POLITECNICO DI TORINO

Collegio di Ingegneria Informatica, del Cinema e Meccatronica

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

Tesi di Laurea Magistrale

Modellazione di ambienti 3D per la realizzazione di un prodotto indipendente di animazione: Reverie Dawnfall

Relatore

prof. Antonino Riccardo Silvio Antonio

Candidato

Ginepro Ilaria
Matricola 233376

Dicembre 2019

"*Do not judge each day by the harvest you reap, but by the seeds that you plants."*

Robert Luis Stevenson

RINGRAZIAMENTI

A tutto il team di produzione con il quale ho lavorato in questi lunghi mesi presso lo studio Robin: mi avete aiutato a superare alcune delle mie insicurezze e a crescere professionalmente e personalmente; al mio professore e relatore Riccardo Antonino, che mi ha dato la possibilità di far parte ad un progetto di animazione di grosse potenzialità e nel quale credo molto.

Ai miei amici universitari più cari, che hanno sopportato le mie ansie in tutti questi anni, e che hanno reso speciale questo percorso di studi.

Ai miei amici di una vita, nei quali trovo sempre un rifugio per sfogarmi e svagarmi e che hanno sempre creduto in me, anche e soprattutto nei miei periodi più bui: siete la mia seconda famiglia.

Ai miei genitori: a voi riservo il ringraziamento più grande. Mi avete sempre sostenuta in questi anni faticosi, sopportandomi e supportandomi; mi avete permesso di intraprendere questo percorso di studi grazie ai vostri sacrifici, mai scontati. A mia madre, da cui ho ereditato la sensibilità e il carattere socievole di cui vado fiera: mi hai trasmesso la forza per affrontare e superare ogni situazione, insegnandomi che per ogni problema esiste sempre almeno una soluzione. A mio padre, perchè è il modello di figura professionale a cui aspiro e a cui ho sempre aspirato: se ho intrapreso questo difficile percorso di studi è grazie a te e a ciò che tu rappresenti per me. Spero possiate essere orgogliosi della persona che sto diventando.

A mia sorella Silvia, che in questi ultimi mesi è diventata la mia più fidata amica; sei fonte di ispirazione per me, la sorella maggiore che tutti vorrebbero avere: grazie per rappresentare una seconda madre per me; hai un cuore grande e vali molto più di quanto tu possa immaginare.

A mia nonna, che con la sua dolcezza e saggezza mi insegna ogni giorno una lezione di vita.

A Giacomo, che è per me ormai un membro della famiglia: grazie per avermi aiutata in alcuni processi di realizzazione di questa tesi, e grazie per l'interesse che hai mostrato sin dall'inizio verso questo progetto

Sommario

1. Introduzione

1.1 Motivazione

Il progetto di tesi che viene presentato in questo elaborato è stato ispirato principalmente da un interesse personale verso il mondo della modellazione 3D, applicata più nello specifico, all'animazione cinematografica. L'attrazione al mondo del cinema e degli effetti speciali mi portò, diversi anni fa, a scegliere di intraprendere il percorso di laurea triennale in *Ingegneria del Cinema e dei Mezzi di Comunicazione;* il corso di *Computer Grafica 3D,* svolto l'ultimo anno della triennale e tenuto dal professor *Andrea Bottino,* rappresenta, oltre il mio primo vero approccio alla grafica e alla modellazione 3D, anche la nascita della mia passione verso questo affascinante mondo che è in grado di rappresentare ogni immaginario pensabile. Successivamente, il corso magistrale di *Realtà Virtuale* per il quale ho dovuto ideare e realizzare un videogame, quello di *Effetti Speciali* tenuto dal mio professore nonchè relatore Riccardo Antonino, e il corso di *Computer Grafica* terminato con la realizzazione di un breve video in animazione 3D, hanno contribuito ad alimentare il mio interesse verso la grafica 3D, più in particolare verso la modellazione di ambienti e scenari virtuali. Il progetto indipendente di una serie animata come *Reverie Dawnfall*, ideata e pensata dal professor Antonino, mi ha permesso di studiare e approfondire le mie conoscenze sui software per la realizzazione di contenuti 3D, e testare approcci professionali diversi al fine di trovare la soluzione più adatta allo scopo richiesto dal team di produzione.

1.2 Obiettivo

Reverie Dawnfall è un progetto indipendente di animazione pensato, ideato e autoprodotto dal professore di effetti speciali del Politecnico di Torino *Riccardo Antonino,* nonchè fondatore di *Robin Studio,* giovane azienda di produzione video e contenuti multimediali per la quale è stata svolta questa tesi*;* serie animata composta da tre stagioni, Reverie raccoglie tutti gli sforzi finora compiuti per raggiungere una produzione di alto livello con le tecnologie più all'avanguardia accessibili per il mercato indipendente.

L'obbiettivo che mi è stato proposto dall'azienda all'interno di questo potenziale progetto è la realizzazione dei modelli 3D degli ambienti presenti nella seconda versione del teaser trailer della serie animata, utilizzando come programma di modellazione il software open source Blender, e sfruttando tutte le innovative potenzialità che la sua più recente versione 2.80 può offrire.

Con questa seconda versione del teaser trailer di Reverie Dawnfall, si desidera introdurre il progetto all'interno del mercato nazionale ed internazionale cercando quindi di realizzare un prodotto multimediale animato innovativo e fortemente competitivo in Italia, con l'aspirazione di espandersi anche al di fuori del confine del nostro Paese.

1.3 Organizzazione dei contenuti

I primi capitoli di questo documento vogliono essere un'introduzione al progetto vero e proprio per cui si è svolta questa tesi; verrà quindi analizzato il concetto di modellazione 3D specifico per gli environments, sottolineandone gli elementi importanti da tenere in considerazione per una buona riuscita dell'obiettivo. Dopodichè verranno elencati e brevemente descritti i principali software esistenti dedicati alla modellazione 3D; si motiverà quindi la scelta effettuata a favore del software *Blender,* delineandone le novità apportate nella sua ultima release 2.80, prima tra tutte il suo nuovo motore di render real time *Eevee*, utilizzato dalla produzione in fase di rendering per il teaser trailer di *Reverie Dawnfall*.

Con la seconda parte del documento si entrerà invece più nello specifico all'interno del progetto indipendente prodotto da *Robin Studio*, azienda per la quale è stata svolta questa tesi: verrà introdotta la serie animata in stile cyberpunk Reverie Dawnfall, descrivendone l'ambientazione e presentandone i principali protagonisti; verrà esposto un confronto tra la prima versione del teaser trailer della serie già esistente, sviluppata e prodotta tra il 2017 e il 2018 e ciò che invece ci si aspetta da questa seconda più innovativa e rappresentativa versione. Prima di procedere con l'analisi dettagliata di ogni singolo environment realizzato, vengono esposti i principali concepts e gli stili grafici utilizzati come reference per la realizzazione del trailer e in generale di tutto il progetto seriale, composto da tre stagioni. Infine, sono stati esaminati nel dettaglio tutti gli steps esecutivi che hanno portato alla nascita di tutti gli ambienti 3D così come presentati nel teaser trailer, evidenziandone le problematiche tecniche avvenute in corso d'opera e le soluzioni trovate per superarle, il tutto lavorando esclusivamente su Blender 2.80 e testandone quindi ogni novità proposta dal software.

2. La modellazione 3D

La modellazione 3D nella computer grafica, è il processo atto a definire una forma tridimensionale in uno spazio virtuale generata su computer; una delle prime rappresentazioni tridimensionali su calcolatore, datata 1960, è stata quella del famoso '*primo uomo'* o '*Boeing Man*' realizzata da *William Fetter,* un insieme di linee che descrivono la sagoma virtuale di un pilota di aereo.

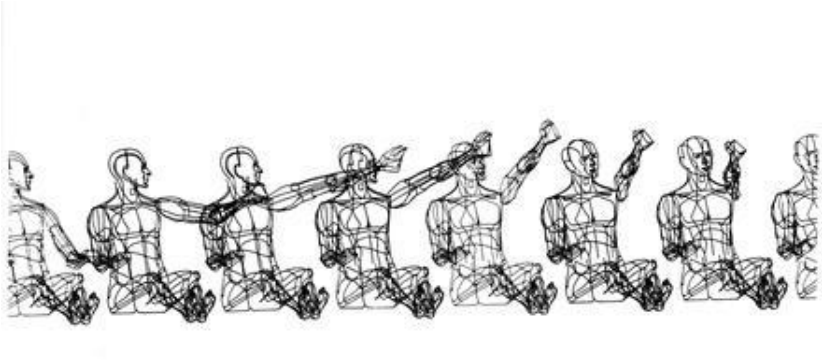


Figura 2.1: Boeing Man di William Fetter, 1960.

La nascita della modellazione 3D è dunque avvenuta in ambito industriale, primariamente come supporto alla progettazione: da allora i campi di utilizzo della stessa si sono enormemente ampliati, uscendo in buona parte dall'ambito teorico; i sistemi di modellazione 3D vengono ora impiegati nei più svariati ambiti della computer grafica, da applicazioni a carattere scientifico o tecnico (come nelle scienze applicate, nella medicina, nell'ingegneria industriale, civile, nell'architettura e nella progettazione di parti meccaniche, e quant'altro) fino alle applicazioni puramente artistiche come l'industria cinematografica e televisiva, l'industria videoludica, la grafica pubblicitaria, il web design e altre svariate applicazioni multimediali.

