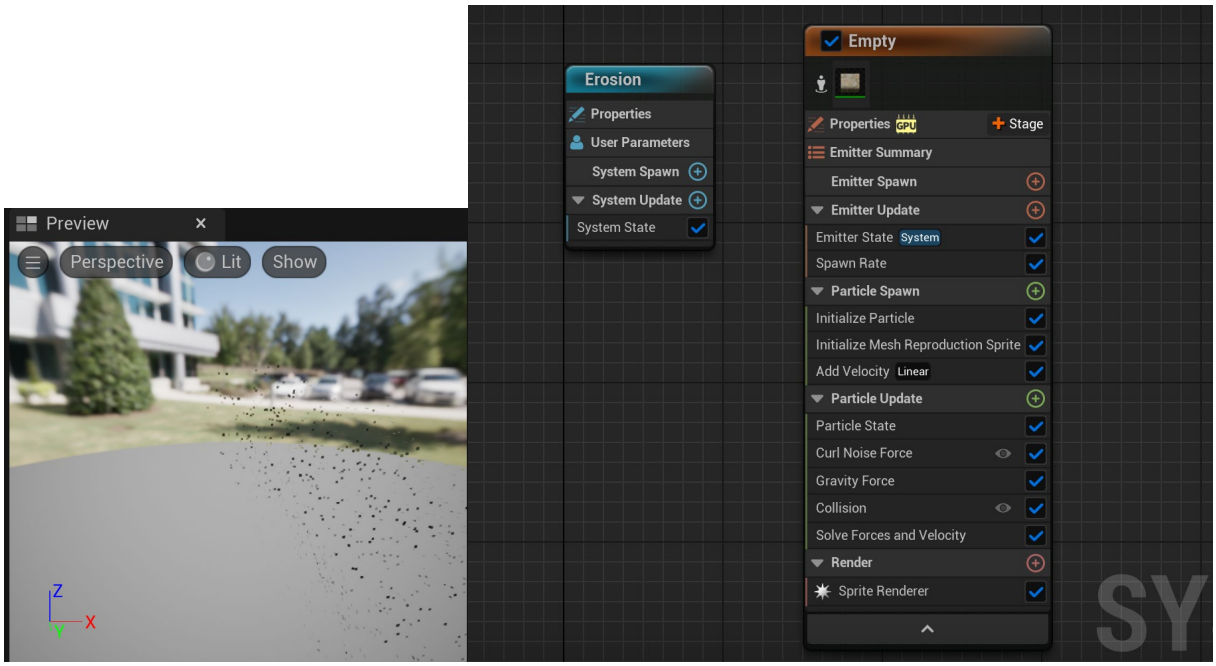

Figura 2.63: Preview e nodi del Niagara "Erosion"

Il Niagara "Erosion", simula, quindi, lo sbriciolamento delle rocce: uno dei moduli chiave di questo sistema è l'"Initialize Mesh Reproduction Sprite", che permette di generare particelle a partire da una forma specifica, in questo caso un cubo, utile per approssimare la forma delle rocce. I moduli di "Particle Update" sono cruciali per determinare il movimento delle particelle, e le loro modifiche dipendono strettamente dalla scena e dal posizionamento del Niagara. Infine, il modulo "Sprite Render" personalizza l'aspetto delle particelle; per simulare lo sbriciolamento delle rocce, le particelle devono avere lo stesso colore della roccia, quindi il materiale della roccia è stato impostato nel parametro Material di questo modulo. Il secondo sistema Niagara, chiamato "Polvere", simula un effetto di fumo attorno alle rocce per accentuare l'erosione e si presenta con la seguente configurazione:

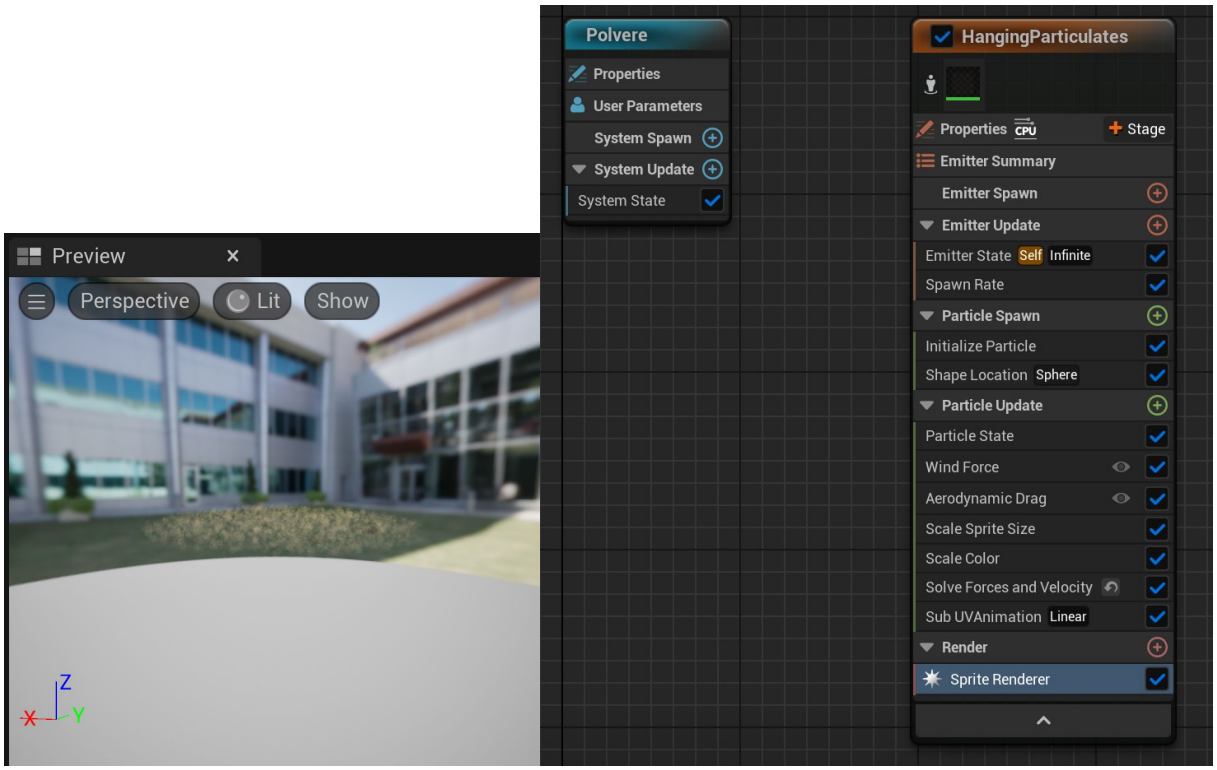

Figura 2.64: Preview e nodi del Niagara "Polvere"

Per creare questo Niagara, si è partiti da un modello precostruito chiamato "HangingParticulates". Sono stati modificati alcuni valori nei moduli, tra cui "Shape Location", dove è stata scelta una forma sferica scalata su x e y invece di una forma cubica di base. È stato aggiunto anche il modulo "Sub UVAnimation", che anima le particelle usando una singola texture contenente molte piccole immagini, cambiando rapidamente tra queste immagini per creare un effetto animato come fiamme, esplosioni o fumo sulle particelle. Nel campo Material del modulo "Sprite Renderer" è stato inserito un materiale denominato "M_Smoke", già presente nella libreria di Unreal, ma modificato per personalizzare l'effetto del fumo desiderato.

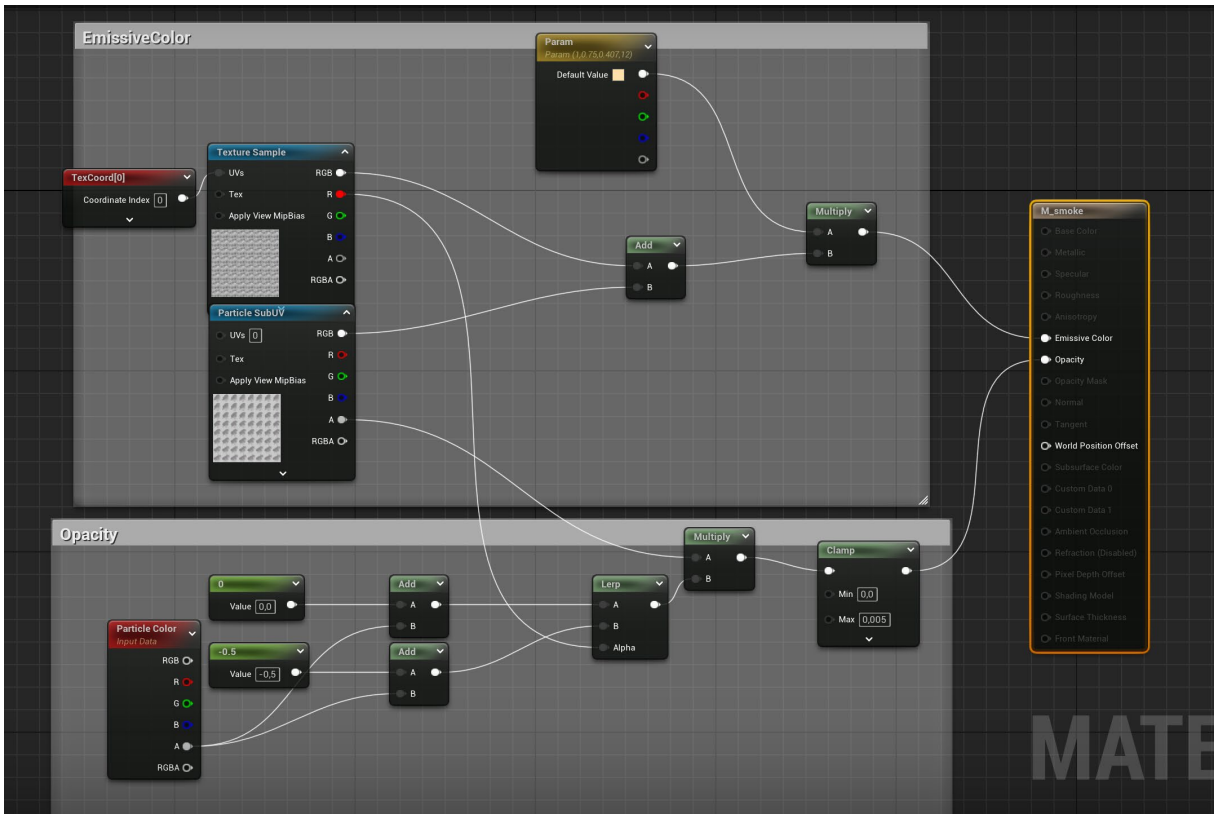

Figura 2.65: Nodi del materiale "M_Smoke"

Innanzitutto, come si può vedere, le texture del materiale in input corrispondono alla descrizione delle texture necessarie al modulo "Sub UVAnimation", ovvero sono composte da tante piccole immagini. Invece, per quanto riguarda le modifiche apportate al materiale "M_Smoke", esse includono l'aggiunta del nodo "Multiply" e del parametro "Param" per permettere la modifica del colore del fumo, rendendolo più simile al colore delle rocce. È stato aggiunto anche il nodo "Clamp" per massimizzare i valori dell'opacità, consentendo un controllo preciso sul grado di opacità/trasparenza del fumo. Questi parametri sono stati personalizzati per ogni roccia in base al loro aspetto specifico, assicurando che gli effetti visivi siano coerenti e realistici.

2.2.6.2 PostProcess Volume

Per simulare una visione subacquea, che differisce notevolmente dalla visione normale, è stato necessario implementare un meccanismo in Unreal Engine che potesse alterare la percezione della camera solo quando questa si trova sott'acqua. La soluzione scelta è stata l'utilizzo di un PostProcess Volume: questo componente di Unreal permette di applicare effetti di post-processing, ovvero effetti che vengono eseguiti come ultimo passaggio prima del rendering finale, quando la camera si trova all'interno di un volume specificato. L'acqua utilizzata nella

scena è un semplice piano, quindi, senza l'applicazione di effetti aggiuntivi, la visione sotto di esso rimarrebbe invariata. L'uso di un WaterBodyRiver, che è volumetrico, avrebbe offerto una visuale subacquea differente, ma per i motivi già discussi, non è stato scelto come modello per l'acqua, pertanto, è stato necessario configurare un PostProcess Volume per alterare i colori e applicare altri effetti fotografici, quando la camera è sott'acqua, per ottenere un risultato visivo realistico. Il PostProcess Volume è stato aggiunto alla scena come qualsiasi altro componente e, attraverso il pannello Details, sono stati modificati diversi parametri per raggiungere l'effetto desiderato.

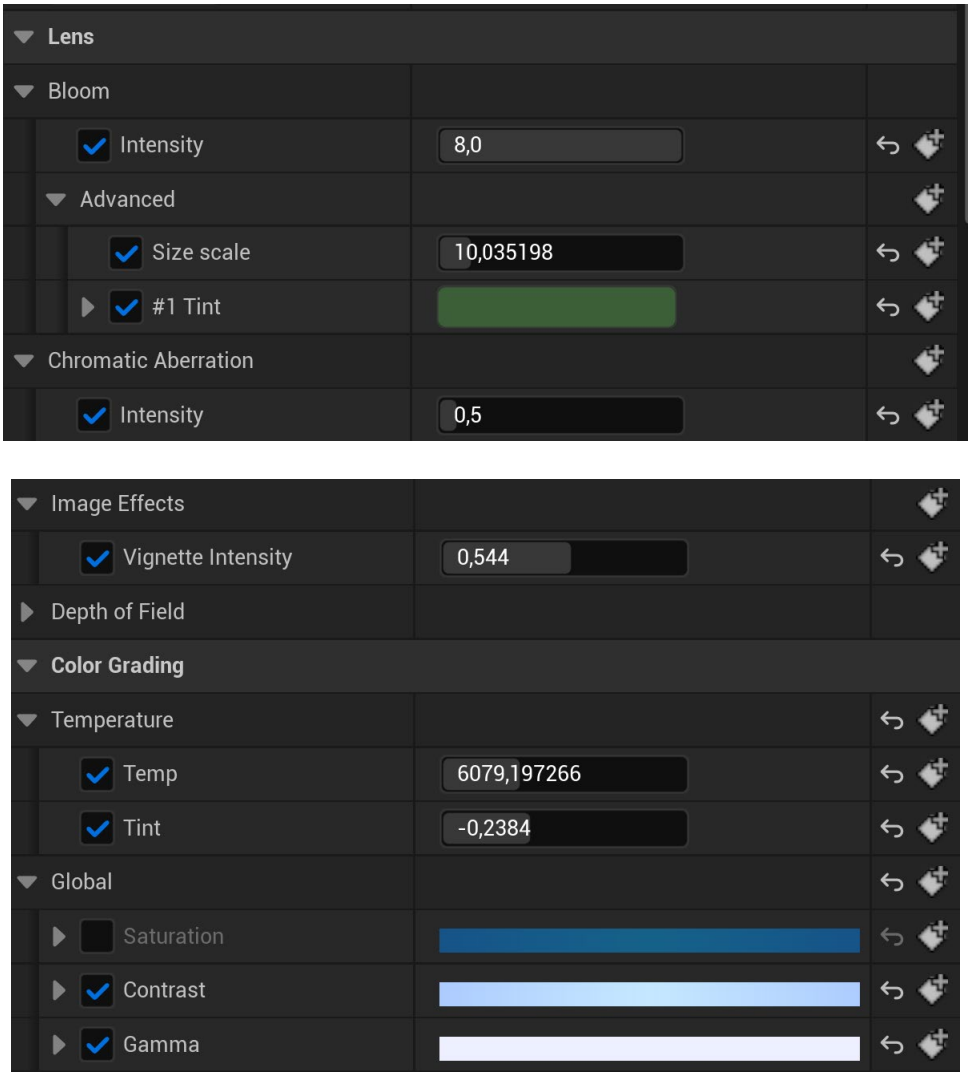

Figura 2.66: Pannello "Details" del "PostProcess Volume"

Tra questi, l'effetto bloom è stato configurato per simulare il fenomeno per cui le aree molto luminose di un'immagine si espandono oltre i loro confini, creando un alone luminoso diffuso: questo aiuta a dare un senso di immersione e brillantezza diffusa tipica dell'ambiente subacqueo. Un altro effetto applicato è stato l'aberrazione cromatica, che provoca bordi colorati attorno ai

contorni degli oggetti nell'immagine e questo effetto è utile per simulare la distorsione ottica che può verificarsi sott'acqua. È stato applicato anche un effetto di vignettatura, che scurisce i bordi dell'immagine rispetto al centro, contribuendo a concentrare l'attenzione verso il centro e a simulare le condizioni di visibilità ridotta sott'acqua. Per ottenere una visione complessivamente più tendente al blu, sono stati impostati i valori di temperatura, tinta, contrasto e gamma. Questo è fondamentale per ricreare l'effetto visivo tipico dell'ambiente subacqueo, dove la luce si diffonde e assume tonalità più fredde. Questi parametri, combinati, hanno permesso di ottenere una visione subacquea convincente e immersiva, migliorando significativamente il realismo della scena e l'esperienza visiva dell'utente.

Figura 2.67: Visione sott'acqua senza PostProcess Volume

Figura 2.68: Visione sott'acqua con PostProcess Volume

Ora che si sono analizzati gli elementi aggiuntivi della scena rispetto alla precedente, possiamo esaminare la scena nel suo complesso. Molti elementi sono i medesimi della scena precedente, come le rocce e la creazione dell'ambiente; tuttavia, come è ovvio, sono state aggiunte le varie coppie di Niagara System per ogni roccia inquadrata, il PostProcess Volume, e anche il Level Sequence, che rimane molto simile al precedente, con tutti gli elementi già visti, con qualche modifica ai valori e al posizionamento dei keyframe. Gli elementi nuovi aggiunti all'animazione sono i Niagara System e il PostProcess Volume.

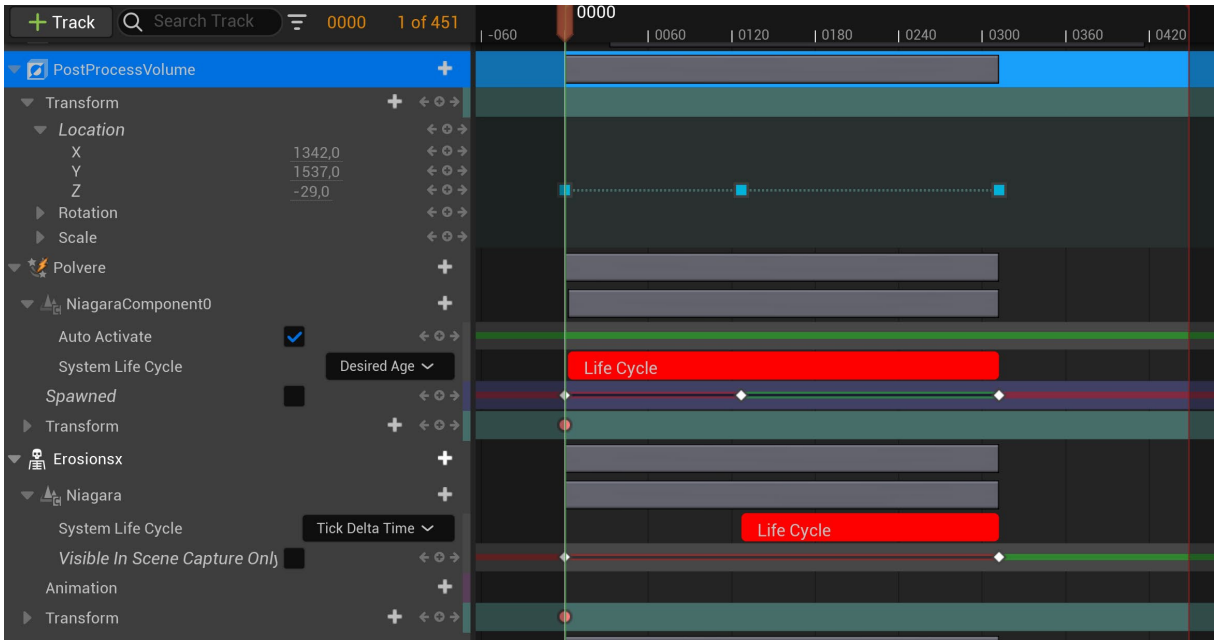

Figura 2.69: Level Sequencer della scena subacquea

Facendo un render di prova, si è osservato frame per frame quale fosse il momento migliore per attivare e disattivare la visione subacquea e l'erosione; in questo caso, la visione subacquea è presente dal frame 127 al 312. Questi due frame sono cruciali perché i Niagara System e il PostProcess Volume saranno visibili solo in questo intervallo. Il PostProcess Volume è inizializzato in una posizione non visibile dalla camera; poi, con una funzione a gradino (come si può notare dai keyframe di forma quadrata anziché romboidale), sale in una posizione visibile dal frame 127, per poi tornare con una coordinata z tale da non essere più visibile dal frame 312. Per il Niagara "Polvere" si è adottata la strategia di renderlo un oggetto "spawnabile", cioè un oggetto la cui presenza in scena è controllata dal Level Sequence tramite il boolean "Spawned". Questo parametro è impostato su true solo dal frame 127 al 312. Il suo Life Cycle parte all'inizio dell'animazione, permettendo al fumo di iniziare a generarsi senza essere visibile, così quando diventa visibile è già formato e non c'è la transizione di crescita del fumo. Per il Niagara "Erosion", prima del frame 127 il Life Cycle è nullo, quindi nessuna particella viene generata; successivamente, dal frame 312 la generazione delle particelle è interrotta, ma quelle vecchie sono ancora visibili. Per questo motivo, viene forzata la loro scomparsa cambiando il flag della variabile "Visible in Scene Capture Only".

In conclusione, l'implementazione del PostProcess Volume e dei Niagara System ha permesso di simulare una visione subacquea realistica e l'erosione delle rocce, arricchendo notevolmente la scena e rendendo l'animazione più coinvolgente e visivamente accurata.

