Politecnico di Torino

Politecnico di Torino

Corso di Laurea magistrale in Ingegneria del Cinema e dei Mezzi di Comunicazione
A.A. 2022/2023
Sessione di Laurea Marzo/Aprile 2023

Modellazione 3D, simulazioni fisiche e compositing per una produzione animata indipendente

Relatore:
Prof.: Riccardo Antonio Silvio Antonino
Tutor aziendale: Matilde Capello

Candidata:
Mariagiusy Simeti

Ad A., senza la quale
questo percorso non
sarebbe mai iniziato.

Abstract

Nell'immaginario collettivo, la tecnica dell'animazione viene quasi sempre associata al racconto di storie rivolte ad un pubblico giovane, ma le possibilità visive infinite di questo medium permettono non solo di raccontare storie puramente di fantasia, ma anche di dare uno sguardo alternativo e originale su eventi realmente accaduti. È questo il caso di *En la tierra de los sueños perdidos*, la storia di un uomo che ha combattuto durante la guerra delle Malvinas, conflitto avvenuto sulle isole Falkland nel 1982 tra Argentina e Gran Bretagna. Non vengono raccontati fatti storici della guerra, bensì le emozioni di un uomo che da giovane fu costretto a prendere parte ad una guerra dalla quale sarebbe uscito sconfitto, abbandonando tutti i suoi sogni.

La storia viene raccontata attraverso un cortometraggio animato realizzato utilizzando tecniche miste di animazione, in cui le ambientazioni sono realizzate in 3D, mentre i personaggi sono disegnati a mano. Com'è possibile mescolare due tecniche di animazione il cui concetto alla base è profondamente diverso? Come si riesce ad ottenere uno stile coerente quando una parte della scena viene realizzata tramite un computer e l'altra parte tramite il disegno a mano?

Il progetto di tesi *Modellazione 3D, simulazioni fisiche e compositing per una produzione animata indipendente* illustra come sono stati realizzati ambienti, esplosioni, effetti visivi in 3D utilizzando Blender, un software gratuito e open-source spesso utilizzato per creare contenuti all'interno di produzioni indipendenti e low-budget, e come sia stato utilizzato il compositing 3D per ottenere la resa stilistica desiderata.

Il cortometraggio nasce da un'idea di Matilde Capello ed è stato realizzato all'interno dell'azienda Robin Studio s.r.l., che ha offerto supporto infrastrutturale.

Ringraziamenti

Ci tengo a ringraziare in primis il professore Riccardo Antonino e Mati per avermi dato la possibilità di lavorare a questo progetto, permettendomi di imparare tantissimo.

Ringrazio i miei genitori e la mia famiglia, per avermi supportato in questo lungo percorso che finalmente (anche se un po' a malincuore) giunge al termine.

Ringrazio Maria che, anche se a distanza, non mi ha mollata un secondo, sopportando i mille minuti di messaggi vocali mandati e ascoltati in x2, per aver creduto sempre in me più di chiunque altro, per avermi sollevata nei miei momenti tristi e per aver riso con me nei momenti felici, ma soprattutto per esserci, da sempre e per sempre.

Ringrazio Simo, per essere l'unica persona sbadata, distratta e ritardataria che sopporto, per avermi regalato momenti bellissimi, per aver cantato con me e per avermi fatto fare il pranzo della domenica più bello del mondo.

Ringrazio i miei amici dell'uni, Ale, Samu, Vincenzo e tutti quelli che mi hanno accompagnata in questo percorso, per le carbonare e per i pettegolezzi e ogni tanto per qualche momento di studio.

Ringrazio gli SOS per aver fatto infiniti chilometri per riaccompagnarmi a casa e per avermi insegnato che dopo la tempesta esce sempre il sole.

Ringrazio tutti i ragazzi di Robin Studio, per avermi aiutata durante questo progetto, per avermi fatto compagnia e per avermi sopportato quando venivo a disturbarvi, ma soprattutto ringrazio "quelli del 3D" che mi hanno fatto tanto ridere.

Infine ringrazio me stessa, per non essermi mai arresa.

Indice

Elenco delle figure

Capitolo 1

Introduzione

L'animazione è l'arte di creare il movimento attraverso l'utilizzo di immagini statiche che si susseguono rapidamente. Questa è resa possibile grazie al principio della *persistenza della visione* dell'occhio umano, una proprietà secondo cui un'immagine persiste sull'occhio dello spettatore per qualche frazione di secondo prima di sparire. Sfruttando questa capacità ottica e utilizzando un numero adeguato di fotogrammi al secondo (un numero maggiore di 10 frame al secondo), è possibile creare l'illusione del movimento. L'animazione può essere realizzata in moltissimi modi differenti, utilizzando tecniche tradizionali come il disegno realizzato a mano (animazione 2D) oppure tecniche digitali come la computer grafica (animazione 3D).

La tecnica dell'animazione 3D è una tecnica largamente diffusa ed è sicuramente un campo in continua evoluzione, che ha rivoluzionato l'industria dell'intrattenimento e della comunicazione. Negli ultimi decenni, l'avvento della tecnologia ha permesso una maggiore accessibilità e una crescente diffusione di questa forma di animazione che oggi viene utilizzata in molti diversi ambiti, dai film d'animazione alle pubblicità, dai videogiochi ai documentari animati.

Per quanto riguarda questa tesi è stato deciso di portare a termine il progetto già avviato nel 2020 del cortometraggio animato *En la tierra de los sueños perdidos*, un cortometraggio realizzato utilizzando una tecnica mista di animazione,

sfruttando l'animazione 2D per quanto riguarda i personaggi, e l'animazione 3D per la realizzazione delle ambientazioni e dei vari effetti visivi.

1.1 *En la tierra de los sueños perdidos*

En la tierra de los sueños perdidos (*Nella terra dei sogni perduti*) è un cortometraggio animato, più nello specifico un *documentario animato*, che racconta le vicende di un uomo di circa cinquant'anni che da giovane ha preso parte alla Guerra delle Malvinas, una guerra durata poco più di 70 giorni nel 1982, avvenuta in Argentina sulle isole Falkland, tra, appunto, Argentina e Gran Bretagna.
Pur essendo le ambientazioni quelle effettive della vicenda, il cortometraggio non vuole raccontare fatti storici di guerra, bensì le sensazioni e le emozioni di un uomo che fu costretto ad abbandonare i suoi sogni per prendere parte ad una guerra dalla quale sarebbe sicuramente uscito sconfitto. Infatti, l'uomo racconta di come il suo tenente, Minguzzi, avesse illuso lui e i suoi compagni di poter vincere la guerra, ed una volta finita, di poter tornare alla vita di sempre, quando in realtà questo non avvenne e le ripercussioni per i soldati furono molto profonde e molto gravi. Il cortometraggio inizia con il protagonista che racconta la sua storia ad un'acquirente che sfogliava una rivista, narrando di come lui sarebbe dovuto diventare un pittore e non un soldato.

1.2 Obiettivi

Come è stato già detto, per la realizzazione del cortometraggio *En la tierra de los sueños perdidos* si è voluto sperimentare nell'utilizzo di 2 tecniche di animazione, realizzando quindi un prodotto ibrido. Questo miscuglio di tecniche lo vediamo innanzitutto nell'effettiva realizzazione delle varie parti, quindi le ambientazioni in computer grafica e i personaggi disegnati e animati a mano, ma anche per quanto riguarda lo stile e l'estetica, in quanto essendo due tecniche di animazione molto

diverse, era necessario rimanere su uno stile coerente per entrambe e che camuffasse la natura delle stesse.

L'obiettivo di questa tesi è quello di analizzare e approfondire le parti di computer grafica, come sono state create le scene, quali sono e come sono stati realizzati gli effetti visivi e le simulazioni fisiche presenti e com'è stato possibile ottenere una resa stilistica che coincidesse e si amalgamasse con quella dei personaggi creati e animati in 2D.

1.3 La pipeline produttiva di un cortometraggio animato

La realizzazione di un prodotto animato, in questo caso un cortometraggio, richiede di seguire delle fasi ben specifiche, affinché si possa ottenere un buon risultato nei tempi prefissati. Le fasi principali della pipeline sono tre e sono *preproduzione*, *produzione* e *post-produzione*.

Preproduzione

La fase di preproduzione di un cortometraggio animato è cruciale per il successo del progetto, poiché in questa fase vengono prese le decisioni chiave che influenzeranno tutto il processo di produzione successivo. Alcune attività che vengono svolte durante la fase di preproduzione di un cortometraggio animato sono:

- *Ideazione e sviluppo della storia*: in questa fase, gli autori sviluppano l'idea e la storia del cortometraggio. Questo include la definizione dei personaggi, del mondo in cui si svolge la storia e degli eventi chiave che si verificano.

- *Sceneggiatura*: una volta che la storia è stata definita, viene scritta la sceneggiatura del cortometraggio, ovvero tutti i dialoghi e tutto quello che accade

nelle varie scene. La sceneggiatura include inoltre tutte le informazioni necessarie per la creazione dello storyboard, come la descrizione delle scene, dei personaggi e delle azioni.

- *Storyboard*: lo storyboard è una sequenza di disegni che rappresenta il cortometraggio nella sua interezza. Lo storyboard aiuta a visualizzare la storia e la regia del cortometraggio e può essere utilizzato come guida durante la produzione. Qui vengono segnati tutti i movimenti di camera, quanto deve durare ogni singola scena e altri dettagli utili per la successiva fase di produzione.
- *Animatic*: una volta che il layout è stato completato, viene creato l'animatic. Questo è un vero e proprio premontato dei disegni realizzati nello storyboard ed è quindi un'anteprima del cortometraggio a cui si aggiunge il dialogo, la musica e gli effetti sonori. L'animatic viene utilizzato come guida per la produzione finale.
- *Pre produzione tecnica*: durante questa fase viene deciso quale software e hardware utilizzare per la produzione del cortometraggio. Inoltre, qualora fosse necessario, vengono sviluppati strumenti specifici per il progetto e viene effettuata la preparazione tecnica per la produzione.
- *Design*: il look finale del lavoro viene deciso a questo punto, disegnando il concept dei personaggi, dei costumi, dei props (oggetti di scena) e dell'ambiente.

Produzione

Questa è la fase in cui tutti i precedenti punti vengono trasformati in prodotti concreti, ma i singoli passaggi saranno approfonditi nel capitolo 4.

Post-Produzione

Nella fase di post-produzione sono messi in atto i passaggi finali della produzione del prodotto prima della distribuzione. Qui attraversiamo le fasi di montaggio video

e audio, *color correction*, l'eventuale aggiunta di effetti visivi in 2D e soprattutto una fase di *compositing*. Nel caso della produzione di *En la tierra de los sueños perdidos*, è stata realizzata sia una parte di *compositing* in 2D, sia una parte di *compositing* in 3D, sulla quale ci concentreremo nel capitolo 5.

Molti di questi passaggi erano stati già portati a termine nel 2020 quando il progetto ha avuto inizio, come ad esempio tutta la fase di preproduzione. Qui ci concentreremo sulla fase di produzione e compositing dei contenuti 3D mancanti al fine di completare definitivamente il progetto.

Capitolo 2

Stato dell'arte

Prima di addentrarci nella spiegazione del lavoro svolto, è necessario analizzare quali altri prodotti audiovisivi sono stati creati fino ad oggi con lo stesso stile scelto per rappresentare il cortometraggio, o comunque simile, ovvero quello di animazione ibrida, e quali sono i prodotti che occupano il genere del documentario animato.

Spiderman: Into the Spider-Verse

Il film d'animazione che ha cambiato definitivamente il modo di vedere la computer grafica è sicuramente *Spiderman: Into the Spider-Verse*, con il suo stile innovativo è stato il primo del suo genere. Il film del 2018 ha conquistato il pubblico con il suo stile visivo non ortodosso, visto come un ritorno alle origini dell'estetica dei fumetti. Ci si trova davanti ad una nuova sfida di animazione in cui "computerizzazione" e personalizzazione si mescolano per creare un'opera di pop art, e pur essendo in un epoca in cui si rincorre a tutti i costi il fotorealismo, i registi hanno investito nell'irreale. Il film propone dei modelli 3D con una grafica 2D come quella dei fumetti, i cui effetti sono disegnati a mano con le loro imperfezioni, lontani quindi dalla macchinosità e perfezione. La combinazione delle linee facciali prima disegnate e poi animate è stata cruciale per il coinvolgimento emotivo dello spettatore, il quale

era l'obiettivo principale degli animatori.

Dall'animazione tradizionale non viene ripresa soltanto l'estetica, ma anche la tecnica di "animazione a passo due". I film di animazione in computer grafica tradizionali sono generalmente animati a 24 frame al secondo, utilizzando quindi la tecnica dell'"animazione a passo uno", ma in questo caso si è deciso di utilizzare 12 frame al secondo per alcune scene, ciò significa che ogni immagine non viene mostrata solo per un frame, bensì per due. In questo modo si riesce ad enfatizzare la dinamicità della corsa del protagonista animando a passo uno, mentre si ha un rallentamento quando, ad esempio, il protagonista fa leva sul pavimento per alzarsi in piedi, enfatizzando il lavoro opposto della forza di gravità, animando a passo due.

Figura 2.1: *Spiderman: Into the Spider-Verse.*

Il gatto con gli stivali 2 – L'ultimo desiderio

Un altro esempio più recente è invece *Il gatto con gli stivali 2 – L'ultimo desiderio*, film della DreamWorks uscito nel 2022 che ripropone uno stile ispirato al film di Spiderman. Già dalle prime sequenze capiamo la volontà del regista di ricreare una cornice artistica che richiama più il fumetto che le pagine di una favola per

bambini. Possiamo notare alcune linee cinetiche che seguono i movimenti dinamici dei protagonisti e uno stile che richiama le pennellate di un artista.

Anche in questo caso ci sono alcune scene, in particolare quelle d'azione e più dinamiche, che presentano 12 frame al secondo, a differenza delle altre parti del film in cui si hanno 24 frame al secondo. Questa è una scelta che si contrappone a quella fatta per il film di Spider-Man, perché in questo caso sono proprio le scene più dinamiche ad essere realizzare a 12 frame al secondo.

Figura 2.2: *Il gatto con gli stivali 2 – L'ultimo desiderio.*

Tartarughe Ninja: Caos Mutante

Sempre su questa scia troviamo il recentissimo trailer di *Tartarughe Ninja: Caos Mutante*, che presenta il film in uscita ad agosto 2023. Come afferma Kellan Jett, un artista dello sviluppo visivo del film, c'è un'altra influenza oltre a Spider-man, ovvero il film *I Mitchell contro le macchine* del 2021. Questo è dovuto principalmente al fatto che una grande percentuale del team artistico del film delle Tartarughe Ninja ha lavorato anche per il film *I Mitchell contro le macchine*, rendendo così quasi inevitabile il fatto di orientare il film verso uno stile "a fumetto".

Figura 2.3: *Tartarughe Ninja: Caos Mutante.*

2.1 Il documentario animato

En la tierra de los sueños perdidos, più che come semplice cortometraggio animato, può essere identificato nel genere del *documentario animato*.

Il documentario animato è un genere di film documentario che utilizza la tecnica dell'animazione per rappresentare i fatti, le persone e gli eventi del mondo reale. A differenza dei documentari tradizionali, che utilizzano principalmente filmati reali, il documentario animato utilizza l'animazione per creare immagini che rappresentano la realtà in modo più creativo e artistico. La tecnica dell'animazione permette inoltre di rappresentare situazioni o fatti difficilmente riproducibili con riprese in presa diretta, come ad esempio le situazioni di guerra, ed è per questo che il genere si presta perfettamente per raccontare la storia di *En la tierra de los sueños perdidos*.

Il genere del documentario animato nasce nei primi decenni del ventesimo secolo, nonostante ci siano alcune opere cinematografiche precedenti che possono essere considerate precursori di questo genere. Tra queste troviamo *The enchanted drawing*, realizzato da J. Stuart Blackton nel 1900, che utilizza l'animazione per mostrare un disegnatore che crea immagini che si animano sulla carta.

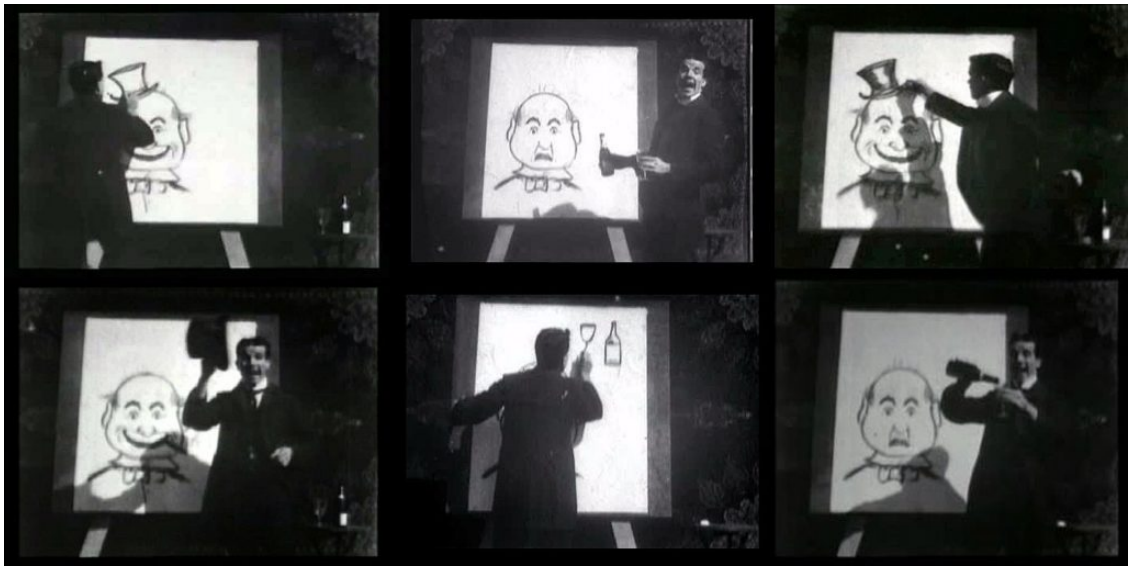

Figura 2.4: *The Enchanted Drawing.*

Uno tra i documentari animati più importanti e innovativi è sicuramente *The sinking of the Lusitania*, del 1918 e diretto da Winsor McCay, che racconta il naufragio del transatlantico britannico.
Il film di McCay utilizza la tecnica dell'animazione, realizzata completamente a mano, per mostrare il naufragio della nave e le conseguenze sulle persone a bordo e sulla popolazione dell'Irlanda. Il documentario ebbe un grande successo e contribuì a sensibilizzare l'opinione pubblica americana riguardo la necessità di entrare in guerra durante la prima Guerra Mondiale. Il film rappresentò un importante punto di svolta nella storia del documentario animato, dimostrando che l'animazione poteva essere utilizzata non solo per scopi ludici, ma anche per raccontare storie

importanti e drammatiche.

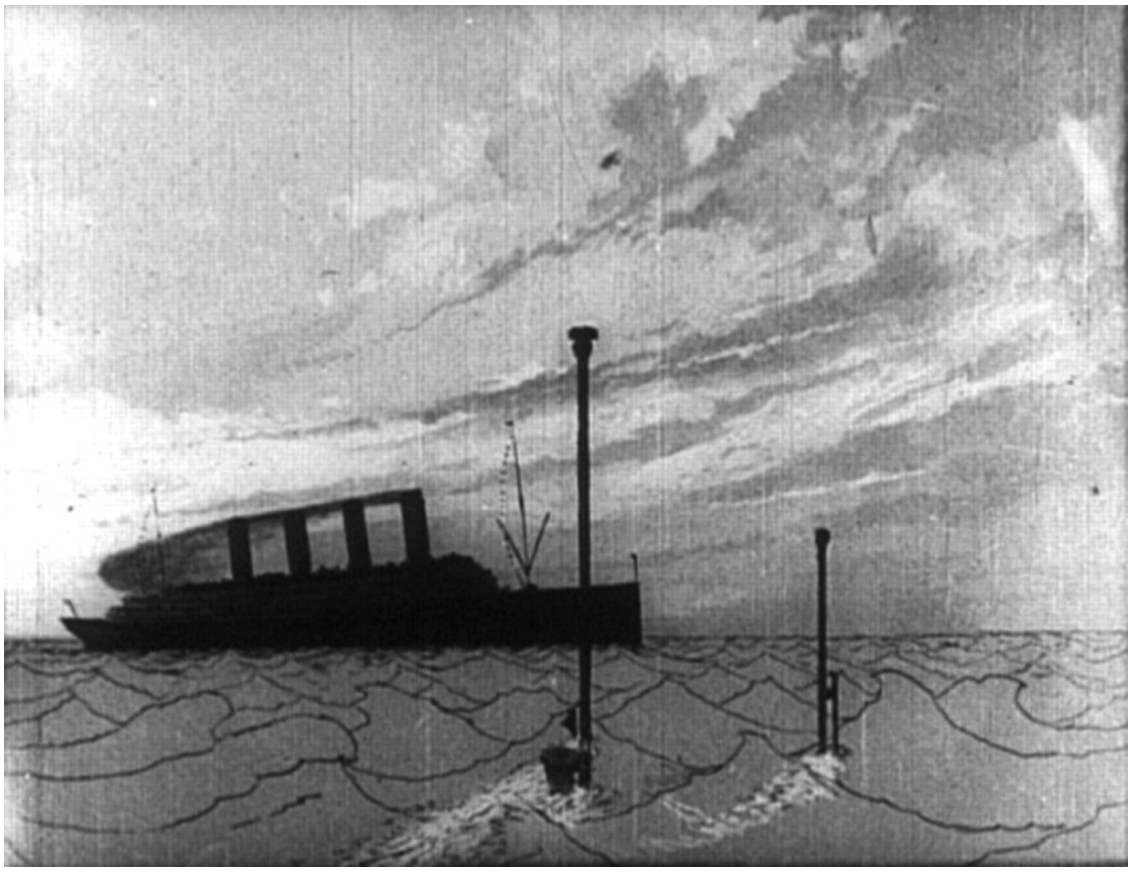

Figura 2.5: *The sinking of Lusitania.*

Il genere del documentario animato ha avuto una crescita significativa negli ultimi anni, grazie alla disponibilità di tecnologie sempre più avanzate e alla creatività dei registi. Tra gli esempi più rilevanti ricordiamo *Tower* (2016) di Keith Maitland, che ricostruisce gli eventi della sparatoria avvenuta nel 1966 alla University of Texas utilizzando animazione, immagini d'archivio e interviste ai testimoni, poi abbiamo *Another day of life* (2018) di Raul de la Fuente e Damian Nenow, che racconta la guerra civile in Angola del 1975 attraverso le esperienze del giornalista polacco Ryszard Kapuściński, e il recente *My favorite war* (2020) di Ilze Burkovska Jacobsen, che racconta la vita della regista in Lettonia durante l'occupazione sovietica e

la lotta per l'indipendenza del paese negli anni '90.

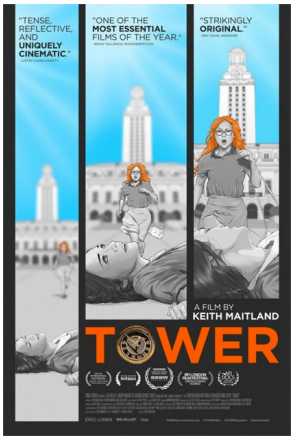

Figura 2.6: *Tower.*

Figura 2.7: *Ancora un giorno.*

Figura 2.8: *My favorite war.*

Capitolo 3

Software

En la tierra de los sueños perdidos è un progetto che nasce nel 2020 e per la quale erano già stati definiti i software da utilizzare. Tra questi, per la produzione dei contenuti 3D vi è *Blender*, il quale, con la sua versatilità e praticità, è stato scelto come software su cui svolgere il lavoro. Ad oggi la produzione dei contenuti è proseguita su questo programma.

3.1 Blender

Blender è un software per la produzione 3D gratuito, open source e multipiattaforma all'interno della quale è possibile attraversare tutte le fasi della *pipeline* di una produzione 3D, dalla modellazione al rigging, dal compositing al rendering. Grazie alla sua flessibilità e alle sue numerose funzionalità, Blender è utilizzato in molti campi, tra cui l'animazione, gli effetti visivi, la progettazione industriale, l'architettura, il design di prodotti e molto altro ancora.

Figura 3.1: Logo di *Blender*.

Con Blender è possibile:

- creare oggetti 3D in modo dettagliato e preciso utilizzando diversi metodi di modellazione, come la modellazione poligonale, la modellazione delle curve, oppure la tecnica dello *sculpting*;
- animare oggetti o personaggi tramite animazione *frame by frame* o con i *keyframe*[1];
- simulare fisica e particelle tramite dei sistemi particellari o strumenti come la simulazione di fluidi, fumo o tessuti;
- fare *compositing*, ovvero "mescolare" immagini e video utilizzando il sistema di nodi di composizione, che consente di creare effetti visivi complessi e integrare le animazioni con le scene 3D in produzioni video;
- renderizzare le scene create in modo realistico, tramite i motori di rendering *Cycles* e *Eevee* che utilizzano funzionalità di *ray tracing*, illuminazione globale e altro ancora.

Figura 3.2: Render di una ciambella realizzata con Blender.