Da un punto di vista tipologico, tutta la modellazione 3D si può facilmente dividere in due macro gruppi, rappresentati dalla *modellazione organica* e da quella *geometrica*. L'ultima è il tipo di modellazione meno recente, utilizzata per realizzare praticamente qualsiasi cosa che abbia una natura artificiale; la modellazione organica invece viene utilizzata per esempio per realizzare characters o soggetti 'naturali'

come piante, rocce o alberi, o comunque per ogni oggetto che abbia forme morbide e naturali.

2.1 Modellare ambienti 3D

In qualsiasi produzione 3D ci sarà sempre la necessità di avere ambienti ed oggetti di scena costruiti per aiutare a preparare al meglio il background e la storia del progetto che si sta realizzando, che si tratti di un videogame, di un film di animazione o di uno spot pubblicitario. Gli environments e gli oggetti presenti in essi rappresentano una parte vitale per qualsiasi produzione, tanto quanto i personaggi; un singolo ambiente può richiedere centinaia di assets organizzati per arrivare al risultato finale.
La missione di un environment artist non è altro che quella di creare gli spazi e i luoghi che creeranno un senso di immersione, e contribuiranno a raccontare una storia.

Il primo elemento fondamentale da tenere in considerazione per una buona riuscita di un ambiente modellato in 3D è la *storia* che si vuole raccontare, e quindi come l'ambiente in considerazione deve adattarsi ad essa. Questo è uno step molto importante, poichè sapere in partenza dove l'environment si inserisce all'interno della storyline, evita che in seguito siano necessarie rilavorazioni e riadattamenti durante la pipeline.
Una volta in mente l'obiettivo e la storia, è importante sviluppare il 'look' che dovrà avere l'intero environment: si procede quidi sviluppando e creando delle librerie di *concept art* e stili da usare come *references* per il proprio progetto. In poche parole questa è la fase cruciale che aiuta a determinare l'idea di come verrà rappresentato l'intero mondo tridimensionale, e specifica quindi come tutti gli altri elementi della storia si integrano con l'ambiente in questione. Nelle alte produzioni, questa fase di lavoro viene realizzata da veri e propri artisti dedicati che si concentrano esclusivamente sul concept design.
Soltanto dopo aver bene in mente lo stile che deve assumere e la storia in cui l'ambiente 3D che si vuol realizzare è ambientato, si può procedere con la fase di modellazione: questa è la fase di sviluppo di tutte le risorse chiave; essa comprende la

progettazione, il blocking, la creazione e il raggruppamento di tutti gli assets necessari per rendere vivo e credibile un environment 3D. Lavorando con una varietà di tools di modellazione e sculpturing, gli artisti si basano su ciò che il concept artist ha creato e prodotto, per realizzarne in modo più coerente possibile una versione tridimensionale; la geometria viene poi resa viva grazie ai materiali ed alle texture, ed in generale allo shader applicato, sia che si voglia realizzare qualcosa di fotorealistico o meno.
Modellando quelli che vengono chiamati probs, gli oggetti di scena, sarebbe bene capire dove essi si troveranno rispetto alla telecamera: è inutile infatti perdere tempo a creare ogni piccolo dettaglio di un particolare prob se la fotocamera non sarà mai abbastanza vicina ad esso da consentire a chi stà guardando la scena la differenza.

Pare dunque naturale che gli environment 3D possono variare a seconda del task richiesto, da una futuristica città di fantascienza ad un interno di un appartamento; spetta al modellatore ed artista creare oggetti di scena e tutto ciò che costituisce il ‘set’ in un modo credibile e funzionale per immergerne al meglio all’interno i personaggi protagonisti e le loro storie.

3. I principali software per la modellazione 3D

Esistono svariati software dedicati alla modellazione ed animazione 3D, ognuno con i propri punti di forza che rispondono alle diverse esigenze del progettista; di seguito verranno elencati e brevemente descritti i programmi più utilizzati per la modellazione tridimensionale.

3.1 ZBrush

Zbrush è un programma di scultura e painting digitale che ha rivoluzionato l'industria del 3D grazie al suo workflow intuitivo ed ai suoi potenti strumenti. Esso utilizza una tecnologia proprietaria che immagazzina informazioni relative al colore, all'illuminazione, al materiale e all'intensità di tutti gli oggetti visualizzati. La differenza sostanziale tra ZBrush e tutti gli altri software in commercio, sta appunto nell'essere più intuitivo e verosimile nella scultura: viene utilizzato come strumento di scultura digitale poichè in grado di lavorare con modelli ad altissima risoluzione, come per esempio personaggi per videogiochi o film di animazione, non a caso infatti è il programma preferito dei 3D Characters designers. ZBrush utilizza livelli dinamici di risoluzione che permettono agli scultori di effettuare cambiamenti locali o globali ai propri modelli; è conosciuto per la facilità nella resa di dettagli a livello medio/alto, che vengono tradizionalmente resi con le *bump map*: la mesh dettagliata risultante può essere esportata come displacement o normal map da usare nelle versione low poly dello stesso modello; questo può risultare necessario per un character che deve essere animato, poichè nell'animazione si è soliti a lavorare con mesh a bassa risoluzione, molto più adatte al calcolo computazionale richiesto. Zbrush, come la maggior parte dei software a seguire, è a pagamento, con un prezzo che varia a seconda delle versioni scaricate.

3.2 Houdini 

Questo software deve la sua popolarità al realismo di effetti organici che è in grado di simulare; si tratta essenzialmente di un programma procedurale, che in quanto tale necessita di un alta potenza di calcolo, anche se questo problema è divenuto trascurabile grazie alle sempre più potenti CPU e GPU dei computer di oggi.

Esiste una versione base gratuita che mette a disposizione tutti i tools principali necessari in una pipeline di animazione, nonostante offra un render limitato rispetto alla versione pro, a pagamento, dedicata ad una produzione più professionale.

La natura procedurale di Houdini è data dal suo caratteristico workflow basato sui nodi: ogni informazione ed azione è registrata in un nodo, chiamato operatore, il quale collegato ad altri nodi crea quello che si definisce *albero nodale* (*nodetree*). Houdini viene visto come un software di programmazione visiva, il che pone la programmazione artist-friendly, ovvero di facile accesso agli utenti, esperti o meno che siano.

Questo software viene principalmente usato per la creazione di effetti speciali nei prodotti cinematografici.

3.3 3D Studio Max 

3Ds Max è un programma di grafica vettoriale tridimensionale ed animazione, realizzato dalla divisione *Media & Entertainemen*t di *Autodesk*. È uno dei software più utilizzati per la creazione 3D per molte ragioni, tra cui le sue potenti capacità di editing e la sua architettura dei plugin, realizzati anche da terze parti. 3Ds Max ha una serie di funzionalità avanzate, tra cui un generatore e renderizzatore built-in normal map, stati inclusi per migliorare le capacità di progettazione dei videogiochi; a partire dalla versione 6.0 poi, è stato introdotto un nuovo motore di render, *Mental-Ray;* include strumenti di illuminazione avanzata come *Radiosity* e *Light Tracer,* ed un software per la creazione di realistiche simulazioni dinamiche, utilizzate in parecchi videogiochi, *Havok Reactor*.

Dispone di sette metodi per la modellazione: NURB, NURMB, Surface tool, modellazione con primitive, modellazione con mesh, modellazione con path e modellazione con poligoni.

3.4 Autodesk Maya

Autodesk Maya è un software di computer grafica 3D, apprezzato soprattutto per l'alta qualità degli strumenti di modellazione, rigging, animazione e rendering. Rappresenta uno dei software maggiormente utilizzati nella realizzazione dei film in CGI.

Una delle categorie in cui eccelle è sicuramente la *Character Animation*, tanto da farne appunto uno dei programmi più usati sia nelle produzioni ad alto budget dei film d'animazione, sia per lo sviluppo di modelli real-time per i videogiochi.

Dispone di grande libertà di personalizzazione dell'interfaccia grafica e implementazione di plugin, grazie ai linguaggi di scripting Python, MEL e C++.

Maya ha grossi vantaggi che riguardano il suo punto forte, ovvero l'animazione digitale: consente infatti di memorizzare nella cache le modifiche apportate alle scene, mentre gli animatori lavorano, velocizzando così i ritmi produttivi; permette inoltre di visualizzare un'anteprima delle animazioni in un ambiente 3D e apportare modifiche in tempo reale. L'interfaccia grafica è totalmente adattabile alle direttive del creativo che adopera allo scopo di massimizzarne l'utilizzo, anche se per un principiante potrebbe risultare un po' complicaato; l'usabilità infatti è un po' più complessa e meno intuitiva rispetto ad altri software, ma non necessariamente può essere considerato un punto a suo sfavore.

3.5 Cinema4D

Cinema4D è il software di maggiore spicco per la motion graphic, che rappresenta l'altro lato della medaglia dell'animazione, oltre ai visual effects. Esso si rivela in grado di offrire riultati pulti, chiari e decisamente professionali. Offre molti plugin, come Mograph, che è un set molto potente di tools che permettono di creare accattivanti animazioni astratte e motion graphic. Il software supporta tecniche di modellazione procedurale, poligonale e solida; è dedicato principalmente alla post-produzione di film per la realizzazione di effetti speciali, principalmente grazie al modulo opzionale *Bodypaint 3D.*

È apprezzato nel mondo della grafica e dell'animazione anche grazie all'integrazione con i più diffusi software del settore come Adobe After Effect e Adobe Illustrator.