Figura 2.70: Frame 0 dell'inquadratura 1

Figura 2.71: Frame finale dell'inquadratura 1

Figura 2.72: Frame 0 dell'inquadratura 2

Figura 2.73: Frame finale dell'inquadratura 2

2.3 Softwares per le fotogrammetrie

Oltre alle scene riguardanti il Nilo, sono state create numerose scene rappresentanti siti archeologici egizi, con l'obiettivo di realizzare dei timelapse giorno/notte di tali siti. Per creare queste scene è necessario importare su Unreal i modelli dei siti archeologici; tuttavia, questi modelli non esistono online e ricrearli manualmente in modo identico è impossibile. Per questo motivo, si è deciso di partire dalle foto scattate ai siti archeologici per ottenere delle fotogrammetrie e, successivamente, generare il modello fotorealistico da inserire nella scena su Unreal, mentre solo alla fine si è proceduto con la creazione del timelapse. Per ottenere questo risultato è stato necessario utilizzare una serie di programmi in sequenza, che saranno brevemente descritti in questo capitolo, focalizzandosi in brevi spiegazioni necessarie per capire come sono stati realizzati i modelli fotogrammetrici. Nel capitolo successivo verrà invece illustrato l'effettivo utilizzo di queste tecniche per le scene realizzate.

2.3.1 DaVinci Resolve 18

DaVinci Resolve 18 è un software all'avanguardia per l'editing video, il color grading, gli effetti visivi e la post-produzione audio, sviluppato da Blackmagic Design. Originariamente creato da DaVinci Systems nel 2004, è stato acquisito da Blackmagic Design nel 2009, che lo ha trasformato in uno dei software di post-produzione più avanzati e completi disponibili oggi. Da allora, DaVinci Resolve ha subito una significativa evoluzione, affermandosi come uno strumento indispensabile per i professionisti del cinema e della televisione. Un momento chiave nella sua storia è stata l'introduzione di una versione gratuita, che ha reso accessibili potenti strumenti di post-produzione a una vasta gamma di utenti, dai professionisti agli appassionati, rivoluzionando così l'industria e rendendo questo software un'opzione preferita e accessibile per molti. DaVinci Resolve 18 si distingue per la sua capacità di integrare in un unico ambiente di lavoro tutte le fasi del processo di post-produzione: il software offre, infatti, potenti funzionalità di editing non lineare, che consentono di assemblare e rifinire video con precisione e flessibilità eccezionali. Inoltre, è rinomato per il suo avanzato sistema di color grading, che permette di manipolare i colori con una precisione senza pari, rendendolo ideale per progetti cinematografici che richiedono un'elevata qualità visiva. Anche le funzionalità di effetti visivi (VFX) sono altamente sofisticate, offrendo strumenti per la composizione e la creazione di effetti speciali. La suite completa di strumenti per la post-produzione audio consente di registrare, editare e mixare audio con la stessa precisione dedicata alla parte visiva. Il software

supporta una vasta gamma di formati e risoluzioni, permettendo di lavorare su tutto, dai video HD ai film in 8K, garantendo una flessibilità che soddisfa le esigenze di qualsiasi progetto.

2.3.1.1 Spiegazione dell'utilizzo fatto del software DaVinci Resolve

DaVinci Resolve è un programma complesso e ricco di funzionalità, quindi ci concentreremo solo su alcuni elementi chiave utili per il nostro obiettivo finale. L'interfaccia del software è suddivisa in diverse sezioni distinte. La prima è la sezione Media, dove è possibile importare immagini e video nel progetto. Se si dispone di sequenze di immagini con nomi identici, ma con un valore numerico finale diverso, il software le interpreta come un unico file, facilitando così la creazione di video a partire da sequenze di immagini o accelerando semplicemente il processo di importazione.

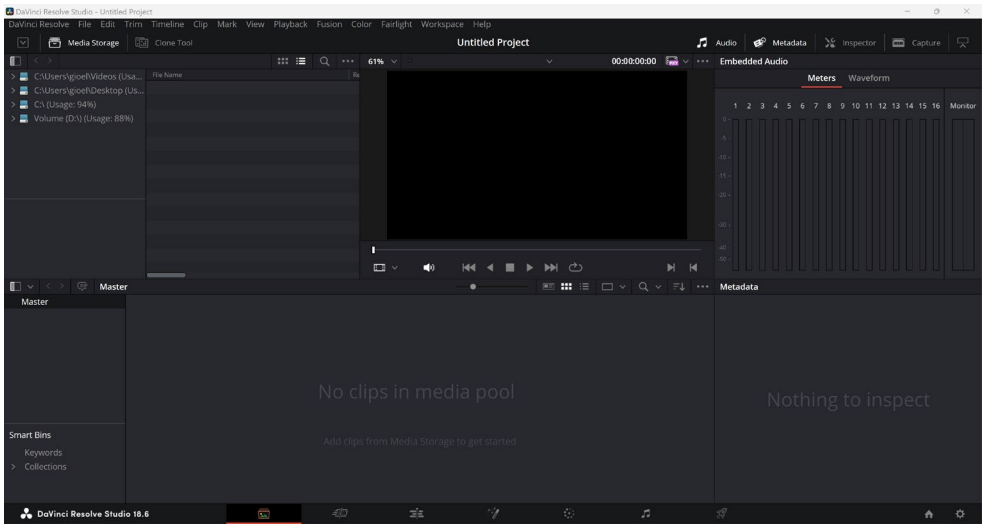

Figura 2.74: Sezione Media

Nelle sezioni successive, Cut e Edit, si possono prendere le timeline, ovvero i "contenitori" dei video o delle immagini importate, e inserirle nella timeline corrente per modificarle, sovrapporle ed eseguire molte altre operazioni. La differenza tra le due sezioni risiede nelle funzionalità specifiche che ciascuna offre.

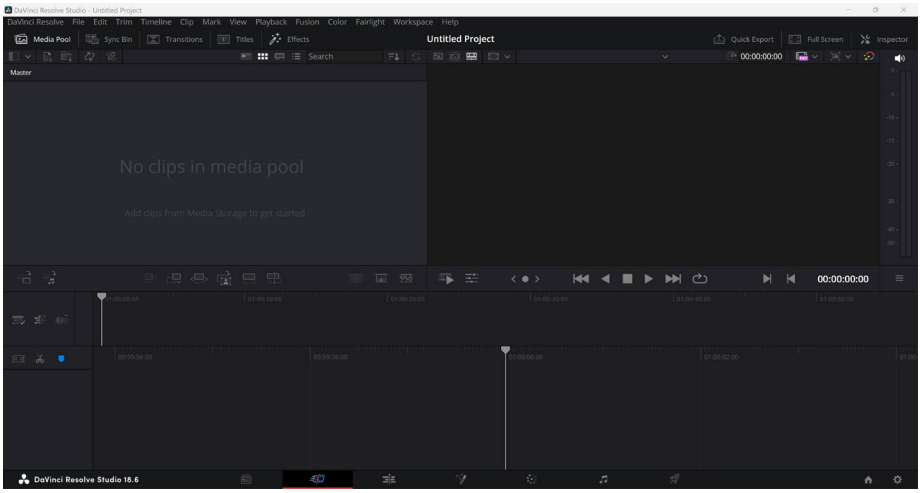

Figura 2.75: Sezione Cut

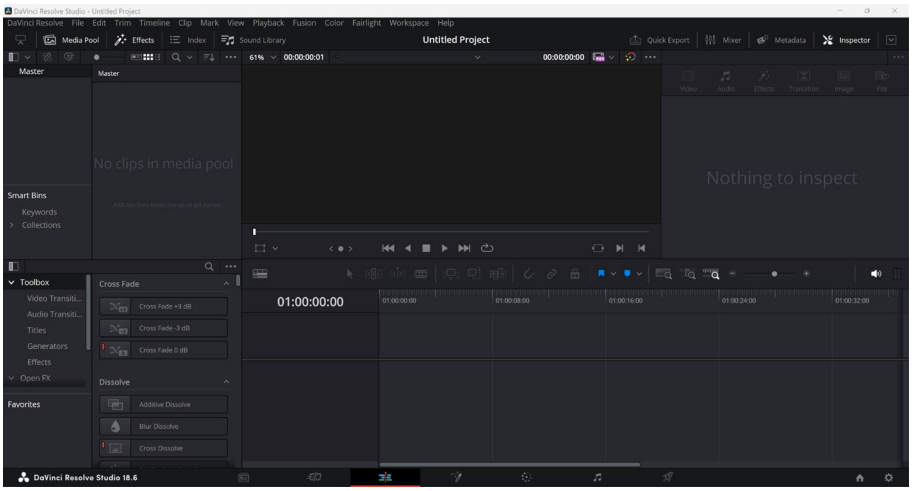

Figura 2.76: Sezione Edit

La sezione Color è dedicata alla modifica dei colori, offrendo tutte le funzionalità avanzate per il color grading, essenziali per ottenere l'aspetto visivo desiderato nei progetti video.

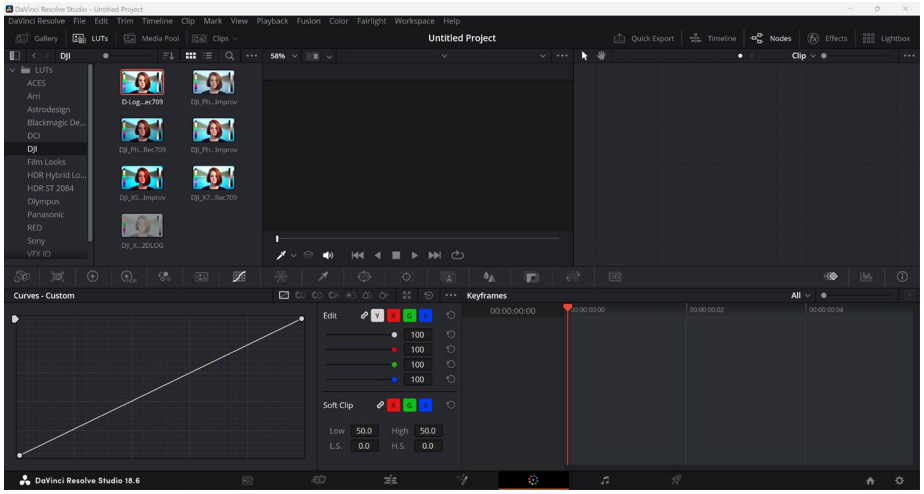

Figura 2.77: Sezione Color

L'ultima sezione, chiamata Deliver, contiene tutte le impostazioni di output per il video, come formato, codec, frame rate e molte altre opzioni. È da questa sezione che si esegue il rendering finale del video del progetto.

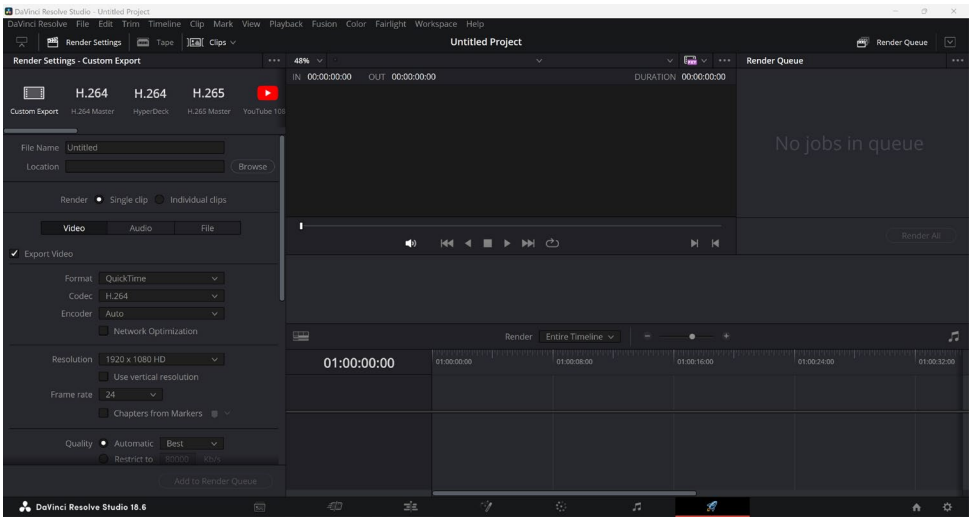

Figura 2.78: Sezione Deliver

2.3.2 Reality Capture

RealityCapture è un software di punta per la fotogrammetria e la scansione 3D, sviluppato da Capturing Reality, azienda acquisita da Epic Games nel 2021. Lanciato per la prima volta nel 2016, RealityCapture ha rapidamente ottenuto riconoscimenti nel settore grazie alla sua capacità di trasformare fotografie e scansioni laser in modelli 3D di alta precisione. Un momento cruciale nella storia del software è stata l'acquisizione da parte di Epic Games, che ha ulteriormente migliorato le funzionalità di RealityCapture e integrato la sua tecnologia nel celebre motore di gioco Unreal Engine, ampliando così le possibilità per la creazione di contenuti 3D. RealityCapture si distingue per la sua efficienza e accuratezza nel processare grandi volumi di dati, generando modelli 3D fotorealistici: infatti, esso utilizza algoritmi avanzati per combinare immagini e dati di scansione laser, permettendo agli utenti di creare repliche digitali estremamente dettagliate di oggetti, edifici, paesaggi e intere città. Questa caratteristica lo rende indispensabile in settori come l'architettura, l'ingegneria, l'archeologia e la conservazione dei beni culturali, dove la documentazione dettagliata e la modellazione accurata sono essenziali. Una delle principali caratteristiche di RealityCapture è la sua velocità: è riconosciuto come uno dei software più rapidi nel campo della fotogrammetria, capace di elaborare grandi quantità di dati in tempi significativamente inferiori rispetto ai concorrenti. Questo è particolarmente vantaggioso per i professionisti che necessitano di consegnare risultati rapidamente senza compromettere la qualità. Inoltre, RealityCapture supporta una vasta gamma

di formati di input e output, garantendo un'ampia compatibilità con diversi flussi di lavoro e facilitando l'integrazione con altri strumenti e piattaforme. Le sue funzionalità includono la gestione automatica delle immagini, la calibrazione delle telecamere, la generazione di mesh e texture, e strumenti di editing per perfezionare i modelli 3D. Il software viene costantemente aggiornato per includere nuove tecnologie e miglioramenti, mantenendo la sua posizione di leader nel campo della fotogrammetria e della scansione 3D.

2.3.2.1 Spiegazione dell'utilizzo fatto del software Reality Capture

L'utilizzo di RealityCapture per il nostro scopo risulta piuttosto intuitivo; pertanto, possiamo partire dall'interfaccia grafica per una panoramica delle funzionalità.

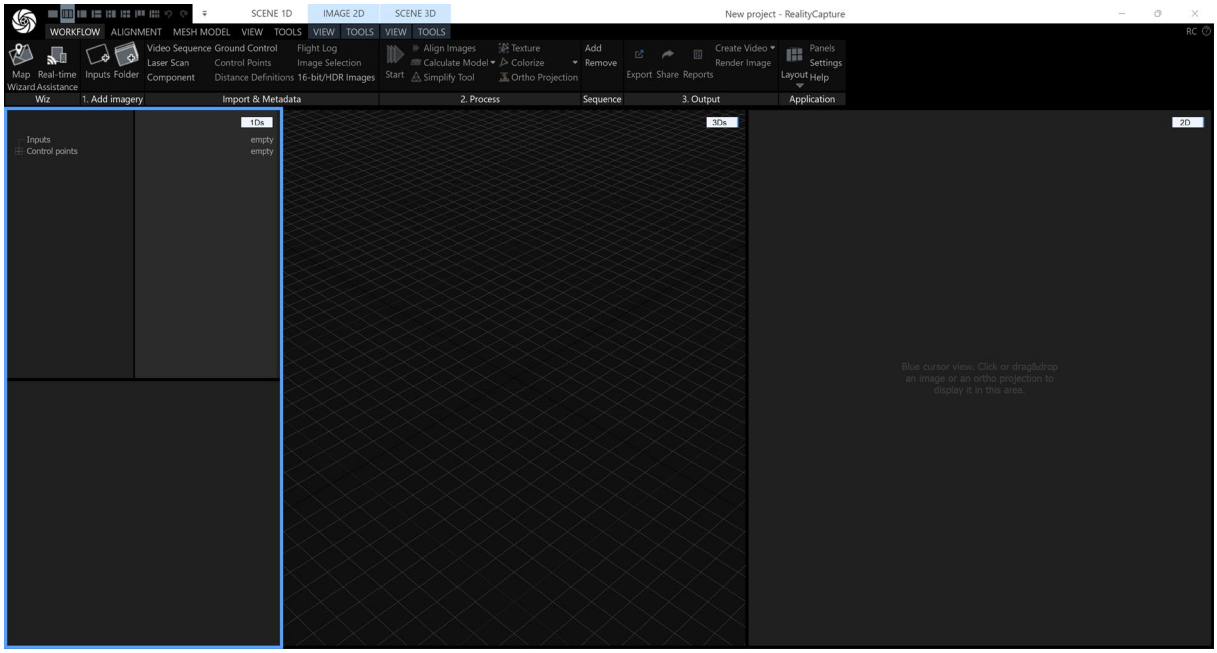

Figura 2.79: Interfaccia Reality Capture

Questa è la sezione principale del programma, denominata Workflow, che offre la possibilità di eseguire molte operazioni tramite la toolbar in alto. Le operazioni principali includono "Inputs" o "Folder" per aggiungere immagini o una cartella contenente immagini da utilizzare per creare la fotogrammetria. Nella seconda sezione della toolbar è presente il tasto "Start", che consente di eseguire tutte insieme le principali operazioni del programma. In alternativa, è possibile eseguire le operazioni una alla volta utilizzando i tasti alla destra di "Start", permettendo così un controllo più preciso del flusso di lavoro.

Cambiando sezione, la toolbar si adatta e propone le operazioni specifiche per quella sezione. Ad esempio, nella sezione "Alignment" è possibile effettuare operazioni di allineamento delle immagini, mentre nella sezione "Mesh Model" si trovano le operazioni

relative alla modellazione della mesh. Per il nostro progetto, abbiamo utilizzato principalmente la sezione "Mesh Model", con un focus particolare sulle impostazioni delle sezioni "Create Model" e "Mesh Color & Texture".

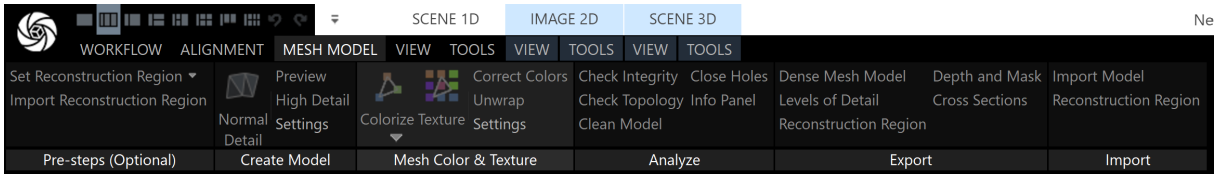

Figura 2.80: Toolbar della sezione Mesh Model

Cliccando sui rispettivi "Settings", si apriranno in basso a destra le impostazioni corrispondenti, che permettono di personalizzare la creazione del modello e delle texture a partire dalle foto.

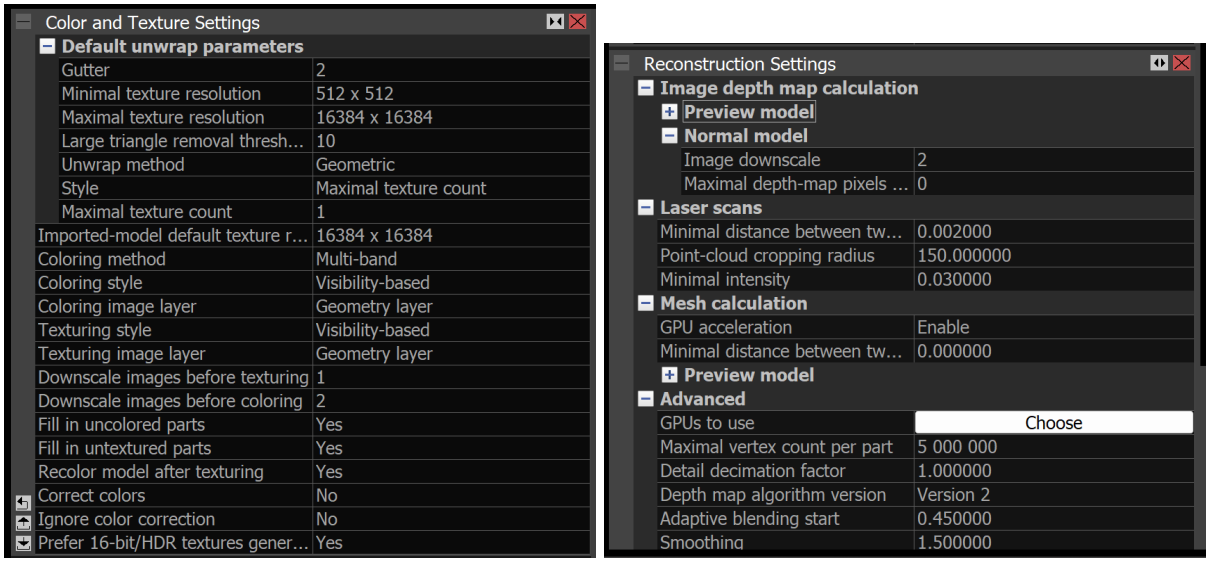

Figura 2.81: Impostazioni per la creazione di textures e modelli

Come si può vedere, ci sono numerose impostazioni, tuttavia, quelle di maggior interesse sono:

- "Minimal/Maximal texture resolution", che serve a impostare la risoluzione minima e massima delle texture generate. Per rappresentazioni fotorealistiche, questi valori devono essere impostati a livelli molto alti.
- "Image downscale", che controlla la qualità della creazione del modello. Più alto è il valore impostato, più la qualità è ridotta.

Attraverso questo sowftware, è quindi possibile ottenere modelli 3D altamente dettagliati e realistici, fondamentali per la creazione di scene fotorealistiche che rappresentano accuratamente i siti archeologici egizi, permettendo di realizzare timelapse giorno/notte che valorizzano al massimo le potenzialità di Unreal Engine.