[1]I keyframe, o fotogrammi chiave, sono dei marcatori in cui è possibile memorizzare alcune proprietà di un oggetto in un determinato istante di tempo. Una volta fissati i keyframe iniziale e finale, vengono creati automaticamente i frame intermedi tramite la tecnica dell'interpolazione.

3.2 Blender 3.4 e le differenze con la versione 2.82

La versione di Blender sulla quale sono state realizzate le scene mancanti del progetto è la versione 3.4, nonché la versione più recente rilasciata dalla Blender Foundation. Poiché il progetto di questo cortometraggio era già stato avviato nel 2020, la versione allora utilizzata fu la 2.82, e inevitabilmente le versioni successive hanno introdotto diversi cambiamenti e nuove funzionalità che hanno facilitato alcune fasi del lavoro e migliorato le prestazioni del programma stesso. Tra le novità più importanti ricordiamo:

- **Interfaccia utente**: è stata in generale migliorata e sono state aggiunte nuove icone e opzioni di personalizzazione, in modo da renderla più semplice e intuitiva. È stato aggiunto inoltre un nuovo sistema di gestione dei file, chiamato *Asset Browser*, che consente di gestire e organizzare le risorse del progetto in maniera più efficiente.

- **Modellazione**: è stato aggiunto un nuovo sistema di gestione delle scene basato sui nodi, chiamato *Geometry nodes*, che consente di creare e gestire le scene in modo più flessibile e creativo.

- **Shading**: sono state introdotte nuove funzionalità per la gestione degli *shader*, come il supporto per le texture procedurali e le immagini HDRI[2].

- **Simulazione**: sono stati inseriti nuovi strumenti di simulazione, come la simulazione di tessuti morbidi, un nuovo sistema di particelle basato su volumi, che consente di creare effetti di fumo, nebbia e fuoco più realistici; è stata aggiunta una nuova funzionalità di simulazione basata su AI[3], chiamata Physics-based

[2]High Dynamic Range Image: immagine in cui l'intervallo dinamico, ovvero l'intervallo tra le aree visibili più chiare e quelle più scure, è più ampio che nelle immagini usuali.

[3]Intelligenza artificiale.

Deep Learning, che consente di creare simulazioni complesse con una maggiore precisione, e successivamente è stata introdotta una nuova funzionalità di simulazione di fluidi basata su *SPH* (*Smoothed Particle Hydrodynamics*) che consente di creare simulazioni di liquidi più realistiche.

- **Animazione**: sono stati introdotti nuovi strumenti di animazione per la creazione di *rigging*[4] automatico, nuove funzionalità come la possibilità di creare animazioni procedurali con il sistema di nodi e la possibilità di utilizzare i *driver* per controllare le proprietà degli oggetti tramite formule matematiche, nuove opzioni di *keyframe* e altre migliorie alla curva di *Bezier*.

- **Rendering**: è stata aggiunta una nuova funzionalità di rendering basata su AI, chiamata *Denoiser*, che permette di rimuovere il rumore presente sull'immagine direttamente in fase di rendering, il supporto per il rendering in tempo reale con l'engine *Eevee* e una maggiore flessibilità nella gestione delle ombre e della luce, significativi miglioramenti al motore di render *Cycles*, che adesso supporta l'accelerazione hardware RTX per il *ray tracing* in tempo reale.

3.2.1 Geometry nodes

I *Geometry Nodes* di Blender sono uno strumento di modellazione procedurale che consente di generare e manipolare geometrie in modo non distruttivo, utilizzando un'interfaccia visuale basata su nodi. Sono stati introdotti per la prima volta nella versione di Blender 2.92 il 25 febbraio 2021 ed è uno strumento creato per semplificare il processo di creazione di oggetti 3D complessi, fornendo un'alternativa alla modellazione manuale.

Come suddetto, i geometry nodes sono uno strumento di modellazione procedurale, ovvero usano un approccio alla creazione di oggetti 3D che si basa sull'utilizzo

[4]Il termine "rigging" nell'animazione deriva da "rig" che rappresenta la struttura ossea costruita per identificare le ossa virtuali che consentono al modello di muoversi.

di algoritmi, regole e funzioni matematiche che generano forme e geometrie. In questo tipo di modellazione, infatti, l'oggetto viene creato attraverso la combinazione di una serie di elementi geometrici di base, come le mesh e le curve, e di modifiche applicate a questi elementi tramite regole predefinite. Grazie ai geometry nodes, l'artista può quindi creare una serie di istruzioni, rappresentate dai nodi, che definiscono il comportamento dell'oggetto, al posto di modellare ogni singolo dettaglio dell'oggetto manualmente. Inoltre, i geometry nodes permettono di manipolare le geometrie in modo *non distruttivo*, il che significa che le modifiche applicate all'oggetto non distruggono o modificano permanente la geometria di base. Ad esempio, se si modifica una variabile all'interno del nodo che crea una sfera, come la dimensione, la forma della sfera verrà modificata, ma non distrutta. Questo significa che è possibile apportare modifiche all'oggetto in qualsiasi momento, senza dover ricominciare da capo ogni volta. Questo rende i geometry nodes un modo di creare delle scene computazionalmente meno onerose rispetto a quelle create con altri metodi, come la modellazione manuale o un *particle system*.

Il geometry nodes inoltre non è altro che un modificatore, e agisce come qualsiasi altro modificatore presente sull'oggetto, dunque è importante l'ordine in cui questi vengono assegnati all'oggetto, perché verranno applicati in ordine dall'alto verso il basso. Essendo dei modificatori, possono essere applicati più geometry nodes sullo stesso oggetto.

Una volta aperto il tab Geometry Nodes, la visuale che si presenta è composta principalmente da tre finestre: in alto a destra troviamo il classico *3D viewport*, in alto a sinistra c'è il cosiddetto "foglio di calcolo", che mostra le coordinate di ogni singolo vertice o faccia, mentre in basso abbiamo la finestra in cui verranno posizionati i veri e propri nodi del Geometry nodes, in grado di creare mesh, texture, animazioni e quant'altro.

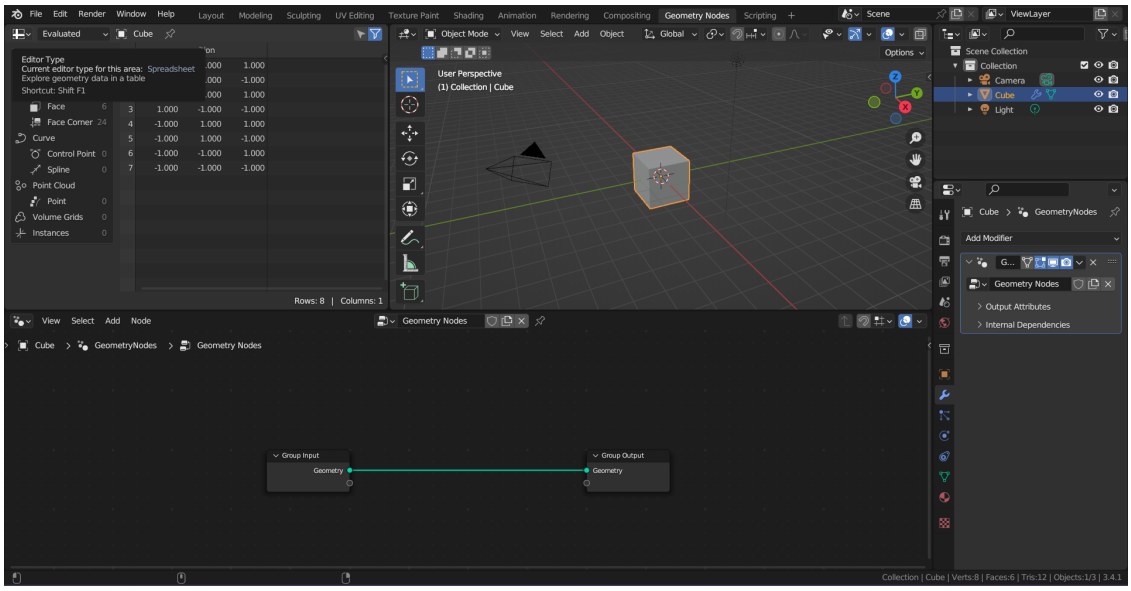

Figura 3.3: Tab *Geometry nodes* in Blender.

La schermata nella quale vengono inseriti i nodi ci mostra di base un nodo di input e uno di output, e qualsiasi altro nodo che verrà inserito tra questi due sarà adibito alla modifica dell'oggetto in input e fungerà come step di post processing. Questo è dimostrabile inserendo, ad esempio, un nodo di *Transform*, il quale modifica la posizione, la rotazione e la scala dell'oggetto, mostrandoci quindi in output un oggetto modificato, ma se passiamo alla *Edit mode*, potremo osservare la forma originale dell'oggetto, a conferma del fatto che i geometry nodes non modificano la mesh in modo distruttivo.

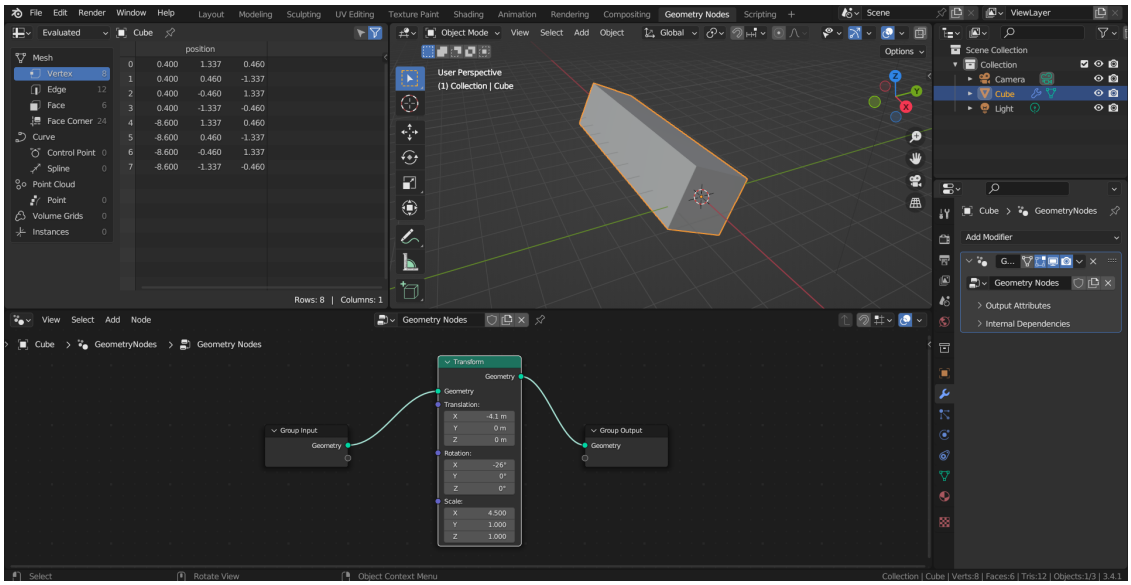

Figura 3.4: Geometry nodes in *Object mode*.

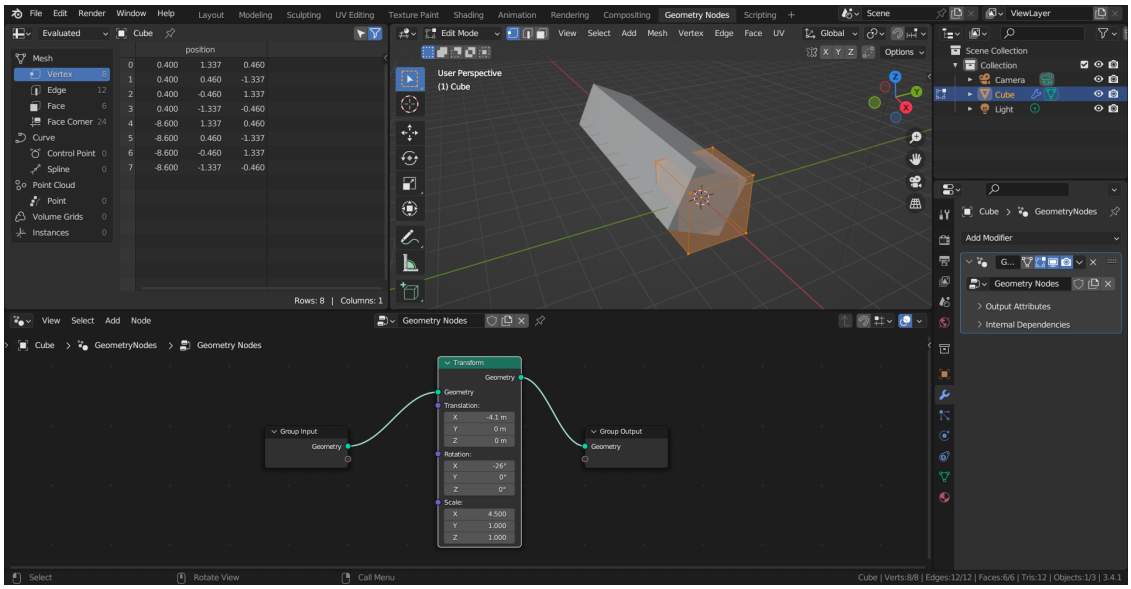

Figura 3.5: Geometry nodes in *Edit mode*.

Capitolo 4

Pipeline di animazione 3D

Dopo aver discusso brevemente della preproduzione nell'introduzione di questa tesi, qui tratteremo la fase della produzione vera e propria, specificando i passaggi necessari per una produzione di contenuti 3D.

La fase di produzione è la fase in cui vengono effettivamente realizzati i contenuti, partendo dalle solide basi costruite durante la fase di preproduzione. Per quanto riguarda la pipeline di animazione 3D, i passi da seguire sono essenzialmente 8.

4.1 Layout 3D

Il layout 3D è per natura strettamente correlato all'*animatic* preparato nella fase di preproduzione, infatti ne è praticamente la trasposizione in tre dimensioni costruita utilizzando dei modelli basici chiamati *proxy*, ma con la giusta scala e le giuste proporzioni che manterrà il prodotto finale. Ad esempio, se nella scena sono presenti dei personaggi che parlano o compiono un'azione, questi verranno rappresentati tramite dei semplici parallelepipedi anziché con la loro forma finale, giusto per avere un'idea dei movimenti e dei volumi occupati dai soggetti.

4.2 Modellazione

È la fase in cui gli ambienti, i props (oggetti di scena) e i personaggi vengono modellati. Questa è una delle fasi su cui si è maggiormente concentrato il lavoro oggetto di questa tesi, non solo modellazione vera e propria, quanto più la composizione delle scene. Sono stati composti gli ambienti di diverse scene e sono state utilizzate molteplici tecniche di modellazione in base alle necessità.

4.2.1 Array modifier

Per realizzare le tegole del tetto nella scena 2 é stato utilizzato l'*Array modifier*, che crea un vettore di copie dell'oggetto.

È stato quindi modellato solo l'oggetto base e poi duplicato attraverso il modificatore, rendendo molto più veloce la composizione del tetto e soprattutto molto più facili eventuali modifiche alla mesh, in quanto sarebbe stato sufficiente modificare solo un oggetto, e non tutti singolarmente. Inoltre, in questo caso sono stati utilizzati due *Array modifier*, uno per le copie in orizzontale e uno per le copie in verticale, e tramite i parametri di *offset* è stato possibile posizionare le tegole più o meno vicine tra loro.

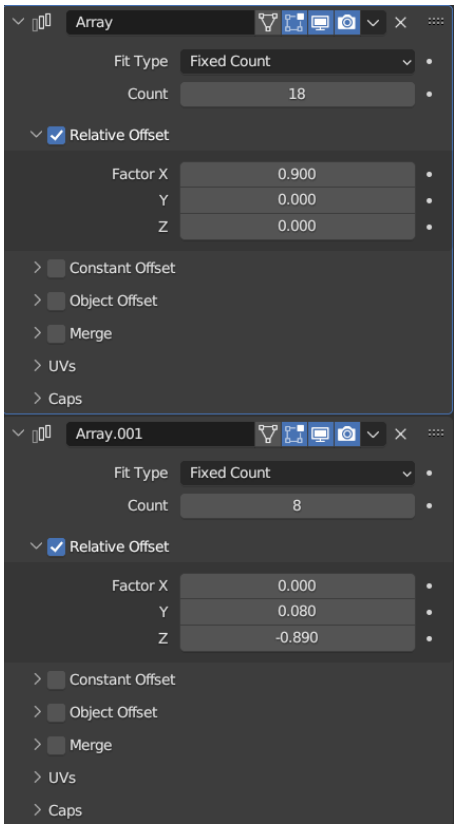

Figura 4.1: *Array modifier* applicato alle tegole della scena 2.

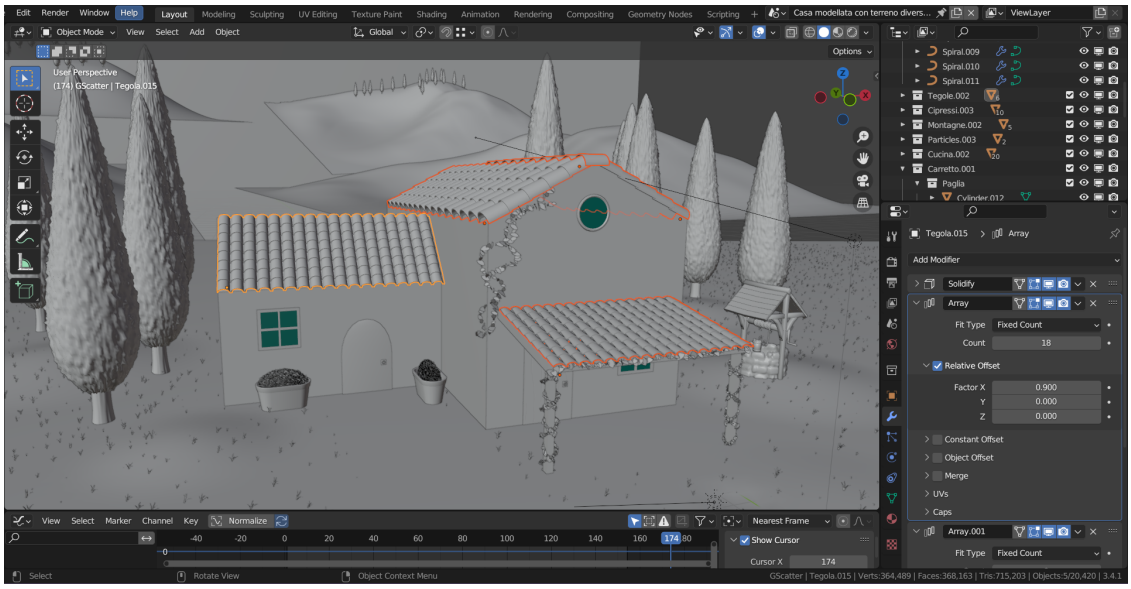

Figura 4.2: Risultato finale delle tegole nella scena 2 con l'*Array modifier*.

4.2.2 Boolean modifier

Questo modificatore crea una nuova mesh partendo da due o più mesh iniziali, utilizzando gli operatori booleani. È necessario definire una mesh target, ovvero l'oggetto che vogliamo utilizzare per applicare gli operatori sulla mesh alla quale è stato assegnato il modifier. È possibile utilizzare come target un oggetto singolo oppure una collezione di mesh.

Per creare le finestre della casa è stato necessario fare un vero e proprio buco nella mesh del muro, e questo è stato possibile aggiungendo il *Boolean modifier* al muro e utilizzando come target una seconda mesh, in questo caso un cilindro. Il modifier prende la parte di volume occupata dal cilindro e la cancella dal volume del muro, creando così il buco.

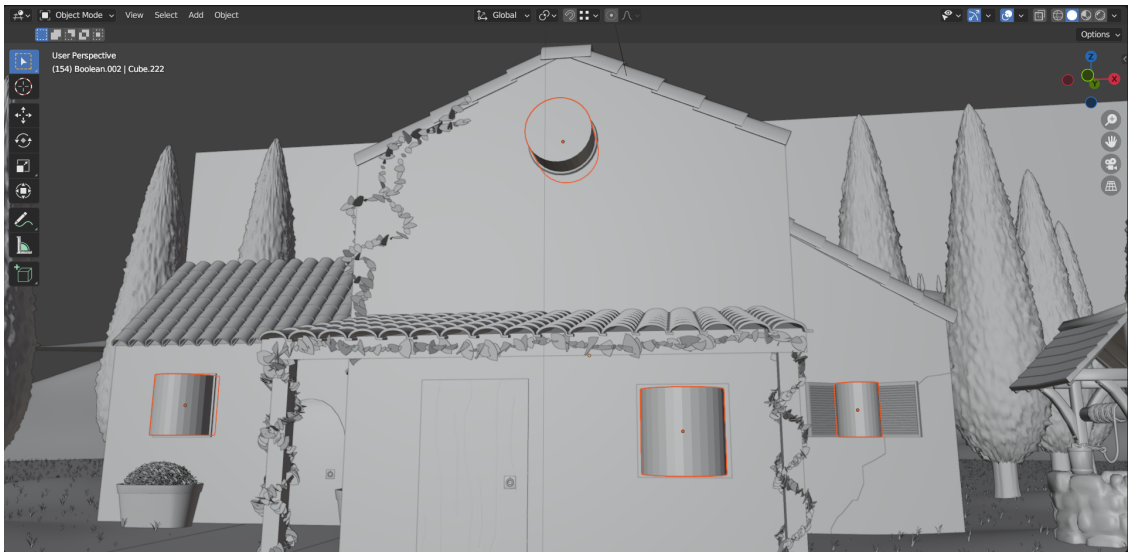

Figura 4.3: Mesh utilizzate dal *Boolean modifier* per creare il buco delle finestre.

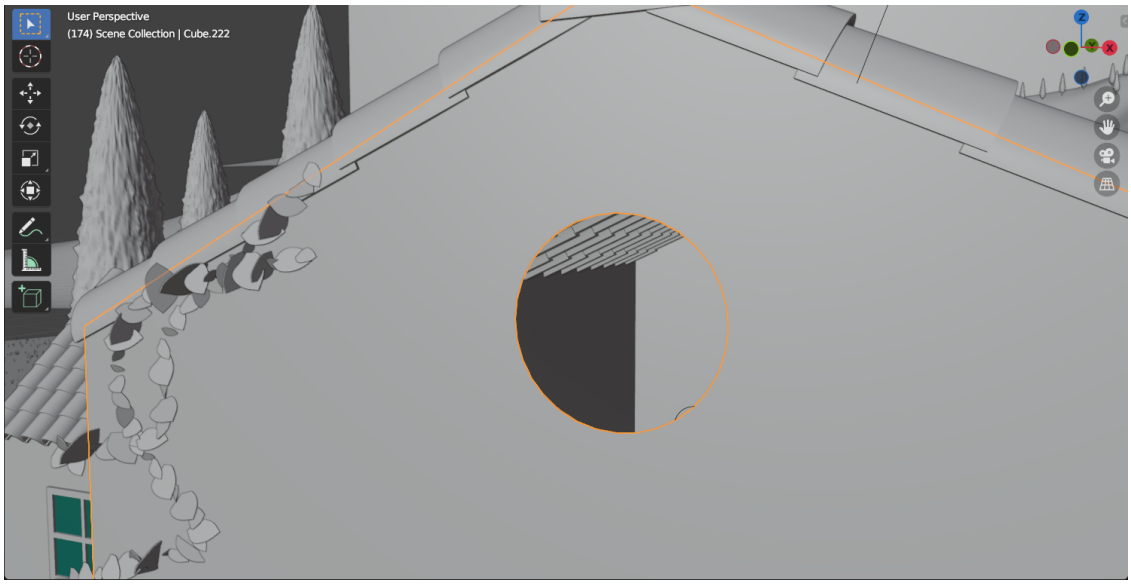

Figura 4.4: Risultato del *Boolean modifier* applicato.

4.2.3 Geometry nodes

I geometry nodes sono stati usati principalmente in due scene: la scena 2 e la scena 9.

Scena 2

Nella scena 2 sono stati utilizzati per realizzare le piante rampicanti che salgono lungo i pilastri della tettoia e sul muro della casa, e per creare le file di alberi presenti sulle colline nello sfondo.

Per la creazione delle piante rampicanti presenti sui pilastri è stata utilizzata come input una curva a spirale, aggiunta aumentando l'altezza e il numero di giri, alla quale successivamente è stato applicato il geometry nodes che comprende i nodi presenti nella figura 4.5. L'oggetto utilizzato come *instance* è una foglia che è stata inserita nel nodo *Instance on Points*, nodo che applica l'oggetto in ingresso su ogni punto presente sulla curva. Il numero di punti sulla curva è un valore che viene recuperato dal nodo di input principale, che si riferisce sempre alla curva di partenza. Sono stati poi aggiunti due nodi di *Random value*, uno sulla rotazione e uno sulla scala, in modo tale che ogni foglia avesse una rotazione e una scala diverse dall'altra, così da creare un effetto più realistico. Successivamente è stata assegnata una texture alla foglia originale e, di conseguenza, a tutte le foglie presenti sul geometry nodes.

Per la realizzazione invece delle piante sulla parete della casa è stata utilizzata come input una curva *Nurbs*[1], in modo tale da spostare i singoli punti e modellare la curva secondo le forme del muro della casa. Il geometry node applicato è pressoché lo stesso del precedente, sono stati modificati solamente i parametri dei vari nodi.