3.6 Blender

Blender è un software libero e multipiattaforma che supporta l'intera pipeline 3D: modellazione, rigging, animazione, simulazione, rendering, compositing e motion tracking, oltre a video editing e game creation. È considerato un ottimo trampolino di lancio per chi decide di entrare nel mondo 3D, proprio per la sua natura open source. Nasce come prodotto interno della società di animazione olandese NeoGeo, fondata nel 1988 da Roosendal; più tardi, lo stesso fondò la società NaN per continuare il suo sviluppo e distribuirlo come freeware. A causa dell'interruzione degli investimenti però, nel 2002, Roosendal creò la fondazione no-profit Blender Foundation, che il 13 ottobre di quello stesso anno, riuscì nell'intento di rendere Blender un progetto open source: da questo momento, il software divenne un progetto libero, sotto la licenza GNU (General Public Licence). Blender si è rivelato essere un prodotto open source molto attivo che vive sostenendo quelli che sono gli obiettivi della corporazione indipendente pubblica no-profit che rappresenta.

Nell'ambito del 3D differisce ancora da programmi come Maya o Maxon Cinema4D, specialmente per l'animazione e le simulazioni.

Il primo grande progetto professionale nel quale Blender è stato usato come strumento primario è stato la previsualizzazione dell'animatic del film *Spiderman2;* ricordiamo inoltre il film d'animatione tutto italiano, interamente realizzato con Blender, *Gatta Cenerentola.*

4. Blender come scelta definitiva

Dopo aver confrontato tra di loro i maggiori software per la computer grafica 3D, la produzione ha deciso di affidarsi a *Blender* per l'intera pipeline di lavoro per la realizzazione del secondo teaser trailer di Reverie Dawnfall.

Le motivazioni che hanno portato a questa scelta sono varie e numerose, prima fra tutte la natura open source del software: come qualsiasi software open source, Blender permette a tutti gli utenti di accedere al codice sorgente, scritto in Python, ed effettuare modifiche sul programma stesso, o creare plugin che permettano di integrare le funzionalità che si ritengono più opportune in base alle necessità; se tali tools si rivelano utili ai più, è probabile che vengano inseriti all'interno della successiva release. Questo particolare, proprio di un software open source, è stato molto utile in fase di produzione, poichè ha permesso di risolvere problemi altrimenti irrisolvibili in un tempo limitato, specialmente in fase di animazione; inoltre essendo la serie in questione un prodotto che trova le sue fondamenta nella *sperimentazione*, avere la possibilità di personalizzare l'applicazione secondo le nostre necessità, aumentava di gran lunga l'opportunità di sperimentare.

Un altro dei motivi che hanno portato alla scelta di Blender per la produzione del teaser è stato sicuramente l'uscita della sua nuova versione, la 2.80: con tutte le sue innovazioni introdotte, questa nuova release ha contribuito ad inserire il software in una categoria altamente professionale, a partire dalla sua nuova interfaccia grafica. Tutte le potenzialità e le novità di questa versione, sono rappresentate nell'ultimo short film (il dodicesimo Open Movie di Blender) *Spring,* interamente realizzato con la 2.80; l'intero processo di produzione di Spring e tutti i file delle risorse sono state poi condivise nella piattaforma di produzione *Blender Cloud,* per dare la possibilità

agli utenti di studiarne i processi di sviluppo relativo ad ogni step della pipeline; ecco quindi che anche per il nostro team di produzione, è risultato estremamente utile capire come sono stati realizzati determinati particolari all'interno del cortometraggio. Inoltre, la possibilità di sperimentare la nuova versione con tutte le novità e i miglioramenti apportati, e realizzare con essa ipoteticamente l'intera serie, può considerarsi un punto di forza per la stessa, poichè non esistono al momento molti prodotti cinematografici o televisivi realizzati interamente con la nuova release di Blender, tanto meno con il suo nuovo motore di render real time, la verà novità del software, *Eevee.*

Figura 4.1: Spring, Open Movie di Blender 2.8.

4.1 I pro e i contro di un software open source

Quando si parla di software open source si rishia spesso di cadere con una facilità disarmante in luoghi comuni o situazioni poco chiare, sia per quanto riguarda il business derivante da questi programmi, sia per le potenzialità che gli vengono associate.

La possibilità che dà un software open source di poter controllare ed esaminare nel profondo ciò che stiamo utilizzando, avendo a disposizione l'accesso libero al codice sorgente, è sicuramente un fattore da non sottovalutare tra i lati positivi della natura del programma. Poter valutare infatti la qualità dei sorgenti, anche in collaborazione con più persone nel caso di contesti collaborativi, è un fattore che può aiutare sviluppatori, tester ed utenti finali a verificare il reale funzionamento dell'applicazione così come può aiutare ad isolarne le lacune per il successivo debugging; inoltre la possibilità di poter modificare un software secondo le nostre esigenze può spesso fare la differenza all'interno del progetto a cui stiamo lavorando. Essendo un programma gratuito, alla base dell'open source vi è un modello di licensing che non prevede il pagamento del pacchetto software ma bensì di chi eroga i servizi, del supporto tecnico, degli aggiornamenti di sicurezza tempestivi, di tutti i plugin annessi e tutti i servizi correlati.
Un altro fattore importante conseguente alla libera disposizione del codice sorgente è sicuramente quello relativo alla sicurezza, poichè tale disponibilità permette di controllare che all'interno di esso non siano presenti backdoor o altre tipologie di codice malevolo. Le licenze d'uso delle soluzioni proprietarie infatti, spesso non sottolineano sufficientemente le implicazioni di un determinato applicativo per la privacy e la tutela dei dati veicolati. L'altro lato della medaglia però, mostra la mancanza di reali garanzie sul funzionamento di un codice, la responsabilità sulle modifiche dello stesso infatti, sono a carico dell'autore che le sta effettuando.
Purtroppo la nomea di software open source stimola spesso l'utilizzatore medio a pensare che si tratti di qualcosa di mediocre, e non aiuta certamente sapere che i programmi a pagamento siano quelli più utilizzati dalle grandi case di animazione internazionali, basti pensare al già citato *Maya* ed a quanto sia ampio il suo utilizzo nel mercato cinematografico mondiale rispetto a *Blender;* 'gratis' sembrerebbe significare 'poco valido'. Sicuramente rispetto ad un programma a pagamento, un software open source può avere dal punto di vista tecnico strumenti e possibilità considerate acerbe, e quindi non 'production ready' all'interno di un ambito lavorativo di alto livello. Blender in effetti ha molto da invidiare a *Maya* per quanto riguarda l'animazione 3D, o a ZBrush per lo sculpturing 3D; inoltre la simulazione

dei *Cloth Material* è ancora molto acerba ed è pesante da processare a livello computazionale per il software; ma gli utenti fedeli del programma in questione sono disposti ad attenderne la versione sempre più aggiornata che garantisca un miglioramento dei tools meno efficienti, aggiornamenti inoltre disponibili ad una velocità nettamente maggiore rispetto a qualsiasi software a pagamento.
Ed è proprio con la versione di Blender 2.80 che forse si inizierà a diffondere maggiormente il programma in ambito professionale.

4.2 Blender 2.80

Con la release 2.80, Blender potrà forse finalmente essere inserito in una categoria di software altamente professionali. La nuova versione ha tante novità, prima tra tutte la sua nuova interfaccia utente molto più intuitiva della precedente, grazie ad una vasta gamma di tools, gizmos e layout coerenti per rendere più semplice la visualizzazione e l'utilizzo di molte proprietà di Blender, ridisegnata ad hoc per porre l'attenzione sull'artwork che si sta creando; il tema ora è scuro e sono state introdotte moderne icone per ogni funzionalità, forse per adattarsi allo stile d'interfaccia grafica dei software 3D più famosi come Maya e Cinema4D. Le interazioni con mouse e tastiera hanno subito una rivoluzione a partire dalla selezione tramite tasto sinistro del mouse di default; quest'ultimo particolare può essere rilevante per gli utenti che si approcciano per la prima volta al software, abituati ad usare quel comando in qualsiasi altro programma. Grazie ai gizmos nella nuova barre degli strumenti è possibile scoprire o accedere più velocemente all'uso di alcuni tools che precedentemente richiedevano delle combinazioni di tasti, noiosi da memorizzare. Viene inoltre introdotto un menù *Quick Favorites* che permette un rapido accesso ai tools maggiormente usati dell'artista.

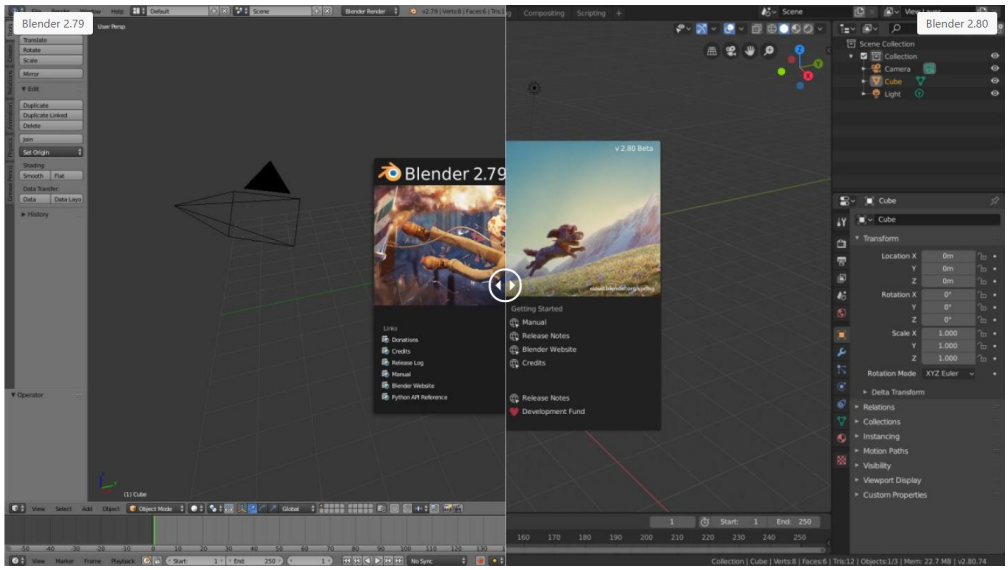


Figura 4.2: Interfaccia dela versione 2.80 a confronto con la precedente 2.79.