2.3.3 Agisoft Delighter

Agisoft Delighter è un software avanzato progettato per migliorare la qualità visiva e l'accuratezza dei modelli 3D fotogrammetrici. Sviluppato da Agisoft, nota per il suo software di fotogrammetria Metashape, Delighter è stato introdotto come strumento complementare per il post-processing dei modelli 3D, con l'obiettivo specifico di rimuovere effetti indesiderati di illuminazione e migliorare la consistenza delle texture. Delighter risponde all'esigenza di ottenere modelli 3D più realistici e coerenti dal punto di vista visivo: nella fotogrammetria, le immagini catturate in condizioni di illuminazione variabili possono generare ombre e riflessi che compromettono la qualità del modello 3D finale. Per affrontare questi problemi, Delighter utilizza algoritmi avanzati di analisi e correzione della luce, che permettono di normalizzare le condizioni di illuminazione sulle texture, eliminando ombre e riflessi indesiderati. Il software sfrutta gli input forniti dall'utente sulle zone d'ombra e di luce sul modello per analizzare le immagini, per poi applicare le correzioni necessarie per uniformare l'illuminazione su tutta la superficie del modello. Questo processo migliora significativamente la qualità delle texture, rendendo i modelli 3D più realistici e visivamente attraenti. In più, l'interfaccia utente di Agisoft Delighter è stata progettata per essere intuitiva e facile da usare, anche per chi non ha una vasta esperienza in fotogrammetria o post-processing. Gli utenti possono caricare i loro modelli 3D e le immagini associate, configurare i parametri di correzione dell'illuminazione e avviare il processo di delighing con pochi clic.

2.3.3.1 Spiegazione dell'utilizzo fatto del software Agisoft Delighter

Come di consuetudine, partire dall'interfaccia è utile per comprendere il funzionamento di base del programma.

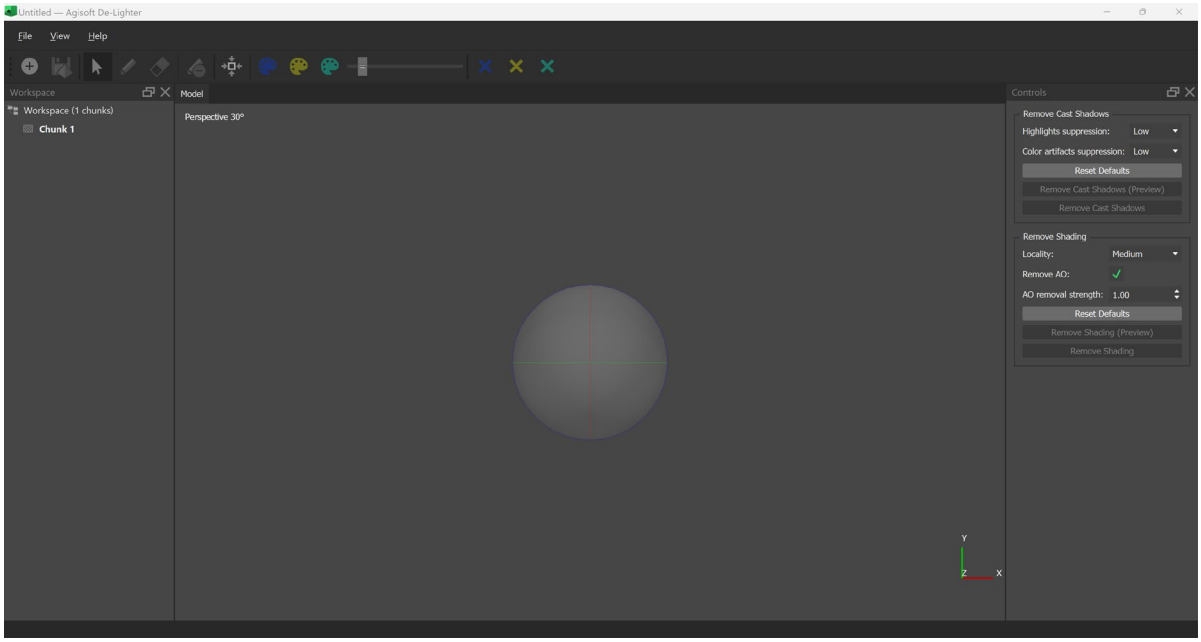

Figura 2.82: Interfaccia di Agisoft Delighter

L'interfaccia di Agisoft Delighter è progettata in modo minimalista e intuitivo. Nella parte superiore, troviamo una serie di operazioni che possono essere eseguite sul modello, il quale appare nella sezione centrale una volta caricato. Il primo pulsante della toolbar consente di aggiungere un modello al progetto. A seguire, ci sono altri pulsanti simili a quelli di Paint, come un pennello, una gomma e la possibilità di scegliere il colore del pennello tra blu, giallo e verde. Il pannello a destra, denominato "Controls", contiene le due principali operazioni che possono essere eseguite con questo programma, ciascuna con le proprie impostazioni per regolare la qualità:

- Remove Cast Shadows: Questa funzione si occupa di eliminare le ombre proiettate da un oggetto su altri oggetti o superfici, migliorando la coerenza visiva tra diversi elementi della scena.
- Remove Shading: Questa operazione si concentra sulle variazioni di ombreggiatura sulla superficie di un singolo oggetto, uniformando l'illuminazione su tutta la superficie per ottenere texture più consistenti e realistiche.

Ogni volta che si esegue una di queste operazioni, il modello viene salvato in cascata nel pannello a sinistra. Questo consente di applicare le operazioni in sequenza, lavorando sul modello precedentemente modificato, e di tornare indietro se il risultato di un'operazione non è soddisfacente. Questa struttura semplice e intuitiva facilita l'uso di Agisoft Delighter, permettendo agli utenti di migliorare i modelli 3D in modo efficiente e preciso.

2.3.4 Blender

Blender è un software di modellazione 3D open source estremamente versatile, sviluppato dalla Blender Foundation. Originariamente creato nel 1995 come progetto interno di un'azienda di animazione olandese chiamata NeoGeo, Blender è stato reso disponibile al pubblico nel 1998. Nel 2002, una campagna di raccolta fondi ha permesso di rendere il codice sorgente open source, consentendo alla comunità globale di partecipare attivamente al suo sviluppo: da allora, Blender ha visto una crescita continua in termini di funzionalità e popolarità, diventando uno degli strumenti preferiti sia dai professionisti del settore che dagli appassionati di grafica 3D. Blender è rinomato per la sua capacità di eseguire una vasta gamma di operazioni nel campo della grafica tridimensionale; infatti, le sue funzionalità includono modellazione, texturing, rigging, animazione, rendering, compositing e motion tracking. Inoltre, Blender offre strumenti avanzati per la simulazione fisica, come fluidi, fumo, particelle e tessuti, e supporta anche il video editing non lineare e la creazione di giochi tramite il Blender Game Engine. Un evento importante nella storia di Blender è stato l'introduzione del motore di rendering Cycles nel 2011: Cycles ha trasformato il flusso di lavoro di rendering in Blender, offrendo un motore di rendering path tracing fisicamente accurato, capace di produrre immagini di qualità fotorealistica e questo ha permesso agli utenti di creare scene con illuminazione realistica e materiali complessi. Un altro sviluppo significativo è stato l'introduzione di Eevee nel 2018, un motore di rendering in tempo reale che consente agli utenti di visualizzare immediatamente i risultati delle loro modifiche, migliorando notevolmente l'efficienza del workflow. Blender continua a evolversi grazie alla collaborazione della comunità globale di sviluppatori e utenti, consolidandosi come uno degli strumenti più completi e potenti per la grafica 3D disponibile oggi sul mercato.

2.3.4.1 Spiegazione dell'utilizzo fatto del software Blender

Anche Blender è un software altamente complesso e ricco di funzionalità; pertanto verrà analizzato in estrema sintesi, osservando l’interfaccia, con un focus sugli elementi chiave utili per il nostro scopo.

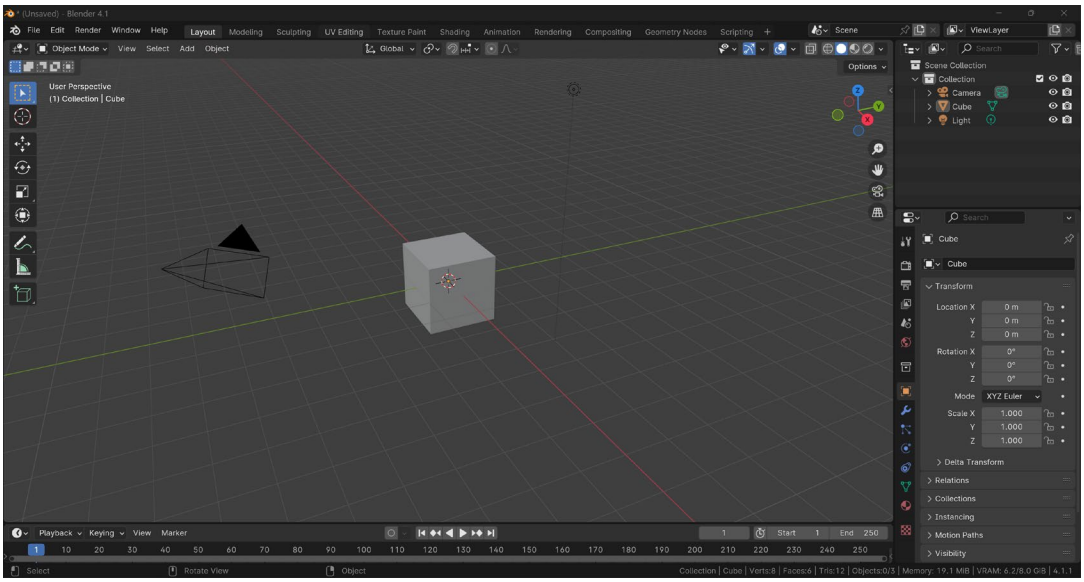

Figura 2.83: Interfaccia della Object Mode di Blender

Questa è l'interfaccia di base nella modalità Object Mode, come si può vedere dall'indicazione in alto a sinistra. Principalmente, in questa modalità, si posizionano gli oggetti in scena e si modificano le proprietà che coinvolgono l'intero oggetto, attraverso il pannello sulla destra, suddiviso in diverse sezioni e, in alto a destra, si ha una panoramica di quali oggetti sono presenti in scena. Oltre alla Object Mode, vi sono altre modalità, in particolare la Edit Mode: questa modalità permette di modellare la mesh modificando non solo i singoli vertici, ma anche le facce e gli edge (spigoli) del modello. Selezionando vertici, facce o edge, si possono spostare, eliminare o modificare per cambiare parti del modello in modo preciso e dettagliato e questa flessibilità consente un completo editing del modello, offrendo infinite possibilità di personalizzazione. Un altro aspetto importante di Blender è la sua capacità di gestire molti formati diversi, sia in import che in export, attraverso il menù a tendina File in alto a sinistra.

2.4 Il workflow per la fotogrammetria: dalle foto reali al modello su Unreal

Ora che abbiamo una panoramica più precisa di tutti gli elementi necessari per la realizzazione del timelapse giorno/notte di una fotogrammetria, possiamo analizzare più in dettaglio il workflow che permette di partire da un insieme di foto di un soggetto fino ad arrivare a un modello finito e rifinito da importare su Unreal Engine. Solo successivamente esamineremo come è stata realizzata la scena su Unreal con quel modello.

2.4.1 DaVinci Resolve: la correzione del colore

Il primo passo consiste nel disporre di un insieme sufficientemente grande di foto dello stesso soggetto e per ottenere una resa fotorealistica, si consiglia di mantenere la risoluzione più alta

possibile. Nel nostro caso, Robin Studio ha effettuato riprese con i droni, ottenendo foto ad una risoluzione di 6016x3200 pixels con una profondità di 24 bit[38]. Per grandi siti archeologici è utile avere una risoluzione così alta, mentre per piccoli oggetti come vasi o piante, riprese in 4K da smartphone possono essere sufficienti se effettuate correttamente: in ogni caso è importante riprendere l'oggetto da più inquadrature possibili per garantire una copertura completa. Le foto, nel nostro caso, sono state salvate in formato D-Log[39], ma anche salvarle in formato RAW – che significa che non hanno il profilo colore della fotocamera – andrebbe bene per mantenere un'alta qualità, ma in ogni caso i colori possono sembrare piatti o neutri finché non vengono processati. Per riottenere i colori originali è necessario elaborare le immagini applicando la LUT[40] della fotocamera, in modo da ottenere una texture con i colori corretti e questo processo si può effettuare su DaVinci Resolve.

Per prima cosa, si importano le foto nel software, facendo attenzione a impostare correttamente il formato della timeline che si andrà a creare, che deve essere uguale al formato delle fotografie per evitare di perdere informazioni; nel nostro caso, 6016x3200. Successivamente, la timeline deve essere inserita nella timeline del software, nella sezione Cut o Edit: nella sezione Cut, premendo con il tasto destro sulla traccia delle immagini nella timeline, si può scegliere l'opzione "Change Speed", che permette di velocizzare la sequenza di immagini. Anche se così facendo vengono persi alcuni frame, per le fotogrammetrie non sono necessarie molte immagini dello stesso punto, ma piuttosto tante immagini da angolature differenti. Nel nostro caso, con riprese a 30fps e un totale di circa 10.000 foto, la sequenza è stata velocizzata di 15 volte, permettendo così di avere "solo" più circa 700 foto, alleggerendo il processo del software successivo. Successivamente, nella sezione Color è possibile applicare la LUT: in alto a sinistra, si può aprire il menù per cercare tutte le LUT presenti nel software o scaricate e inserite in esso. Nel nostro caso, è stata scaricata la LUT "D-Log to Rec.709",

[38] La profondità in bit in informatica grafica rappresenta la capacità di un sistema di codificare i colori utilizzando, in questo caso, 24 bit per pixel, permettendo di visualizzare fino a 16,777,216 colori distinti (2^{24}). Questo livello di profondità di colore è standard per immagini ad alta definizione, garantendo una riproduzione cromatica molto precisa e dettagliata.

[39] Il formato D-Log, disponibile su alcune videocamere e droni DJI, è una modalità di registrazione video che cattura una vasta gamma dinamica di luminanza e colore, simile al formato RAW per le fotografie. Utilizzando una curva logaritmica, D-Log preserva maggiori dettagli nelle alte luci e nelle ombre, offrendo una maggiore flessibilità in post-produzione. Tuttavia, senza l'applicazione di una LUT (Look-Up Table), le immagini D-Log appaiono con colori piatti e desaturati, poiché la rappresentazione logaritmica riduce il contrasto e la saturazione per catturare l'intera gamma di luminanza della scena. La LUT è quindi fondamentale per convertire questa immagine logaritmica in una rappresentazione visivamente piacevole, ripristinando contrasto e colori naturali.

[40] Le LUT (Look-Up Table) sono strumenti utilizzati per modificare i colori e il contrasto di immagini e video. Trasformano un set di colori di input in un set di colori di output, migliorando l'aspetto visivo del contenuto, specialmente in post-produzione per ottenere un look specifico o correggere il colore.

strettamente vincolata al modello della fotocamera, che salva le foto in formato D-Log, quindi necessitava di quella specifica LUT per ottenere i colori nello standard Rec.709, tipicamente usato per la televisione ad alta definizione.

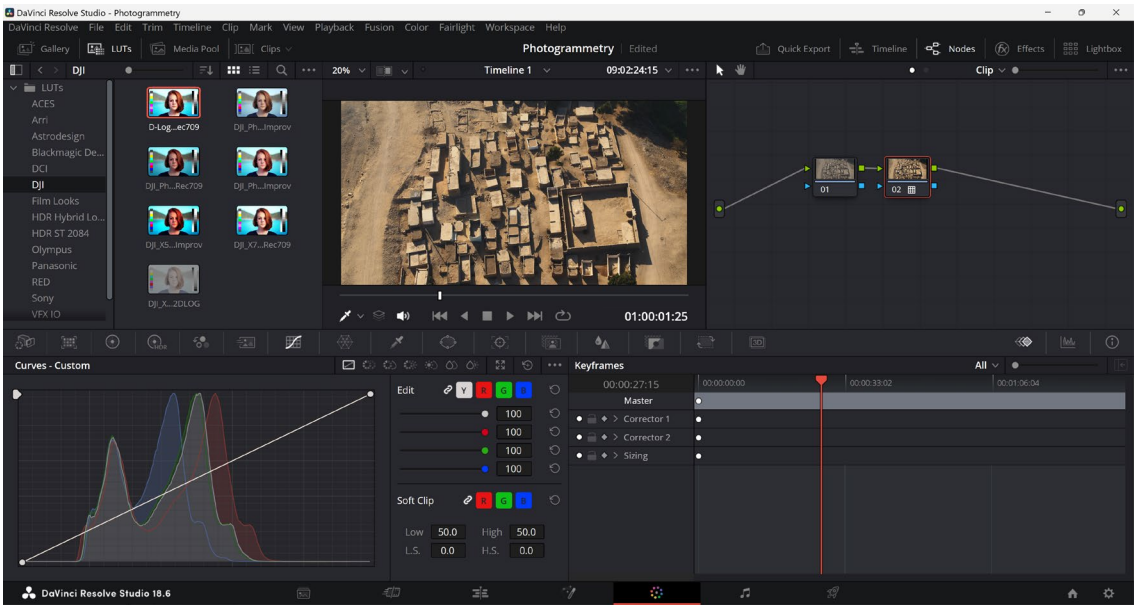

Figura 2.84: Sezione Color di DaVinci Resolve, con LUT applicata

Per applicarla alle immagini, basta cliccare con il tasto destro sul nodo presente nel pannello in alto a destra, scegliere “Add Node” e poi “Add Serial” per aggiungere un nodo successivo a quello già presente. Trascinando la LUT desiderata dal pannello delle LUT sopra il nodo appena creato, la LUT sarà applicata a tutte le immagini presenti nella timeline.

Figura 2.85: Foto originale e foto con LUT applicata

A questo punto si può procedere con l’export della sequenza delle immagini, scegliendo con attenzione i parametri. La risoluzione deve essere quella delle foto originali, e solitamente si sceglie il formato PNG poiché è un formato di immagini senza perdita di informazione durante la compressione, a differenza del JPEG. Nella sezione Codec, selezionando PNG, si può scegliere tra RGB 8 o 16 bits, e ovviamente si preferisce 16 bits per mantenere la maggiore qualità possibile.

2.4.2 Reality Capture: la creazione del modello

Ora che le immagini hanno la gradazione di colore corretta, possiamo procedere con l'utilizzo di Reality Capture per creare il modello 3D e le relative texture. Il primo passo consiste nell'importare le immagini nel progetto e anche se ci fossero più sequenze di immagini, il processo è lo stesso: si importa ogni sequenza individualmente: generalmente, utilizzare più sequenze da diverse angolazioni porta a risultati più precisi. Una volta importate tutte le foto, è fondamentale configurare le impostazioni del software per ottenere il risultato desiderato: le impostazioni principali che potrebbero interessare sono quelle relative alla qualità delle operazioni. Per il nostro caso, puntiamo a mantenere una qualità molto alta:

- Nelle impostazioni di allineamento, l'opzione "Image Downscale Factor" è impostata a 1, per evitare perdite di qualità.
- Nella sezione Create Model, l'opzione "Image Downscale" è impostata a 2, mentre "Minimal Distance between Two Vertices" è settata a 0, per ottenere una mesh il più dettagliata possibile.
- Nella sezione Mesh Color & Texture, le impostazioni per la risoluzione delle texture ("Texture resolution") sono settate ad alta qualità (16K). I parametri di "Downscale" per la qualità delle foto utilizzate nella creazione delle texture e dei colori sono rispettivamente impostati a 1 e 2.

Dopo aver configurato queste opzioni, si può procedere con l'allineamento delle immagini, la creazione delle texture e del modello. Sebbene sia possibile eseguire tutte queste operazioni in sequenza automaticamente premendo il pulsante "Start", questo approccio potrebbe comportare la perdita di alcune informazioni. Per questo motivo, è spesso più sicuro eseguire prima solo l'allineamento delle immagini: in questo modo, il software identifica la posizione di ogni foto (rappresentata da piccole icone di telecamere bianche nella sezione 3D) e crea una versione puntiforme del modello. Questo processo aggiunge anche un parallelepipedo che circonda il modello, noto come bounding box, il quale limita la creazione di texture e modello allo spazio interno ad esso e, a volte, questo bounding box potrebbe non essere delle dimensioni appropriate, risultando troppo grande o troppo piccolo; pertanto, è utile regolarlo manualmente, spostando i suoi lati per racchiudere esattamente la zona di interesse e solo a questo punto si procede con la creazione delle texture e del modello.

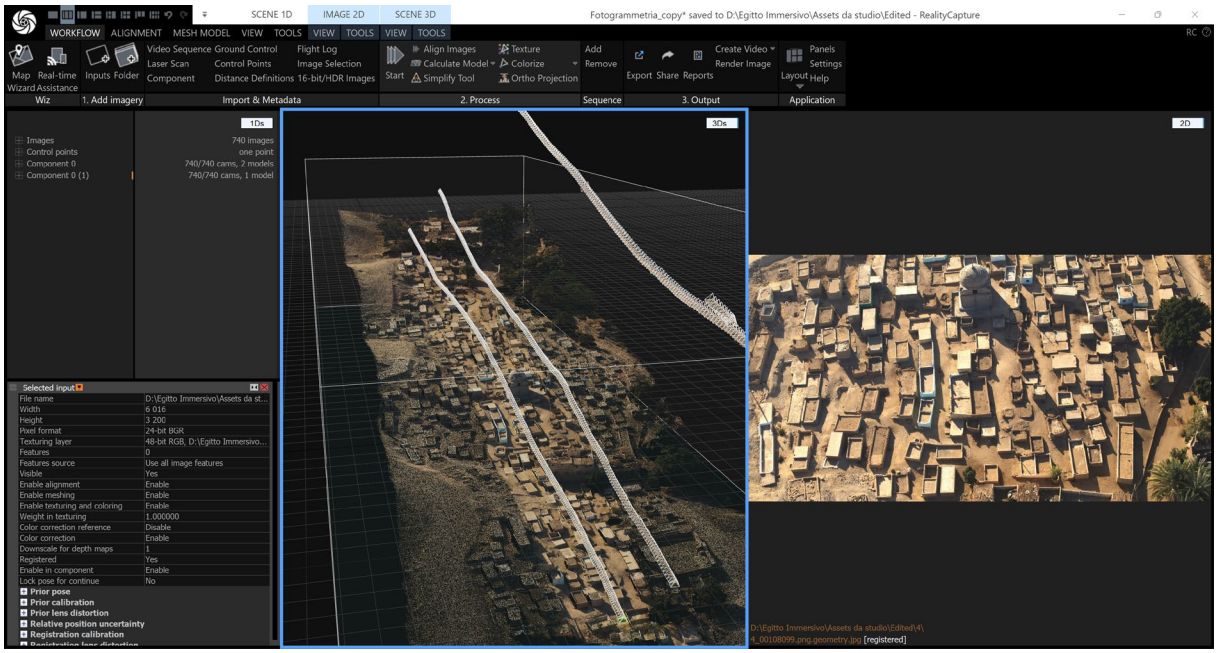

Figura 2.86: Interfaccia di Reality Capture con modello, posizione telecamere e bounding box

Nel caso dell'immagine, il modello e le texture sono già state create, e l'ultimo passo è esportarle tramite l'opzione nella sezione Export del menù Mesh Model, scegliendo il formato che si vuole.