[1]Non-uniform Rational Basis Spline: sono delle curve a controllo locale, è possibile muovere un singolo punto della curva senza modificare la geometria circostante.

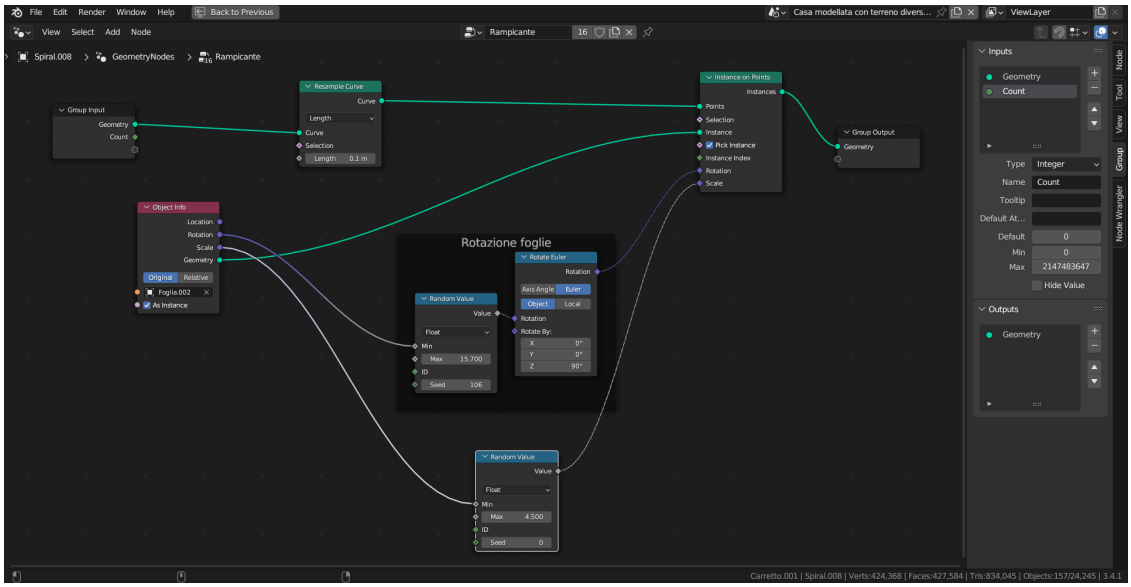

Figura 4.5: Geometry nodes del rampicante.

Gli alberi sulle colline nello sfondo sono stati anch'essi realizzati con una *Nurbs curve* come input, alla quale è stato applicato il geometry nodes la cui struttura di nodi è molto più semplice di quelle viste in precedenza per le piante rampicanti.

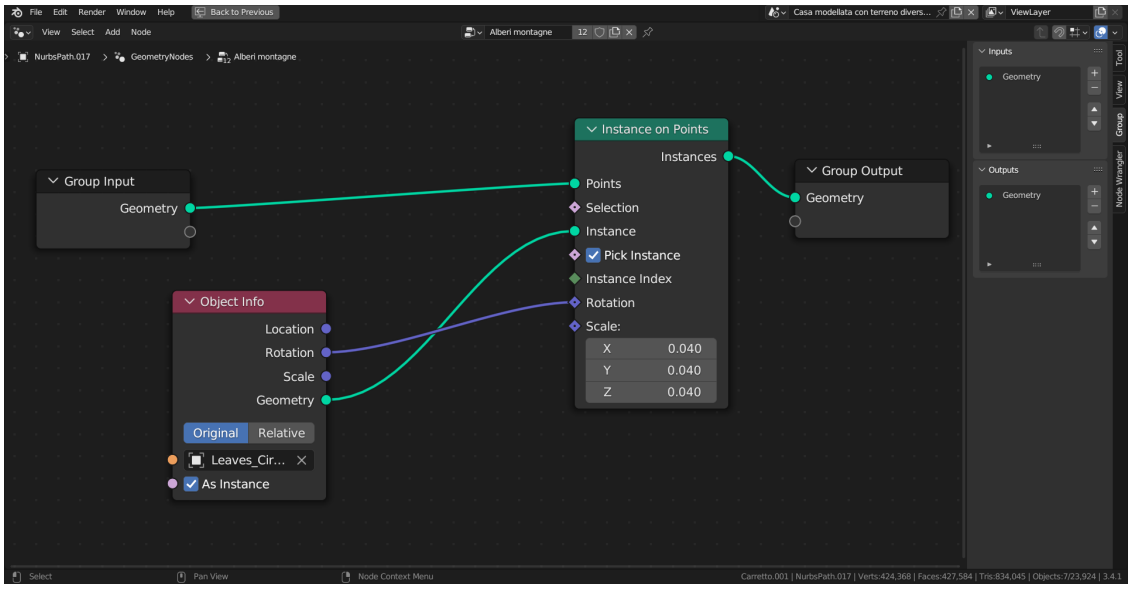

Figura 4.6: Geometry nodes degli alberi sulle colline nello sfondo.

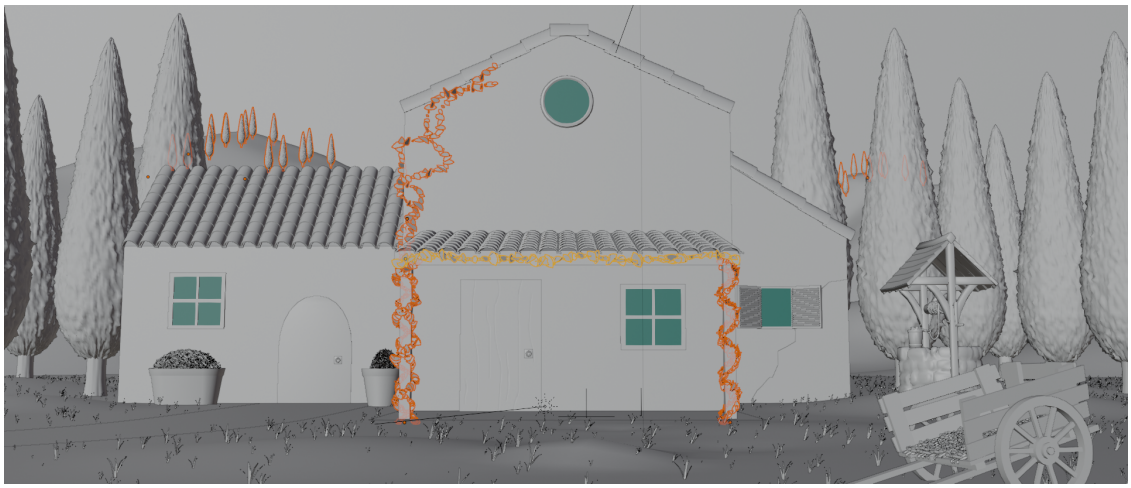

Figura 4.7: Risultato finale dei geometry nodes utilizzati nella scena 2.

Scena 9

Nella scena 9 i geometry nodes sono stati utilizzati per realizzare la staccionata e la neve posata su di essa. È stato utilizzato un solo geometry node sia per la staccionata che per la neve in quanto quest'ultima è relazionata alla staccionata e ne segue le forme e la posizione.

Per la realizzazione della staccionata si parte da una curva *Nurbs* alla quale è stato aggiunto il modificatore *Shinkwrap*, utile per fare in modo che la curva aderisca al terreno su cui è posizionata e ne segua la geometria. Questo è possibile modificando il parametro *Wrap method* su "*Nearest Vertex*" e mettendo come target proprio il terreno su cui poggerà la staccionata.

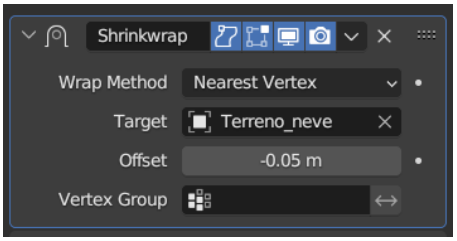

Figura 4.8: *Shrinkwrap modifier.*

Successivamente, sulla curva viene applicato il geometry nodes, che prende come oggetto di instance il modello di una staccionata. Con questi due *modifier* applicati

alla curva, ogni qualvolta viene spostato un punto della curva, o un punto sul terreno target, si sposta coerentemente anche la staccionata collegata ad essa. Possiamo notare però che la rotazione della staccionata non segue quella del terreno, ma le singole instance mantengono la rotazione della staccionata originale. Per ottenere una rotazione dinamica della staccionata, ovvero che segua la forma del terreno sulla quale è posizionata, è necessario inserire altri tre nodi, il nodo *Sample curve*, il nodo *Align Euler to Vector* e il nodo *Spline Parameter*.

Il primo prende alcune informazioni dal nodo di input, come la posizione e le normali, e ne restituisce altrettanti. Prendendo il valore *Tangent rotation* e collegandolo alla rotazione del nodo *Instance on point*, noteremo che questo non lavora in maniera corretta perché l'informazione della tangente circa l'allineamento dei punti della curva non viene tradotta efficacemente dal vettore rotazione del nodo *Instance on point*, quindi abbiamo bisogno di un altro nodo che traduca meglio queste informazioni, ovvero il nodo *Align Euler to Vector*. Questo fondamentalmente definisce come ruotare l'oggetto sui 3 assi, ruotandolo prima sull'asse z, poi sull'asse y e poi sull'asse x, e permette di connettere il vettore tangente che proviene dalla sorgente con la reale rotazione dell'oggetto posizionato sulla curva. Per migliorare ulteriormente il risultato ottenuto, utilizziamo il terzo nodo, *Spline parameter*, che prende l'informazione della rotazione della curva dal modificatore *Shrinkwrap* applicato in precedenza e la passa come informazione al nodo *Sample curve*. Adesso la rotazione della staccionata segue perfettamente quella del terreno su cui è posizionata. Il nodo *Resample curve* invece serve solamente ad aumentare agilmente il numero di punti presenti sulla curva, in modo da gestire il numero di staccionate in maniera adeguata alle esigenze.

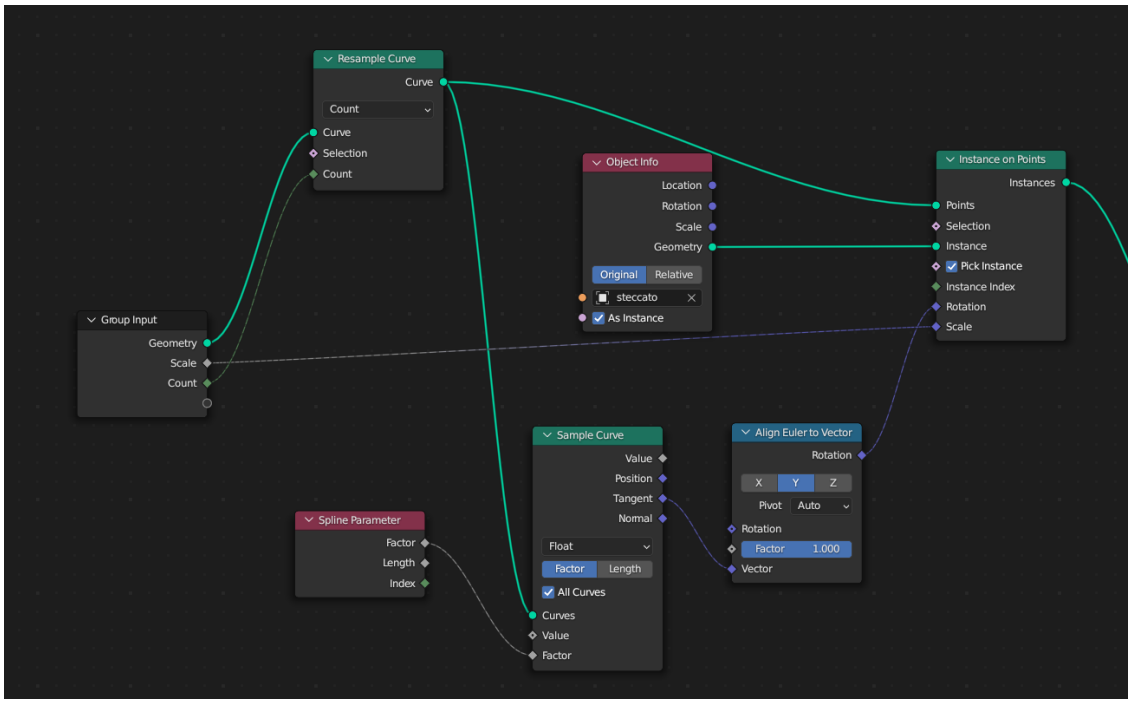

Figura 4.9: Geometry nodes dello steccato nella scena 9.

Per la realizzazione della neve sulla staccionata, è bastato duplicare alcuni nodi visti in precedenza ed applicare le giuste trasformazioni, ad esempio la rotazione e la scala per ottenere il risultato desiderato.

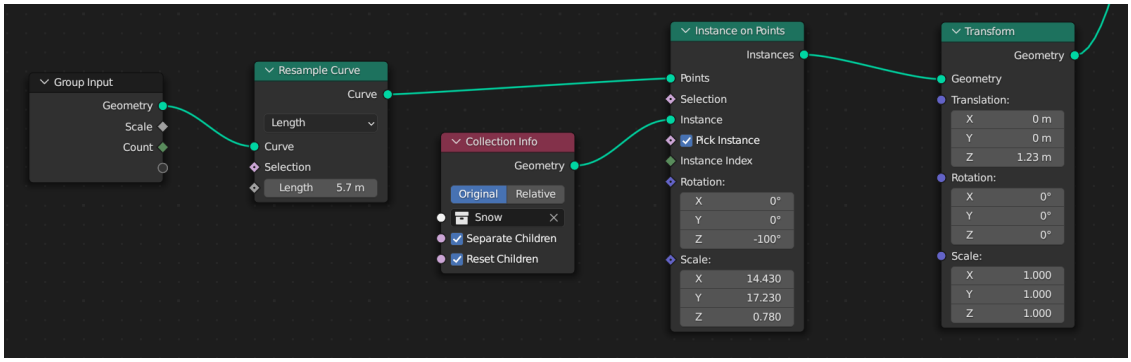

Figura 4.10: Geometry nodes della neve sullo steccato nella scena 9.

Dopo di che è stato sufficiente unire i due gruppi di nodi verso un unico output utilizzando il nodo *Join Geometry*.

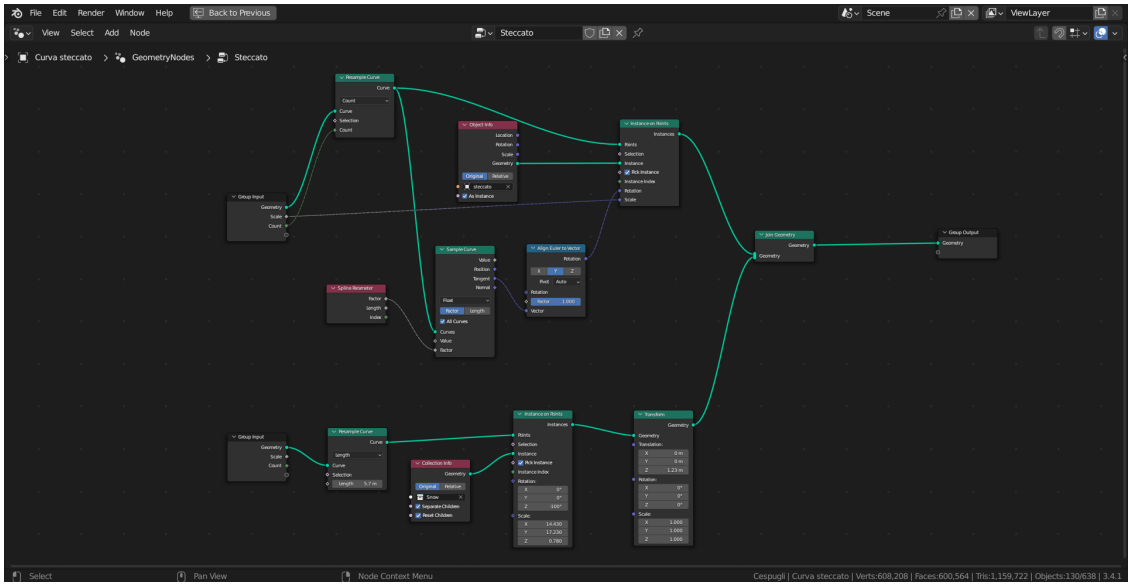

Figura 4.11: Geometry nodes completo dello steccato con la neve nella scena 9.

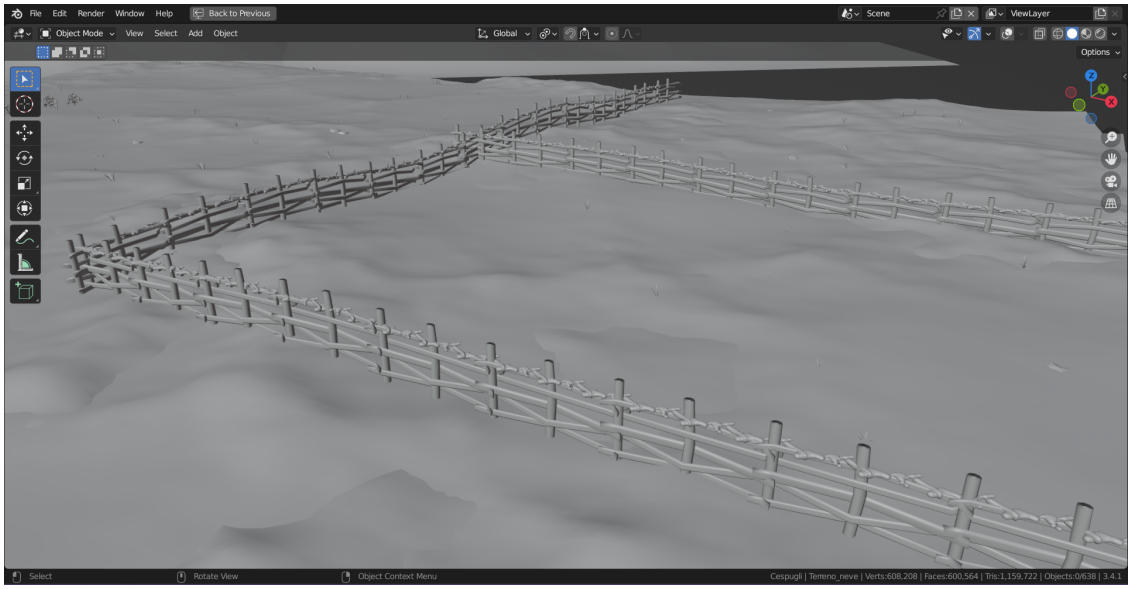

Figura 4.12: Risultato finale dello steccato con la neve nella scena 9.

4.2.4 GScatter

La scena 2, in cui troviamo una cascina toscana sopra una collina, presenta una serie di elementi e texture che appesantivano molto il progetto. Per ottimizzare quindi il lavoro e accorciare i tempi di render, si è cercato di ridurre o eliminare alcuni elementi ove possibile.
Il terreno sottostante la casa presentava un *particle system* con dell'erbetta e delle rocce che si estendeva e occupava tutto il piano, anche dove la camera effettivamente non renderizzava. Allora si è pensato di eliminare il *particle system* e di introdurre l'erbetta attraverso l'utilizzo dell'add-on[2] *G-scatter*. Questo è un add-on di terze parti che consente di creare scene con distribuzioni casuali di oggetti in modo veloce.

L'add-on offre molte opzioni per personalizzare la disposizione degli oggetti, la loro posizione, rotazione e scala, nonché per creare una variazione di colori e forme. L'add-on offre di base una libreria da cui attingere per prendere alcuni modelli di erba o piccole rocce, ma è possibile utilizzare un modello personalizzato come modello sorgente. Una volta selezionati la superficie che fungerà da emitter e il modello da utilizzare, è possibile gestire valori come la densità di distribuzione, la scala o la rotazione. È inoltre presente un tab di ottimizzazione, in cui è possibile selezionare la camera della scena corrente e utilizzarla come "limite" per l'emissione delle particelle, in modo tale da non crearle in punti che non saranno poi visibili nel render finale. È necessario inserire alcune informazioni

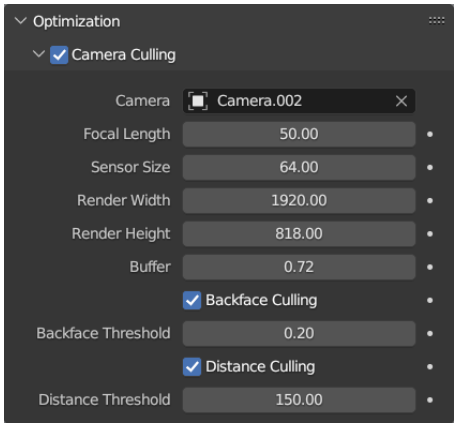

Figura 4.13: Tab di ottimizzazione dell'add-on GScatter.

[2]Componente di un software che aggiunge delle funzionalità a un programma. Può essere direttamente integrato al software oppure di terze parti.

della camera, come la lunghezza focale[3] e la dimensione del sensore[4] della camera, e il formato della scena, che nel caso del cortometraggio è di 1920x818 pixel.

Attraverso l'aggiunta di un *Effect layer*, in questo caso *Weight Mask*, è possibile selezionare delle porzioni di piano in cui le particelle non devono essere generate. Questo è realizzabile tramite la *Weight paint*, una funzionalità di Blender che permette di assegnare dei "pesi" ai vertici di un oggetto 3D. Questi pesi indicano l'influenza che un determinato oggetto ha su un altro, in molti casi è utilizzata per realizzare il *rigging* di un personaggio, indicando, ad esempio, quali vertici di un braccio devono muoversi per rendere il movimento più realistico possibile. In questo caso viene utilizzata per indicare in quali zone non creare le particelle di erbetta o in quali crearne di più o di meno.
I pesi vengono assegnati ai vertici della mesh utilizzando lo strumento come un pennello, avente un determinato raggio, intensità e flusso, e il valore assegnato ad ogni singolo vertice spazia in un intervallo tra 0 e 1, in cui 0 significa nessun peso mentre 1 significa peso massimo. Per una visione più facile ed intuitiva, i pesi sono rappresentati tramite dei colori, in cui il valore 0 viene rappresentato con il colore blu, mentre il valore 1 viene rappresentato con il colore rosso, e gli altri valori compresi sono rappresentati tramite una scala sfumata di colori, in modo da creare una transizione più fluida tra le parti pesate e quelle non pesate.

[3]La lunghezza focale di un obiettivo è la distanza ottica (espressa generalmente in mm) tra il punto in cui si concentra la luce all'interno dell'obiettivo e il sensore della fotocamera.

[4]Il sensore è un dispositivo elettronico che converte un'immagine ottica in un segnale elettrico.

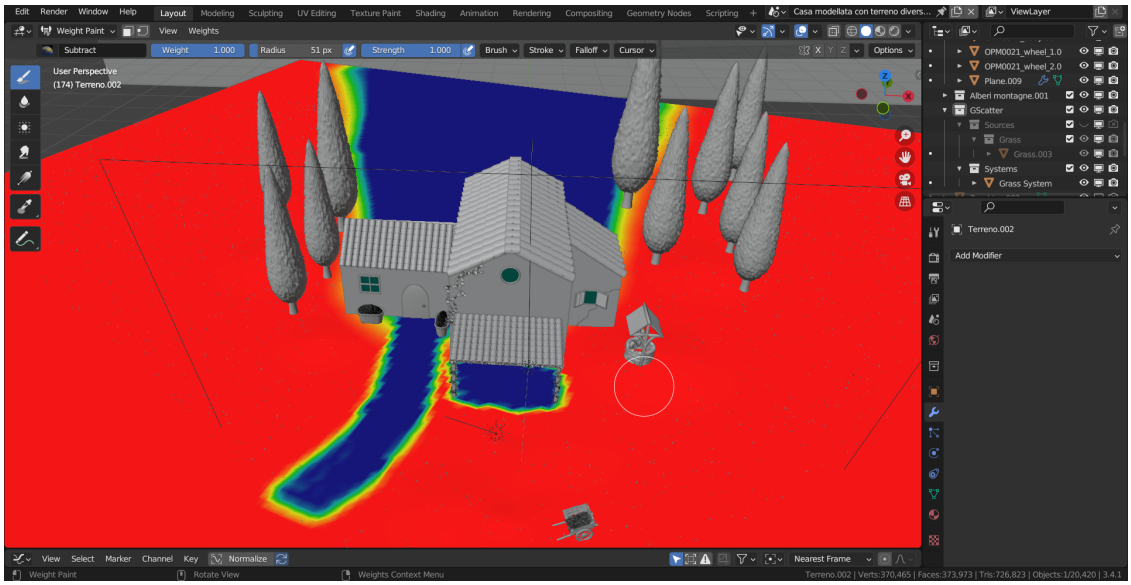

Figura 4.14: Weight paint dell'add-on GScatter.

I punti scelti in cui assegnare peso 0 ai vertici sono punti dove è presente altra geometria sulla quale non deve essere presente l'erbetta, come il pavimento sotto il porticato e la stradina davanti la porta, e la zona retrostante la casa che non viene mai inquadrata dalla camera, e nella quale quindi non era necessario creare le particelle.