Grazie al nuovo moderno viewport 3D si è in grado di visualizzare una scena ottimizzata per l'attività che si sta eseguendo; è stato progettato un nuovo motere di rendering di Workbench per eseguire il lavoro nel viewport, supportando attività come il layout della scena, la modellazione e la scultura.
Questa versione del software, ha poi trasformato Blender in un programma di disegno ed animazione 2D grazie alle novità apportate al *Grease Pencil:* viene introdotto l'oggetto Grease Pencil, a cui vengono associati tutti i tools e le operazioni apportati agli oggetti già esistenti nel software; ora gli '*strokes*' si possono animare, possono essere inseriti in un ambiente 3D ed interagire con esso, vi si possono applicare materiali e texture, ed hanno modificatori simili alle texture, senza contare che un oggetto Grease Pencil può essere trasformato in curva che a sua volta può essere trasformata in geometria tridimensionale.
Modifiche sono state apportate anche al motore di render Cycles, che ora offre funzionalità standard del settore come *Cryptomatte,* che semplifica notevolmente la fase di compositing facilitando la creazione di *maschere* veloci, o il nuovo shader physically-based *Principled Hair BSDF* per un render più fotorealistico dei capelli e pelo. Sono state fatte inoltre molte ottimizzazioni di rendering tra cui un render combinato tra CPU e GPU, che ne velocizza le prestazioni, ed un supporto CUDA per scene che non rientrano nella memoria GPU.

Ma la novità più importante e strabiliante apportata a Blender 2.80 è sicuramente il nuovo motore di render real time *Eevee,* che può essere usato sia come renderer per i fotogrammi finali, sia come motore che guida la visualizzazione per la creazione di risorse.

4.3 EEVEE: il nuovo motore di render real time

Esistono principalmente tre tipologie di motori di render: i motori di render *Raytracing,* in cui tutti i calcoli delle luci, delle riflessioni e delle ombre vengono ottimizzate ed approssimate per ottenere un buon risultato con calcoli veloci; motori di render *Unibiased*, come *Cycles* per esempio, che utilizzano algoritmi che si sforzano di riprodurre in maniera realistica il comportamento della luce, a scapito dei tempi di calcolo. Nel caso particolare di Cycles, la sua potenza rispetto ad altri motori di render Unibiased è rappresentata dalla possibilità di utilizzare schede video in modalità OpenCL o CUDA, per poter velocizzare notevolmente il calcolo altrimenti svolto dal processore. Infine esistono i *motori di render real time,* che hanno la capacità di calcolare, e quindi di mostrare a schermo le immagini in brevissimo tempo: in questo caso il progettista può verificare in tempo reale il risultato e le modifiche delle sue scelte progettuali; quando si cambiano per esempio i materiali, l'illuminazione della scena, cambia la loro rappresentazione nel rendering istantaneamente. Questo tipo di motore di render è stato, fino a poco tempo fa, principalmente un'esclusiva prevalente delle produzioni videoludiche; ma la caratteristica del real time di ridurre di gran lunga i tempi di produzione, rispetto a quanto è possibile con i workflow tradizionali, ha introdotto l'utilizzo del render in tempo reale anche in altri ambiti. Blender con la sua versione 2.80 ha introdotto il nuovo render engine real time *Eevee,* il quale segue il trend della game industry e del PBR (Physically Based Rendering) supportando schede video high-end per avere un viewport estremamente reattivo.

Di seguito verranno esposte le caratteristiche tecniche di Eevee in comparazione con il motore di render Cycles, già precedentemente integrato in Blender.

Path Tracing, Rasterizzazione e Ray Tracing

Il motore di render *Cycles* funziona proiettando raggi di luce da ciascun pixel della camera presente in scena: i raggi riflettono, rifrangono e vengono assorbiti dagli oggetti fino a quando non colpiscono una sorgente di luce o raggiungono il limite di rimbalzi settato; ma seguire soltanto un raggio di luce da un pixel non è un'operazione così accurata poichè molti dettagli della texture degli oggetti per esempio, possono essere più piccoli di un pixel, per questo motivo Cycles genera ulteriori raggi randomizzati (samples) da quello stesso pixel e ne fa la media del risultato nel tempo. Questo approccio brute-force della media dei campioni randomizzati è chiamato *simulazione Monte-Carlo,* e quando è applicato a percorsi di luci viene chiamato *path tracing.* Sebbene il path tracing non sia del tutto accurato per rappresentare un alto grado di realismo (ha infatti un limite di rimbalzo e non è comunque in grado probabilmente di simulare ogni fotone), si traduce nella tecnica per ottenere le immagini in computer grafica il più possibile fotorealistiche, nonostante richieda un alto costo computazione per essere processato, risultando quindi piuttosto lento.

Il metodo di rendering utilizzato dalla maggior parte dei 3D game engines e quindi dai motori di render real time, è chiamata *rasterizzazione*; rappresenta uno dei metodi più antichi per il rendering nella computer grafica ed è incredibilmente veloce, specialmente se si tratta di scene semplici. Esso funziona proiettando le facce di un modello sui pixel che compongono l'immagine 2D visibile sullo schermo; Eevee utilizza la rasterizzazione tramite OpenGL 3.3, motivo per cui può essere velocissimo rispetto ad altri tipi di rendering. La cosa fondamentale da ricordare è che questa velocità ha naturalmente un costo in termini di accuratezza, proprio perchè si sta lavorando con le informazioni relative ai pixel e non sugli effettivi percorsi di luce.
Ma alla rasterizzazione si sta combinando sempre più spesso il ray tracing anche per i motori di render real time; alcuni giochi infatti hanno iniziato a sfruttare le nuove schede RTX di NVIDIA che hanno un'architettura specifica costruita apposta per tracciare i raggi di luce. Ma ciò che viene utilizzata non è propriamente una tecnica di

rendering vera a propria, ma piuttosto un ibrido: viene usata una versione di path tracing con un solo sample e un numero limitatissimo di rimbalzi (il ray tracing ha bisogno in realtà di un solo raggio per pixel per essere definito tale), per rendere solo i riflessi e le ombre , processandolo attraverso un denoiser di apprendimento automatico (motion learning), combinando il risultato con la rasterizzazione per ottenere la versione finale dell'immagine.

Ambient Occlusion

L'occlusione ambientale di Cycles si basa sulla distanza tra le singole superfici e lo spazio 3D in cui sono collocate; l'AO di Eevee invece, viene calcolata a seconda della distanza tra le immagini 2D e le viste dello schermo; il risultato è soddisfacente ugualmente, poichè viene utilizzata l'informazione sulla profondità per capire dove applicare il 'f*ade*'. Eevee può occludere soltanto nel raggio di una distanza, settata a priori, da un oggetto. Nella Figura 3.3 si può notare come nel render effettuato con Cycles i lati del cubo abbiano un AO con un ampio gradiente di grigio, e le superfici in cui si trovano le scimmie *Suzanne* abbiano invece poco gradiente: la combinazione di transizioni sia nitide che ampie tra buio e luce risulta essere molto naturale. Nel render effettuato con Eevee invece, in entrambe le aree i gradienti risultano essere identici, facendo apparire piatti (*flat*) i lati del cubo, mentre l'occlusione causata dalle due Suzanne risulta essere eccessivamente scura.

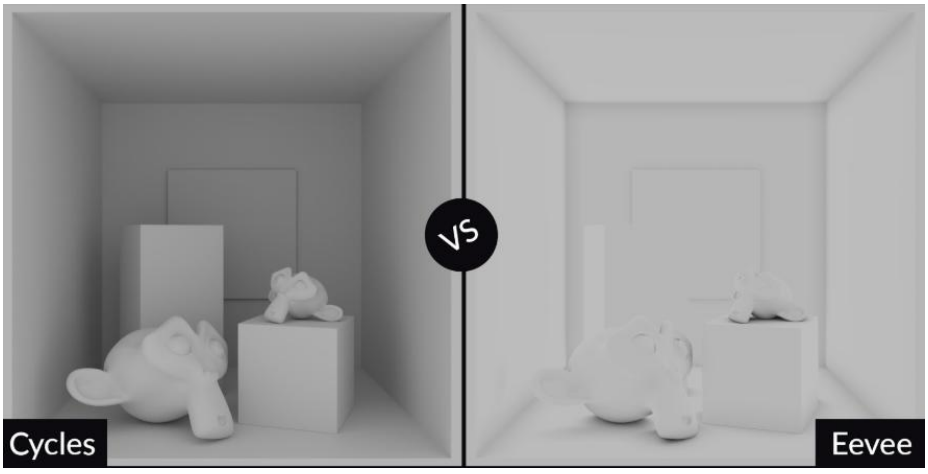


Figura 4.3: Ambient Occlusion di Eevee a confronto con Cycles.