2.4.3 Agisoft Delighter: l'eliminazione delle ombre

A questo punto il modello sembrerebbe essere pronto; tuttavia, esiste un problema dovuto alle ombre presenti su di esso, causato dal fatto che le foto originali erano state scattate con ombre reali. Per utilizzare il modello in modo ottimale e poter scegliere l'illuminazione più adatta su Unreal Engine, è necessario rimuovere le ombre già esistenti, così da poterle ricreare in modo coerente con la nuova scena.

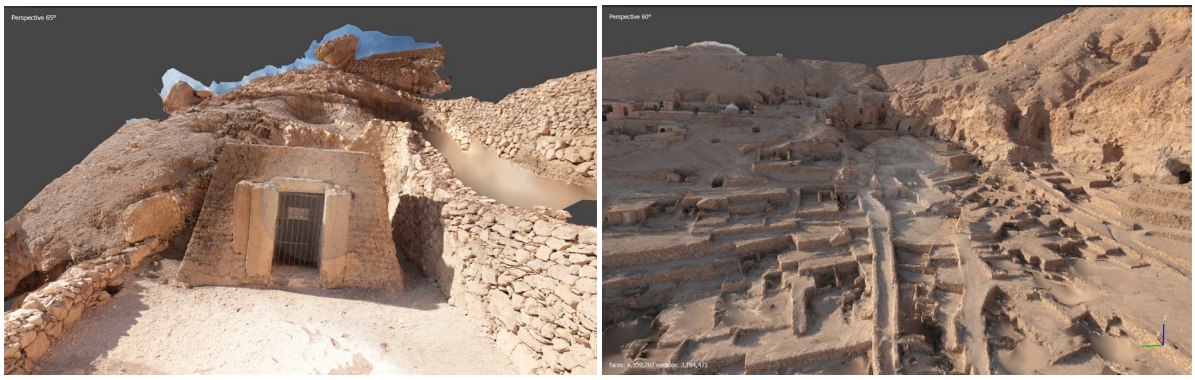

Figura 2.87: Modelli su Agisoft Delighter con ombre ancora presenti, a sinistra la tomba di KHA, a destra il sito archeologico di Medina

Come si può vedere dalle immagini, i modelli presentano ombre che variano a seconda dell'ora del giorno in cui sono state scattate le foto. Per eliminare queste ombre e uniformare l'illuminazione su tutta la mesh, è necessario utilizzare i pennelli giallo e blu di Agisoft Delighter. Il pennello blu identifica le aree in ombra, mentre quello giallo indica le zone con maggiore illuminazione. Questo processo aiuta il software a comprendere la direzione della luce nella scena, consentendogli di rimuovere correttamente le ombre.

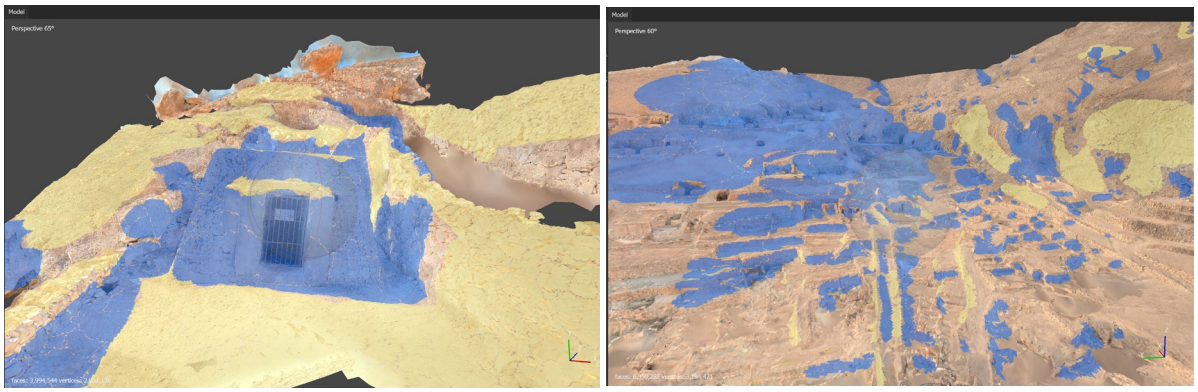

Figura 2.88: I modelli su Agisoft Delighter con le evidenziazioni

Non è necessario colorare l'intera mesh, soprattutto per modelli grandi e complessi come quello del sito archeologico di Medina. È sufficiente evidenziare le ombre principali e i punti con illuminazione più intensa. Dopo aver segnato le aree con i pennelli, si procede con l'operazione di "Remove Shading", che inizia a uniformare l'illuminazione e i colori del modello.

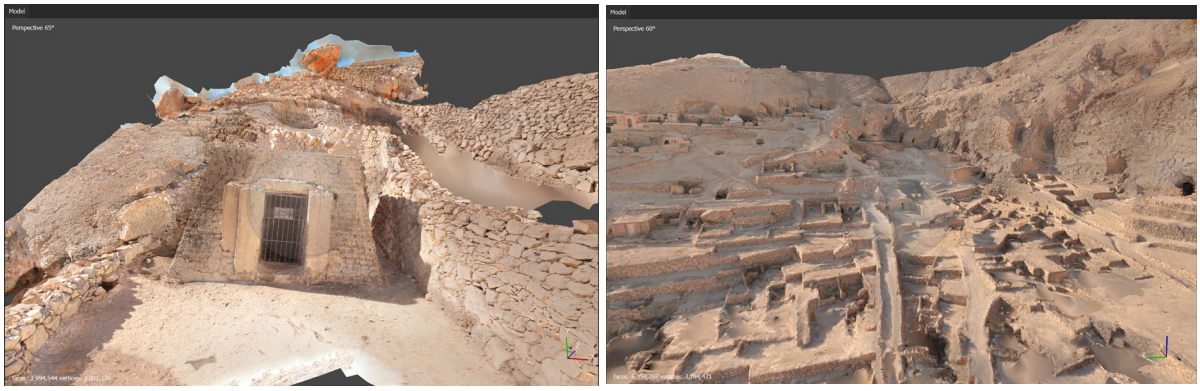

Figura 2.89: I modelli con applicato "Remove Shading"

Successivamente, si utilizza la funzione "Remove Cast Shadows", che elimina il più possibile le ombre dal modello.

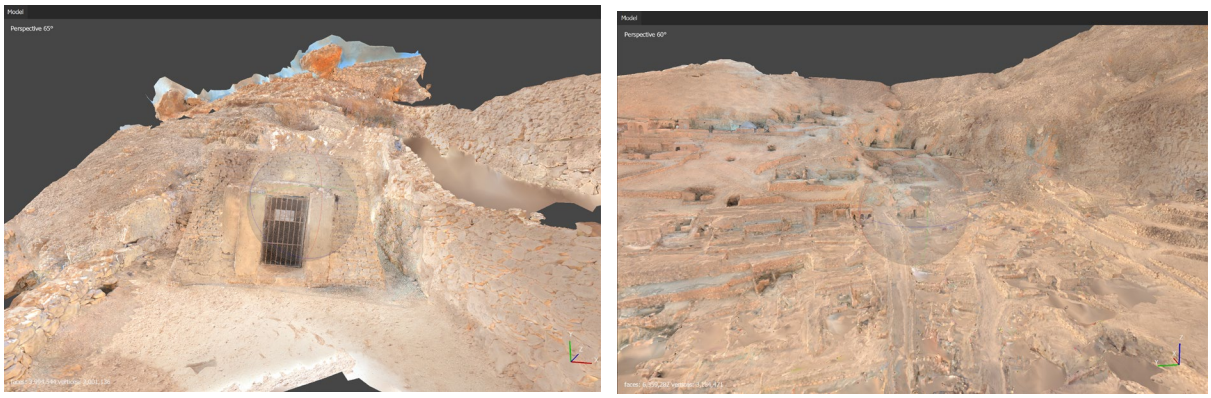

Figura 2.90: I modelli con anche applicato "Remove Cast Shadows"

Il risultato potrebbe inizialmente apparire meno realistico, poiché l'assenza di ombre può ridurre l'effetto di profondità. Tuttavia, questo è un passo necessario per garantire che il modello possa essere illuminato correttamente in Unreal Engine: una volta importato nel motore grafico, con un'illuminazione adeguata, il modello tornerà a sembrare realistico, recuperando l'aspetto dettagliato e la profondità visiva che ci si aspetta. L'ultimo passaggio è quello di premere nel menù a tendina su File, e cliccare Export Model per esportare il modello.

2.4.4 Blender: ultimi ritocchi

I modelli ottenuti fino a questo punto potrebbero essere già utilizzabili; tuttavia, spesso presentano imperfezioni come zone non fotografate che risultano mancanti nel modello, buchi, o sporgenze mal definite. Prima di importarli su Unreal, è utile passare attraverso Blender per poterli editare e correggere questi difetti.

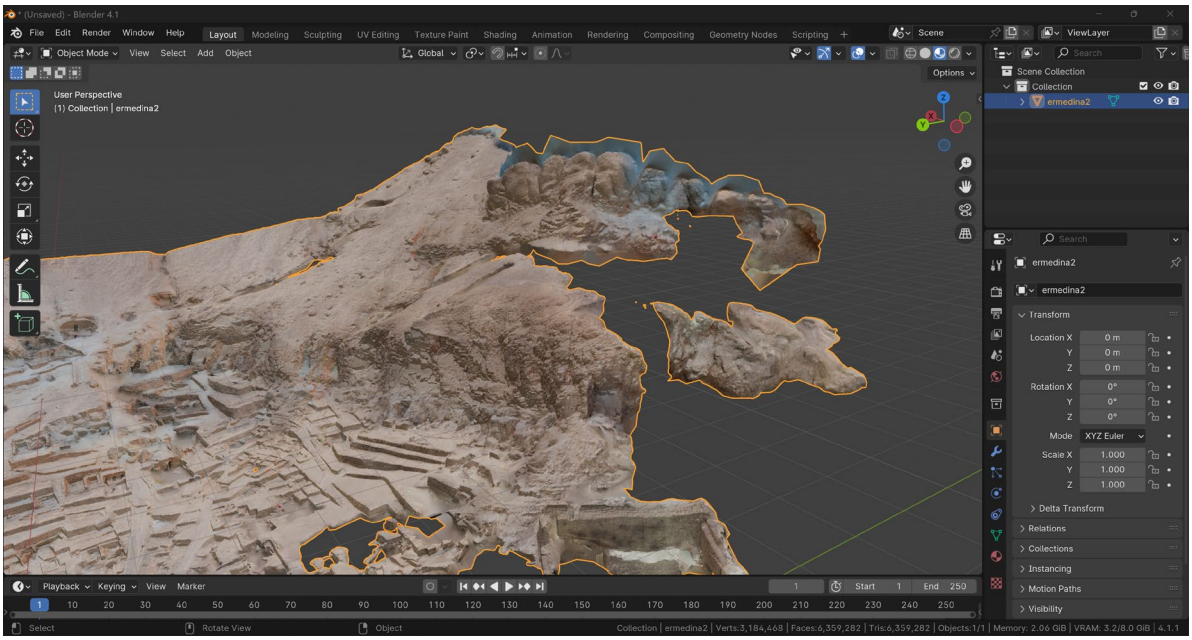

Figura 2.91: Il modello di Medina su Blender

Ad esempio, se si desidera inquadrare una specifica zona del sito archeologico di Medina, si potrebbero riscontrare gravi imperfezioni come parti blu in cima alla collina o cumuli di terra separati dal resto, quindi, in base al tipo di inquadratura e alle scelte artistiche, si può editare il modello per migliorarlo. Aprendo l'Edit Mode, è possibile eliminare i vertici in eccesso e aggiungere vertici o facce dove ci sono dei buchi, rendendo il modello più coerente e realistico.

Figura 2.92: Il modello di Medina su Blender, con applicate alcune correzioni

In una possibile inquadratura del sito archeologico, si può vedere come metà della zona blu sia stata eliminata, rendendo la cima della montagna più realistica, mentre l'altra metà è stata lasciata intatta per mostrare il miglioramento: continuando in questo modo, si può perfezionare il modello. Una volta soddisfatti del risultato, il modello deve essere esportato; per fare ciò, si va su File e poi su Export, scegliendo il formato desiderato, che nel caso di Unreal è FBX. Si aprirà un menù che permetterà di scegliere il luogo in cui salvare il file e di personalizzare alcune impostazioni di export. Una delle impostazioni consigliate è “Apply Transform”, che applica tutte le trasformazioni sul modello prima di esportarlo, garantendo che il modello esportato sia identico a quello visibile nella viewport di Blender.

2.4.5 Unreal Engine: la scena finale

I passaggi precedenti garantiscono che il modello finale sia pronto per essere importato in Unreal Engine, e per fare ciò basterà trascinare il file nel Content Browser nell’interfaccia di Unreal; una volta fatto questo, si può prendere la static mesh appena aggiunta e trascinarla nella scena. Oltre al modello, per realizzare una transizione giorno/notte di alta qualità e ottenere un'atmosfera realistica, è stato inserito nella scena anche il blueprint dell'UltraDynamicSky.

Il primo passaggio consiste nel posizionare correttamente la camera per creare una scena ben inquadrata e nel creare il consueto Level Sequence per il rendering della scena. Per realizzare il timelapse delle ore del giorno, si deve includere nel Level Sequence anche il blueprint dell'UltraDynamicSky. Questo blueprint contiene una variabile essenziale chiamata "Time of the Day", che regola il tempo, muovendo il sole e facendo sorgere la luna e le stelle. Sarà sufficiente impostare i keyframe per far avanzare questo valore da un'ora all'altra, in base alle esigenze della scena. Il secondo passaggio è la modifica dei parametri dell'UltraDynamicSky per rendere la scena il più realistica possibile: per la scena che raffigura Medina, seguendo i consigli di chi ha esperienza diretta con l'Egitto, sono stati regolati alcuni parametri per ottenere un cielo privo di nuvole e con una quantità discreta di foschia. Inoltre, ricercando online le coordinate esatte del sito archeologico e impostandole nel pannello Details, è stato possibile riprodurre fedelmente il ciclo diurno e notturno e il posizionamento del sole, migliorando così l'accuratezza della scena.

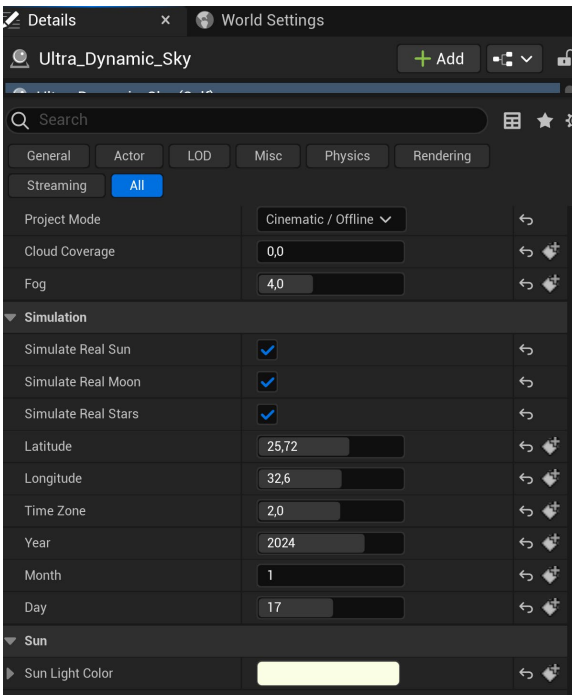

Figura 2.93: Pannello Details dell'UltraDynamicSky

Si è anche impostato un colore del sole leggemente sul giallo, e una modalità di progetto “Cinematica/Offline” per ottenere la massima resa artistica. A seguire tre momenti della giornata su Medina:

Figura 2.94: Medina in pieno giorno

Figura 2.95: Medina al tramonto

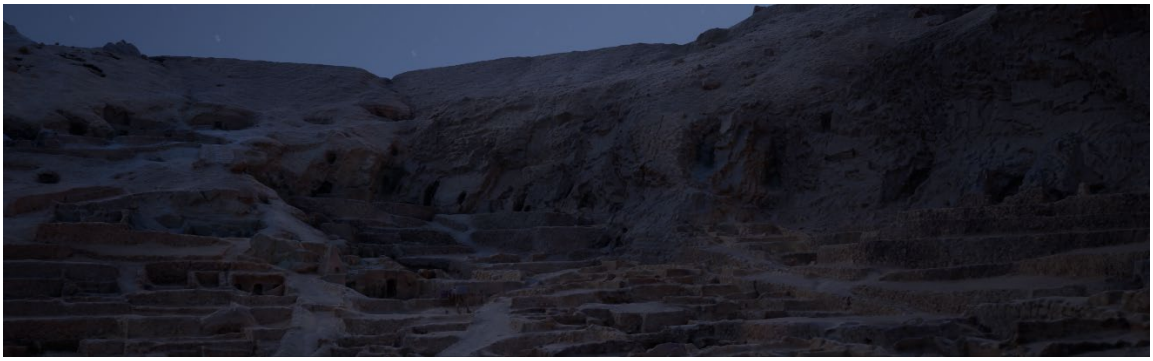

Figura 2.96: Medina di notte

2.5 Specifiche tecniche del rendering

Come ultimo capitolo riguardante Unreal, si illustrerà il processo di rendering delle scene e le metodologie utilizzate. Per effettuare rendering professionali, è stato necessario installare il plugin “Movie Render Queue” su Unreal Engine, che consente un'alta personalizzazione del tipo di rendering e della qualità desiderata.

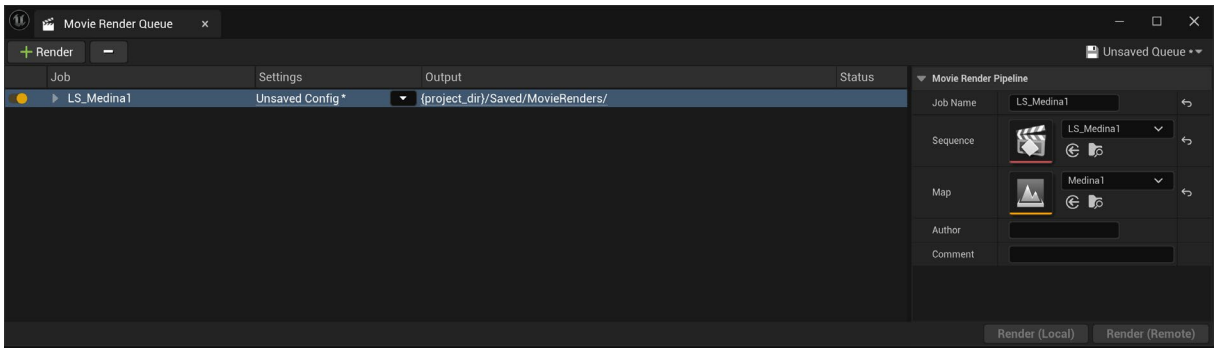

Figura 2.97: Finestra su Unreal Engine del "Movie Render Queue"

La finestra del Movie Render Queue, accessibile tramite il pulsante di rendering nel Level Sequence Editor, si presenta come nell'immagine. Se si premesse il tasto Render in basso, verrebbe renderizzata la sequenza "LS_Medina1" del livello "Medina1", utilizzando le impostazioni di default presenti in "Unsaved Config" nella colonna Settings, e aprendo queste ultime, è possibile modificare il comportamento del rendering.