Inoltre, è stato aggiunto un movimento alle particelle applicando un livello di *Wind* nel tab di Rotazione del *GScatter*, che agisce esattamente come un *force field*, simulando anche un effetto di *Noise* applicato alla forza in modo tale che il movimento risulti più naturale e non costante.

4.2.5 Real Snow

Per simulare i vari accumuli di neve presenti nelle diverse scene è stato utilizzato l'add-on *Real snow*, uno strumento il cui principio alla base è quello di aggiungere un

particle system di *metaball*[5] sulla superficie dell'oggetto selezionato e di trasformarlo successivamente in mesh. Infatti, una volta aggiunta la neve tramite l'add-on, è possibile modificare la mesh in *Edit mode*.

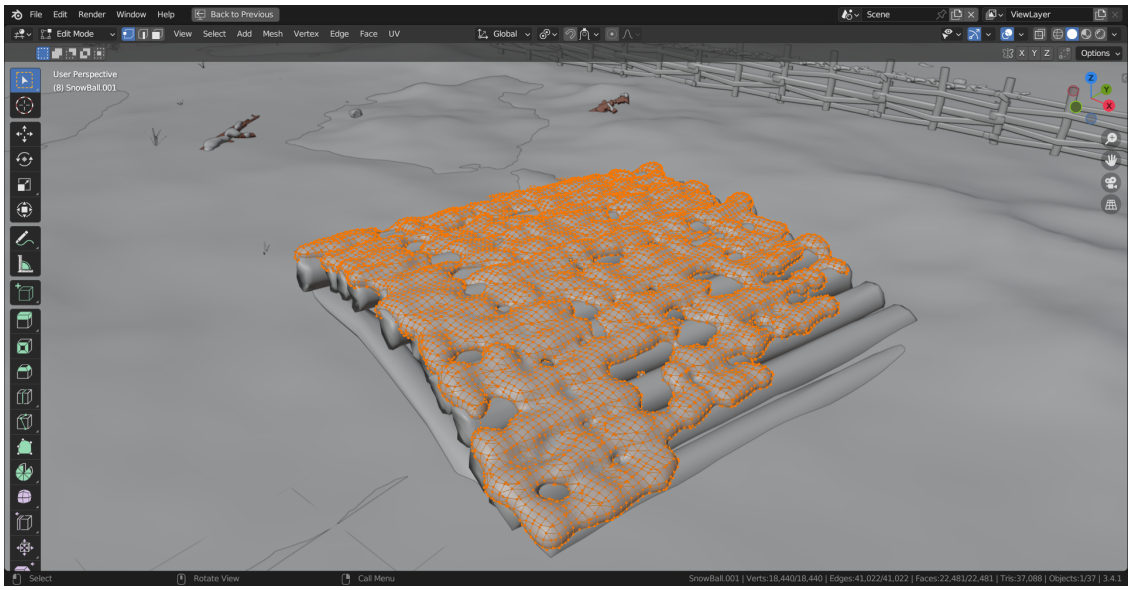

Figura 4.15: Mesh della neve aggiunta dall'add-on, editabile.

L'interfaccia dell'add-on presenta pochi elementi di personalizzazione, solamente la quantità di copertura della superficie di interesse e l'altezza della neve, ma un elemento particolarmente interessante è la checkbox *Selected Faces*. Questo permette di aggiungere la neve solo sulle facce della mesh precedentemente selezionate in *Edit mode*, in modo da permettere una maggiore

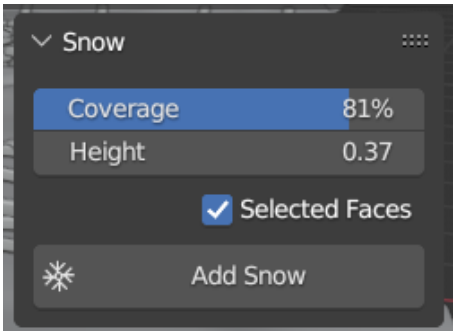

Figura 4.16: Tab dell'add-on Real snow.

[5]Una metaball, in computer grafica, è un oggetto che non è definito da vertici o punti controllo, bensì da espressioni matematiche pure. In altre parole sono superfici implicite, che vengono calcolate e messe in relazione tra loro da operazioni logiche additive o sottrattive.

personalizzazione e un maggiore realismo.

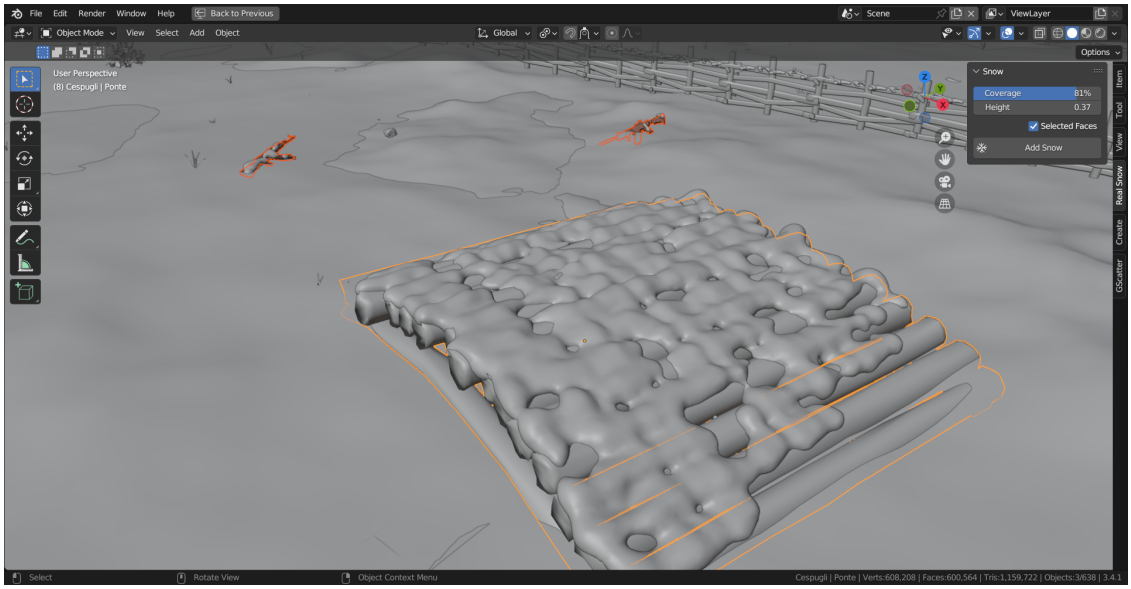

Figura 4.17: Oggetti su cui è stato utilizzato l'add-on *Real snow* nella scena 9.

4.3 Texturing

I modelli 3D creati nella fase di modellazione vengono mostrati tendenzialmente con un materiale grigio privo di profondità. In questa fase vengono quindi create tutte le texture, i materiali, i colori da applicare ai singoli oggetti per dare loro un aspetto quanto più realistico possibile. La fase di *texuring* è talvolta l'arte di "vestire" i modelli 3D con delle immagini 2D, immagini che possono anche essere create su software esterni e successivamente importate.

Generalmente il *3D texturing* comprende tre proprietà principali per ogni superficie: il **materiale**, in modo da mostrare allo spettatore di cosa sia realmente fatto l'oggetto, gli **effetti di luce**, ovvero la gestione della riflessione, della rifrazione e così via, e infine i **dettagli terziari**, in quanto, se si dovessero modellare tutti i

singoli dettagli di una mesh, il modello sarebbe pieno di vertici, rendendolo pesante e difficile da renderizzare in termini di tempo. Quindi, è possibile creare molti dettagli attraverso le texture, come venature del legno, cicatrici e altro.

Il *workflow* per il *texturing* inizia con la fase di *unwrap*, nient'altro che il "taglio" della mesh 3D per ottenerne una versione 2D, in questo modo si avrà un'applicazione della texture sulla mesh molto più precisa.

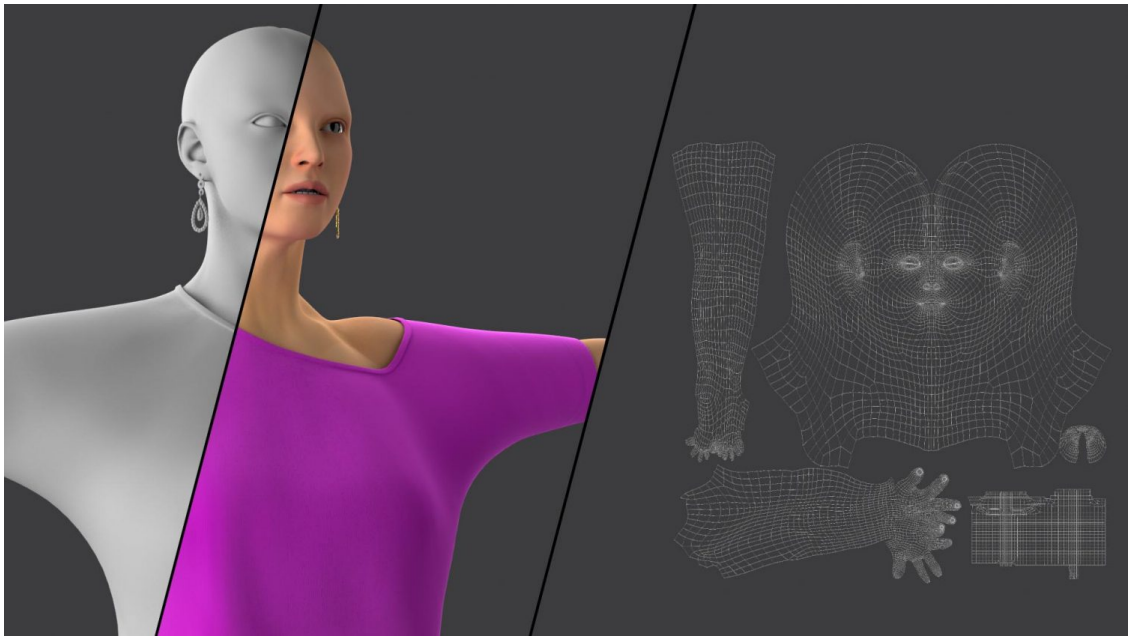

Figura 4.18: Esempio di *unwrap* di una mesh.

Dopodiché si passa alla fase di *texture painting e shading*, in cui la texture è, come abbiamo già detto, un'immagine 2D, mentre lo *shading* è un insieme di funzioni che determinano come la luce agisce sulla texture.
Il processo di definizione del colore, dei dettagli della superficie e delle proprietà visive è definito *texture mapping*, e tra le texture maps più utilizzate troviamo la *Base color map*, la *Normal map* e la *Roughness map*, ma anche l'*Ambient occlusion map* e la *Displacement map*.

Pavimento della tettoia esterna

Come possiamo notare dal tab *shading* nella figura 4.19, per la creazione della texture del pavimento sono state utilizzate quattro immagini 2D, una per il *base color*, una per la *roughness*, una per la *normal* e una per il *displacement*, e, poiché le immagini della *normal map* e del *displacement* non vengono tradotte nel modo corretto dal nodo principale, sono stati utilizzati i nodi *Normal map* e *Displacement* per ottenere un risultato soddisfacente.

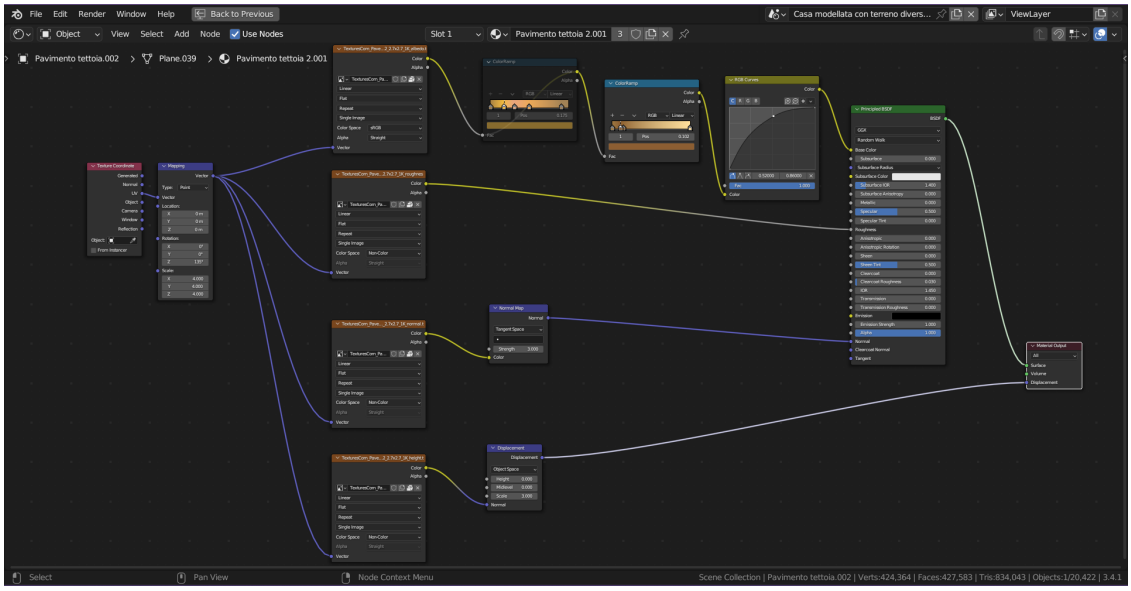

Figura 4.19: Nodi dello *shading* del pavimento della scena 2.

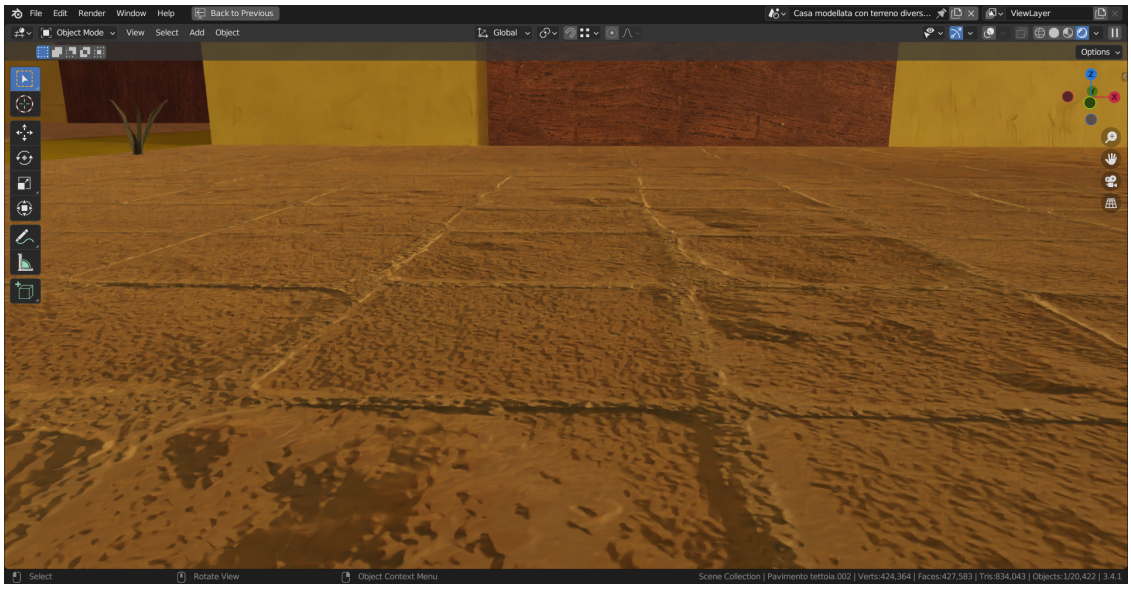

Figura 4.20: Texture del pavimento della scena 2.

Muro della casa

Per dare ancora un maggior senso di realismo alla scena 2, in alcune parti della casa è stato creato un muro rovinato, come se avesse perso l'intonaco, mostrando il mattone sotto.

Per realizzarlo è stata presa la mesh del muro in questione ed è stata suddivisa tramite *Subdivide*, questo passaggio è utile per il successivo *unwrap*, passaggio come abbiamo detto necessario per un'applicazione più precisa della texture. Dopodiché è stata creata un'ulteriore geometria utilizzando lo strumento *knife* (coltello), che effettivamente "taglia" gli *edge* e crea nuovi vertici in modo da poter separare quella parte di muro per creare una nuova mesh.

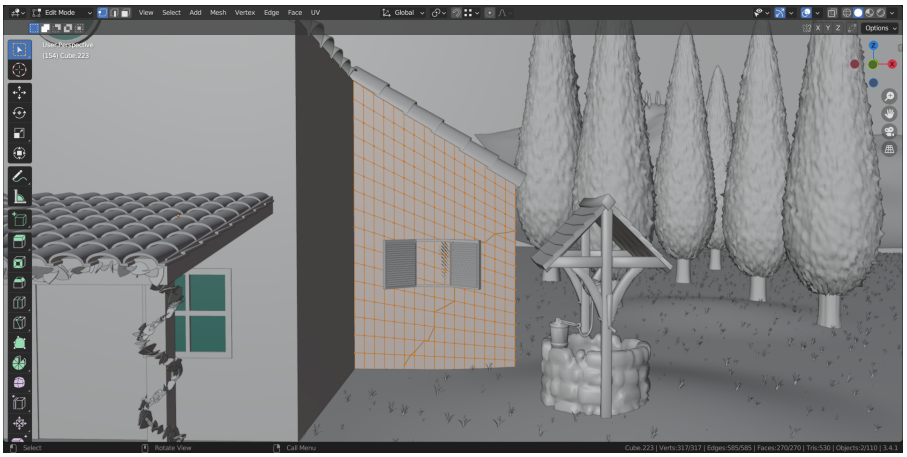

Figura 4.21: Creazione della nuova mesh del muro nella scena 2.

Successivamente è stata applicata la texture dei mattoni, ma poiché il colore della texture non era adeguato alla scena e presentava dei colori che non erano in palette, è stata applicata una *Color ramp*, un nodo che prende in input il colore originale della texture e lo mappa tra 0 e 1, portandolo quindi in bianco e nero. In questo modo è possibile scegliere degli altri colori a piacimento per rendere la texture più coerente con il resto della scena.

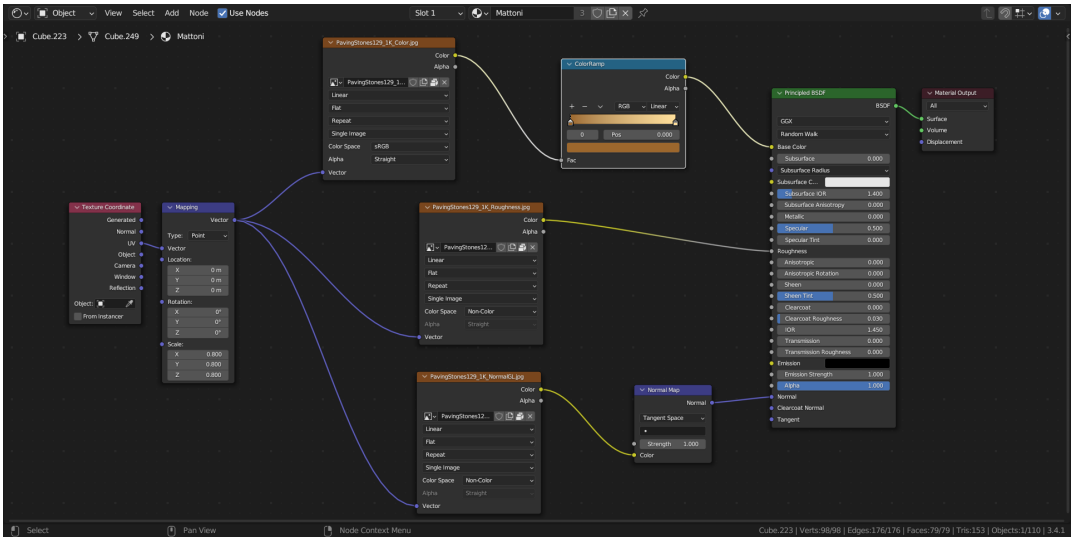

Figura 4.22: Nodi dello *shading* del muro della casa nella scena 2.

Anche in questo caso è stato possibile dare profondità alla geometria pur non dovendo modellare i singoli dettagli dei mattoni, grazie alla *Normal map* e all'azione delle luci sulla texture.

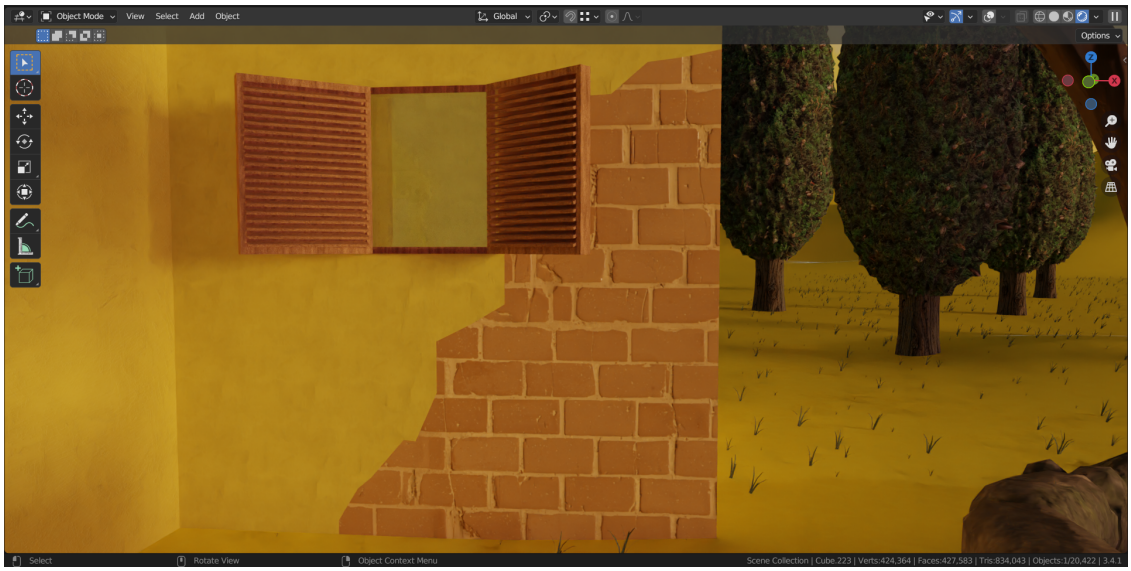

Figura 4.23: Risultato del muro della casa nella scena 2.

Tegole

Per il materiale delle tegole del tetto è stata utilizzata una struttura di nodi abbastanza semplice in cui vengono utilizzati due *Principled BSDF node* ai quali sono stati assegnati due colori leggermente diversi in modo da poter conferire una sfumatura alla tegola. Questa sfumatura è possibile crearla assegnando al materiale una *Noise texture*, che consiste in una texture rumorosa come mostrato in figura 4.24.

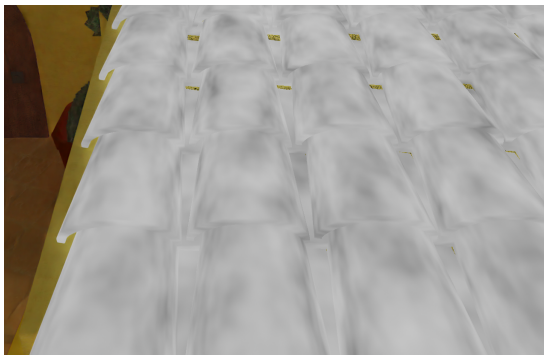

Figura 4.24: *Noise texture*.

Questa texture passa poi dentro una *Color ramp* che definisce in quali punti le sfumature devono essere più scure e in quali più chiare, e, una volta connessi insieme i due *Principled BSDF node* e la *Noise texture* all'interno del nodo *Mix shader*, le sfumature prenderanno il colore assegnato ai due nodi principali.

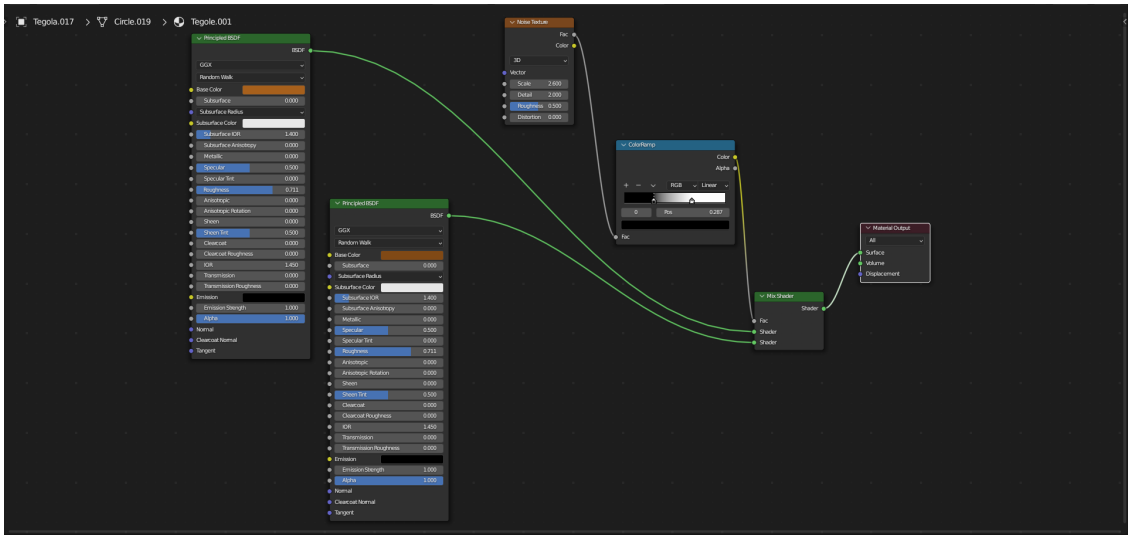

Figura 4.25: Nodi del materiale delle tegole.