Global Illumination

L'illuminazione Globale di Cycles (ovvero tutti i processi di '*indirect lighting*' e '*radiosity*') viene calcolata esattamente nel modo in cui funziona fisicamente il fenomeno: nei percorsi compiuti dalla luce vengono raccolti dei dati, come il colore che è già stato assorbito, nei punti in cui i raggi rimbalzano, e queste informazioni vengono usate per determinare cosa fare quando il raggio colpisce la superficie successiva.

Eevee sicuramente non sarebbe in grado di ottenere tali informazioni di rimbalzo, quindi utilizza quelle che vengono chiamate '*lights probes*' per approssimarle, delle vere e proprie sonde luminose; esse catturano informazioni istantanee della scena dal loro punto di vista nello spazio e proiettano i colori appropriati sugli oggetti nelle loro vicinanze: questo processo è chiamato *baking* e i dati vengono così memorizzati e non aggiornati ogni frame. Per questo motivo non conviene applicare il bake di un indirect lighting ad oggetti che dovranno poi essere animati.

Si potrebbe ottenere un risultato simile a quello di Cycles, se si introducesse un alto numero di light probes, ma questo causerebbe un elevato tempo di baking.

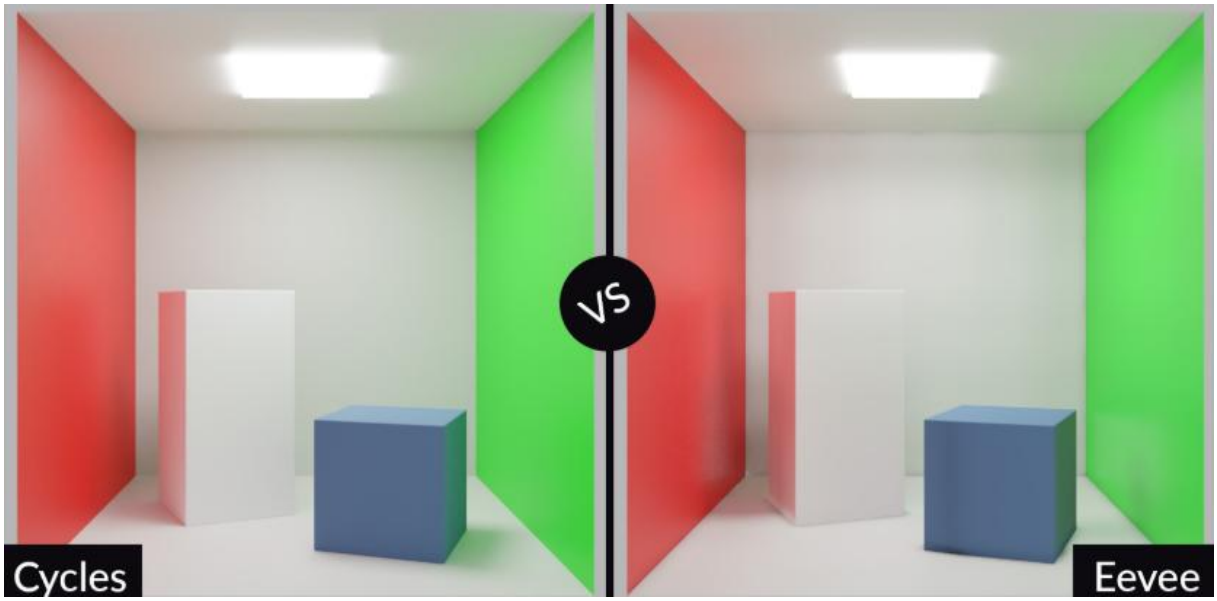


Figura 4.4: Global illumination di Eevee a confronto con Cycles.

Reflection

Come si può evincere fino ad adesso, qualsiasi caratteristica di Cycles si basa sui rimbalzi che compiono i raggi di luce all'interno dello spazio tridimensionale, dunque anche per quanto riguarda la riflessione i risultati sono ottimi. Eevee usa un paio di trucchi per cercare di imitare il più possibile il fenomeno fisico della riflessione, anche se ovviamente non danno un risultato preciso; le riflessioni vengono create prendendo l'immagine risultante dalla camera e 'ruotandola': questo rappresenta sicuramente un calcolo veloce, ma la maggior parte delle volte è artefice di un risultato comunque gradevole, nonostante non possa funzionare per qualsiasi cosa non vista dalla camera come la parte posteriore o inferiore degli oggetti. Esistono anche in questo caso, dei probes per la riflessione, che sono sostanzialmente cubi o sfere che agiscono come telecamere virtuali: in poche parole essi renderizzano ciò che li circonda e quindi mappano quell'immagine sull'oggetto.

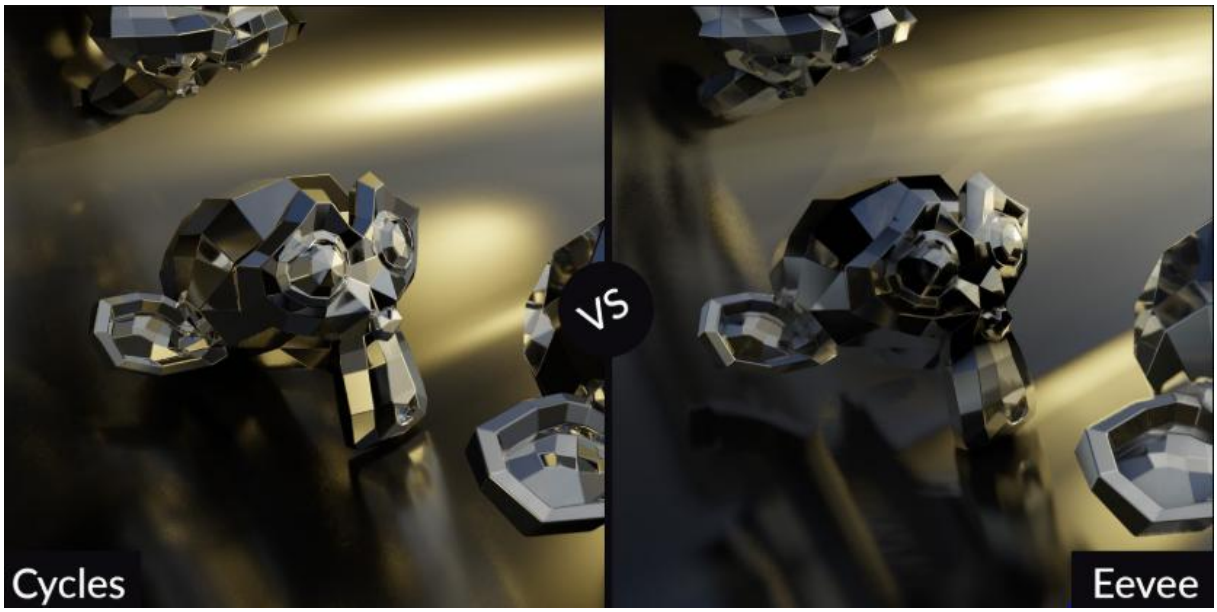


Figura 4.5: Reflection di Eevee a confronto con Cycles.

Refraction

La rifrazione in Eevee viene calcolata deformando ciò che si trova dietro l'oggetto o ciò che viene visto da una mappa cubica in base alle sue normali e al suo spessore (*thickness*). questo significa che la luce non rimbalza all'interno dell'oggetto, come effettivamente avviene in Cycles, e non si ha nemmeno l'informazione completa sulla quantità di luce entrante da qualsiasi punto dell'oggetto; ciò può dare risultati fantastici per rappresentare una sfera di vetro, od una bolla, o ancora una pozza d'acqua; ma non si può dire lo stesso quando si tratta di oggetti complessi, come si può facilmente notare dalla Figura 3.6.

Figura 4.6: Refraction di Eevee a confronto con Cycles.

Shadows

Le ombre in Eevee vengono calcolate essenzialmente catturando un'immagine di ciò che ogni sorgente luminosa 'vede', e controllando successivamente ogni pixel per scoprire se è incluso in quella luce o meno: maggiore è la risoluzione delle ombre, maggiore sarà il numero di pixel che devono essere controllati e quindi più accurato sarà il calcolo ed il risultato. Impostando poi le ombre come *soft* e attivando le ombre di contatto (*contact shadows*) si aiuta a compensare i soliti limiti visivi delle tecniche di mappatura dell'ombra, anche se i rimbalzi di luce considerati da Cycles assicurano un risultato molto più naturale.

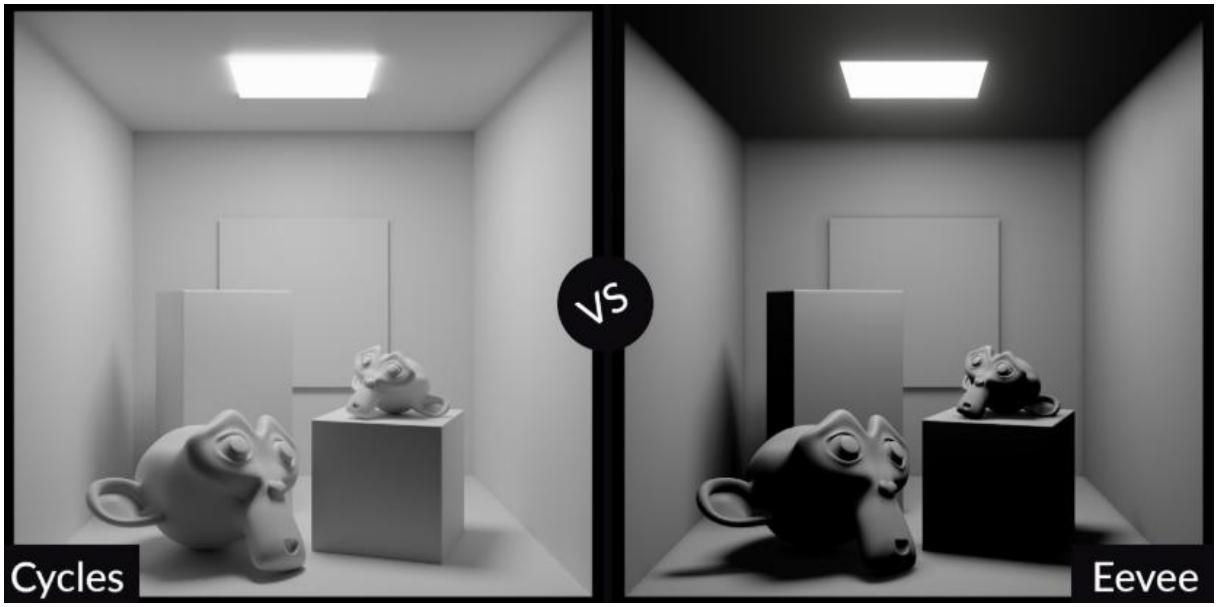


Figura 4.7: Ombree di Eevee a confronto con Cycles.