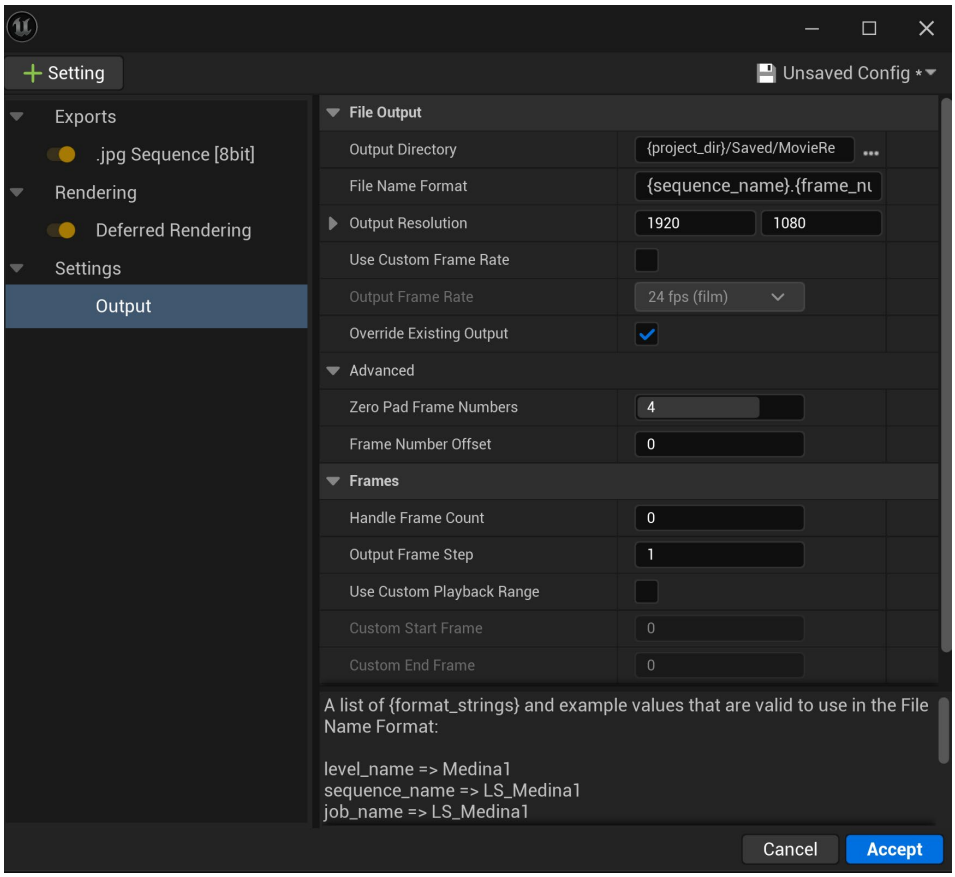

Figura 2.98: Finestra delle impostazioni di rendering

Una volta aperte, le impostazioni mostrano varie opzioni di configurazione. In questo esempio, vengono visualizzate le impostazioni predefinite, ma è possibile personalizzare il comportamento in modo significativo. Oltre al modulo Output, sempre presente e necessario

per definire parametri di base come risoluzione, frame rate e nome del file, sono disponibili ulteriori moduli che possono essere aggiunti, rimossi o disattivati premendo su "+Setting", infatti, attraverso questa finestra, si può ottimizzare ogni dettaglio del rendering per garantire la massima qualità visiva e la perfetta corrispondenza con le esigenze del progetto. Nel nostro progetto, sono state impiegate due configurazioni di rendering distinte a seconda delle esigenze grafiche delle scene: Path Tracing e Lumen. Il Path Tracing è una tecnica di rendering avanzata che simula il comportamento della luce in modo fisicamente accurato, producendo immagini di elevato realismo e, a differenza dei metodi tradizionali, che utilizzano algoritmi di approssimazione per calcolare l'illuminazione, il Path Tracing segue i percorsi dei raggi di luce dall'occhio dell'osservatore attraverso ogni pixel della scena, rimbalzando su varie superfici fino a raggiungere una fonte luminosa e questo permette di simulare effetti complessi come riflessioni, rifrazioni, ombre morbide e caustiche[41] con un elevato grado di realismo. Il Path Tracing inizia con l'emissione di raggi di luce dalla camera virtuale, che percorrono la scena e interagiscono con gli oggetti, rimbalzando in modo stocastico[42] e ad ogni interazione, vengono calcolate la riflessione e la rifrazione in base alle proprietà fisiche delle superfici colpite. Questo metodo utilizza un modello di illuminazione globale, che tiene conto della luce indiretta proveniente da altre superfici, contribuendo a creare immagini con una illuminazione naturale e coerente. Tuttavia, il Path Tracing è computazionalmente molto intensivo poiché richiede un gran numero di campioni per pixel per ridurre il "noise", ovvero il rumore, e ottenere un'immagine pulita. Questo può rendere il processo di rendering molto lento, ma con l'avanzamento dell'hardware e delle tecniche di ottimizzazione, come l'uso di GPU e algoritmi di denoising, il Path Tracing sta diventando sempre più praticabile per applicazioni real-time. Lumen, invece, è un sistema di illuminazione globale dinamica introdotto in Unreal Engine 5, progettato per fornire un'illuminazione realistica in tempo reale senza la necessità di pre-computazioni. Questa strategia utilizza tecniche avanzate per calcolare l'illuminazione indiretta e le riflessioni, adattandosi dinamicamente ai cambiamenti nella scena, come il movimento degli oggetti o le variazioni delle sorgenti luminose. Sebbene Lumen offra un'illuminazione globale dinamica che si aggiorna in tempo reale, consentendo di vedere immediatamente come le modifiche all'ambiente influenzano la scena, la qualità delle ombre e della luce riflessa può essere leggermente inferiore rispetto al Path Tracing: questo è dovuto al fatto che Lumen

[41] Le caustiche sono pattern di luce concentrata formati da riflessioni o rifrazioni attraverso superfici curve, come l'acqua o il vetro, creando effetti luminosi distintivi.
[42] Il termine "stocastico" si riferisce a processi o modelli che incorporano variabilità casuale o probabilistica, spesso utilizzati in simulazioni e algoritmi.

utilizza tecniche di approssimazione per mantenere alte le prestazioni, sacrificando un po' di precisione e fedeltà visiva rispetto ai metodi basati su simulazioni fisicamente accurate. Tuttavia, questa compromissione è compensata dalla velocità di rendering, rendendo Lumen particolarmente adatto per applicazioni real-time come i videogiochi e le esperienze interattive. Lumen migliora notevolmente la qualità delle scene interne, dove la luce naturale entra attraverso finestre e fessure, rimbalzando sulle superfici e illuminando l'ambiente; con esso, Unreal Engine 5 offre agli sviluppatori la possibilità di creare mondi virtuali altamente realistici senza compromessi significativi sulla qualità visiva, bilanciando al contempo prestazioni e realismo.

In generale, considerando che le nostre scene sono tutte esterne e richiedono un'alta qualità visiva, e dato che non è necessario renderizzare velocemente poiché il rendering è offline, la scelta principale è ricaduta sul path tracing. Le impostazioni di rendering per questa modalità sono state le seguenti:

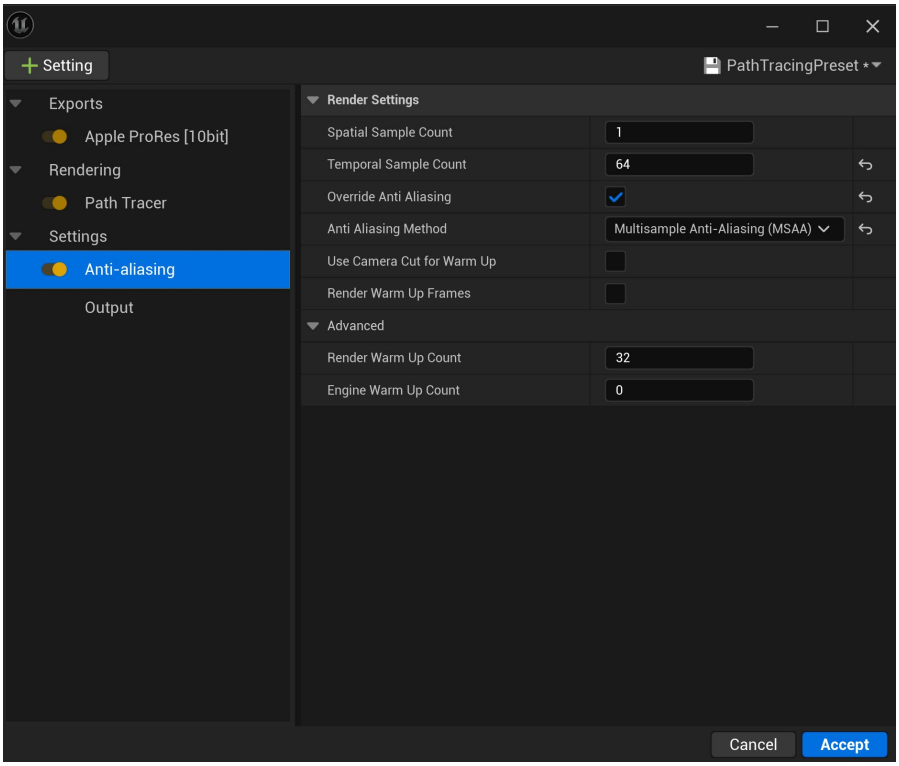

Figura 2.99: Impostazioni di rendering per Path Tracing

- Apple ProRes [10 bit]: Questo è il formato in output del render finale, indipendentemente dal tipo di render scelto (Lumen o path tracing). È stato necessario scaricare un plugin per avere questa opzione tra i Setting disponibili, ma è uno dei pochi

formati che permettono una profondità in bit di 10 (rispetto ai consueti 8 bit). All'interno di questo modulo, è stata scelta la configurazione del codec "Apple ProRes 422HQ"[43]

- Path Tracer: Questo modulo attiva il render in path tracing, e al suo interno è stata attivata la spunta "Reference Motion Blur", che migliora il risultato finale nella fase di eliminazione del rumore.
- Anti-Aliasing: Questo modulo contiene le impostazioni riguardanti l'anti-aliasing[44]. Viene utilizzato anche per altre modalità di render per eliminare il rumore, ma i valori cambiano: con il path tracing, sono stati scelti 64 "Temporal Sample" ovvero vengono renderizzati 64 frame ad ogni frame da combinare insieme per ottenerne uno unico finale. Come metodo di anti-aliasing è stato selezionato il Multisample Anti-Aliasing (MSAA)[45]. Inoltre, per ridurre il rumore fin dai primi frame, sono stati impostati a 32 i frame di warm-up, il che significa che il motore di render produce 32 frame preliminari per sfruttare le loro informazioni e migliorare la qualità dei frame successivi.
- Output: Questo è il modulo sempre presente, in cui si impostano i parametri di base. Nel nostro caso, sono stati scelti 30 fps e una risoluzione di 3550x1080. La risoluzione è in questo formato particolare perché la sala immersiva ha pareti estremamente larghe ma un'altezza contenuta.

Il path tracing è stato utilizzato per la maggior parte delle scene, tuttavia presenta dei problemi con l'acqua. Fino ai primi mesi del 2024, l'acqua non veniva renderizzata con il path tracing, indipendentemente dal modello scelto, ma fortunatamente, con i nuovi aggiornamenti, l'acqua è ora renderizzata, ma le scene subacquee presentano un grave problema: quando la camera passa sotto il livello dell'acqua, la luce non penetra più, rendendo la scena completamente nera. Per questo motivo, è stato utilizzato anche Lumen come strategia di rendering alternativa, con le seguenti impostazioni:

[43] Apple ProRes 422HQ utilizza la sottocampionatura della crominanza 4:2:2, una tecnica che comprime le informazioni di colore riducendo la risoluzione delle componenti cromatiche (Cb e Cr) rispetto alla luminanza (Y). La luminanza rappresenta la luminosità dell'immagine, mentre la crominanza rappresenta le informazioni di colore. Con il campionamento 4:2:2, la luminanza è campionata a piena risoluzione, mentre la crominanza è campionata a metà della risoluzione orizzontale, ottimizzando la compressione senza una significativa perdita di qualità percepita, visto che l'occhio umano è molto più sensibile alla luminosità piuttosto che al colore.

[44] Tecnica utilizzata per ridurre gli effetti di scalettatura (aliasing) che si verificano nei bordi diagonali o curvi delle immagini digitali, rendendoli più lisci e naturali.

[45] Una tecnica di anti-aliasing che campiona più punti all'interno di ogni pixel per ridurre l'aliasing spaziale, combinando i risultati per produrre un'immagine più liscia e dettagliata. Questo metodo è il migliore per scene non dinamiche rispetto alle altre strategie.

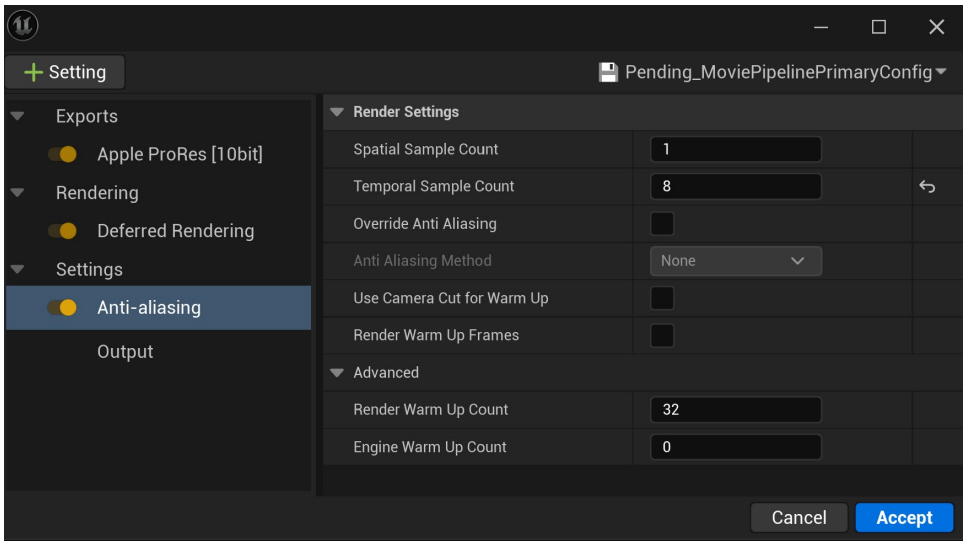

Figura 2.100: Impostazioni di rendering per Lumen

In questo caso, il modulo del path tracing è stato omesso, ma è stato aggiunto il modulo "Deferred Rendering"[46] per gestire l'illuminazione e le ombre in modo più efficace. Nelle impostazioni dell'anti-aliasing sono sufficienti 8 campioni temporali, senza la necessità di specificare un metodo di anti-aliasing, poiché Lumen non è particolarmente suscettibile al rumore.

Figura 2.101: Inquadratura 1 della scena subacquea, in Path Tracing

[46] Deferred Rendering è una tecnica di rendering che posticipa il calcolo della luce e del colore dei pixel fino a dopo che tutte le geometrie della scena sono state processate. Questo permette una gestione efficiente di scene complesse con molte luci dinamiche.

Figura 2.102: Inquadratura 1 della scena subacquea, in Lumen

L'illuminazione della scena differisce notevolmente tra i due metodi di rendering, specialmente per quanto riguarda le ombre: quelle generate da Lumen risultano eccessivamente nere e innaturali, come si può osservare dalla roccia in primo piano e dalla qualità delle ombre e delle luci riflesse sulla vegetazione sulla destra; anche l'aspetto dell'acqua varia significativamente se renderizzata con Lumen o path tracing.

Figura 2.103: Inquadratura 1 della scena subacquea, in Lumen e con le correzioni

Per migliorare leggermente la scena, pur dovendo utilizzare Lumen, è stata adottata la strategia di aggiungere una luce puntuale leggera in basso a sinistra. Questa luce schiarisce leggermente l'ombra della roccia in primo piano, simulando una migliore riflessione della luce e correggendo il difetto più evidente e aggiustabile.

TERZA PARTE: TOUCHDESIGNER PER IL PROGETTO PAESAGGI – LANDASCAPES

Oltre al motore di gioco Unreal Engine, per la realizzazione di questo progetto è stato anche utilizzato il software TouchDesigner – oggetto di questa ultima parte della mia tesi. In particolare, questo software è stato da me sfruttato, oltre che per il lavoro sul Blob Tracking[47], per la realizzazzione di un sistema particellare a forma del tempio di Philae, nonchè per il flusso finale del video.

3.1 TouchDesigner

3.1.1 Il software per lo sviluppo visivo

TouchDesigner è un software per lo sviluppo visivo in tempo reale creato da Derivative, una società canadese fondata nel 1991 da Greg Hermanovic, uno dei co-fondatori del noto software di animazione 3D Houdini. Nato nei primi anni 2000, TouchDesigner è diventato uno strumento essenziale per artisti visivi, designer interattivi e sviluppatori multimediali. Progettato per fornire una piattaforma versatile e potente, TouchDesigner consente la creazione di contenuti interattivi, installazioni artistiche, spettacoli dal vivo e visualizzazioni di dati complessi. Un aggiornamento significativo nella storia di TouchDesigner è stata la versione 099 rilasciata nel 2017, che ha introdotto numerose nuove funzionalità e miglioramenti: questo aggiornamento ha ampliato le possibilità creative del software, integrando il supporto per Vulkan, una moderna API di rendering grafico, e migliorando il supporto per la realtà virtuale, l'input/output di dati in tempo reale e la compatibilità con vari hardware e dispositivi. TouchDesigner offre una combinazione unica di strumenti per la creazione di contenuti visivi e interattivi, infatti la sua interfaccia basata su nodi, dove ogni nodo rappresenta un'operazione o un effetto, permette agli utenti di collegare e manipolare questi nodi per costruire flussi di lavoro visivi complessi. Questa architettura modulare rende TouchDesigner accessibile sia ai principianti che ai professionisti, facilitando la prototipazione rapida e l'adattamento delle creazioni alle esigenze specifiche di un progetto. Il software è rinomato per la sua capacità di gestire contenuti

[47] Il blob tracking - su cui non si approfondirà in questa tesi ma a cui ho contribuito insieme ad altri colleghi - è una tecnica per rilevare e monitorare oggetti in movimento all'interno di un video. Questa tecnica sfrutta la segmentazione per identificare le aree di interesse, denominate "blob", e l'etichettatura per distinguere ciascuna area. In TouchDesigner, il blob tracking consente di tracciare la posizione degli oggetti rilevati e, nel caso specifico del mio progetto, il blob tracking è stato utilizzato per seguire i movimenti degli oggetti, sovrapponendo contorni di rettangoli bianchi che contengono parole attinenti all'Egitto. Questa tecnica arricchisce l'esperienza visiva, integrando elementi artistici che evocano il tema dell'antico Egitto.

multimediali in tempo reale, inclusi video ad alta risoluzione, audio, grafica 3D e dati sensoriali: questa caratteristica lo rende ideale per una vasta gamma di applicazioni, dalle performance live ai media generativi, dalle installazioni interattive alle visualizzazioni di dati. Infatti, TouchDesigner è particolarmente apprezzato nelle arti visive e negli spettacoli, dove la capacità di creare e manipolare contenuti in tempo reale è cruciale. È utilizzato in eventi di grande scala, festival, musei e spazi pubblici, grazie alla sua capacità di integrare vari tipi di media e di interagire con il pubblico, creando esperienze coinvolgenti e dinamiche. In sintesi, TouchDesigner è un potente strumento per la creazione di esperienze visive e interattive, combinando flessibilità e potenza. La continua innovazione e sviluppo del software lo hanno reso un programma molto usato nel campo della creazione multimediale.

3.1.2 L'interfaccia grafica di Touchdesigner

In questo capitolo verrà analizzata brevemente l'interfaccia del software, per poi proseguire con gli aspetti fondamentali.

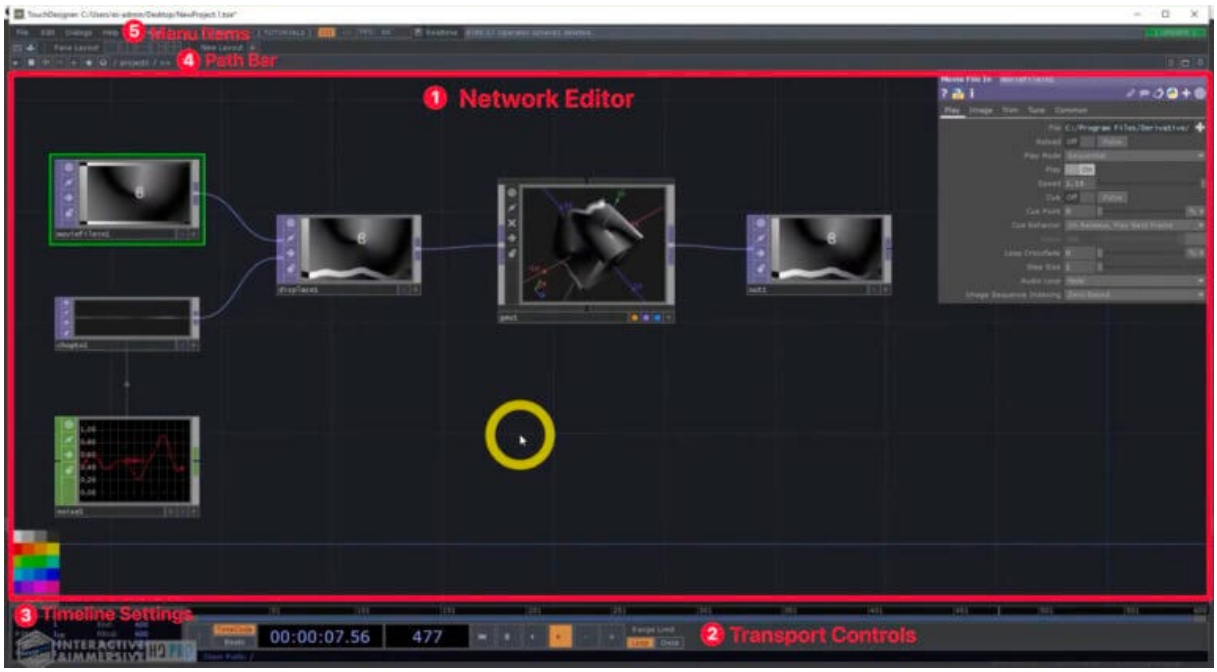

Figura 3.1: Interfaccia di TouchDesigner

1. Network Editor: è la parte principale del programma, dove si gestiscono i nodi e si crea l'intera logica. Il pannello a destra è il Parameter Window, che permette, una volta cliccato su un nodo, di vedere tutte le sue proprietà e modificarle per personalizzare il comportamento del nodo.
2. Transport Controls: offrono i controlli base dell'animazione, come i pulsanti di play, stop, e consentono di visualizzare i secondi e il frame corrente dell'animazione.

3. Timeline Settings: contiene le informazioni essenziali per gestire l'animazione, come il frame di partenza, quello finale (sia per l'anteprima che per la registrazione), gli fps e la durata totale. Per il nostro progetto, abbiamo sempre usato 30 fps.
4. PathBar: indica la posizione corrente nel progetto. In progetti complessi, infatti, ci possono essere nodi che fungono da funzioni, contenenti altri nodi al loro interno. La PathBar permette di navigare e vedere dove ci si trova all'interno del progetto.
5. Menu Items: costituiscono la classica barra dei menu, con le impostazioni del software, gestione dei file e altre opzioni. Un'impostazione importante da menzionare è la spunta su "Realtime": se attiva, quando viene effettuato il rendering, potrebbe non risultare correttamente, in quanto viene renderizzato a velocità reale, senza elaborazione frame by frame. Questo può causare problemi se il computer non è molto potente o il progetto è complesso, risultando in un'animazione a scatti poiché non tutti i frame vengono renderizzati correttamente.