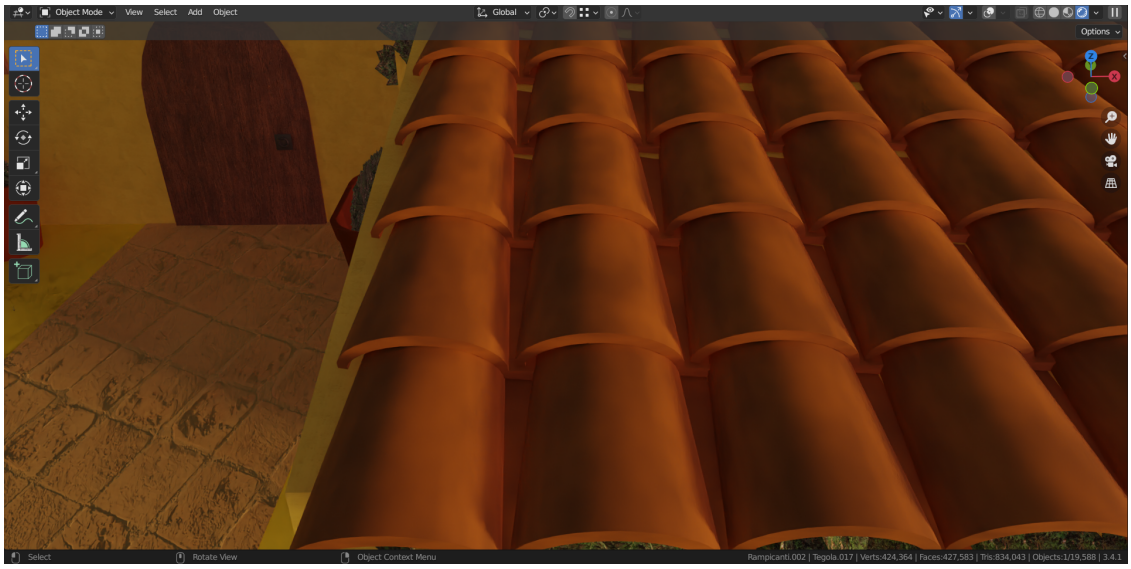

Figura 4.26: Risultato delle tegole nella scena 2.

Esplosione

A differenza delle texture appena descritte, la texture che è stata applicata all'esplosione ha una natura molto diversa, in quanto viene applicata ad un *volume*, cioè una cosa "non tangibile", quindi non ad una mesh.

Il *Volumetric shading* è diverso dallo *shading* degli altri oggetti 3D, perché con questi ultimi, è possibile applicare una texture basandosi sulle *Uv maps*, adattando la texture alle cordinate della mappa, ma per quanto riguarda lo shader di un'esplosione, il comportamento dinamico della stessa rende difficoltosa l'applicazione di una texture perché questa dovrebbe seguire il movimento della simulazione.

Al posto delle coordinate della *Uv map*, nel *Volumetric shading* sono presenti degli altri attributi che sono generati in concomitanza con la simulazione. È importante sottolineare che questi attributi fanno parte del dominio dell'esplosione e non dell'emitter, e sono visibili tra le impostazioni del *Viewport display* del dominio stesso. Tra gli attributi troviamo la *temperature*, ovvero la temperatura alla quale si innesca la combustione, che genererà calore (*heat*) che, se la reazione procede a velocità molto elevata, innescherà il *fire/flames*, che si tradurrà in *smoke* (fumo), anche chiamato *density*. Tutti questi attributi contibuiranno a dare una texture all'esplosione finale.

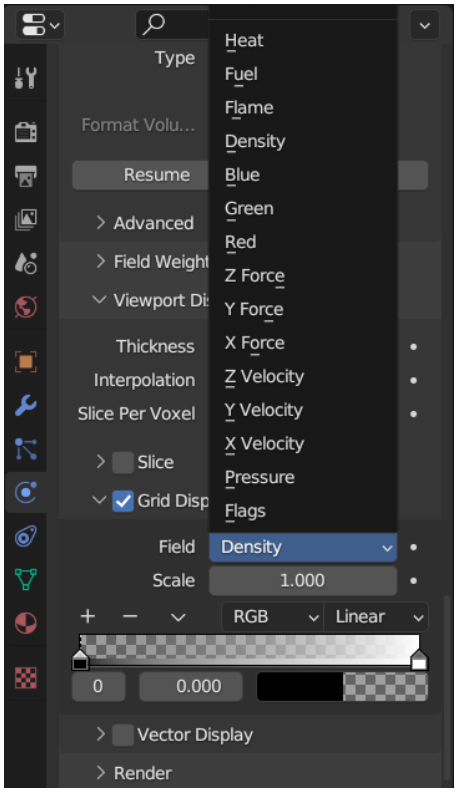

Figura 4.27: Attributi dell'esplosione.

Quando creiamo un nuovo materiale in Blender, di default viene creato un nodo di *Principled BSDF* e un nodo di *Material output*, ma queste impostazioni creano un

materiale sulla superficie del dominio che contiene l'esplosione, e non sull'esplosione vera e proria. Di conseguenza è necessario cambiare il nodo da *Principled BSDF* a *Principled volume* e collegarlo al nodo di output sul parametro del volume (e non sulla superficie).

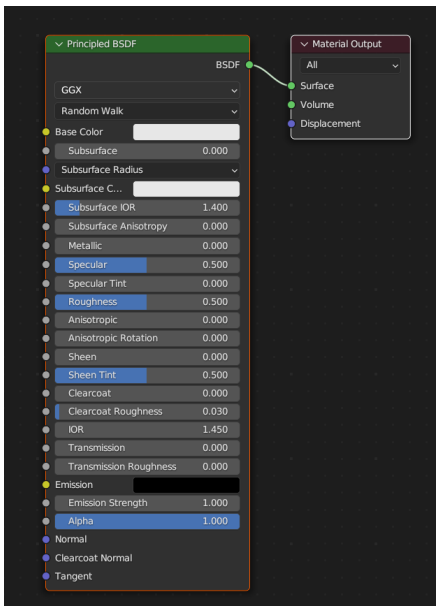

Figura 4.28: Nodo *Principled BSDF*.

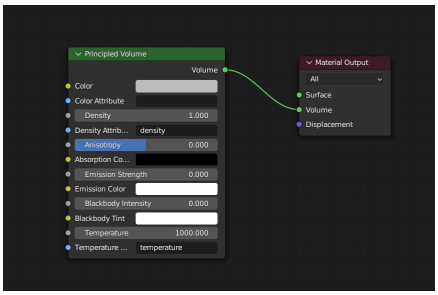

Figura 4.29: Nodo *Principled volume*.

Come possiamo vedere, sul nodo *Principled volume* sono presenti alcuni degli attributi (ad esempio *density* e *temperature*), ma è altrettanto possibile inserire uno o più attributi dall'esterno tramite il nodo *Attribute*. Se da un lato è possibile lasciare il campo *Color attribute* vuoto, dall'altro non è possibile farlo con il campo *Density attribute*, perché in quel caso non attribuirebbe più la texture all'esplosione bensì all'intero volume del dominio. Il parametro *Emission strength* è uno tra i più importanti perché definisce quanta luce emette l'esplosione, nella maggior parte dei casi è consigliabile lavorare con delle impostazioni esterne e poi applicare l'output sul valore, in modo da essere più facile da gestire.

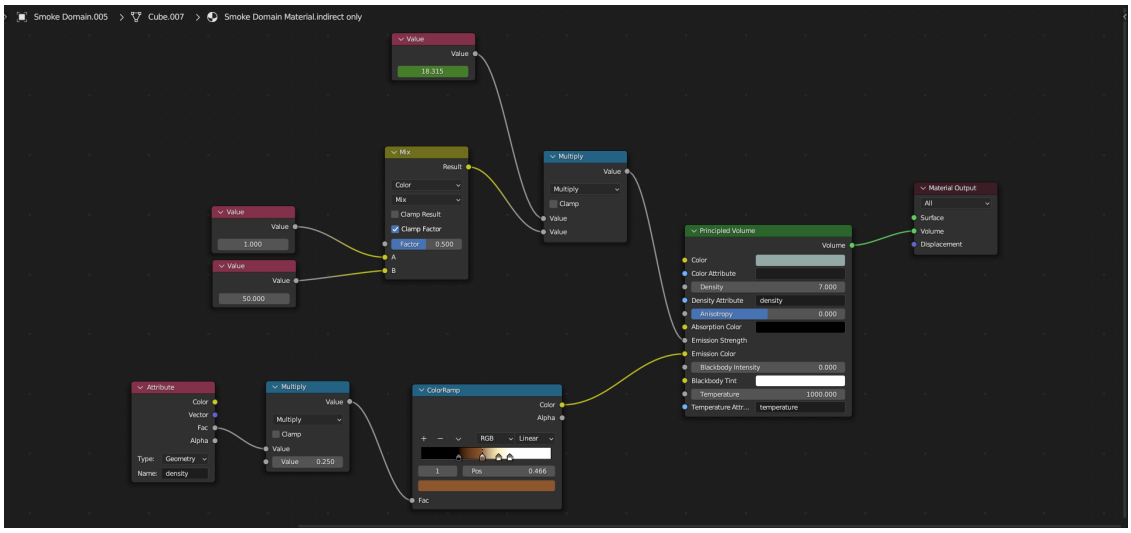

Figura 4.30: Nodi del materiale dell'esplosione.

Per creare il materiale dell'esplosione sono stati utilizzati due "gruppi" di nodi come vediamo nella figura 4.30, uno che agisce sull'*Emission strength* e l'altro che agisce sull'*Emission color*. Per l'*Emission strenght* abbiamo un paio di nodi *Value* che vengono mescolati tramite il nodo *Mix* il cui risultato viene moltiplicato tramite il nodo *Multiply* con un altro valore proveniente da un altro nodo, che però questa volta è stato animato, come si può notare dal colore verde sulla variabile. Animare questo valore aiuta a gestire il tempo in cui il colore che simula l'esplosione vera e propria debba essere presente nella scena, infatti, dopo qualche secondo, questo scompare lasciando solo il fumo di colore bluastro.

Sotto invece troviamo la serie di nodi che attribuisce il colore vero e proprio alla combustione, gestito tramite una *Color ramp*. Il colore generale dell'esplosione e soprattutto del fumo è, come abbiamo detto, un bluastro, perché l'esplosione si trova all'interno di una scena di ricordi di guerra, scene che sono state realizzate con una palette di colori tendenti al blu/grigio.

4.4 Rigging e Animazione

Il *rigging* è il processo tramite cui si definisce il range di azione di un modello 3D, utilizzando una serie di "ossa" interconnesse collegate ai vertici della mesh. Una volta che un modello è stato "riggato" è possibile animarlo, muovendo le ossa secondo le posizioni desiderate e inserendo i keyframe nei punti di interesse, così da ottenere il movimento della parte di mesh collegata all'osso in questione.

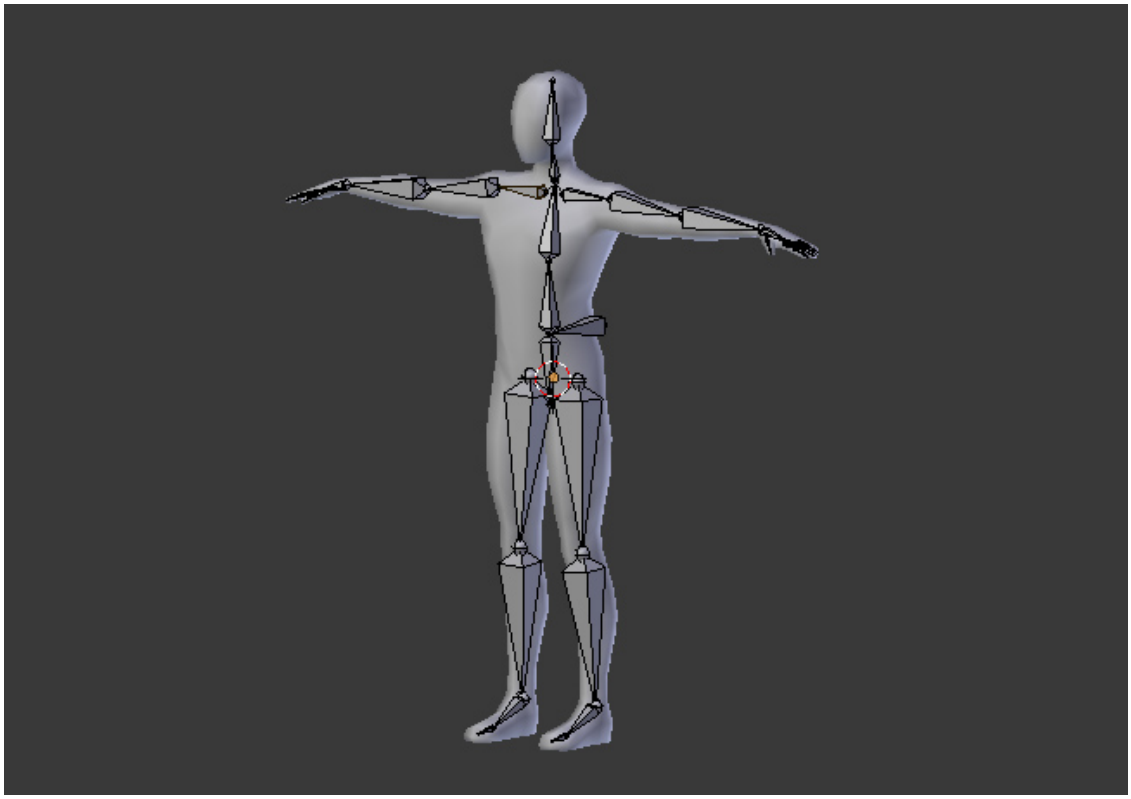

Figura 4.31: Esempio di un modello 3D "riggato".

Per quanto riguarda *En la tierra de los sueños perdidos*, è stato deciso di creare le animazioni dei personaggi utilizzando tecniche 2D, quindi non è stato necessario il processo di rigging e animazione in 3D. I movimenti presenti nelle scene sono principalmente dovuti a simulazioni fisiche, ad eccezione dei movimenti di camera e di qualche altro oggetto.

È stata animata una rivista le cui pagine vengono sfogliate da un personaggio che invece è stato animato in 2D, e il movimento è stato creato aggiungendo alle singole pagine il *Simple deform modifier*, impostato su *Bend*, che piega le pagine sull'asse Z secondo l'angolo scelto. È stato quindi applicato un *keyframe* sul valore dell'angolo ed è stata poi ruotata la pagina, applicando un altro *keyframe* sulla rotazione.

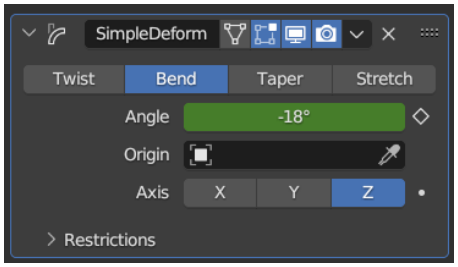

Figura 4.32: *SimpleDeform modifier*.

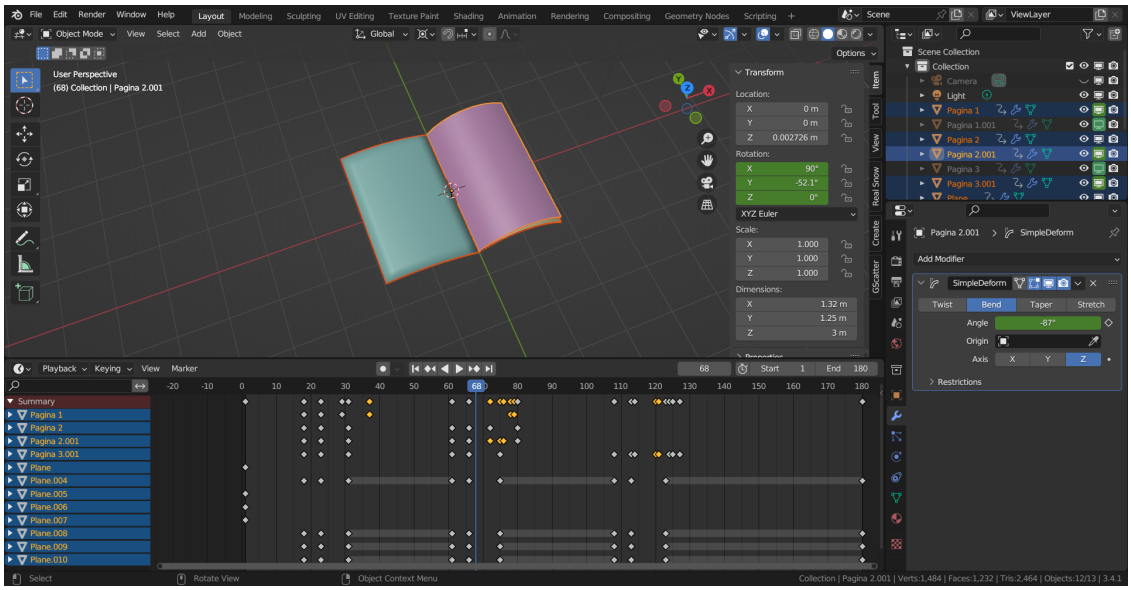

Figura 4.33: *Timeline* con i *keyframe* dell'animazione della rivista.

I *keyframe* che vengono aggiunti all'animazione in Blender hanno di default un'interpolazione di tipo *Bezier*, ovvero le curve di interpolazione presentano una fase di accelerazione e decelerazione, chiamate rispettivamente *Ease in* e *Ease out*, in prossimità dei *keyframe*, a differenza di un'interpolazione lineare che invece ha una curva costante tra due *keyframe*: questo rende l'animazione più naturale e meno "scattosa".

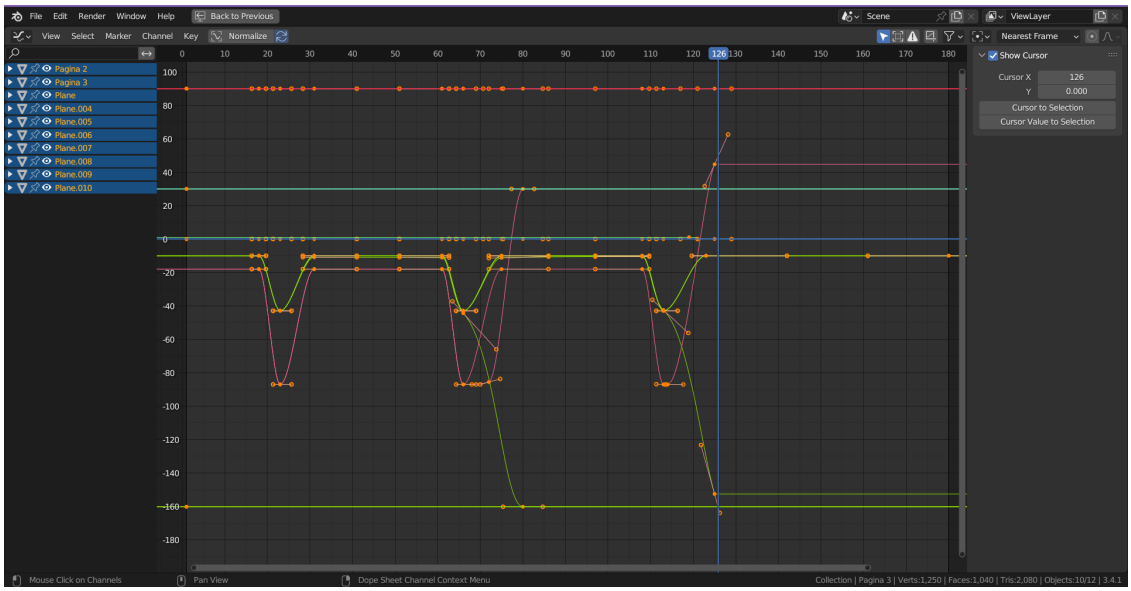

Figura 4.34: *Graph editor* dell'animazione della rivista.

Le curve di Bezier sono facilmente riconoscibili anche guardando il grafico perché presentano dei **punti di controllo** tramite cui è possibile regolarle, in modo da dare più o meno accelerazione. Queste punti si chiamano *handle* e definiscono la direzione e la curvatura della curva.

4.5 Effetti visivi

Quando si parla di effetti visivi nel 3D non si parla solo, ad esempio, di esplosioni, ma anche di simulare elementi come capelli, peli, acqua o vestiti, quindi tutti elementi che sarebbe molto complicato animare manualmente. L'utilizzo delle simulazioni 3D nella produzione di cortometraggi animati sta diventando sempre più popolare negli ultimi anni. Le simulazioni 3D permettono agli animatori di creare mondi virtuali realistici e di far interagire personaggi e oggetti in modo più credibile, aumentando la qualità e l'impatto emotivo dell'animazione.

4.5.1 Particle simulation system

I *particle system* lavorano utilizzando delle particelle che interagiscono con alcune forze naturali, come il vento o la gravità, per creare effetti come la pioggia, la neve oppure uno stormo di uccelli che si muove. Per la realizzazione di questi elementi è assolutamente conveniente utilizzare il particle system in quanto sarebbe un lunghissimo processo quello di animare ogni singolo elemento manualmente, perché in molti casi si parla di centinaia di particelle.

Neve

Nell'ambito del cortometraggio, il particle system è stato utilizzato per creare la neve che cade in diverse scene. È stato inizialmente creato un piano e su di esso è stato applicato un particle system, di tipo *emitter*, rendendo quindi la superficie in grado di emettere le particelle, secondo alcuni parametri specifici. Ogni particellare di tipo *emitter* ha un frame di partenza e uno di fine emissione delle particelle e queste hanno un *lifetime* (tempo in cui le particelle sono "vive", quindi visibili nella simulazione), che può essere uguale per tutte oppure reso randomico. È inoltre possibile definire il tipo di particella che viene emessa, cioè quale forma debba avere, selezionando una mesh modellata a parte come *instance object* della simulazione.

Un aspetto estremamente importante di tutti i particle system è la possibilità di fare il *bake* della simulazione, ovvero l'azione di calcolare tutti i dati necessari per la simulazione e memorizzarli in memoria, in modo da averli già pronti ogni volta che viene lanciata l'animazione, senza doverli calcolare real time volta per volta. Questa è tendenzialmente un'operazione molto lunga e va fatta ogni qual volta viene fatto un cambiamento nel sistema particellare, in quanto ogni cambiamento (ad esempio variare il numero di particelle o la quantità di forza che agisce su di esse) può alterare il risultato finale.

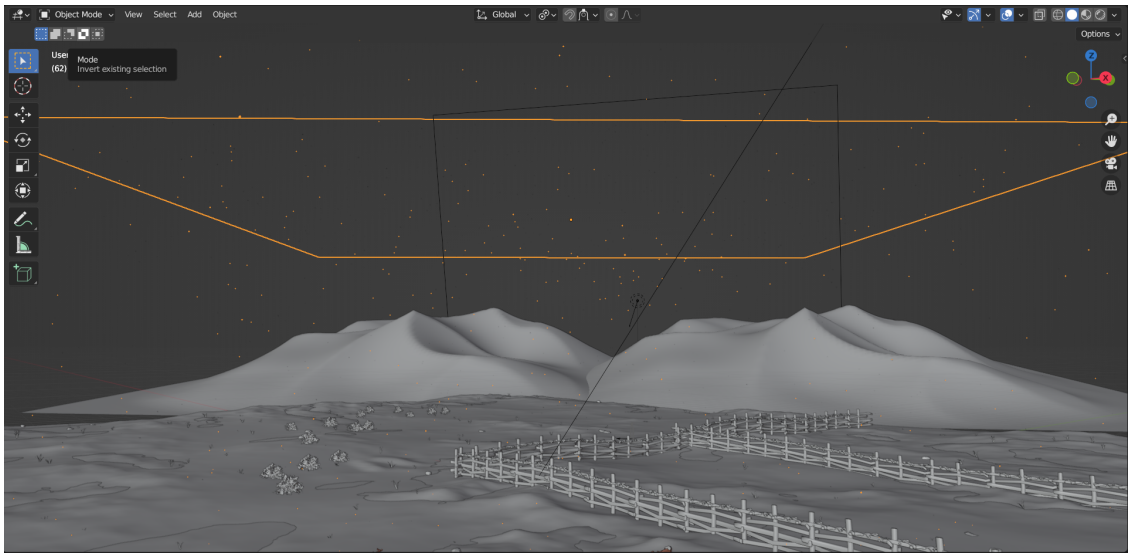

Figura 4.35: *Emitter* del *particle system* della neve nella scena 9

Erbetta e rocce

Per creare invece l'erbetta e le piccole rocce presenti sul terreno è stato usato un *particle system* di tipo *hair* che, a differenza del *particle system* di tipo *emitter*, non rilascia delle particelle, bensì crea una sorta di "capelli" che rimangono ancorati alla superficie. Infatti, in questo caso le particelle non hanno un frame di inizio e uno di fine, tantomeno un tempo di vita, perché non vengono mai distrutte. Questo tipo di *particle system* è molto utile nel caso si vogliano creare dei peli, dei capelli, o in questo caso dell'erbetta.