Transparency

Entrambi Cycles ed Eevee supportano materiali trasparenti, nonostante sia qualcosa che qualsiasi motore di render non ama particolarmente gestire. La trasparenza in Cycles è limitata dal numero di rimbalzi della luce, i quali se superati, porteranno ad una visualizzazione del materiale come nero pure, quindi come puramente trasparente.

In Eevee è necessario settare un impostazione specifica del materiale se si vuole ottenere un materiale trasparente: *Alpha Blend* funziona sovrapponendo i materiali uno sopra all'altro, da quello dietro a quello davanti, con un limite specificato; superato tale limite non verranno più disegnate e sovrapposte facce. *Alpha Clip* invece, stratifica i materiali in modo opposto, da quello davanti a quello dietro, risultando nel complesso un aspetto più gradevole e nitido; il compromesso in questo caso è che il materiale è completamente opaco o completamente trasparente e non si possono gestire fusioni tra i due. La trasparenza tende anche ad aumentare man mano che ci si allontana con la vista dall'oggetto su cui è applicata; in Figura 3.8 si può notare che l'albero nella parte posteriore mostra chiaramente i suoi rami attraverso le foglie, anche se dovrebbero essere quasi completamente occlusi.

Figura 4.8: Transparency di Eevee a confronto con Cycles.

Subsurface Scattering

Eevee utilizza le informazioni relative allo spessore di un oggetto e alle sue normali per approssimare la dispersione della luce sotto la propria superficie. Ciò risulta ottimo se usato in piccole quantità, per rappresentare per esempio il materiale della pelle. Non va invece troppo d'accordo con le superfici troppo sottili e i dettagli nitidi, problema presente meno in Cycles. Nell'immagine 3.9 si nota nel render effettuato con Eevee, mancanza di definizione specialmente nella parte relativa alla bocca del drago.

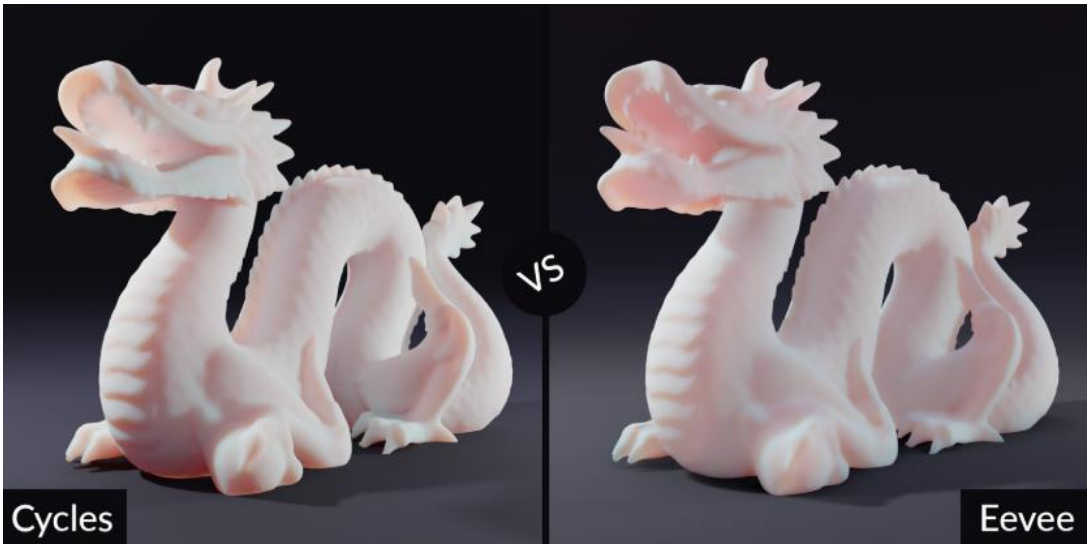


Figura 4.9: Subsurface scattering di Eevee a confronto con Cycles.

Volumetric Light

La feature in cui Eevee supera di gran lunga Cycles è rappresentata dal calcolo delle luci volumetriche: esse vengono simulate valutando tutti gli oggetti volumetrici all'interno del frustum visivo, ottenendo un risultato quasi completamente privo di rumore; è possibile aggiungere anche texture 3D agli oggetti volumetrici rendendo l'effetto ancora più interessante.

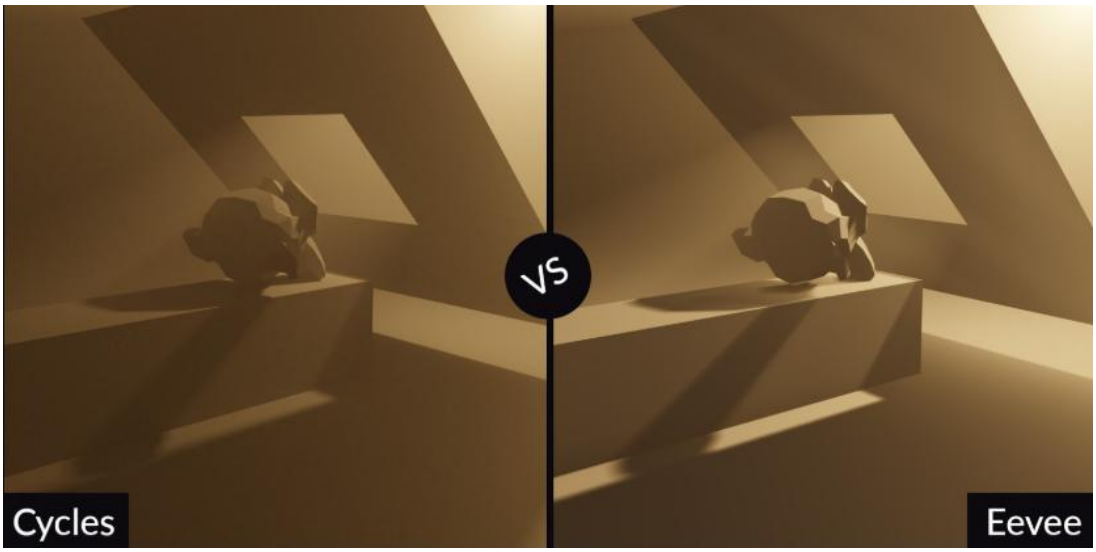


Figura 4.10: Volumetrico di Eevee a confronto con Cycles.

Hair

Il sistema particellare Hair in Eevee, per la simulazione di capelli e pelo, è un altra novità strabiliante all'interno del software, garantendo un risultato così realistico da non sembrare quasi renderizzato in real time; le limitazioni entrano in gioco quando si considerano le ombre che un Hair proietta sugli oggetti circostanti, e la mancanza di dispersione attraverso gruppi di ciocche che si otterrebbe usando il Principled Hair BSDF già citato di Cycles. In Figura 3.11 si può notare infatti le differenze delle ombre sulla fronte e la gradazione di colore lungo i capelli.

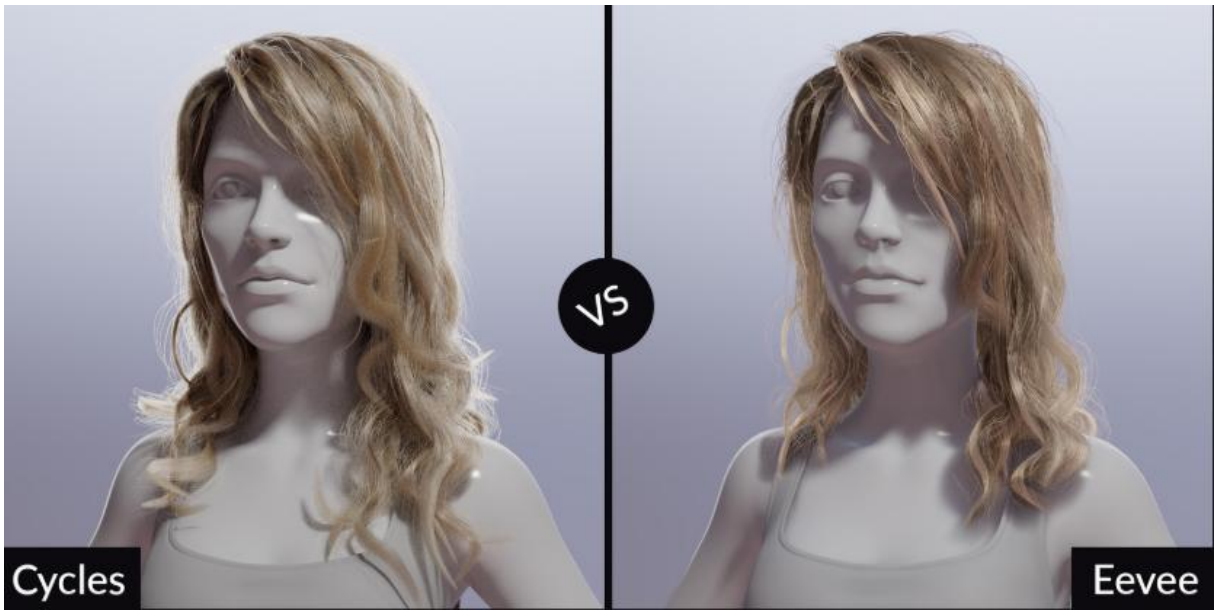


Figura 4.11: Hair di Eevee a confronto con Cycles.