3.1.3 I nodi

I nodi, come nel blueprint di Unreal Engine, creano la logica del progetto. Essi si collegano tra di loro con link che si collegano dall'output di uno all'input dell'altro, per creare il flusso del progetto, oppure si trascina un nodo o una sua variabile, visualizzabile dal Parameter Window, all'interno di un altro nodo o variabile interna, per creare riferimenti di valori tra punti diversi del programma. Per aggiungere un nuovo nodo, si può premere il tasto TAB, e si aprirà un menu con l'elenco di tutti i nodi, divisi in sei categorie. I TOPs (Texture Operators) lavorano esclusivamente con le texture, gestendo immagini, filmati, disegni 2D e video in entrata e in uscita da vari tipi di hardware. Questi operatori, fondamentali per qualsiasi operazione che coinvolge contenuti visivi bidimensionali, sono riconoscibili dal loro colore viola. I CHOPs (Channel Operators), identificati dal colore verde, sono invece utilizzati per gestire segnali, dati numerici e di controllo: essi possono creare oscillatori a bassa frequenza (LFO) e gestire dati di controllo come MIDI (Musical Instrument Digital Interface) e OSC (Open Sound Control), protocolli per la comunicazione tra computer, sintetizzatori e altri dispositivi multimediali, essendo essenziali per creare interfacce utente con pulsanti e trigger. Inoltre, ci sono i DATs (Data Operators), di colore rosa, che gestiscono dati a livello di stringa, come testi e tabelle, supportando la programmazione Python, risultando cruciali per l'interazione con API web e per gestire richieste HTTP, permettendo di ricevere e manipolare dati esterni. Passando ai SOPs (Surface Operators), riconoscibili dal colore azzurro, si occupano della geometria 3D, focalizzandosi sulla creazione e manipolazione di superfici tridimensionali in modo

procedurale, consentendo di costruire e modificare oggetti 3D in modo dinamico e in tempo reale. Poi ci sono i COMPs (Component Operators), caratterizzati dal colore nero, che sono versatili e suddivisi in quattro sottocategorie: Pannelli, Oggetti 3D, Altro e Dinamica. I Pannelli sono usati per elementi dell'interfaccia utente, gli Oggetti 3D gestiscono il rendering di scene tridimensionali, la categoria "Altro" include nodi generici con varie funzionalità, mentre i nodi nella categoria Dinamica si occupano di fare operazioni rigardanti simulazioni fisiche. Infine, i MATs (Material Operators), di colore giallo, sono dedicati alla gestione dei materiali applicati alle geometrie 3D durante il rendering, supportando vari tipi di materiali, come texture importate o generate, e materiali specifici come il materiale costante, che definiscono l'aspetto visivo finale degli oggetti tridimensionali.

Ci sono alcuni nodi, come il nodo “Null”, che appartengono a più categorie e la cui scelta dipende dai nodi a cui si collegano, pur mantenendo lo stesso funzionamento. Il nodo “Null” è particolarmente interessante e frequentemente utilizzato, poiché, nonostante non compia alcuna operazione sulle informazioni ricevute, funge da punto di osservazione intermedio per verificare i risultati prodotti fino a quel punto. Inoltre, può essere impiegato come nodo di transito per collegare successivamente altri nodi, o come input per ulteriori operazioni. In alcuni casi, questo nodo, viene rinominato per fornire una chiara indicazione del processo in corso, rendendo più comprensibile la funzione svolta dai nodi precedenti.

3.2 Touchdesigner per il tempio di *Philae* particellare

Durante il progetto, è stata sviluppata una versione particellare del tempio di *Philae* utilizzando vari software, tra cui TouchDesigner. Avendo a disposizione due video per produrre altrettante versioni del tempio da angolazioni differenti, in TouchDesigner è stato creato un sistema che non solo permettesse di visualizzare il tempio in versione particellare, ma anche di realizzare una transizione tra le due visualizzazioni. Prima di esplorare il progetto in TouchDesigner, è però utile descrivere il workflow adottato, partendo da un video fino ad arrivare a un file particellare in TouchDesigner.

3.2.1 Il sito Luma AI

Luma AI è una società tecnologica fondata nel 2021, specializzata nell'intelligenza artificiale e nella computer vision. L'azienda si concentra sullo sviluppo di strumenti avanzati per la creazione e l'animazione di contenuti 3D realistici. Utilizzando tecnologie come la fotogrammetria e l'apprendimento automatico, Luma AI permette di trasformare immagini e video 2D in modelli 3D dettagliati e dinamici. Per il nostro progetto, Luma AI è stata utilizzata

per creare i Gaussian Splats[48] del tempio di Philae, che costituiscono la prima fase di questo workflow. È stato sufficiente caricare sul sito di Luma AI i due video del tempio, e dopo qualche ora è stato possibile scaricare i due file dei Gaussian Splats ottenuti dai video.

3.2.2 SuperSplat: un programma su browser

SuperSplat è uno strumento avanzato per l'editing e l'ottimizzazione di 3D Gaussian Splats, sviluppato da PlayCanvas. Questo editor, che funziona interamente nel browser, è progettato per manipolare file PLY, un formato comune per le nuvole di punti 3D. SuperSplat offre strumenti potenti di selezione e un'interfaccia intuitiva per la pulizia e la modifica delle nuvole di punti, rendendolo ideale per creare scene 3D fotorealistiche a partire da dati di fotogrammetria.

I Gaussian Splats creati da Luma AI presentano spesso imperfezioni e difetti, come la creazione del cielo in versione Gaussian, che non è necessario per il nostro scopo ma utilizzando SuperSplat, è stato possibile pulire il modello per renderlo successivamente utilizzabile.

Figura 3.2: Interfaccia di SuperSplat con caricato il tempio di Philae

Il modello, nonostante sia molto accurato, presenta anche altre imperfezioni che devono essere eliminate. Per far ciò, quindi, si utilizza il sito SuperSplat, che fortunatamente è abbastanza intuitivo: nel menu a sinistra ci sono strumenti per selezionare e eliminare gli Splats non necessari. Si possono eliminare selezionandoli uno ad uno, oppure utilizzando un "Brush", che permette di selezionarne più contemporaneamente, oppure ancora utilizzando un piano o una

[48] I 3D Gaussian splat sono tecniche di rendering che utilizzano funzioni gaussiane per rappresentare e visualizzare nuvole di punti 3D, creando transizioni morbide e naturali tra i punti. I punti, in questo caso, sono chiamati Splats.

sfera come riferimento per selezionare tutti i punti superiori o all'interno di tali forme. Aumentando il valore "Splat Size" è possibile visualizzare meglio gli splat per capire quali eliminare. In figura, è stato utilizzato il brush per selezionare una parte del tempio, mostrando quali punti sarebbero eliminati se venisse premuto "Deleted Selected Splats" o il tasto "Canc". Infine, tramite il pulsante a forma di foglio in alto a sinistra, si può esportare il file modificato e scaricarlo.

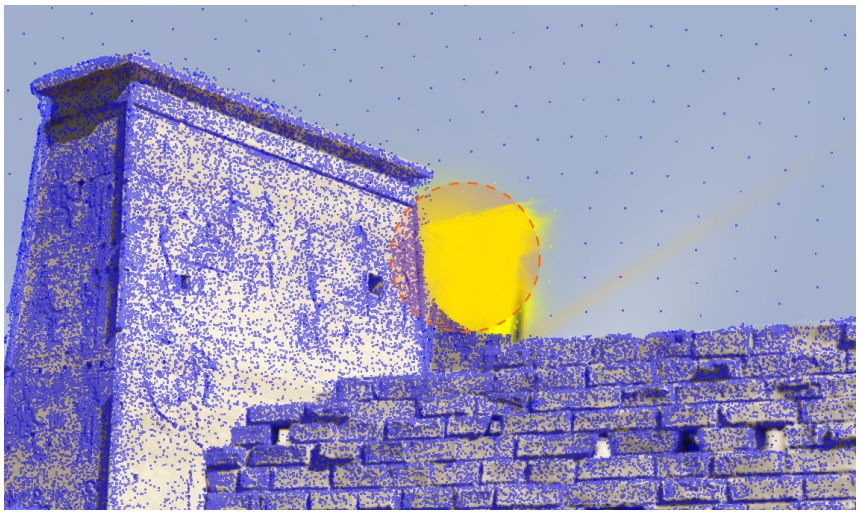

Figura 3.3: Sezione del tempio, con acluni splat selezionati con il brush

È fondamentale tenere presente che la rappresentazione tramite Gaussian Splat non crea un modello a 360° dell'oggetto, a meno che non venga inquadrato da ogni angolatura nel video di input. Nel nostro caso, visualizzando il modello da un punto di vista molto diverso da quello del video, il modello risulta inutilizzabile.

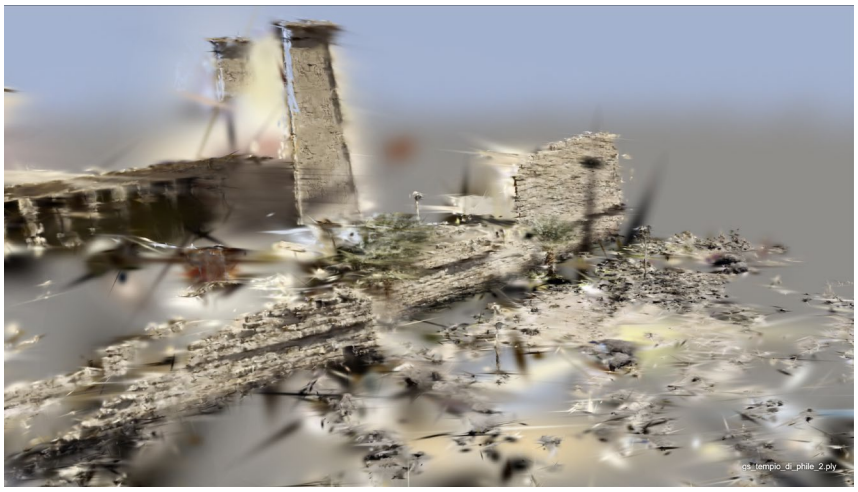

Figura 3.4: Il tempio di Philae visto da un'altra angolazione

Pertanto, questo tipo di rappresentazione è uno strumento molto potente, ma va usato con cura e presenta comunque delle limitazioni.

3.2.3 Il tentativo su Unreal Engine

Prima di procedere direttamente su TouchDesigner, tuttavia, si è pensato di rimanere su Unreal Engine per realizzare una scena fotorealistica del tempio di Philae con un timelapse giorno/notte. Per fare ciò, è stato semplicemente scaricato il plugin ufficiale di Luma AI per Unreal Engine, che consente di importare file di tipo PLY. Dopo l'installazione del plugin, il file è stato trascinato nel Content Browser per l'importazione e, una volta posizionato in scena, è stato necessario trovare l'angolazione della camera ideale per evitare di inquadrare parti imperfette del modello. Per aumentare il realismo della scena, è stato aggiunto il Nilo, con i colori modificati per corrispondere a quelli del video originale.

Figura 3.5: Il tempio di Philae in Gaussian Splat su Unreal Engine

Nonostante i contorni degli oggetti mostrino ancora imperfezioni da eliminare, la scena risulta molto realistica e accurata. Tuttavia, per creare un timelapse giorno/notte, è essenziale che le ombre si muovano e scompaiano in assenza di luce su Unreal; purtroppo, come si può notare, le ombre sono già presenti nel modello, esattamente come lo erano nella realtà al momento della ripresa del video di input, ma essendo un modello Gaussian Splat e non un modello Mesh, non è possibile rimuovere queste ombre preesistenti. Questo rappresenta il principale motivo per cui questo tipo di scena è stato abbandonato a favore di una rappresentazione a nuvola di punti del tempio, rinunciando al fotorealismo.

3.2.4 Il particellare su TouchDesigner

Poiché il metodo precedente non è risultato soddisfacente, si è deciso di utilizzare TouchDesigner per gestire e animare le particelle. Inizialmente, è stato importato nel progetto il modello del tempio di Philae trascinando il file PLY nel Network Editor; questo ha generato automaticamente un nodo DAT denominato “Point File In”. Per visualizzare il modello, è sufficiente cliccare sul simbolo “+” in basso a destra del nodo, premere con il tasto destro all’interno di esso e selezionare “View as Points”. Sono stati importati i due modelli del tempio

da due angolature diverse e un modello di una sfera, chiamato “Earth”, utilizzato per la transizione tra i due.

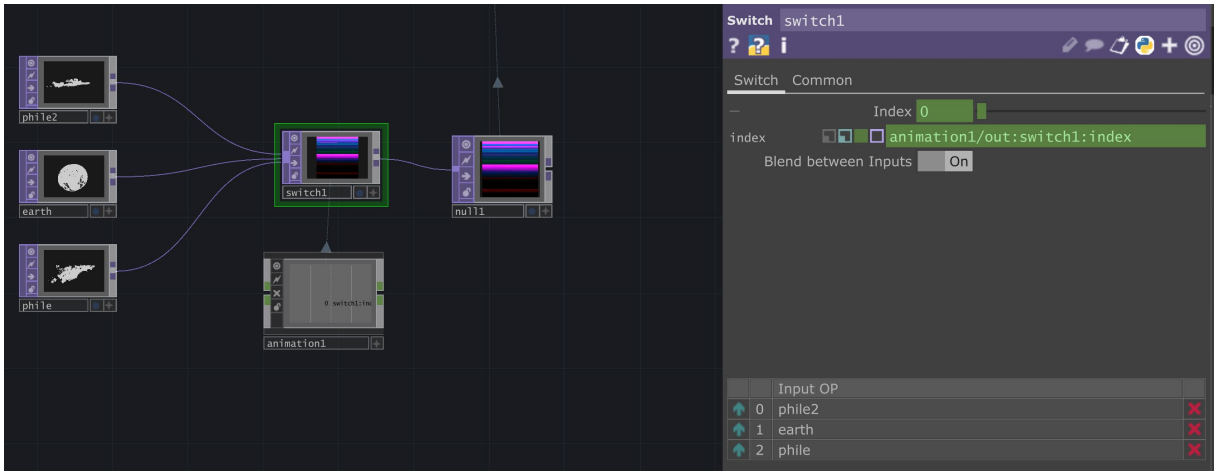

Figura 3.6: Progetto TouchDesigner sul tempio di Philae

I tre nodi di input sono stati collegati a un nodo “Switch”, che consente di passare da una rappresentazione all’altra modificando il valore “Index”, contenuto nel Parameter Window del nodo. Anche l’ordine degli input, situato in basso nel pannello, può essere cambiato: con l'ordine corrente, il valore 0 rappresenta “Phile2”, 1 rappresenta “Earth” e 2 rappresenta “Phile” e le transizioni tra le rappresentazioni avvengono quando l'indice si trova tra questi valori. Per animare il valore di index, si utilizza un’espressione pilotata da una variabile contenuta in un’animazione, simile ai Level Sequence di Unreal. Per creare l’animazione e il collegamento, è stato sufficiente cliccare con il tasto destro sul valore di “Index”, selezionare “Keyframe Parameter in” e poi “New Animation”. In alternativa, si poteva aggiungere manualmente il nodo “Animation” e scrivere l’espressione “nomenodoanimazione/out”. Il collegamento è confermato dalla comparsa di una freccia grigia tra i due nodi.

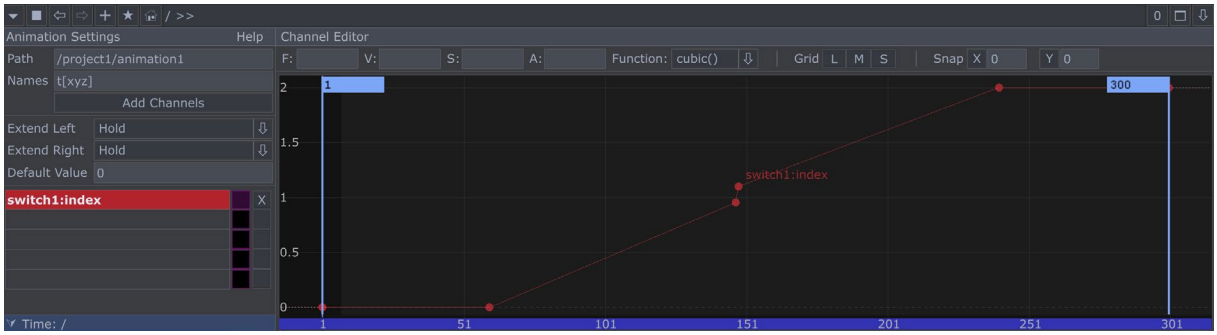

Figura 3.7: Pannello dell'animazione su TouchDesigner

A questo punto, si è aperto il pannello di editing dell’animazione, simile al Graph Editor di altri software, compreso Unreal. A sinistra vi è l’elenco delle variabili, mentre a destra si trova il

grafico con i keyframe della variabile selezionata, con l'asse x rappresentante i frame e l'asse y il valore della variabile; infine, in alto al centro è indicato il tipo di curva utilizzata, in questo caso "cubic()". Dopo aver gestito l'animazione, l'output dello switch è stato collegato a un nodo null di passaggio, utilizzabile come input per il nodo successivo.

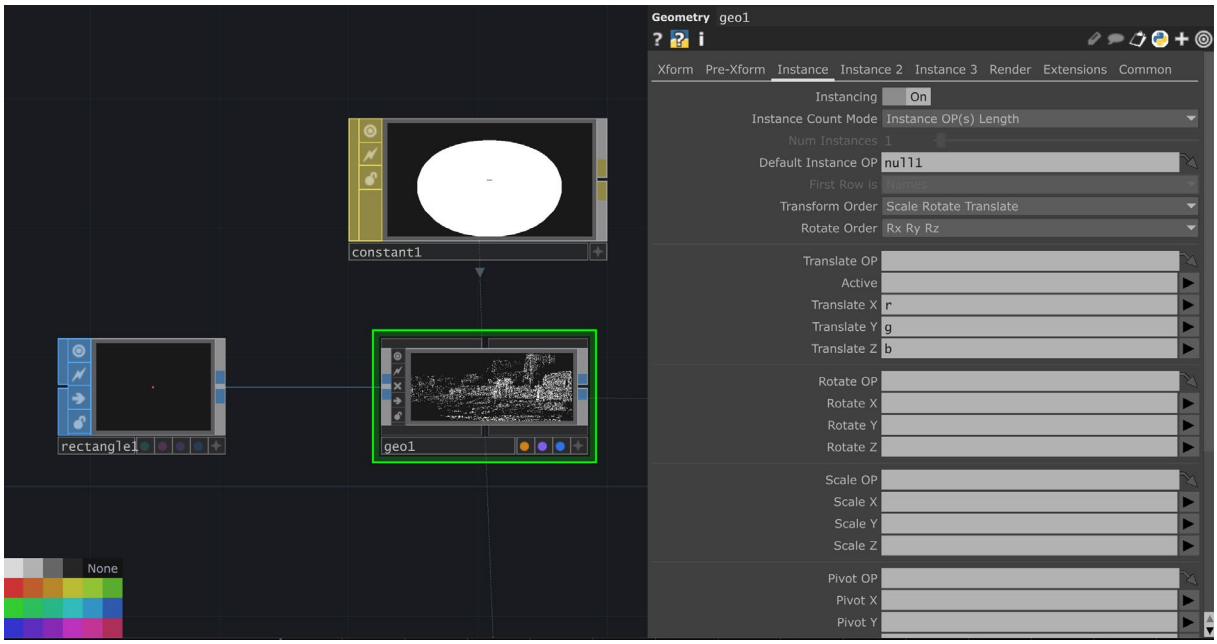

Figura 3.8: Progetto TouchDesigner sul tempio di Philae

La seconda parte del progetto riguarda l'aspetto e il posizionamento delle particelle: fino ad ora sono state trasmesse le informazioni sulla loro posizione, ma non sono ancora state effettivamente posizionate in scena. Per fare ciò, è stato necessario utilizzare il nodo MAT "Constant", che produce un materiale di base che in questo caso, è stato lasciato bianco. Successivamente, è stato utilizzato il nodo "Rectangle" per dare una forma alle particelle; all'interno di questo nodo, è stata selezionata l'opzione "Size" per definire la dimensione 2D delle particelle, con valori molto piccoli (0.009 e 0.004), che dipendono dal risultato desiderato e dall'input fornito. Da questo nodo, si è collegato il nodo "Geometry": esso combina tutte le informazioni relative alla forma, al materiale e alla posizione dell'oggetto. Nella sezione "Instance" del nodo "Geometry", è stato aggiunto il collegamento con il nodo "null1" precedentemente creato, trascinandolo nella posizione corretta. È stato poi impostato il modo in cui i dati di posizione devono essere letti: la x è rappresentata da r, la y da g e la z da b. Questo cambiamento è necessario perché i nodi precedenti utilizzavano il sistema rgb, mentre ora le informazioni sulla posizione sono gestite in termini di x, y e z. L'informazione sulla forma è automaticamente trasferita grazie al collegamento tra il nodo "Rectangle" e il nodo "Geometry". Invece, per quanto riguarda il materiale, è stato necessario trascinare il nodo

"Constant" nel parametro "Material" della sezione "Render" del nodo "Geometry", per applicarlo alle particelle. Questo completa la preparazione della scena, ma, come avviene anche in Unreal Engine, è ora necessario impostare la camera e i parametri di rendering.