Una volta selezionata la superficie sulla quale applicare il *particle system*, sono stati modificati i vari parametri per raggiungere il risultato desiderato, come ad esempio la quantità di "capelli" generati, la posizione e altro ancora. Inizialmente era stata utilizzata una *collection* di oggetti come *particle instance*, che comprendeva un ciuffetto d'erba e una roccia. Poiché tutti i tipi di *particle system* reagiscono alle forze naturali, è stato notato che, non solo l'erbetta, ma anche le rocce reagivano al vento presente nella scena, muovendosi. Allora è stato deciso di applicare sullo stesso terreno un altro *particle system*, in modo da utilizzare su uno solo l'*object instance* dell'erbetta, e sull'altro solo quello della roccia, per fare in modo che la roccia non reagisse al vento.

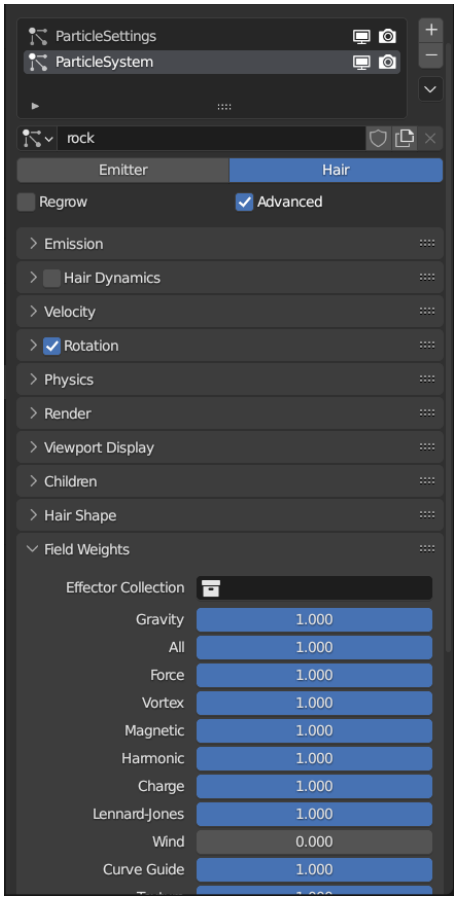

Figura 4.36: *Particle system* applicato al terreno della scena 9.

Come vediamo nella figura 4.36, è stata portata a zero la quantità di forza che agisce sulle particelle all'interno del *particle system* delle rocce.

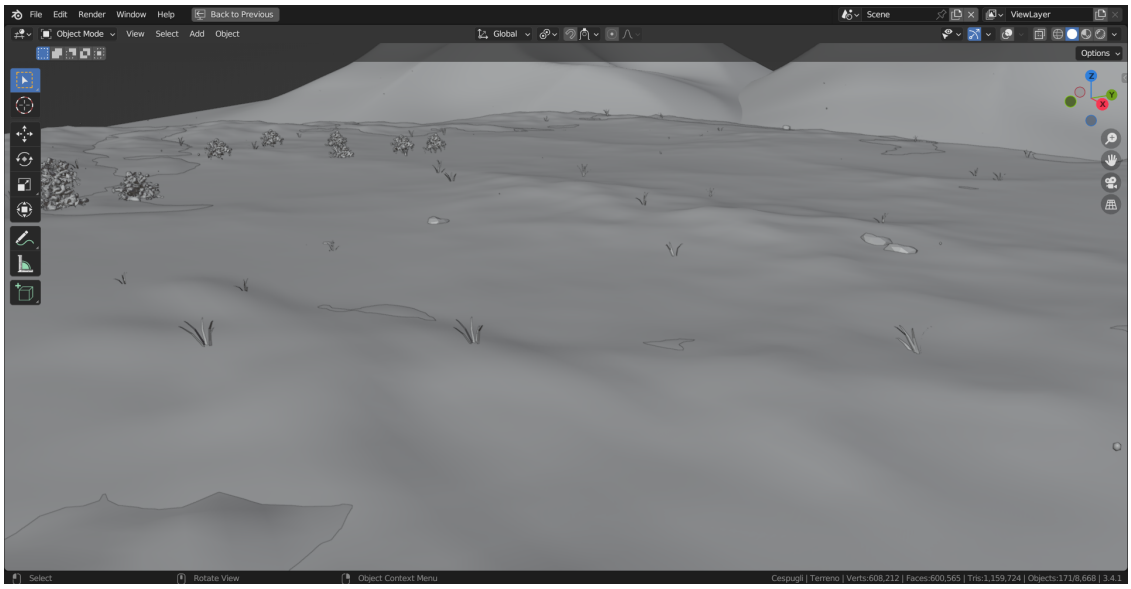

Figura 4.37: Risultato dell'erbetta e delle rocce nella scena 9.

Cespugli

Nella creazione della scena 9 sono stati inseriti dei cespugli che, come per il particellare dell'erbetta, si muovono con il vento. Poiché questi alberi non sono un particellare ma sono mesh vera e propria, non è stato possibile farli muovere utilizzando la stessa forza di tipo *Wind* che è stata utilizzata per muovere l'erbetta, ma è stato utilizzato un altro stratagemma per simulare il movimento. È stato applicato su ogni cespuglio un modificatore *SimpleDeform* di tipo *Bend* in cui è stato animato il valore dell'angolo di deformazione. Una volta applicato un keyframe sul parametro, viene generata automaticamente una curva nel *Graph editor*, a cui è possibile applicare un modificatore. In questo caso è stato scelto un *Noise* (rumore) che assegna un valore casuale all'angolo, ottenendo un risultato simile al movimento prodotto dal vento. Sono poi stati modificati i parametri di *Scale*, *Strenght*, *Offset* e *Phase* in modo da non omologare i movimenti dei singoli cespugli.

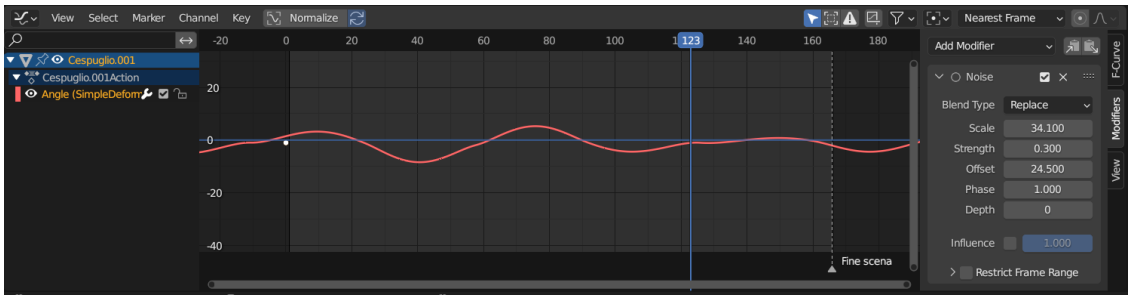

Figura 4.38: *Graph editor* dei cespugli della scena 9.

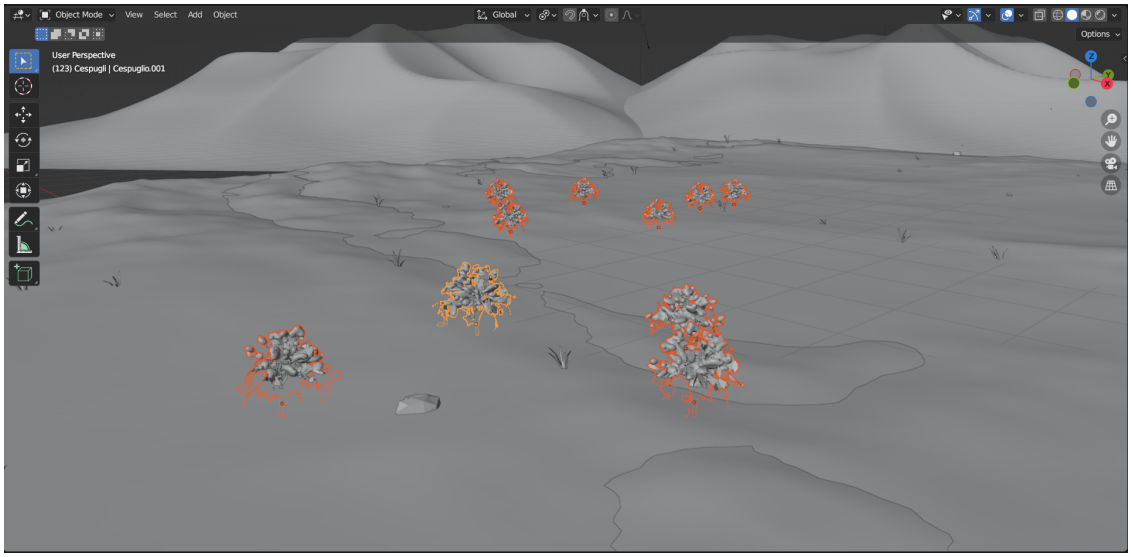

Figura 4.39: Risultato dei cespugli nella scena 9.

4.5.2 Fluid simulation system

I fluidi in questo contesto non si riferiscono esclusivamente a dei materiali simili all'acqua, ma anche a fumo, fuoco o altre sostanze liquide.

Eslposione

In *En la tierra de los sueños perdidos* è stato necessario ricreare un'esplosione ed è stata la fase più difficile e lunga di tutto il lavoro.

Per simulare dei fluidi, sono necessarie due componenti: un *dominio*, la porzione di spazio che contiene la simulazione, e l'*emitter*, la mesh dalla quale parte la vera e propria simulazione.

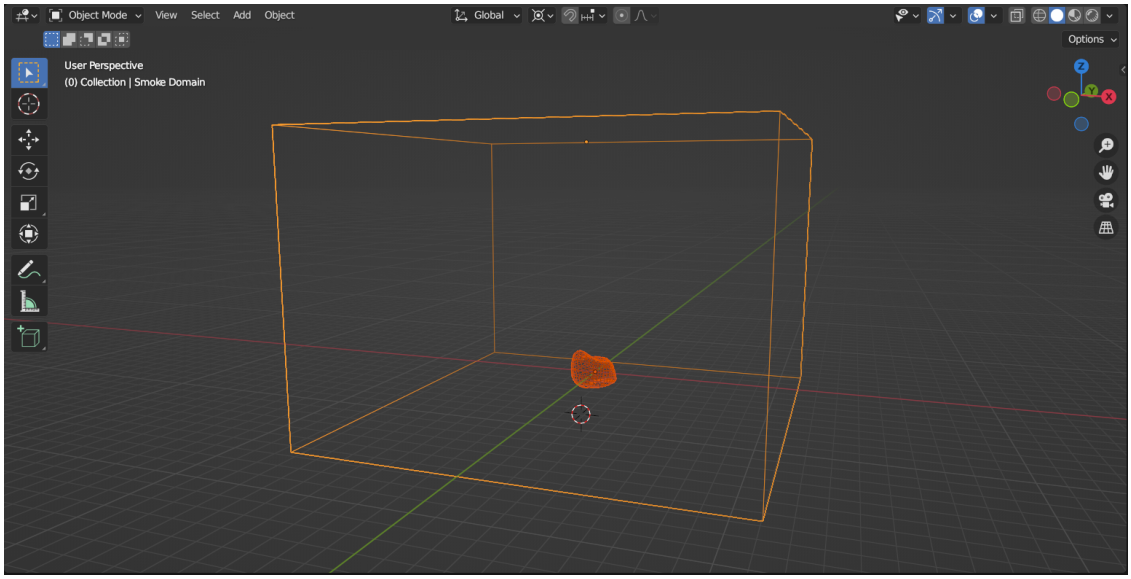

Figura 4.40: *Emitter* e *dominio* per la creazione della simulazione di un fluido.

L'*emitter* può essere un oggetto qualunque, e in questo caso è stato scelto un quarto di sfera i cui vertici sono stati leggermente spostati in modo da conferire alla mesh una forma più casuale. Tra i vari parametri dell'*emitter*, è possibile scegliere il comportamento che questo deve avere, ad esempio *inflow*, ovvero il fluido parte dalla mesh come l'acqua che scorre dal rubinetto, *outflow*, quando il fluido entra nella mesh e scompare come l'acqua che scende nello scarico del lavandino, oppure *geometry*, quando il fluido è la mesh stessa.
Nel caso dell'esplosione del cortometraggio, l'emitter è di tipo *inflow* ed utilizza come tipo di sorgente un *particle system*, così da ottenere un'esplosione abbastanza realistica. Le particelle vengono emesse solo per qualche frame, in modo da dare la sensazione di impulsività dell'esplosione, mentre il fumo che si crea successivamente è dovuto alle impostazioni del dominio.

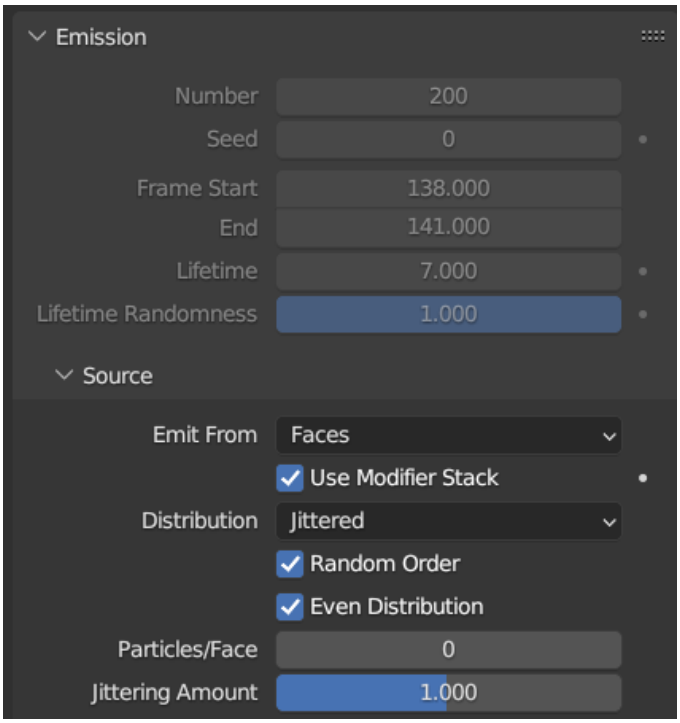

Figura 4.41: Settings del particellare dell'esplosione.

Il dominio, come abbiamo detto, è l'oggetto che contiene la simulazione e quest'ultima non può esistere senza un dominio. È necessario innanzitutto adattare la dimensione del dominio al volume occupato dalla simulazione, poiché, se dovesse essere troppo piccolo, ci sarebbe una collisione con il fluido e la simulazione non potrebbe espandersi come dovrebbe, creando dei bordi netti per niente realistici.
È possibile rendere il dominio *adaptive*, ovvero che si espande o comprime in base al volume occupato dalla simulazione, ma è comunque necessario impostare una dimensione generale del dominio perché l'*adaptive domain* si espande solo fino alla dimensione assegnata.

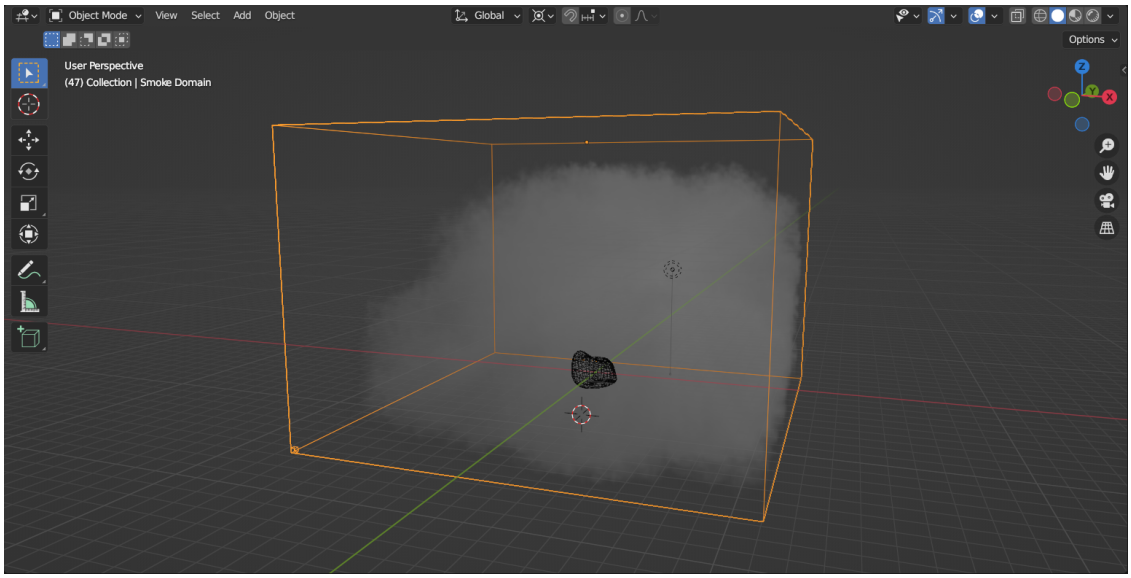

Figura 4.42: Esempio di dominio troppo piccolo per la simulazione e conseguente collisione del fluido con il bordo del dominio.

Un fattore estremamente importante per quanto riguarda una simulazione è la *risoluzione* del dominio, cioè in quante piccole celle, chiamate *Voxel*, può essere suddiviso l'intero volume per poter calcolare in maniera più o meno precisa la simulazione al suo interno. La risoluzione dipende anche dalle dimensioni del dominio stesso, in quanto, per ottenere lo stesso risultato in un dominio più piccolo e in uno più grande, è necessario gestire adeguatamente la risoluzione di entrambi. Nel caso dell'esplosione in considerazione è stata utilizzata una risoluzione di 128 *samples* perché con questa impostazione è stato ottenuto il risultato desiderato. Più si aumenta il numero di *samples*, più sarà precisa la simulazione e viceversa.

Tra i parametri principali per la gestione del fumo e del fuoco abbiamo *Bouyancy density*, che si basa sulla quantità di fumo che si sposta verso l'alto in base alla densità del fluido in cui è immerso l'oggetto, la *Bouyancy heat*, che gestisce la quantità di fumo in base alla temperatura del flusso, in particolare alla *temperatura iniziale*: se entrambi i valori sono positivi, quindi sia la *Bouyancy heat* che la temperatura iniziale, allora il fumo salirà verso l'altro, mentre se uno è positivo e l'altro è

negativo il fumo scenderà verso il basso. Poi abbiamo il parametro *Vorticity*, che controlla la quantità di turbolenze nel fumo.

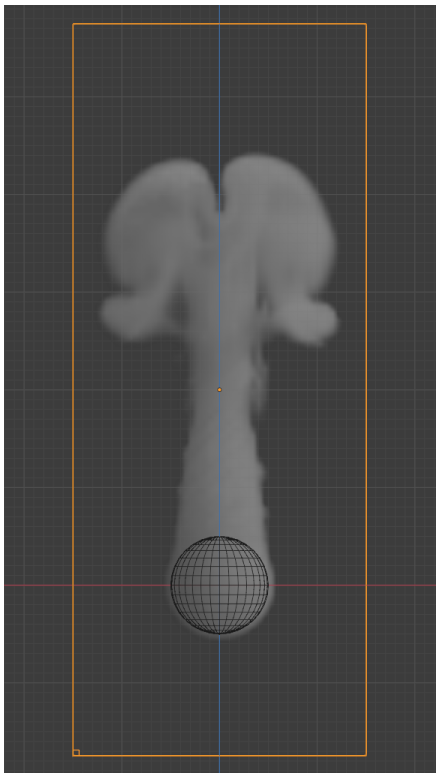

Figura 4.43: Esplosione con *vorticity* uguale a 0.

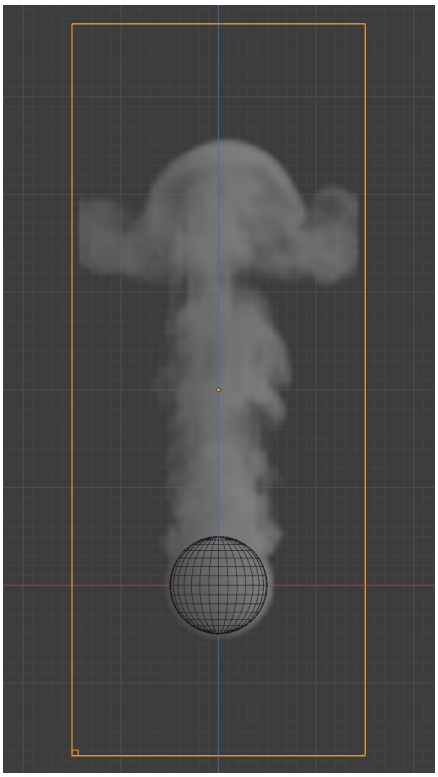

Figura 4.44: Esplosione con *vorticity* uguale a 0.2.

Un altro elemento che aumenta la realisticità dell'esplosione è sicuramente il *noise*, in quanto aggiunge più dettagli al fumo senza cambiare tutto il movimento del fluido. Anche per quanto riguarda il *noise* abbiamo un fattore chiamato *Upres factor* che lavora in maniera simile alla risoluzione che abbiamo visto per quanto riguarda il dominio. Sono comunque due fattori non equivalenti, quindi si possono ottenere risultati diversi usando differenti combinazioni dei due valori.

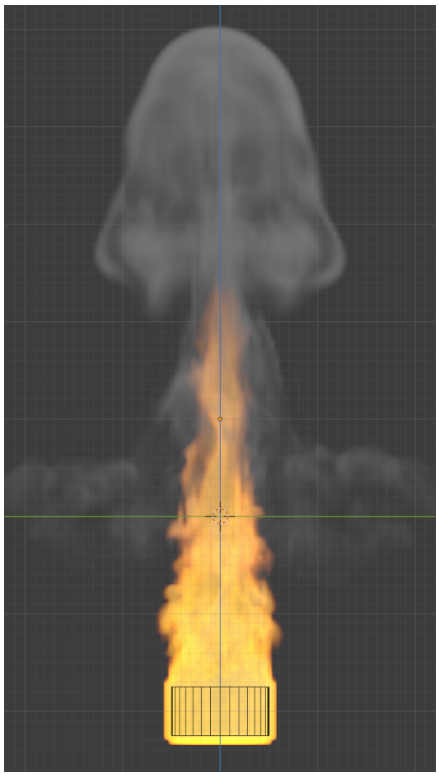

Figura 4.45: Risoluzione del dominio a 200 samples, senza *noise*.

Figura 4.46: Risoluzione del dominio a 100 samples, con *noise scale* uguale a 2.

Il fattore che ha portato via la maggior parte del tempo per la creazione di questa esplosione è stato sicuramente il *bake* della simulazione. Il tempo richiesto per fare il *bake* dipende da tutti i parametri settati in precedenza, quindi dal numero di particelle, dalla quantità di fumo che si vuole creare, da quanto tempo quest'ultimo deve restare nella scena, ma soprattutto dalla quantità di samples assegnati alla risoluzione del dominio. Quindi, per vedere il risultato di un singolo cambiamento bisognava aspettare il tempo del *bake*, che poteva spaziare tra un tempo di 10 minuti a uno di 30.

Figura 4.47: Risultato finale dell'esplosione senza texture.

4.5.3 Cloth modifier

Un altro genere di simulazione che viene spesso utilizzata nel mondo dell'animazione 3D è sicuramente quella dei *cloth* (tessuti). La simulazione dei tessuti è uno degli aspetti più difficili della computer grafica, perché ha una serie di interazioni, sia interne che con l'ambiente esterno, molto complesse.

Questo *modifier* permette alla mesh sulla quale viene applicato di reagire ad eventi esterni come il vento o la forza di gravità, e grazie a questo può essere di grande aiuto per risparmiare tempo nella modellazione. Ad esempio, se si deve modellare una tovaglia su un tavolo, con tutte le sue pieghe, oppure un telo che copre un oggetto, sarebbe un lavoro tedioso da fare manualmente. In questo caso, applicando il *Cloth modifier* alla mesh e lasciandola cadere facendo agire la forza di gravità sull'oggetto, e tenendo conto di altri accorgimenti (come, per esempio, applicare un *Collider* sul tavolo), la mesh si deformerà da sola seguendo le forme del tavolo e si otterrà un risultato molto naturale. Una volta trovata la forma desiderata,

basterà stoppare la simulazione e fare l'*Apply* del modificatore, per "congelare" il movimento e la mesh.

La fisica dei tessuti è stata utilizzata in particolare in due scene, nella scena 10 è presente una bandiera che svolazza con il vento, mentre nella scena 4 è stata applicata a dei fogli di giornale che si muovono dentro un vortice.