Motion Blur

Mentre Cycles supporta qualsiasi tipo di *motion blur,* Eevee al momento supporta soltanto quello della fotocamera: gli oggetti in movimento o la deformazione delle armature non avranno alcun effetto in termini di motion blur. Si tratta di una limitazione che non è intrinseca alla rasterizzazione, quindi è probabile che venga aggiornata in futuro; dopotutto anche per Cycles esistevano molti limiti all'inizio per questa feature, dunque ci si aspetta anche per Eevee miglioramenti futuri. Il motion blur non è un problema banale infatti, e la sua implementazione può richiedere parecchio tempo.

Depth of field

Eevee supporta la profondità di campo con risultati gradevoli; tuttavia è ancora calcolata in modo approssimato rispetto a Cycles, che è fisicamente più preciso. La differenza si percepisce nella transizione tra aree nitide e sfocate e nella distorsione anamorfica. Ma nella maggior parte dei casi, la profondità di campo di Eevee funzionerà benissimo, e sarà molto veloce da calcolare in termini di render. Sono previsti inoltre grandi miglioramenti su questa feature in Eevee nelle prossime versioni di Blender 2.81 e 2.82, che dovrebbero rendere la depth of field più simile e comparabile a Cycles.

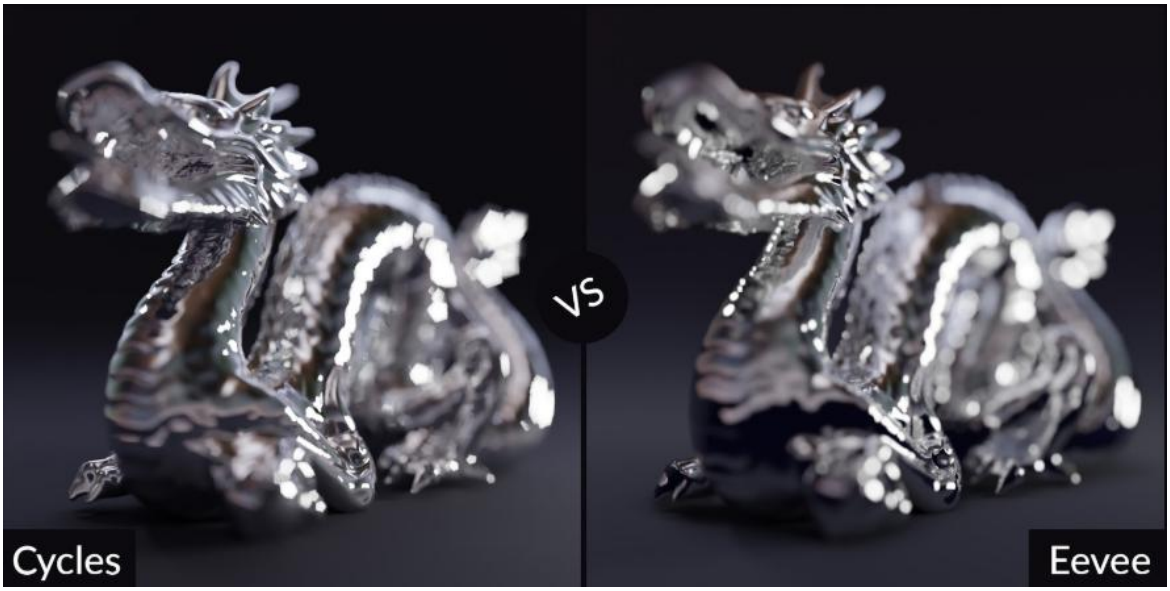


Figura 4.12: Profondità di campo di Eevee a confronto con Cycles.

5. Produzioni ‘Major’ vs produzioni ‘Indie’

Di seguito verranno analizzate alcune differenze tra il workflow e l’organizzazione all’interno di una grande produzione, come quella delle aziende leader o ‘Major’, e quello all’interno di una produzione più piccola ed indipendente, ‘Indie’ appunto, essendo l’azienda per cui è stata svolta la tesi descrivibile come quest’ultima.

5.1 Pipeline e personale

Le differenze tra uno studio di produzione major ed uno indie si riflettono tutte all’interno delle rispettive pipeline di produzione. Bisogna considerare che molti studi indipendenti o appunto indie, provengono da realtà nuove e giovani, a volte nate come società o start-up: il risultato è un ambiente di lavoro di dimensioni ridotte, popolato da un numero contenuto di persone. Le *major* sono invece suddivise in sedi, in settori e poi ancora in dipartimenti, contano un numero molto elevato di dipendenti e di collaboratori esterni: è naturale che si sviluppino pipeline molto differenti tra loro: in uno studio major esse sono frazionate maggiormente e le figure lavorative richieste hanno mansioni molto specifiche. Puntano a sfruttare al massimo la specializzazione dei dipendenti, che solitamente hanno ruoli molto precisi e definiti, inoltre nel loro dipartimento possono avere più gradi di specializzazione: *entry level*, *stagisti*, *tesisti* e *junior*, per passare poi a posizioni regular, e successivamente *senior*, *supervisor* e capo dipartimento. Negli studi *indie* invece, tutto è molto più ‘compresso’ e capita spesso che una stessa persona debba gestire più mansioni, questo anche perchè minori sono i dipartimenti e di questi solitamente esistono solo i principali e non quelli specifici; chi lavora in uno studio indie può spesso godere di un’esperienza e di una conoscenza di diverse parti della pipeline e può quindi gestire più mansioni e spostarsi da una all’altra in base alle esigenze della produzione in corso. Questi studi infatti possono gestire progetti di diverso tipo e spesso lavorano per più settori, solitamente i dipendenti sono più flessibili al cambiamento ed esistono meno livelli di

classificazione professionale. Uno studio di grande dimensioni tende invece a specializzarsi nel settore produttivo di appartenenza per spingerlo ad un livello superiore.
La differenza più sostanziale però, tra un'azienda Major ed una Indie, si situa sicuramente nei fondi d'investimento a disposizione: la grossa discrepanza di budget tra queste due realtà, crea infatti dinamiche di mercato molto differenti. Per esempio, uno studio indipendente è più probabile che lavori per una serie di clienti, piuttosto che investa su un progetto personale; questo non significa però che uno studio indie non possa decidere di investire su un prodotto inedito, ma per farlo è necessario senza dubbio cercare investitori, sponsor, occasione di crowdfunding o partecipare a bandi e concorsi.

Si è deciso di dedicare un paragrafo a puntualizzare la differenza tra i due tipi di produzione, poichè punto fondamentale e cruciale per descrivere il lavoro svolto all'interno di questo progetto: *Robin Studio* può essere definita a tutti gli effetti un'azienda *Indie,* e il team di lavoro a cui ho fatto parte era organizzato tramite un workflow ed una pipeline non sempre lineari, essendo *Reverie Dawnfall* un progetto sperimentale il cui stile diventava sempre più corroborato in corso d'opera, e in cui ogni persona non ha una vera e propria specializzazione per un unico determinato dipartimento, ma può appunto ricoprire diverse mansioni in base alle esigenze del lavoro e del teeam di produzione.

6. Reverie Dawnfall

6.1 Robin studio

Robin Studio è un giovane studio di produzione con sede a Torino, fondato nel 2016 dal professore del Politecnico, nonchè relatore di questa tesi, *Riccardo Antonino.* La peculiarità di Robin è senza dubbio il fatto che ogni singola persona che vi lavora ha frequentato o frequenta il corso di *Ingegneria del Cinema e dei Mezzi di Comunicazione*, compresi i fondatori stessi; l'idea di fondo era infatti quella di dare voce e visibilità alla figura professionale di Ingegnere del Cinema, dimostrandone la capacità di creare *'robe incredibili'*.

"Siamo 15 giovani esploratori che lavorano in una sinergia di competenze e idee nel campo della video produzione, animazione, brand identity e i nuovi media, sviluppando strategie integrate che rispondono ai processi e le esigenze di innovazione a livello tecnologico e della comunicazione dei contesti culturali-territoriali, dell'intrattenimento e dell'impresa."

6.2 Che cos'è Reverie Dawnfall?

Reverie Dawnfall è una serie di animazione ideata, pensata e autoprodotta da *Riccardo Antonino* iniziata insieme al progetto di tesi di due miei colleghi, *Melissa Coarezza* e *Michele Cannata,* e portata avanti dal nuovo team di produzione: tale serie raccoglie tutti gli sforzi finora compiuti per raggiungere una produzione di alto livello con le tecnologie più all'avanguardia accessibili per il mercato indipendente.
L'obbiettivo che si vuole raggiungere è la realizzazione di una *seconda versione* del teaser trailer della serie, tramite sperimentazione ed analisi sulla resa degli ambienti 3D e dei personaggi protagonisti, avendo a disposizione come punto di partenza la prima versione del teaser, precedentemente realizzato da Coarezza.