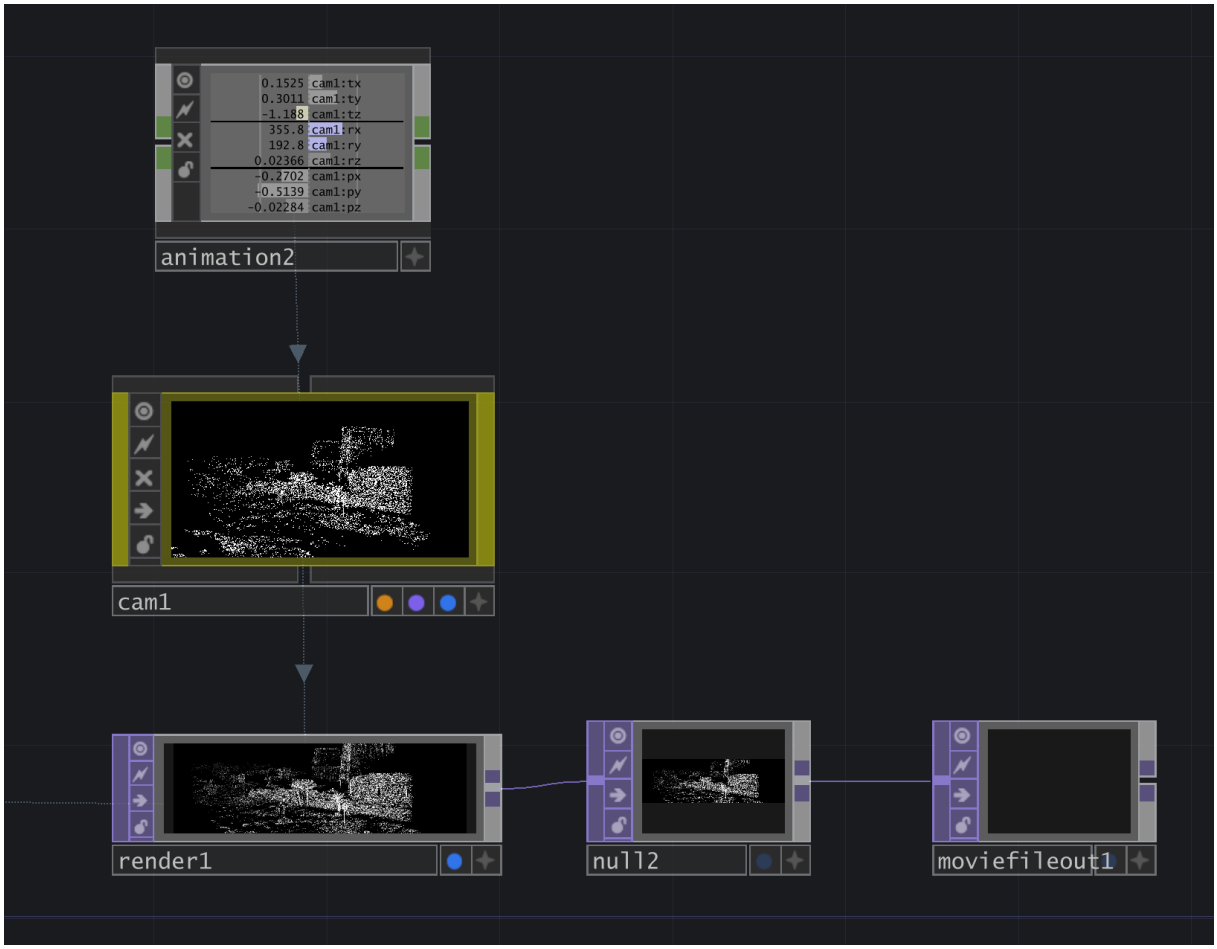

Figura 3.9: Progetto TouchDesigner sul tempio di Philae

Per gestire questi aspetti, è stato utilizzato il nodo "Render", che permette di impostare parametri di rendering come la risoluzione (nel nostro caso 3510x1080) e il formato dei pixel, inclusa la profondità in bit e i canali da mantenere. Nel nostro caso, è stata scelta una profondità di 32 bit e un formato Mono per ottenere una rappresentazione in bianco e nero. Per specificare cosa deve essere renderizzato, è stato aggiunto un nodo COMP "Camera2, che funziona come una vera e propria telecamera, con tutte le impostazioni relative alla sua posizione e orientamento nello spazio 3D. In questo progetto, la posizione e l'orientamento della camera sono stati animati utilizzando un secondo nodo "Animation", con valori distinti per ogni coordinata, permettendo alla camera di ruotare leggermente intorno al modello per rendere la scena più dinamica e visivamente interessante. Il collegamento tra il nodo "Render" e il nodo "Camera" è stato effettuato trascinando quest'ultimo sopra il primo. Successivamente, per

renderizzare l'intera animazione, è stato utilizzato il nodo “MovieFileOut”, che consente la registrazione e il salvataggio dei frame; in questo nodo, è possibile impostare il tipo di file di output (video o sequenza di immagini), il formato, il nome del file, gli fps e altre opzioni. Quando tutto è stato impostato correttamente, si è configurato il frame corrente su 1 e il Range Limit su “Once” (per iniziare il render dall’inizio e per evitare che ricominci una volta finito) nel Transport Controls, quindi si è attivata la spunta “Record” del nodo “MovieFileOut” e si è avviato il rendering utilizzando il tasto play o la barra spaziatrice.

Figura 3.10: Frame iniziale dell’animazione, con il tempio di Philae da un'angolazione

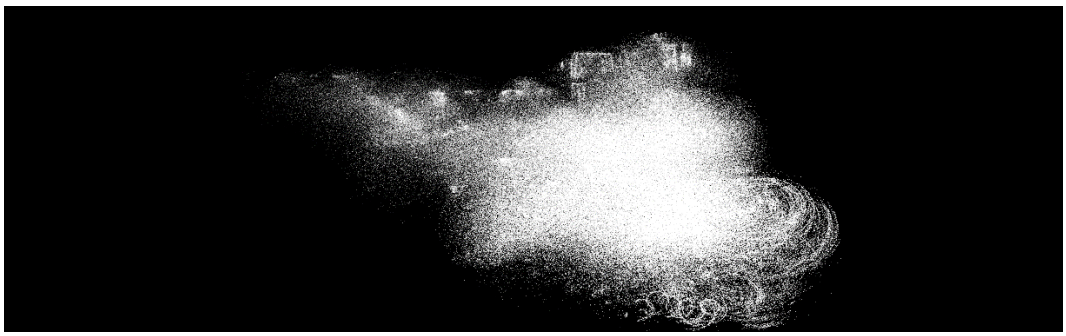

Figura 3.11: Frame intermedio dell'animazione, dal tempio alla sfera

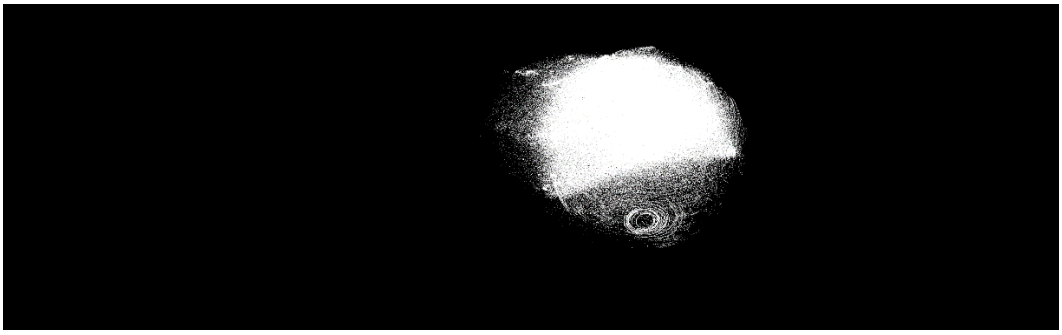

Figura 3.12: Frame a metà dell'animazione, quando vi è la sfera

Figura 3.13: Frame intermedio dell'animazione, dalla sfera al tempio

Figura 3.14: Frame finale dell'animazione, con il tempio visto da un'altra angolazione

L'animazione così ottenuta mostra inizialmente il tempio di Philae da un'angolazione, con la nuvola di punti che si deforma fino a rappresentare una sfera, per poi formare nuovamente il tempio da un'altra angolazione.

3.3 Il Flusso finale

Come parte finale del video, si è pensato a un flusso di particelle che partisse da un punto della sala per espandersi sulle pareti circostanti, rappresentando un flusso di pensieri e idee mescolate insieme. Questo flusso è stato realizzato con TouchDesigner, mantenendo un'idea visiva in bianco e nero: l'idea generale del progetto è quella di partire da un piano e renderlo particellare, per poi creare un movimento casuale delle particelle aggiungendo nodi di noise. Per ottenere l'effetto desiderato, in cui le particelle si espandono principalmente a destra e sinistra, è stata animata la larghezza del piano, inquadrando il piano lateralmente. Questo progetto di TouchDesigner è complesso, quindi verrà analizzato in gruppi di nodi accomunati da scopi specifici, senza entrare nei dettagli di ogni singolo parametro modificato, poiché molti valori sono stati determinati empiricamente per soddisfare l'aspetto grafico desiderato.

Innanzitutto, si devono descrivere due valori che sono connessi a molti nodi e hanno due funzionalità diverse.

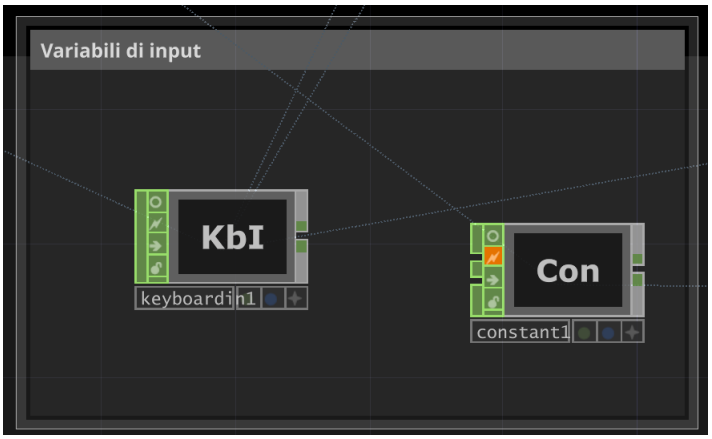

Figura 3.15: Progetto Touch Designer sul flusso finale: variabili di input

Il primo, “Keyboard In”, è un nodo che reagisce alla pressione di un tasto della tastiera, in questo caso ‘1’; infatti in TouchDesigner, è prassi che il tasto ‘1’ venga usato per resettare l’animazione e farla ripartire da capo. I nodi che lo permettono hanno infatti un tasto “Pulse” tra i loro parametri, che, se premuto, resetta il nodo stesso per far ripartire l’animazione o valori che servono per essa; collegando, quindi, il nodo “Keyboard In” al pulsante “Pulse” o simili dei vari nodi, si può resettare l'intera animazione con la pressione di un solo tasto. Il secondo valore, dal nodo “Constant”, indica il numero di particelle in scena ed è comodo da usare in quanto si può cambiare il valore in un solo punto, modificandolo in tutti i nodi a cui è connesso.

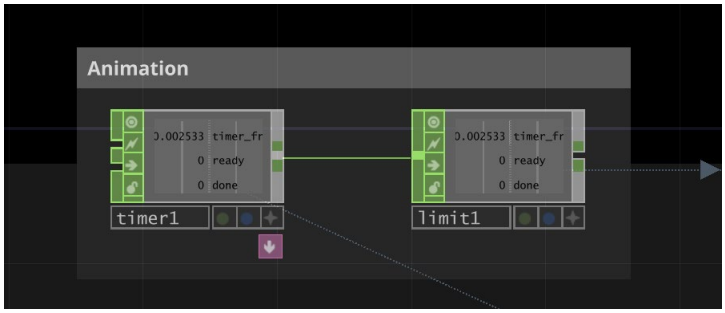

Figura 3.16: Progetto Touch Designer sul flusso finale: animazione

Questi due nodi gestiscono l’animazione dell’aumento della larghezza del piano. È stato utilizzato il nodo “Timer”, che fa aumentare il suo valore di uscita da 0 a 1 mentre scorre il tempo, con parametri che gestiscono il tempo necessario per arrivare a 1, se deve essere in loop o meno, e altre opzioni. Inoltre, il nodo “Keyboard In” è stato collegato al pulsante “Start” del timer, in modo che premendo '1' il timer ricominci. Il valore in output è stato limitato grazie al nodo “Limit”, che, attivando la funzione di Clamp, restituisce in output il valore di ingresso solo se compreso tra i due valori selezionati, altrimenti restituisce il valore precedente: questo

serve per evitare che la dimensione del piano sia eccessiva, mantenendo il flusso entro i bordi dello schermo.

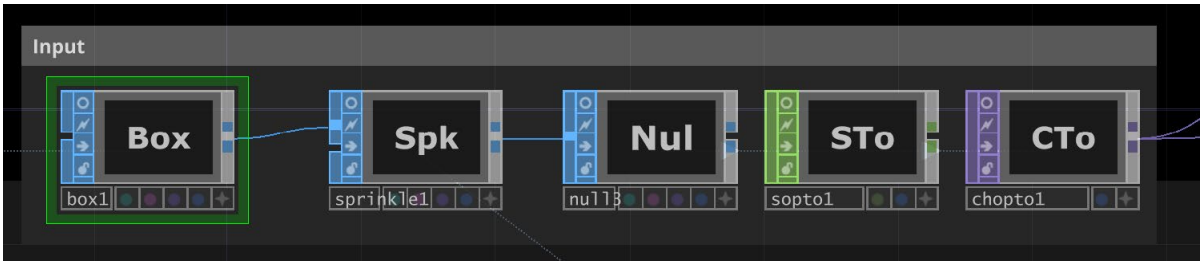

Figura 3.17: Progetto Touch Designer sul flusso finale: input

Il passo successivo è creare il piano di partenza e convertirlo in particelle. Questo si può fare con il nodo “Box” per creare un parallelepipedo, impostando le dimensioni e assegnando a un asse un valore estremamente piccolo per simulare un piano, e a un altro asse il valore in uscita del nodo “Limit”. Successivamente, si collega il nodo “Sprinkle” per convertire il piano in particelle: essendo un piano quadrato, la quantità di particelle è data dal valore di “Constant” moltiplicato per sé stesso. Per passare da un livello di superfici a uno di texture, le informazioni vengono convertite da SOP a CHOP e poi da CHOP a TOP, utilizzando i nodi “SopTo” e “Chop To”, attraverso un nodo “Null” di passaggio.

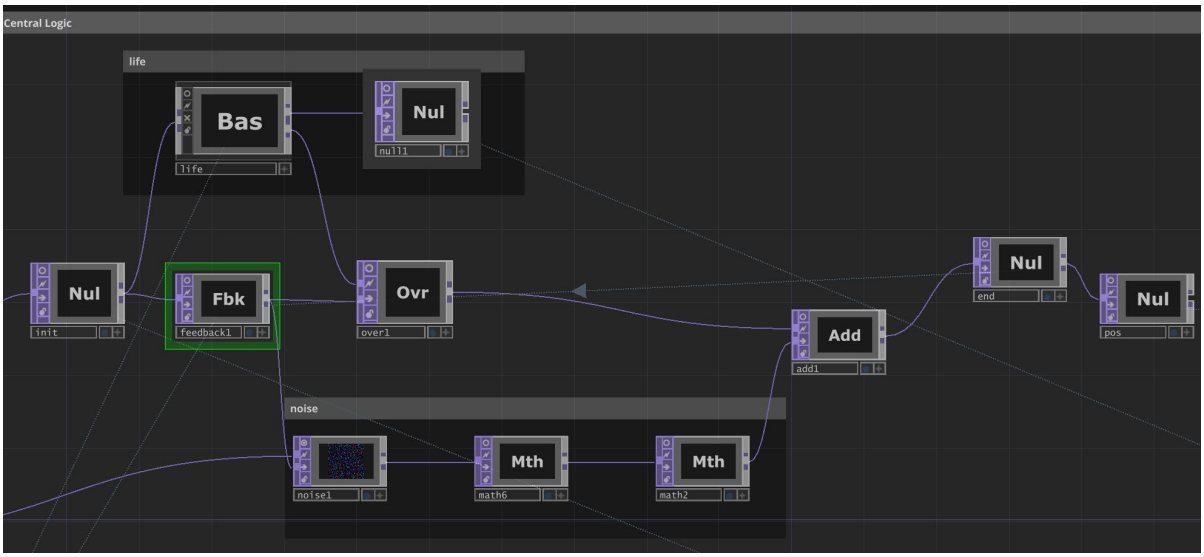

Figura 3.18: Progetto Touch Designer sul flusso finale: logica centrale

Questa parte riguarda il movimento delle particelle: gli input in “Init” e in “Noise” (entrambi output di “ChopTo”) rappresentano l’insieme delle posizioni di tutte le particelle. Il nodo “Init”, che è un nodo “Null”, serve solo come nodo di passaggio, trasferendo informazioni al nodo

"Life" [49] e al nodo "Feedback". Dopo "Init" vi è il nodo "Life", che gestisce la durata e la rigenerazione delle particelle, creando un movimento di base delle singole particelle; ha un parametro per il reset e uno con un valore per indicare la lunghezza di vita delle particelle. Connesso a "Init" vi è anche il nodo "Feedback", con il parametro "Pulse" per essere resettato, recupera informazioni dai frame precedenti per creare effetti visivi unici, sovrapponendo informazioni passate e presenti; in questo caso, le informazioni passate sono prese dal nodo "Null" chiamato "End". Entrambi questi nodi ("Life" e "Feedback") confluiscono in "Over", che unisce in sovrapposizione le due informazioni. Il nodo "Null1" serve come input per una parte successiva, trasmettendo informazioni delle particelle dopo il nodo "Life".

Parallelamente al nodo "Over" viene creato il movimento generale delle particelle. Al nodo "Noise" (con molti parametri per gestire il tipo di rumore) si connette sia l'output delle informazioni di base delle particelle, sia quello delle informazioni del "Feedback". Successivamente, due nodi "Math" eseguono operazioni matematiche sui valori, gestendo ulteriormente le caratteristiche del rumore. Ad esempio, in "Math2" è impostato un valore di moltiplicazione pari a 0.002, controllando così il movimento delle particelle, che deve essere molto limitato. Per applicare il rumore, si sommano i valori delle particelle senza rumore (output del nodo "Over") e quelli di "Math2" utilizzando un nodo "Add". Viene poi aggiunto il nodo "End" per passare il valore al precedente "Feedback" e un nodo "Null" rinominato in "Pos" per indicare che a questo punto si hanno tutte le informazioni sulla posizione.

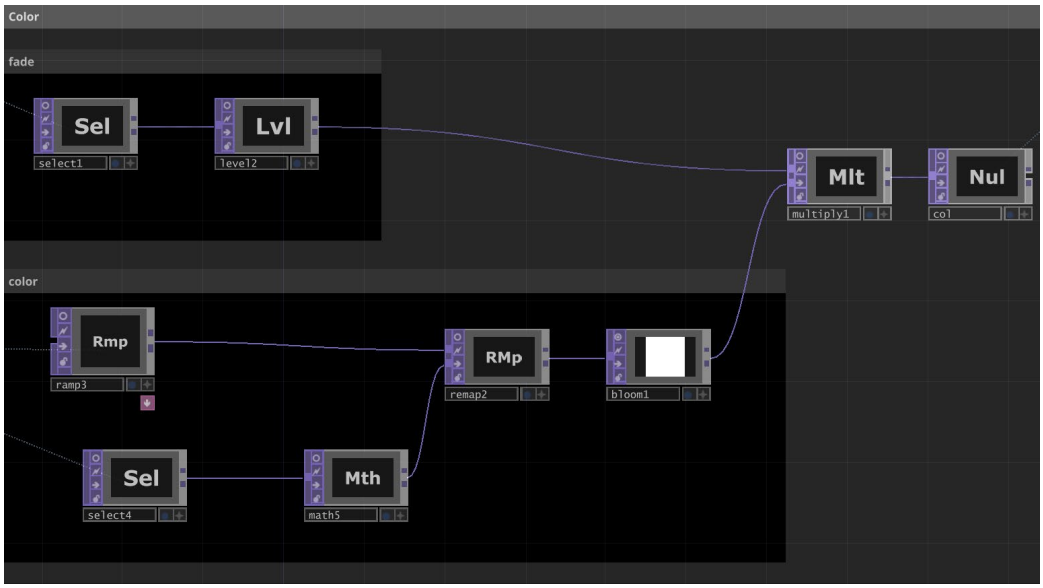

Figura 3.19: Progetto Touch Designer sul flusso finale: colori

[49] Questo nodo è stato creato da Simon Savid Ryden, e pubblicato nel suo patreon https://www.patreon.com/supermarketsallad/

Nella successiva parte, visibile nella precedente immagine, si gestiscono i colori: i valori in input per i tre nodi di sinistra provengono da "Null1" (particelle già soggette al nodo "Life"), da "Constant" (numero di particelle) e da "Init" (informazioni delle particelle iniziali). Due flussi paralleli vengono poi uniti tramite il nodo "Multiply": uno riguarda la sfumatura e l'altro il colore vero e proprio. Per la sfumatura, i valori vengono recuperati tramite il nodo "Select1" e regolati con il nodo "Level" per modificare contrasto, luminosità, gamma dei colori e opacità. Per i colori, si utilizza il nodo "Ramp" per selezionare i colori desiderati e, per ottenere una quantità di informazioni pari alla quantità di particelle, si imposta la sua risoluzione su x e y con il valore "Constant". I valori di base delle particelle sono presi con "Select4" e modificati con il nodo "Math" per ottenere l'aspetto desiderato, cambiandone proprietà come luminosità e contrasto. Queste informazioni di colori e particelle vengono unite tramite il nodo "Remap", che utilizza il primo input come immagine di base e il secondo come UV map. Infine, il nodo "Bloom" aggiunge un leggero effetto di alone intorno agli oggetti. Alla fine, il nodo "Null" rinominato in "Col" rappresenta tutte le informazioni riguardanti i colori.

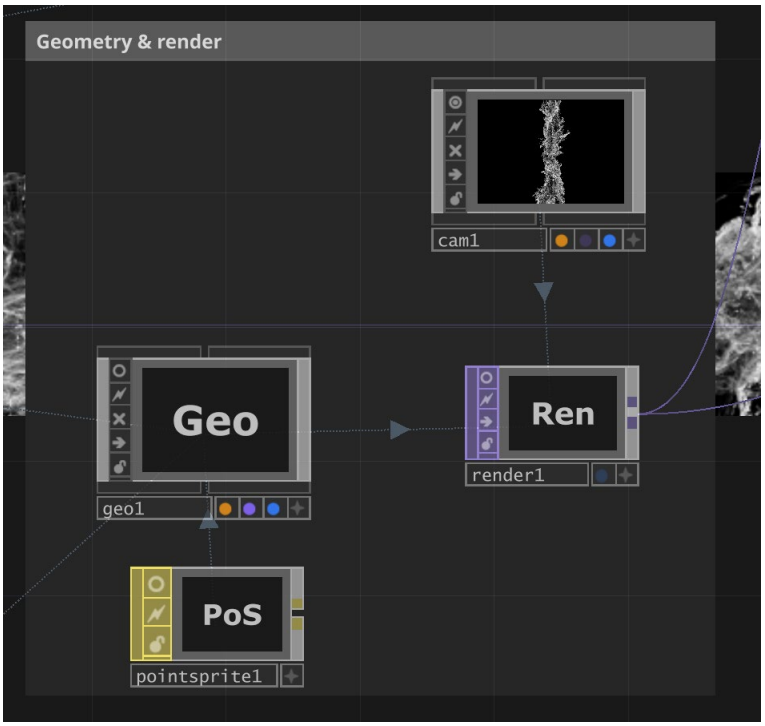

Figura 3.20: Progetto Touch Designer sul flusso finale: geometria e render

A questo punto, come nel progetto precedente relativo al tempio di *Philae*, è necessario unire tutte le informazioni riguardanti posizione, materiale e colore: questo si ottiene attraverso il nodo COMP "Geometry". Nella sua sezione Instance, nel parametro "Translate OP", viene collegato il nodo "Pos", mappando i suoi valori r, g, b come traslazione su x, y, e z. Successivamente, nella sezione Instance2, il parametro "Color OP" è impostato con il nodo

"Col", con una corrispondenza diretta dei valori r con r, g con g, e così via. Infine, per aggiungere il materiale, si utilizza il nodo "PointSprite", collegato al nodo "Geometry" nella sezione Render. Successivamente, vengono aggiunti i nodi "Camera" e "Render" per posizionare la camera nella posizione ottimale e impostare la risoluzione. Per il nodo "Render", si cerca di utilizzare la risoluzione quadrata più alta possibile per mantenere le proporzioni corrette; nel caso di questo progetto, tutto si basa su proporzioni quadrate, quindi una risoluzione con valori diversi su x e y deformerebbe le particelle. Utilizzando un nodo successivo, è possibile ottenere la risoluzione desiderata. Considerando la complessità del progetto, è necessario trovare un equilibrio tra il numero di particelle e la risoluzione per evitare il crash del programma.