Bandiera

Per quanto riguarda la bandiera, è stata prima modellata la mesh, realizzando un bordo frastagliato con lo strumento *knife* di cui abbiamo già parlato precedentemente, e successivamente è stato applicato il *Cloth modifier*, rendendo così la mesh in grado di reagire alle due forze di tipo *Wind* che sono state inserite nella scena. Una volta settati i parametri delle due forze, è stato fatto il *bake* della simulazione della bandiera, "fissandone" il movimento.

In una delle estremità della bandiera sono stati fissati i vertici in modo da evitarne il movimento, in quanto collegati all'asta. Selezionati i vertici di interesse della mesh, sono stati assegnati ad un *Vertex group* e poi nel tab di impostazioni del *Cloth modifier*, sotto la voce *Shape* è stato indicato che quel gruppo di vertici fosse "pinnato", quindi non si sarebbe mosso insieme al resto della mesh. Senza questo passaggio, l'intera mesh sarebbe volata via reagendo con il vento come farebbe un pezzo di carta.

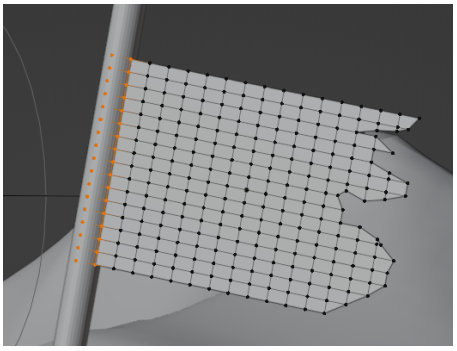

Figura 4.48: Vertici selezionali della mesh della bandiera.

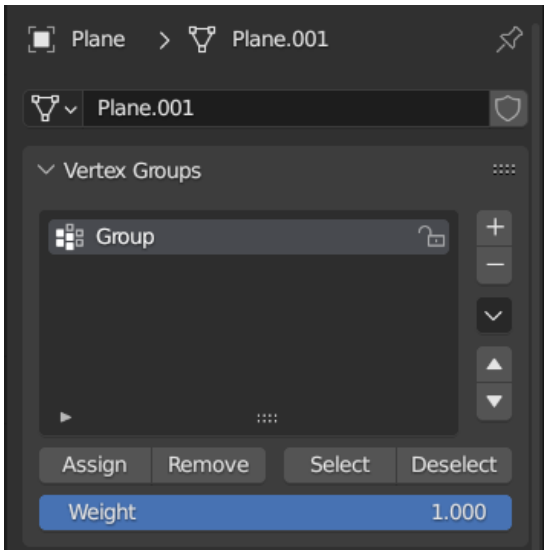

Figura 4.49: Creazione del *Vertex group*.

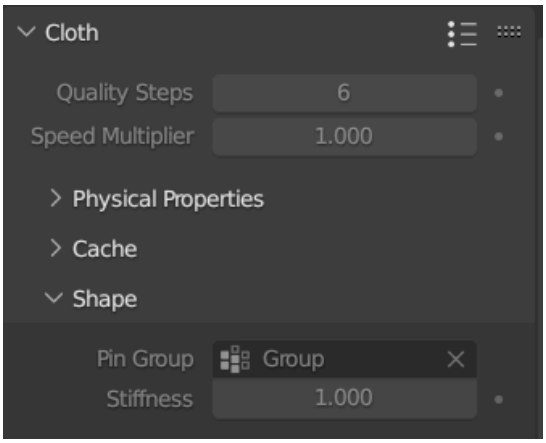

Figura 4.50: *Pin group* nel *Cloth modifier*.

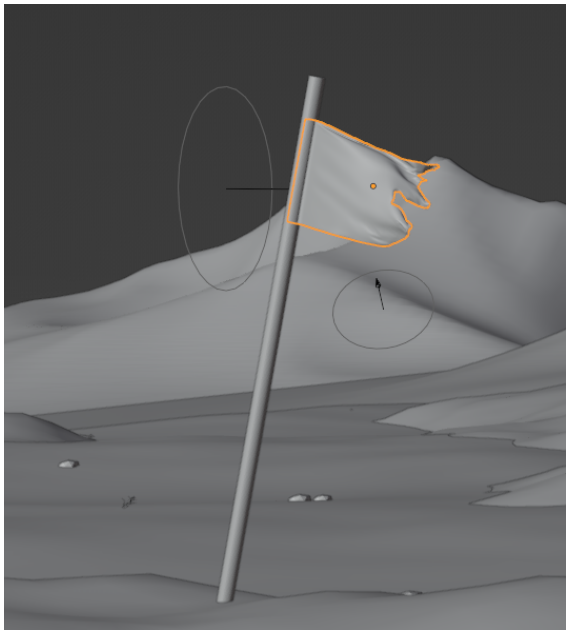

Figura 4.51: Risultato della bandiera in movimento nella scena 10.

Giornali e vortice

Nella scena 4, invece, il *Cloth modifier* è stato applicato a dei fogli di giornale, che se mossi con il vento sembrano simulare il movimento di un tessuto.

Innanzitutto è stato inserito nella scena un piano "di appoggio", sulla quale è stato applicato il modificatore di tipo *Collision*, in modo tale che, una volta che i giornali avessero toccato il piano, non sarebbero passati attraverso. Sono state successivamente inserite tre forze, una di tipo *Vortex*, una di tipo *Wind*, e una di tipo *Turbolence*, in modo da simulare un vortice il più verosimile possibile. Poiché i fogli di giornale, interagendo con queste 3 forze, non davano una vera e propria sensazione di vortice in quanto finivano per volare troppo lontano, è stato inserito un cilindro che contenesse tutti i fogli di giornale, alla quale è stato applicato nuovamente il *Collision modifier*, cosicché quando i giornali vengono a contatto con il bordo del cilindro, restano confinati al suo interno, continuando a girare e dando maggiormente la sensazione di vorticosità.

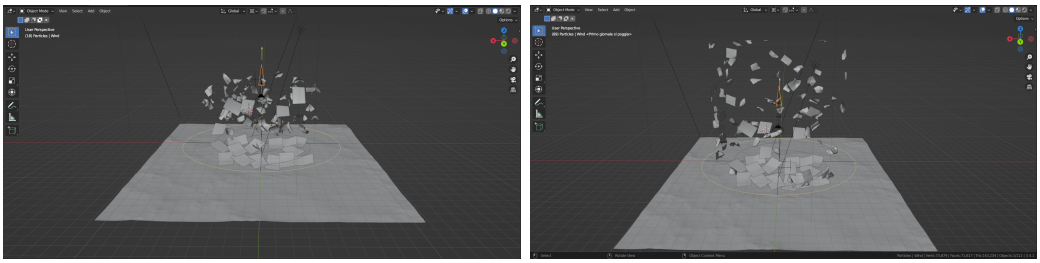

Figura 4.52: Evoluzione della simulazione del vortice di giornali nella scena 4.

4.6 Illuminazione

L'illuminazione è uno degli elementi più importanti di ogni prodotto visivo, serve ad impostare il *mood* generale della scena e soprattutto a dare naturalezza agli oggetti che osserviamo. È infatti molto importante che il *3D lighting artist* mantenga una coerenza visiva in ogni sequenza dell'animazione.

In *En la tierra de los sueños perdidos*, è stato deciso in fase di preproduzione di mantenere un certo tipo di illuminazione e di palette di colori in base al tipo di scena. Ad esempio, nella scena della cascina è stato deciso di mantenere dei

colori caldi che richiamassero il calore di un bel ricordo, infatti sono state inserite due luci di tipo *Sun*, una di colore giallo e una di colore rosa, per simulare le luci di un tramonto. È stata poi aggiunta una luce di tipo *Area* davanti alla casa in modo tale da illuminarla maggiormente, perché è stato notato che solo con le luci *Sun*, risultava parecchio in ombra. Infine sono state inserite delle luci di tipo *Point* all'interno dell'edificio, per dare la sensazione che la luce passasse attraverso le finestre e illuminasse l'interno della casa, mostrando gli oggetti presenti. Anche lo sfondo, che è un piano con una texture, è stato reso *Emitter* in modo da simulare un vero e proprio cielo che illumina la scena.

Figura 4.53: Risultato dell'illuminazione nella scena 2 senza compositing.

Al contrario, nelle scene dei ricordi di guerra, la palette scelta è fatta di colori freddi, tendenti al blu e al grigio.
Nella scena 6 sono state inserite diverse luci, alcune di tipo *Sun* con orientazioni diverse, altre di tipo *Area* posizionate dietro le montagne in modo da enfatizzare l'effetto della *Rim light* che verrà inserita in *compositing*, ed è inoltre presente la luce emanata dall'esplosione che illumina le rocce davanti e parte del terreno.

Figura 4.54: Risultato dell'illuminazione nella scena 6 senza compositing.

4.7 Rendering

Il *rendering* è l'ultima fase della pipeline di una produzione 3D, nonché la più complessa e importante ai fini di ottenere un ottimo risultato. Qui, tutto quello che è stato fatto nelle fasi precedenti della produzione viene mescolato insieme e tradotto in immagini 2D, calcolando ogni dato proveniente da ogni fase della produzione e trasformandolo in una serie di pixel che comporranno l'immagine finale. Questo rende il processo di rendering computazionalmente molto complesso e molto lungo.

Esistono due tipi di rendering, *Real-time rendering*, che, come suggerisce il nome, è un metodo abbastanza veloce da calcolare tutte le componenti della scena e mostrarle in tempo reale sullo schermo, ed è infatti il metodo utilizzato nelle produzioni come videogiochi, con un minimo di 20 frame al secondo; e il *Non-real-time rendering*, utilizzato per produzioni cinematografiche, che produce un risultato molto più dettagliato ma che di contro richiede molto più tempo per ottenere il risultato finale.

Blender include due motori principali di render differenti, *Eevee* e *Cycles*.

Eevee

Eevee (Extra Easy Virtual Environment Engine) è un motore di rendering in tempo reale, quindi è possibile visualizzare in anteprima le animazioni e i progetti, prima di renderizzarli. Tendenzialmente, i *3D artist* hanno molta familiarità con il concetto di *ray-tracing*, ovvero il calcolo della luce nella scena, che viene emessa dalla fonte luminosa e rimbalzata sulle superfici degli oggetti un numero definito di volte. Poiché *Eevee* non esegue questi calcoli per velocizzare i tempi di visualizzazione, diventa di conseguenza un motore di rendering in tempo reale, adatto al mondo videoludico o semplicemente ideale per la previsualizzazione e la prototipazione. *Eevee* esegue invece il calcolo della scena attraverso un metodo chiamato *rasterizzazione*, che prende le informazioni della scena e prevede una stima di come dovrebbero essere, e non di come appariranno poi realmente. Con questa tecnica di stima della scena è possibile falsificare effetti come i riflessi.
Eevee ha una serie di funzionalità che lo rendono un motore di rendering in tempo reale molto potente. Ad esempio, supporta l'illuminazione globale, l'illuminazione ambientale, l'illuminazione volumetrica, l'ombreggiatura normale, l'ombreggiatura di contorno, l'ombreggiatura di *cel-shading*[6] e l'ombreggiatura di schizzo. È anche in grado di gestire la trasparenza, la riflessione, la rifrazione e l'emissione. Tuttavia, presenta alcune limitazioni rispetto al motore di rendering *Cycles*, ad esempio, non supporta la riflessione caustica, la rifrazione caustica, la dispersione della luce, la diffusione della luce e l'illuminazione indiretta avanzata, ovvero i rimbalzi della luce sulle superfici. Grazie al suo lavoro con le luci, Eevee è particolarmente adatto per renderizzare scene notturne e con luci artificiali, ma per la creazione di immagini fotorealistiche è necessario utilizzare il motore di rendering *Cycles*.

[6]Il cel-shading o cel shading è uno stile non fotorealistico di visualizzazione di modelli 3D. Finalizzato a far apparire questi ultimi come se fossero disegnati a mano, con spessi bordi neri a simulare un segno tradizionale e colorazione a tinte unite. Questa tecnica è quindi utilizzata spesso per imitare lo stile grafico dei fumetti o dei cartoni animati tradizionali.

Cycles

Cycles è un motore di rendering fotorealistico, che utilizza la tecnica del *ray-tracing*. Quest'ultimo è essenzialmente un algoritmo che traccia il percorso della luce dalla sorgente alla superficie degli oggetti e simula il modo in cui questa interagisce con gli oggetti virtuali, calcolando ogni singolo rimbalzo della luce su di essi. Il risultato ottenuto sarà sicuramente più preciso e realistico di quello ottenuto con Eevee, ma a scapito di una maggiore quantità di tempo per ottenere il render finale.

Per la fase di rendering delle scene di questo cortometraggio è stato usato il motore di rendering *Cycles*.

Tra i parametri fondamentali da settare prima di lanciare un render c'è sicuramente il numero di *samples* con la quale si vuole renderizzare la scena. Più è alto il numero di *samples*, maggiore sarà il dettaglio dell'immagine, e di conseguenza più lungo sarà il tempo impiegato per completare il render. Per l'intero cortometraggio è stato deciso di impostare un valore di *samples* pari a 32, un numero piuttosto basso ma che comunque attribuisce l'effetto desiderato alla scena. È molto utile anche abilitare l'opzione di *Denoise*, che si occupa della rimozione del rumore dall'immagine senza dover passare l'immagine uscente attraverso strumenti appositi.

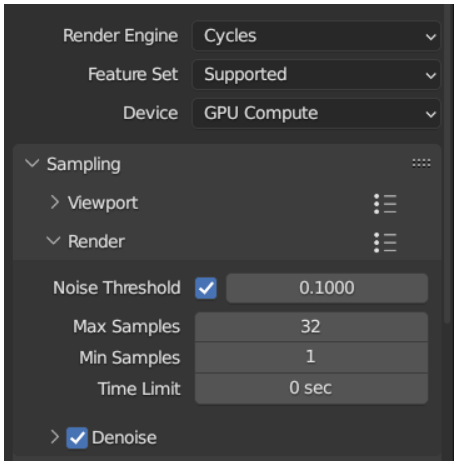

Figura 4.55: Alcune opzioni di *rendering*.

Una delle caratteristiche fondamentali della fase di *rendering* è quella di poter renderizzare le scene "per passi", ovvero renderizzare separatamente i vari aspetti, dagli oggetti, alle texture, all'illuminazione e così via, per poi compositarle, cioè metterle insieme, in post-produzione. Questo processo è stato particolarmente utile per quanto riguarda la scena 6 dell'esplosione, in cui è stato necessario renderizzare

degli oggetti a parte, e in più, di questi stessi elementi è stato utile renderizzare il parametro *Alpha*.

In questo insieme di nodi del compositing che riguardano la scena 6, notiamo che dal nodo *Render layer* è possibile selezionare alcuni dei parametri da renderizzare. Il primo è *Image* che renderizza tutta la scena in una volta, con tutti i calcoli necessari sulla luce e quant'altro, subito dopo troviamo invece il parametro *Alpha*, che permette di renderizzare un'immagine contenente le informazioni sulla trasparenza.

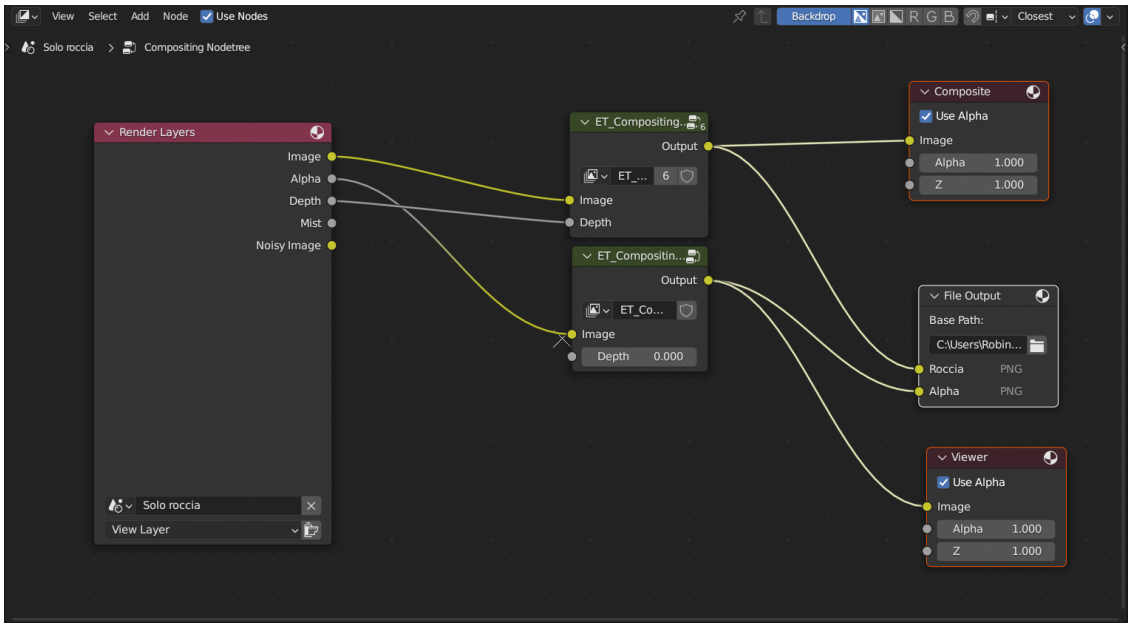

Figura 4.56: Nodi del compositing per creare render separati.

Per quanto riguarda questa specifica scena, in cui è necessario renderizzare una roccia separatamente dallo sfondo, dopo aver impostato il *background* del mondo su *Transparent*, avremo un'immagine del canale *Alpha* in cui il volume occupato dalla roccia sarà rappresentato da un colore chiaro, mentre tutto il resto, che è trasparente, sarà rappresentato da un colore nero. Un'immagine di questo tipo è estremamente utile per creare il cosidetto "mascherino" in fase di compositing su software come After Effects, tramite cui è possibile isolare un elemento ed utilizzarlo

in maniera separata dal resto della scena.

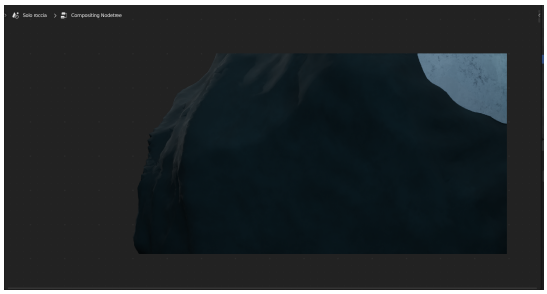

Figura 4.57: *Rendering* dell'*Image* della roccia della scena 6.

Figura 4.58: *Rendering* dell'*Alpha* della roccia della scena 6.

La stessa cosa è stata fatta per l'esplosione, renderizzata separatamente dal resto della scena per permettere di apportare modifiche specifiche sul colore, per gestire meglio l'aggiunta della nebbia in post-processing e anche per facilitare la fase di rendering.

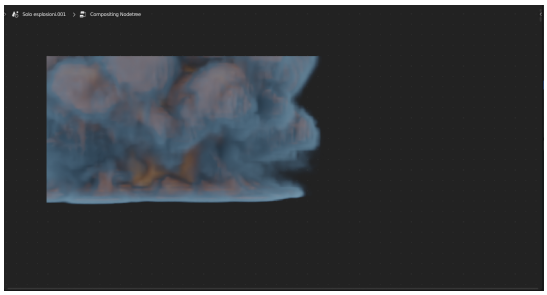

Figura 4.59: *Rendering* dell'*Image* dell'esplosione scena 6.

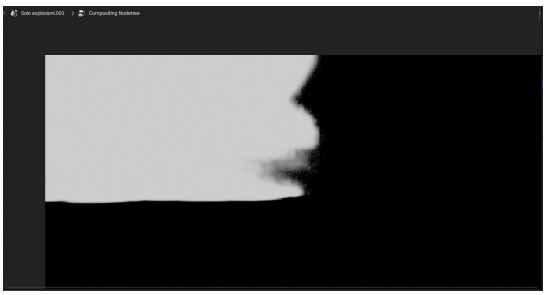

Figura 4.60: *Rendering* dell'*Alpha* dell'esplosione della scena 6.

Capitolo 5

Post-produzione

Nella terza e ultima fase della pipeline produttiva di un prodotto animato vengono aggiunti al progetto gli ultimi tocchi per perfezionare il risultato. Qui gli artisti della post-produzione hanno a loro disposizione una vastissima serie di strumenti con la quale possono dare al progetto qualsiasi tipo di look loro vogliano, senza dover renderizzare nuovamente tutti i frame e risparmiando così tantissimo tempo. I passi generali di questa fase sono il *compositing*, gli *effetti visivi 2D*, il *montaggio* e la *color correction*.

5.1 Compositing 3D

Come è stato anticipato nella sezione 4.7, è possibile renderizzare i vari livelli di una scena separatamente, ed è proprio in questa fase che vengono *compositati* per ottenere il look finale. Il *compositing*, quindi, combina layer di immagini separati in un'unica immagine, permettendo così all'artista di manipolare selettivamente gli elementi senza dover rigenerare la scena complessiva. Questa può essere definita una delle fasi più importanti nella realizzazione di un prodotto come un cortometraggio animato, ma non solo, in quanto è qui che si riesce a dare coerenza a tutto quello che è stato fatto finora e, in particolare, nel caso di un cortometraggio che prevede

l'utilizzo di due stili di animazione differenti, è necessario fare in modo che l'estetica sia mescolata perfettamente per dare un senso di continuità allo spettatore.

Il tab del *Compositing* in Blender si presenta con un interfaccia simile a quella del tab *Geometry nodes*, oppure a quella dello *Shading*, perché anche in questo caso lavoriamo con una struttura a nodi, partendo da un input per ottenere un output finale.

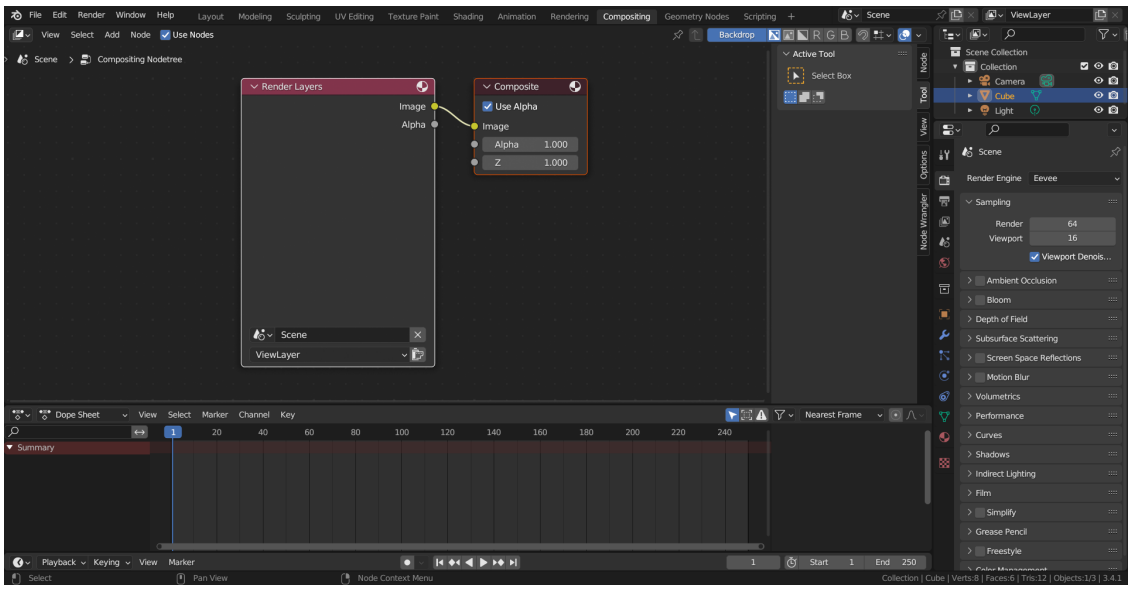

Figura 5.1: Tab del *Compositing* in Blender.

L'utilizzo del *Compositing* non è obbligatorio, quindi per far si che il render passi anche per questa fase è necessario attivare la dicitura *Use Nodes* in alto a sinistra, cosicché ad ogni frame renderizzato vengano applicate tutte le modifiche del caso. Di default compare un nodo di input chiamato *Render Layers* che prende come immagine di input proprio il render della scena, ma è possibile utilizzare un qualsiasi altro genere di input, come un'immagine esterna, una texture e così via, mentre in output troviamo il nodo di *Composite* collegato all'output del render e che mostra su di esso tutte le modifiche aggiunte in questo processo.

Per una questione di comodità, è stato utilizzato anche il *Viewer Node* che permette

di mostrare l'immagine che si sta compositando nel background del tab, senza dover controllare continuamente l'immagine nel tab *Rendering*. Questo nodo è molto utile perché mostra anche l'azione vera e propria del nodo selezionato, in questo modo è possibile capire come questo stia agendo sull'immagine di input.