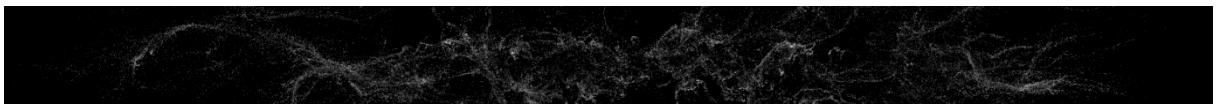

Figura 3.21: Risultato parziale, flusso solo particellare

Il risultato alla massima espansione potrebbe soddisfare le aspettative se si desidera un effetto altamente particellare. Tuttavia, per ottenere un effetto leggermente più denso senza sovraccaricare il sistema, è stato aggiunto un altro nodo "Feedback" per sovrapporre il movimento delle particelle con la scia di quelle precedenti, creando una sorta di patina.

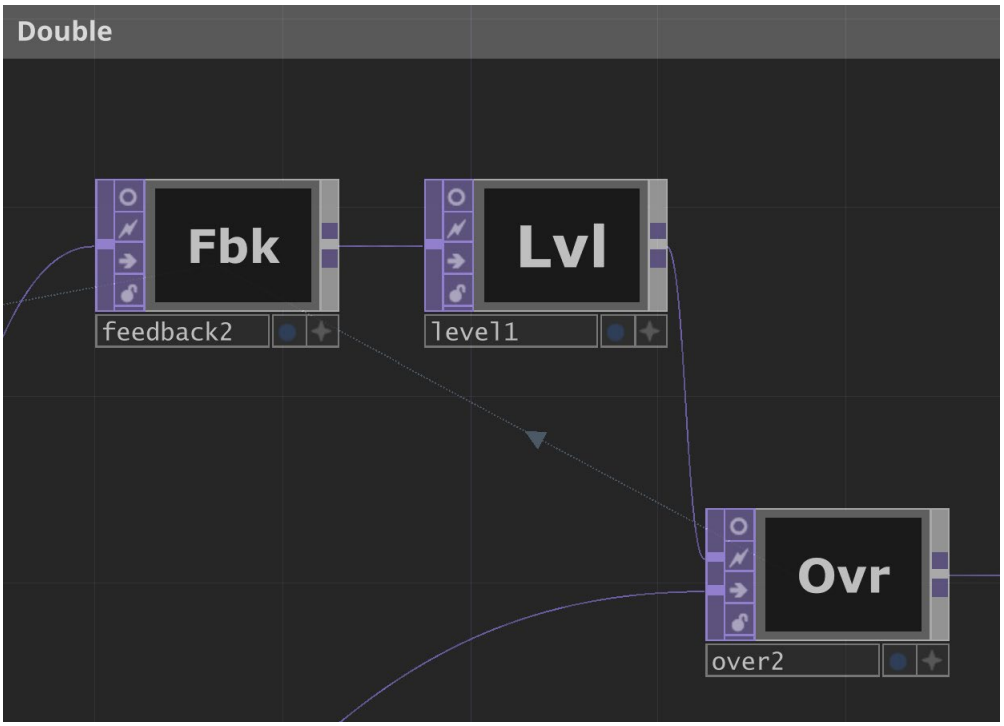

Figura 3.22: Progetto Touch Designer sul flusso finale: secondo feedback

Questo effetto è stato ottenuto collegando il nodo “Render” al nodo “Feedback”, che riceve informazioni dal nodo “Over”. Quest'ultimo è connesso sia al ramo del “Feedback” sia direttamente al nodo “Render”, unendo i due livelli. Dopo il nodo “Feedback”, è stato aggiunto il nodo “Level” per gestire gamma, opacità, luminosità e altre proprietà della sovrapposizione.

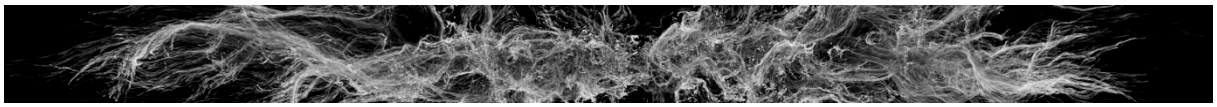

Figura 3.23: Risultato finale, con la sovrapposizione del secondo feedback

Il risultato ottenuto è meno particellare, ma più adatto ai nostri scopi.

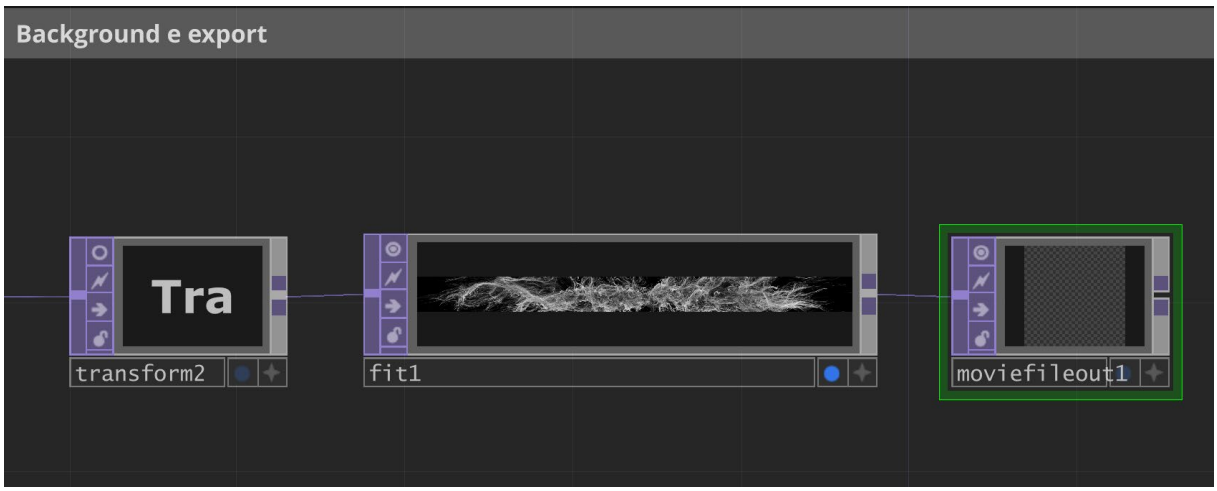

Figura 3.24: Progetto Touch Designer sul flusso finale: background e export

Successivamente, gli ultimi aspetti dell'uscita del video sono definiti attraverso il nodo “Transform”, che permette di impostare il colore del background e applicare ulteriori traslazioni e rotazioni finali. Poi, il nodo “Fit” consente di ottenere la risoluzione finale desiderata, indipendentemente da quella del nodo “Render”, che serve solo a stabilire la qualità della visione delle particelle. Infine, il nodo “MovieFileOut”, già descritto nel capitolo precedente, viene utilizzato per renderizzare l’animazione. Di seguito sono riportate alcune immagini delle fasi principali del processo.

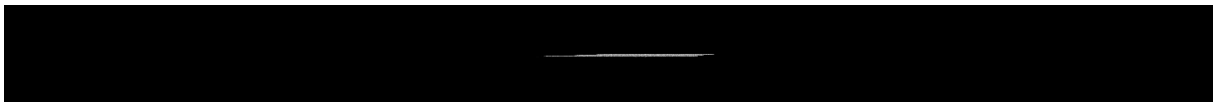

Figura 3.25: Render del flusso: frame iniziale

Figura 3.26: Render del flusso: inizia l'espansione

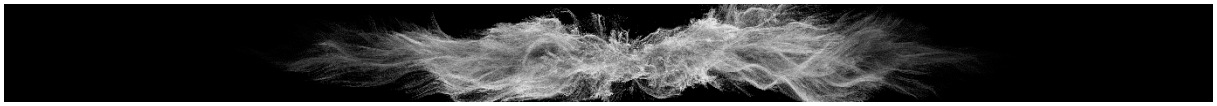

Figura 3.27: Render del flusso: l'espansione continua

Figura 3.28: Render del flusso: il flusso è quasi alla larghezza massima

3.3.1 Topaz Gigapixel AI: alcune informazioni addizionali

Il flusso finale, destinato a occupare mezza sala, necessita di una risoluzione di 13500x1080. Quindi, per mantenere la qualità visiva, anche le particelle dovrebbero avere una risoluzione elevata, idealmente vicina ai 13500, per evitare che risultino sfocate su un'enorme parete. Tuttavia, aumentare la risoluzione del nodo “Render” incrementa notevolmente il carico computazionale, riducendo la dimensione delle particelle e necessitando un aumento del loro numero; purtroppo nessuno dei computer disponibili poteva gestire una risoluzione di 13500x13500 con un migliaio di particelle. Per risolvere questo problema senza compromettere la qualità, è stato deciso di utilizzare il software Topaz Gigapixel AI, che sfrutta l'intelligenza artificiale per aumentare la risoluzione delle immagini, mantenendo un'alta qualità senza sgranature. Si è optato per un render a metà della risoluzione desiderata, impostando il nodo “Render” a 6750x6750 e il nodo “Fit” a 6750x540, consentendo a Gigapixel di raddoppiare la risoluzione. Per rendere l’animazione più densa possibile, il numero di particelle è stato aumentato fino al massimo consentito dai computer disponibili, ovvero circa 1500 particelle.

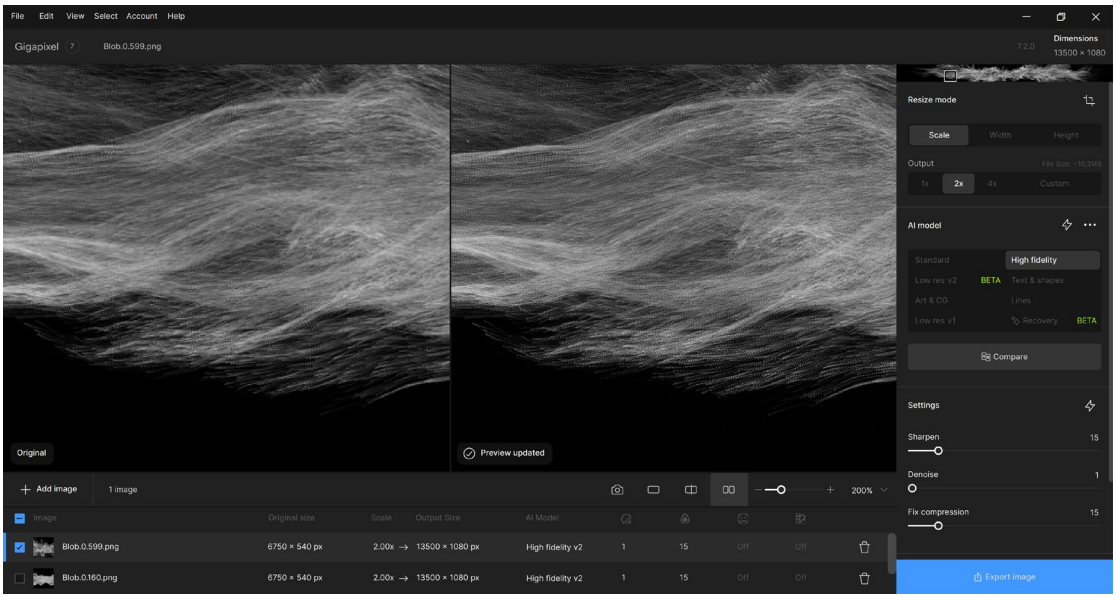

Figura 3.29: Interfaccia di Gigapixel AI, con l'anteprima dell'upscaling di un frame

L'interfaccia di Gigapixel AI è molto intuitiva: dopo aver aggiunto le immagini, si possono impostare le proprietà per le operazioni nel menù a destra. Il parametro fondamentale è il 2x dell'output, mentre gli altri parametri variano in base al tipo di immagine e alla qualità desiderata. Si è scelto di utilizzare il modello AI per l'alta fedeltà dell'immagine, con i parametri visibili in basso a destra: “Sharpen” aumenta la nitidezza dell'immagine rendendo i dettagli più chiari; “Denoise” riduce il rumore digitale, migliorando la qualità delle immagini scattate in condizioni di scarsa illuminazione, parametro impostato a 1 poiché non è il nostro caso; “Fix Compression” corregge gli artefatti di compressione, come blocchi e bande di colore, migliorando la qualità visiva delle immagini compresse. Utilizzati insieme, questi strumenti migliorano significativamente la nitidezza, la pulizia e la qualità complessiva delle immagini ingrandite. Una volta selezionati i parametri e verificata la preview, che permette di vedere un leggero miglioramento, si può esportare l'immagine scegliendo il formato del file, come JPEG o PNG, e la qualità desiderata.

CONCLUSIONE

In conclusione, la presente tesi ha esplorato un ampio spettro di tecniche avanzate per la creazione di scene realistiche e visivamente accattivanti. L'obiettivo principale è stato, infatti, quello di sviluppare ambienti virtuali dettagliati e dinamici, con particolare attenzione agli effetti visivi e alla realisticità dell'ambiente, per le scene su Unreal Engine, e alla gestione delle particelle, per le scene su TouchDesigner. Per raggiungere questo scopo, sono stati utilizzati metodi innovativi di gestione per l'acqua e per le roccie, ed è stato creato un workflow funzionale per ottenere fotogrammetrie realistiche da poter utilizzare su Unreal Engine.

Per quanto riguarda Unreal Engine, ci si è focalizzati, infatti, sulla simulazione dell'acqua e dei processi di erosione delle rocce. Per prima cosa si è sfruttato il modello dell'acqua già presente in Unreal, modificando il suo complesso materiale in modo da ottenere effetti specifici come i Ripple e l'uso di spline per gestire la forma dell'acqua. Successivamente, attraverso algoritmi specifici, nonché aiuti grafici – quali Niagara System e PostProcess Volume –, è stato possibile generare l'erosione delle rocce in modo realistico, tenendo conto della profondità dell'acqua e della posizione dei vertici delle mesh. In questo caso – lavorando su video renderizzati offline –, questa tecnica di erosione è stata più che funzionale, nonostante il carico computazionale elevato; infatti, se si volesse usare questa tecnica per la realizzazione di scene real-time, si produrrebbe, al contrario, un brusco calo di framerate. In più, l'adozione del Path Tracing ha permesso di ottenere immagini di alta qualità, grazie alla simulazione accurata dell'interazione della luce con i materiali, producendo effetti di riflessione e rifrazione molto realistici. Tuttavia, sono emerse alcune limitazioni del Path Tracing nella gestione dell'acqua, portando all'uso complementare del sistema di rendering Lumen per determinate scene. E purtuttavia, sebbene Lumen offra prestazioni superiori in tempo reale, la qualità delle ombre e delle riflessioni risulta leggermente inferiore rispetto al Path Tracing, rendendo necessarie ulteriori ottimizzazioni.

Un altro aspetto chiave della tesi è stato l'utilizzo della fotogrammetria per creare modelli 3D dettagliati: si è quindi giunti alla creazione di un workflow che parte dalle immagini, fino a giungere su Unreal, con l'uso di svariati software. La fotogrammetria ha permesso di trasformare un insieme di fotografie ad alta risoluzione in modelli 3D dettagliati, utilizzando software avanzati come Reality Capture. Questo processo ha richiesto un'attenta preparazione delle immagini, incluso il trattamento del colore e la rimozione delle ombre indesiderate, per garantire che il modello finale fosse il più realistico possibile; infatti strumenti come DaVinci Resolve e Agisoft Delighter hanno giocato un ruolo cruciale in questa fase, migliorando la

qualità delle texture e uniformando l'illuminazione. Una volta ottenuto il modello 3D perfezionato, Blender è stato utilizzato per ulteriori modifiche e rifiniture, correggendo eventuali imperfezioni e preparando il modello per l'importazione in Unreal Engine. L'integrazione del modello in Unreal ha richiesto una cura particolare nella gestione dell'atmosfera e dell'illuminazione, per cui si è utilizzato il blueprint dell'UltraDynamicSky: questo strumento ha permesso di simulare accuratamente il ciclo e le caratteristiche del sole e della luna, aggiungendo un ulteriore livello di realismo alla scena. Il processo ha avuto il suo culmine con la creazione di un Level Sequence in Unreal, dove sono stati animati tutti gli elementi necessari della scena, in modo da creare animazioni realistiche – come per il ciclo giorno/notte. Grazie a questi strumenti e tecniche, è stato quindi possibile ottenere un risultato finale che non solo rappresenta visivamente il sito archeologico con grande accuratezza, ma che riesce anche a trasmettere l'atmosfera e le condizioni ambientali del luogo, offrendo un'esperienza immersiva e coinvolgente.

Per di più, si è voluto sfruttare alcune riprese realizzate in Egitto da Robin Studio per ottenere i Gaussian Splat di siti archeologici. Tuttavia, per quanto realistici, a questi modelli non è possibile eliminare le ombre per creare scene dinamiche, quindi, per superare tali limitazioni, si è optato per una rappresentazione particellare dei modelli tramite TouchDesigner. Questo approccio ha permesso di visualizzare i modelli in modo dinamico, non realistico, ma visivamente accattivante, con transizioni fluide tra visuali differenti. Un esempio significativo è riscontrabile in quella che è stata la realizzazione della versione particellare del tempio di *Philae*, il cui flusso di particelle animate – ottenuto utilizzando il nodo di “Switch” in TouchDesigner – ha permesso una trasformazione della struttura, visibile da due angolazioni differenti.

Un ulteriore esempio di utilizzo avanzato di TouchDesigner è fornito da quella che è stata la creazione di un flusso di particelle – quasi rappresentante il flusso di idee – che si espande dal centro verso i lati. Partendo da un piano particellare, animato attraverso nodi di noise e timer, è stato possibile creare un movimento casuale e crescente delle particelle. L’utilizzo del nodo di “feedback” ha permesso di ottenere una rappresentazione densa del flusso di particelle, per un effetto molto coinvolgente. Per gestire la complessità di questa parte del progetto e ottimizzare le prestazioni, è stato adottato il software di post-produzione Topaz Gigapixel AI, che ha permesso di aumentare la risoluzione delle immagini senza perdere qualità, pur renderizzando ad una risoluzione più bassa di quella voluta.

In conclusione, – nonostante la necessità futura di ottimizzare le soluzioni illustrate, esplorando nuove metodologie di memorizzazione delle posizioni dei vertici, per poter erodere real-time, e adottare tecniche di animazione tramite blueprint – il workflow elaborato ha dimostrato la capacità di combinare diverse tecnologie e strumenti per ottenere risultati visivi di alta qualità e realisticità. L'uso di tecniche avanzate di rendering, algoritmi di erosione, e gestione delle particelle hanno portato ad un alto livello di immersività. L'integrazione tra Unreal Engine e TouchDesigner ha permesso di sfruttare al meglio le potenzialità di entrambi i software, creando scene dinamiche e realistiche per vari contesti, dalla visualizzazione architettonica alle installazioni artistiche.

BIBLIOGRAFIA E SITOGRAFIA

DEL VESCO Paolo, POOLE Federico, TÖPFER Susanne, *Deir el-medina through the kaleidoscope*, Franco Cosimo Panini Editore S.p.A., 2022

DITNER Arthur, FISSOUN Daria, ROBERTS Chris, SCOPPETTUOLO Dion, *The Editors Guide to DaVinci Resolve 18,* Blackmagic Design Training Series, 2023

RAMOS Brais Brenlla, *Unreal Engine 5 Shaders and Effects Cookbook: Over 50 recipes to help you create materials and utilize advanced shading techniques*, Packt Publishing, 2023

ROMERO Marcos, SEWELL Brenden, (con prefazione di) CATALDI Luis, *Blueprints Visual Scripting for Unreal Engine 5: Unleash the true power of Blueprints to create impressive games and applications in UE5,* Packt Publishing Limited, 2022

SORKHABI, Elburz, *Introduction to TouchDesigner 099*, LeanPub, 2019

SPINA, Giorgio, *La civiltà sul Nilo. Storia e cultura dell'antico Egitto,* De Ferrari, 2008

https://gallerieditalia.com/it/torino/mostre-e-iniziative/in-evidenza/2024/06/13/paesaggi-landscapes-museo-egizio/#

https://www.museoegizio.it/esplora/notizie/il-museo-egizio-e-gallerie-ditalia-presentano-paesaggi-landscapes/

https://www.storicang.it/a/il-pane-e-le-rose-i-lavoratori-di-deir-el-medina_16252

https://www.vanillamagazine.it/deir-el-medina-il-villaggio-degli-artisti-della-valle-dei-re/

https://italiano.memphistours.com/Egitto/Guida/Aswan/wiki/Il-Tempio-di-File

https://www.touregitto.com/guida-viaggio-egitto/tempio-diphilae#:~:text=Il%20Tempio%20di%20Philae%20%C3%A8,preg%C3%B2%20per%20la%20sua%20resurrezione

https://dev.epicgames.com/documentation/it-it/unreal-engine/unreal-engine-5-3-documentation?application_version=5.3

https://medium.com/unreal-engine-technical-blog/pure-impure-functions-516367cff14f

https://learn.microsoft.com/it-it/windows/win32/direct3dhlsl/dx-graphics-hlsl

https://www.treccani.it/

https://catlikecoding.com/unity/tutorials/flow/waves/

https://agisoft.freshdesk.com/support/solutions/articles/31000158376-agisoft-texture-de-lighter-general-workflow

https://interactiveimmersive.io/touchdesigner-user-interface/

https://docs.derivative.ca/Main_Page

(siti consultati per l'ultima volta domenica 23 giugno)