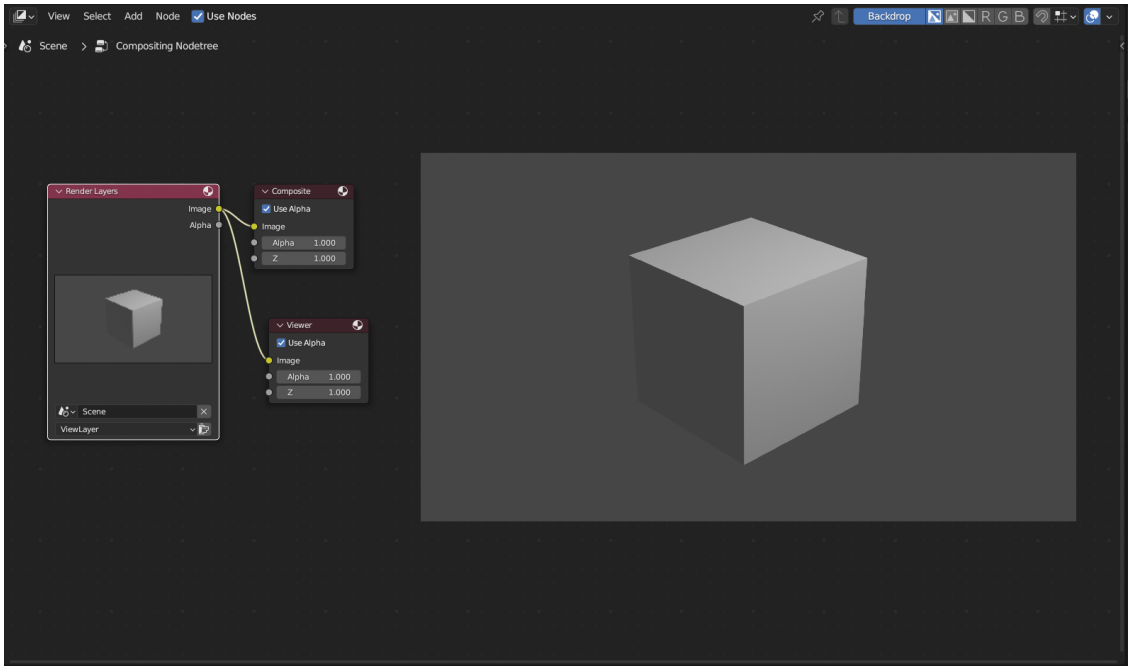

Figura 5.2: Nodi di default nel tab *Compositing*.

Tra le modifiche più importanti che si vogliono aggiungere in questa fase troviamo principalmente quelle che riguardano l'estetica del prodotto, in quanto si voleva dare alle scene uno stile quasi fosse un dipinto, aggiungendo una sorta di pennellate sulle texture già presenti.

È possibile creare dei gruppi di nodi in modo da avere una gerarchia ordinata di livelli di compositing, e questo è quello che è stato fatto per quanto riguarda il cortometraggio in questione.

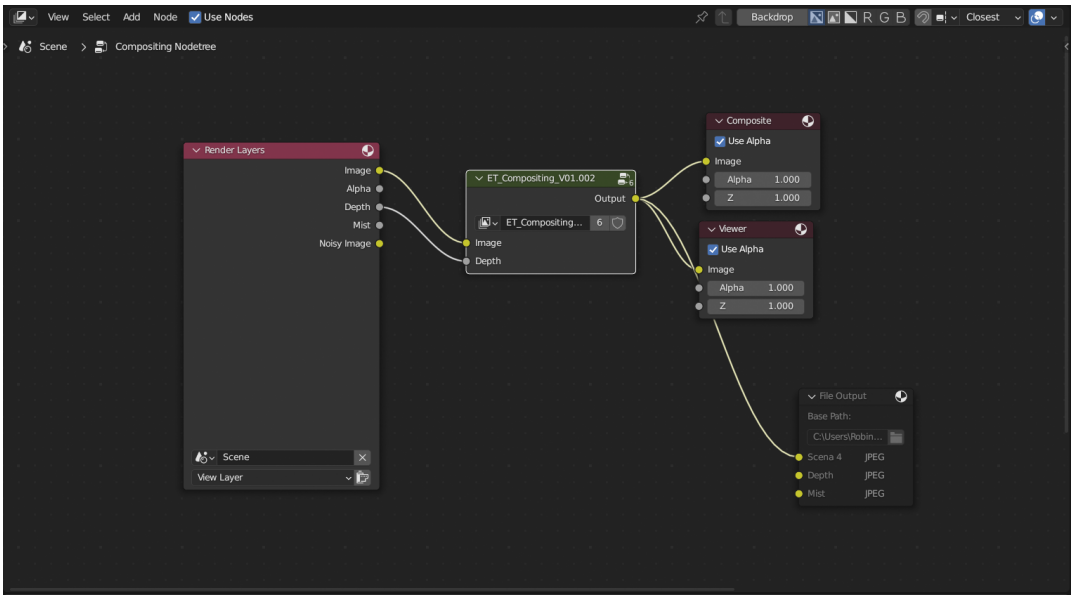

Figura 5.3: Nodi utilizzati per compositare la scena 6.

All'interno del nodo "ET Compositing V01" troviamo tutti i nodi che effettivamente agiscono sull'immagine del *Render Layer*.

Il nodo presenta al suo interno altri nodi suddivisi in sezioni in base all'effetto che aggiungono sull'immagine, e la struttura generale è la seguente, nella figura 5.4.

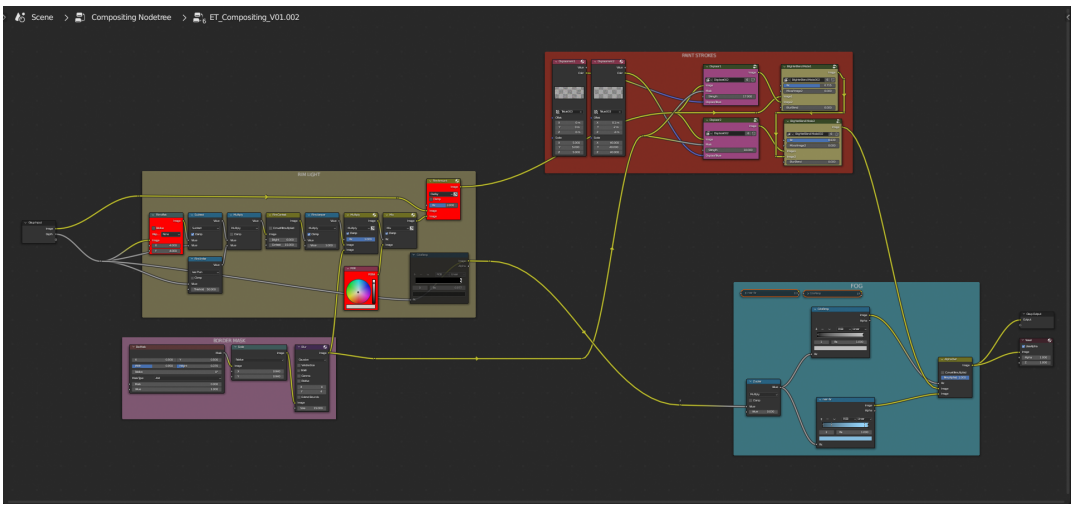

Figura 5.4: Nodi utilizzati per compositare la scena 6.

Rim Light

La prima sezione che troviamo è quella della *Rim Light*. La "Luce di contorno" è un effetto di illuminazione che si ottiene posizionando una fonte luminosa dietro un soggetto in modo da illuminarne il contorno, creando un bordo luminoso intorno ad esso per aiutare a separarlo dallo sfondo e per creare una sensazione di profondità e tridimensionalità. In questo caso è stato utilizzato il dato proveniente dal passo di *rendering z-depth*, ovvero il valore pari alla distanza che ogni oggetto ha rispetto alla camera, ed è stato utilizzato per creare un bordo leggermente luminoso intorno agli oggetti della scena.

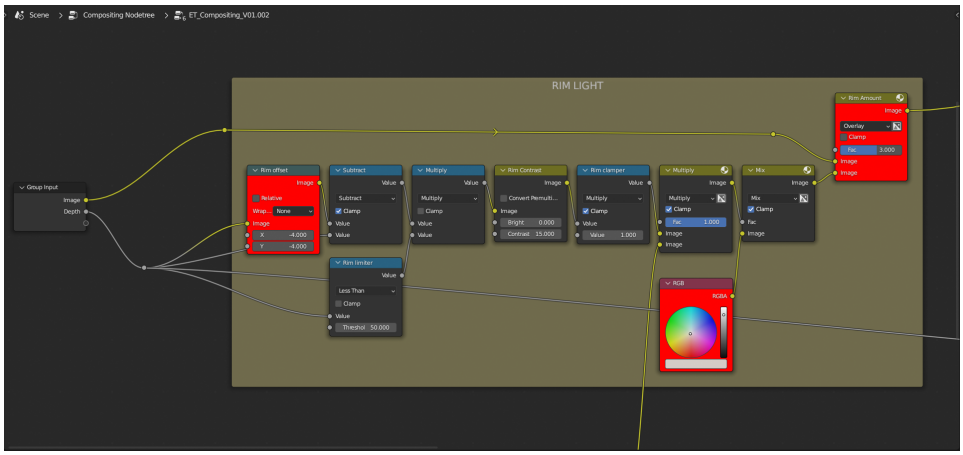

Figura 5.5: Nodi per creare la *Rim light*.

Figura 5.6: Scena 6 senza *Rim light*.

Figura 5.7: Scena 6 con *Rim light*.

Paint strokes

La sezione di *Paint stokes* è quella che crea l'effetto dipinto desiderato.

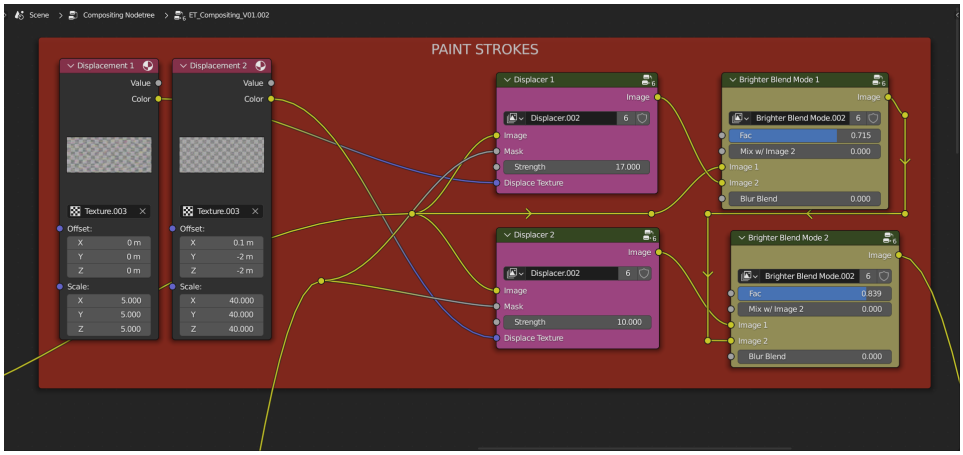

Figura 5.8: Nodi per creare il *Pain strokes*.

Abbiamo in input due texture di rumore che vengono utilizzate come *displace* per creare l'effetto frastagliato e non preciso tipico di un disegno fatto a mano. Dopodiché viene aggiunto un ulteriore bordo più chiaro per accentuare questo effetto, come possiamo maggiormente notare sulle montagne dello sfondo dove l'effetto è più evidente.

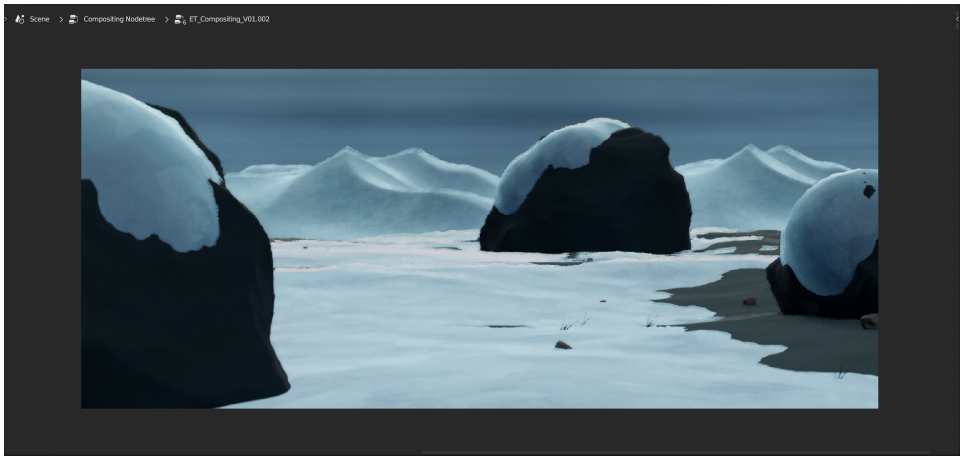

Figura 5.9: Risultato del *Pain strokes* applicato alla scena 6.

Border mask

Poiché questo effetto, tramite il *Displacer*, crea il suddetto bordo frastagliato degli oggetti, è stato notato che nei bordi dell'immagine, questa veniva tagliata, lasciando un sottile bordino vuoto. Per evitare questo effetto che sarebbe stato sgradevole in fase di montaggio, è stata inserita una sezione che creasse una maschera in modo da far agire i nodi del *Paint strokes* solo nella zona desiderata.

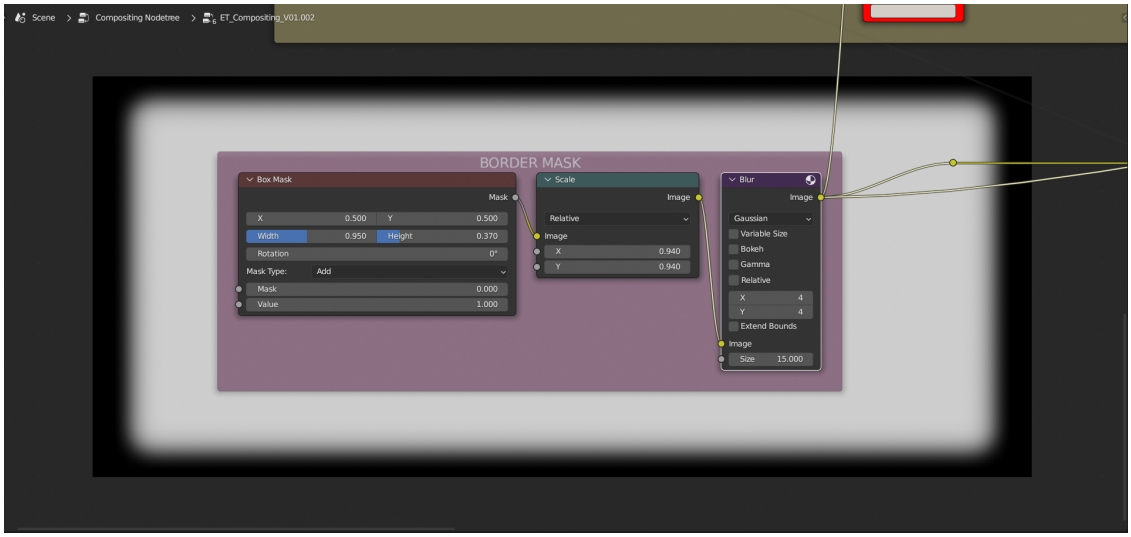

Figura 5.10: Nodi per creare la maschera.

Nebbia

L'ultima sezione del *compositing* che agisce sull'immagine è quella che crea una sorta di nebbia nella scena, per enfatizzare l'effetto dei ricordi offuscati del protagonista, presente soprattutto nelle scene ambientate sulle isole che hanno ospitato la guerra. Anche in questo caso è stato utilizzato il valore dello *z-depth*, che è stato moltiplicato per un determinato valore all'interno del nodo *Z scaler* e che è diventato successivamente l'input di due *Color ramp*, una che definisce in quali punti c'è più o meno visibilità (quindi più o meno nebbia) assegnando un colore scuro alle zone con meno nebbia e un colore chiaro nelle zone dove deve essere presente

più nebbia, mentre la seconda *Color ramp* definisce il colore bluastro che si vuole assegnare alla scena.

Le informazioni uscenti dalla seconda *Color ramp* e quelle che provengono dalla sezione di *Paint strokes* convergono nel nodo *Alpha Over*, che sovrappone le due immagini. L'input *Fac* prende il valore dalla prima *Color ramp*, quella che indica la quantità di nebbia da creare.

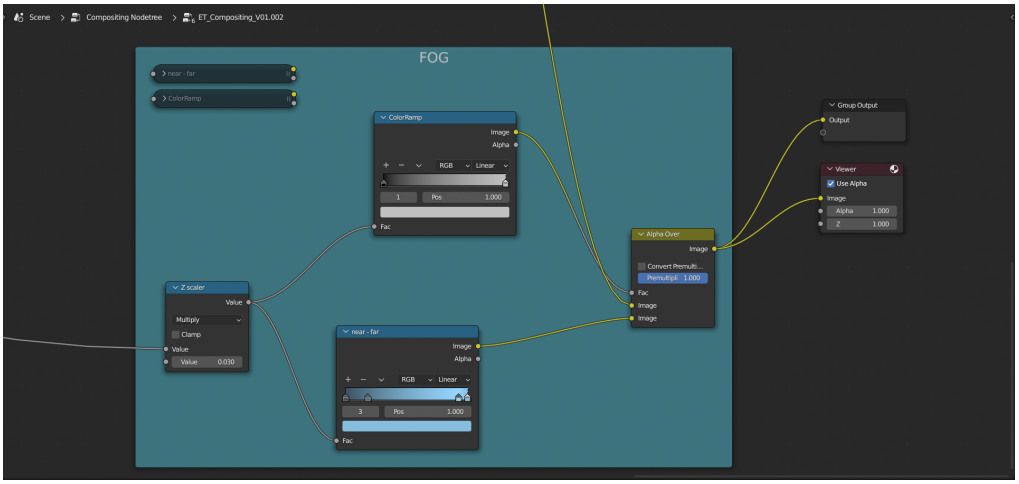

Figura 5.11: Nodi per creare la nebbia.

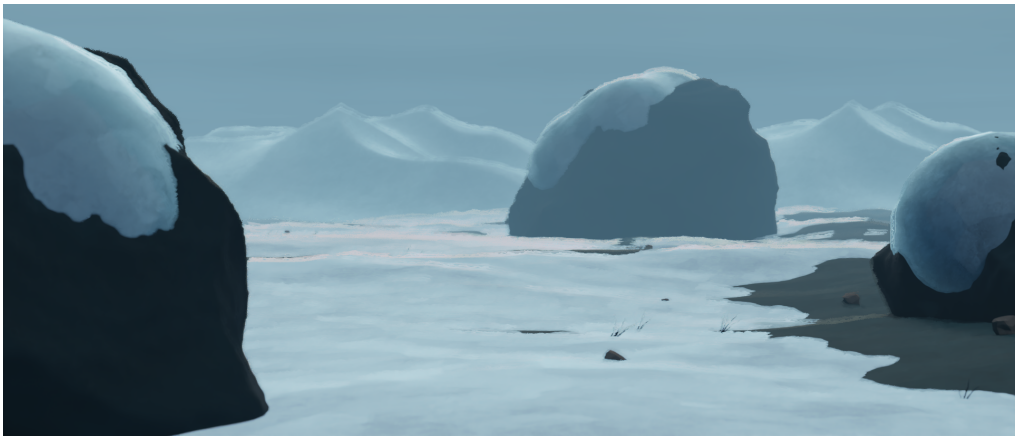

Figura 5.12: Risultato finale del *compositing* applicato alla scena 6.

Il nodo "ET Compositing V01" è stato utilizzato in tutte le scene, con i dovuti accorgimenti del caso, quindi nelle scene in cui non era necessaria la nebbia sono stati bypassati i nodi che si occupavano di crearla, e sono stati apportati gli aggiustamenti del caso per ottenere un'estetica convincente per ogni scena.

5.2 Effetti visivi 2D

È tipico inserire in post-produzione alcuni elementi visivi per dare alla scena un particolare mood, come ad esempio polvere, fumo, pioggia o neve. Per quanto riguarda la realizzazione di questo progetto, tutti gli effetti visivi sono stati realizzati in 3D in Blender e l'argomento è stato approfondito nella sezione 4.5.

Capitolo 6

Conclusioni

Dopo aver attraversato e analizzato tutte le fasi della produzione dei contenuti 3D e la fase del compositing, è possibile affermare che, tramite gli strumenti presenti nel mercato odierno, non per forza ad alto budget, siamo in grado di mescolare 2 tecniche diverse di animazione all'interno di un unico prodotto. La creazione di materiali e di texture che ricordassero il disegno realizzato a mano è stato fondamentale per riuscire nell'intento di mantenere una coerenza tale da camuffare la differenza delle due parti presenti nella scena.

Di seguito sono riportati alcuni render finali che hanno attraversato anche lo step del *compositing 3D* e che mostrano le ambientazioni nella loro forma finale, insieme alle immagini di reference alle quali sono ispirate.

Figura 6.1: Risultato finale della scena 9.

Figura 6.2: Reference per la creazione della scena 9.

Figura 6.3: Risultato finale della scena 9 inquadratura 2.

Figura 6.4: Risultato finale della scena 2.

Figura 6.5: Reference per la creazione della scena 2.

Figura 6.6: Risultato finale della scena 7.

Figura 6.7: Reference per la creazione della scena 7.

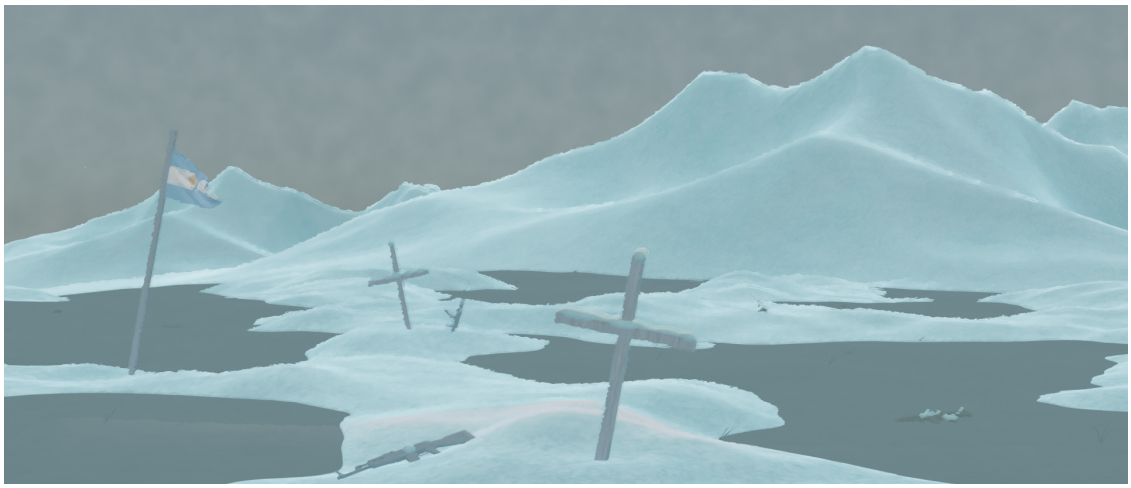

Figura 6.8: Risultato finale della scena 10.

Le difficoltà più grandi incontrate nella realizzazione di questo progetto vedono sicuramente coinvolta la scena dell'esplosione, sia per quanto riguarda la natura della simulazione, sia a causa della scarsa efficienza del computer a disposizione, che ha sicuramente allungato tantissimo i tempi dei *bake* e non ha reso facile la manipolazione delle scene non permettendomi di vedere i cambiamenti in tempo reale. Nonostante ciò, queste difficoltà sono state comunque uno strumento per cercare altri escamotage e modi per rendere le scene più efficienti e facili da maneggiare, portando a soluzioni di cui abbiamo parlato nel capitolo della modellazione.

Per gli sviluppi futuri del progetto è prevista, una volta ultimato il montaggio, l'iscrizione del cortometraggio al festival *ANIMA – Festival Internacional de Animación de Córdoba* e ad altri concorsi nazionali e internazionali.

Bibliografia

[1] Payam Adib. «3D Animation Pipeline: A Start-to-Finish Guide». In: *Dream Farm Studios* (2023). URL: https://dreamfarmstudios.com/blog/3d-animation-pipeline/.

[2] Taher Ahmed. «'Puss in Boots: The Last Wish' revels in fairy-tale style animation». In: *Deccan Herald* (2023). URL: https://www.deccanherald.com/entertainment/entertainment-news/puss-in-boots-the-last-wish-revels-in-fairy-tale-style-animation-1190105.html.

[3] Giacomo Beretta. «Teenage Mutant Ninja Turtles: Mutant Mayhem, l'artista dell'animazione in stile Spider-Verse». In: *Nerdpool.it* (2023). URL: https://www.nerdpool.it/2023/03/09/teenage-mutant-ninja-turtles-mutant-mayhem-lartista-dellanimazione-in-stile-spider-verse/.

[4] Charles Bramesco. «How Spider-Man: Into the Spider-Verse Changed the Animation Game». In: *Vulture* (2019). URL: https://www.vulture.com/2019/01/how-spider-man-into-the-spider-verse-changed-animation.html.

[5] Sammie Crowder. *Shading, Lighting, and Rendering with Blender EEVEE*. Packt, 2022. URL: https://subscription.packtpub.com/book/game-development/9781803230962/2/ch02lvl1sec04/what-is-eevee-eevee-as-a-rendering-engine.

[6] Jackie Thomas. «What is ray tracing? The games, the graphics cards and everything else you need to know». In: (2019). URL: https://www.techradar.com/news/ray-tracing.