Reverie Dawnfall è ambientata in un futuro distopico, dove l'umanità è ridotta allo stremo da guerre biochimiche e dalle condizioni ambientali ormai degenere. In questa particolare ambientazione si mescolano elementi futuristici e cyberpunk, caratterizzando tutti i personaggi e le loro storylines in maniera unica. L'idea di questa ambiziosa serie nasce dai classici della cultura pop giapponese contemporanea, come *Il Mistero della Pietra Azzurra* (1991)*, Ergo Proxy* (2006)*, Battle Angel Alita* (1991) e altri simili di stampo più occidentale come *Saga* (2012), ed è stata sviluppata per un target principalmente *teen-adult*. All'interno della storia si svolgono diverse dinamiche che spingono ad una profonda riflessione su diverse tematiche, alcune delle quali molto vicine al nostro 2019 : l'ecosistema al collasso, il genere umano pesantemente mutato dalle radiazioni lasciate dalla guerra e il desiderio innato di riscoprire il passato sono solo alcuni pezzi di una storia che promette di essere avvincente e adatta per un pubblico maturo.

Attualmente l'arco narrativo è composto da tre stagioni di una decina di episodi ciascuno, e vedono come protagonista una giovane e brillante studentessa di entomologia di nome ***Nadya Sinkamen***. La sua storia è principalmente ambientata in una città protetta da una cupola gigante, *Dome City,* mentre al suo esterno l'aria è tossica ed il mare abitato da un tappeto di meduse mortalmente velenose; la popolazione è alle prese con continue rivolte popolari sotto l'influenza di cinque

grandi superpotenze che coincidono e si riflettono in multinazionali dal controllo monopolistico. In una società dove ogni essere umano soffre di qualche malattia o alterazione genetica, la protagonista crede di soffrire di *sinestesia e iperestesia,* condizione che la porta ad avere visioni di un mondo simile a quello che conosce ma vivo e luminoso: nel momento in cui queste visioni si fanno più nitide e insistenti, con attacchi violenti che la dissociano completamente dalla realtà, Nadya inizierà a dubitare del proprio universo, e con l'aiuto dei suoi amici intraprenderà un viaggio pieno di pericoli e imprevisti alla ricerca di un mezzo per salvare il loro mondo.

Il progetto nasce in un ambiente totalmente indipendente: di fatto al momento è auto-prodotto da Riccardo Antonino, professore di *Effetti Speciali* al Politecnico di Torino e uno dei fondatori di *Robin Studio*, presso il quale si svolge la produzione. Questa serie nasce quindi dal forte desiderio di realizzare un prodotto multimediale animato innovativo e fortemente competitivo nel panorama televisivo nazionale, con l'aspirazione di espandersi anche al di fuori del confine del nostro Paese.

6.3 Descrizione dei personaggi

Di seguito si vogliono presentare brevemente i due personaggi che verranno mostrati all'interno della seconda versione del teaser trailer e che insieme ad altri characters animeranno le avventure di *Reverie Dawnfall;* verranno descritti brevemente il loro background e le loro caratteristiche principali così come delineati dallo scrittore della serie animata *Mark Gore*, e mostrandone il profilo grafico così come disegnato dal nostro concept artist *Edoardo Audino*.
I due personaggi in questione sono la protagonista della serie, *Nadya Sinkamen* e la sua migliore amica *Jameela Rani*.

6.3.1 Nadya Sinkamen

"Figlia di due dipendenti della Pharmacopia (una delle cinque superpotenze) residenti nella seconda città più grande del Paese, Nadya ha 19 anni e studia entomologia comportamentale all'università di Dome City. Presenta un carattere curioso e tenace, velato da un cinico distacco, accompagnato da una nota di sarcasmo, ed è profondamente legata ai suoi amici. Soffre di sinestesia e iperestesia, una condizione che la porta ad avere vere e proprie allucinazioni psichedeliche: per questo possiede alla base della nuca un impianto biomeccanico che le consente di iniettare la sua dose giornaliera di farmaco e tenere a bada il suo male, per il quale lei prova una forte paura. È sempre accompagnata dal suo fedele compagno a sei zampe Alep, un insetto simile a uno scarabeo ma alto circa venti centimetri e in grado di camminare in posizione semi-eretta, geneticamente creato da lei".

Nadya è il personaggio su cui si è maggiormente lavorato fino ad oggi, non solo perché si tratta della protagonista ma perché in quanto tale è la prima a comparire sullo schermo, gli esperimenti grafici e di animazione dei personaggi sono stati effettuati su di lei, in modo da stabilire un processo di creazione uniforme per tutti i characters futuri.

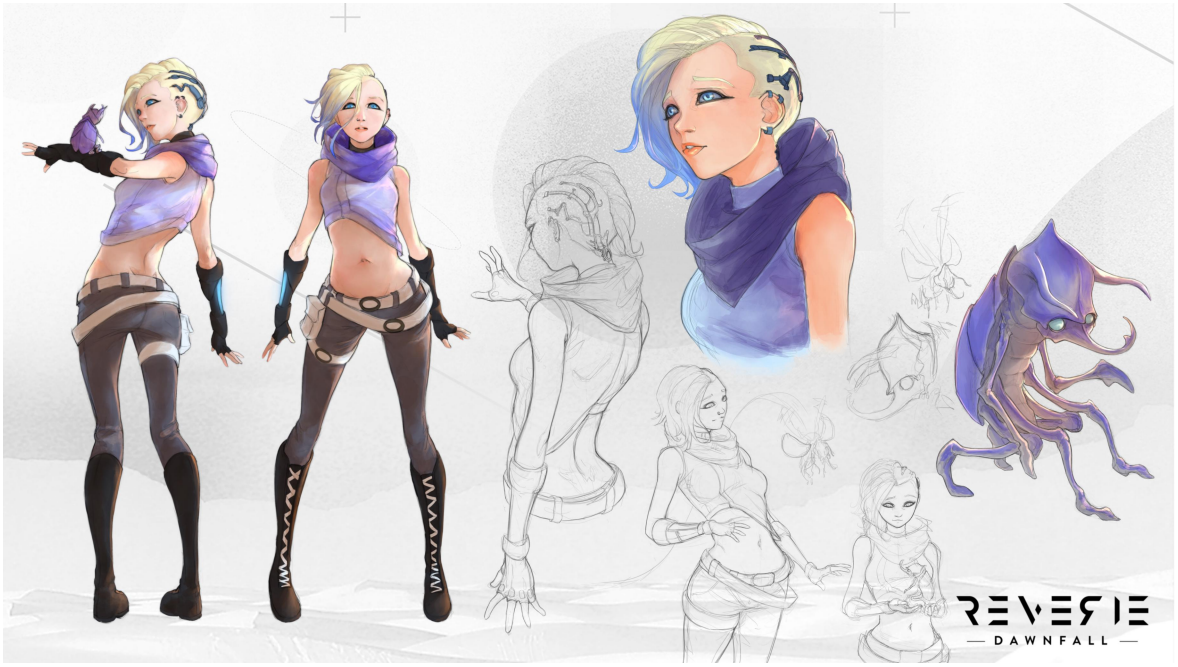


Figura 5.1: Character Design di *Nadya* e di *Alep*

6.3.2 Jameela Rani

"Jameela Rani è la migliore amica di Nadya di un anno più giovane e sua compagna di stanza all'università. Ha un carattere molto pragmatico e intuitivo, seppur mostri un'indole rassegnata e disfattista che rasenta un certo nichilismo. Nata con un solo arto sano, quello del braccio sinistro, Jameela ha origini molto umili, e non potendosi permettere delle vere protesi cyborg si costruisce gambe e braccia con tutte le componenti meccaniche che riesce a riciclare: parti robotiche, elettrodomestici, circuiti, fanno tutti parte del suo corredo che subisce continui cambiamenti a causa dei materiali di fortuna con il quale vengono assemblati. In confronto a una Nadya dalla pelle quasi diafana, lei ha la carnagione abbronzata sulla quale risaltano i suoi occhi verdi e i capelli castano rossiccio".

Figura 5.2: Character design di *Jameela*

6.4 Ambientazione, Dome City

Reverie Dawnfall è una serie ambientata su un esopianeta simile alla Terra come morfologia e caratteristiche, ma con la differenza sostanziale di assenza del moto di rotazione, un elemento che rende perennemente notturna una metà del pianeta, e perennemente giornaliera l'altra metà, esattamente come il nostro satellite. La linea di confine delle due facce vive in un crepuscolo immutabile, e questo particolare momento della giornata detta i colori e lo stile dell'illuminazione globale dell'intera serie animata.

La città protagonista di Reverie, *Dome City*, si trova sul margine estremo della zona crepuscolare, e sorge su una baia denominata *Poisoned Seas,* abitata da meduse velenosissime. Una cupola trasparente protegge poi la città e ne preserva l'atmosfera vivibile in un contesto altamente inquinato e tossico in cui si trova il modo esterno: la conformazione della città quindi si erge secondo la cupola che la protegge, con gli edifici più alti verso il centro e via via sempre più bassi verso i confini della cupola.

"Dome City è demograficamente la quarta città più grande della nazione, e ospita al suo interno la più antica università del pianeta poichè la città era stata strategicamente costruita sul confine con l'eterna notte per studiarne la flora e la fauna. Grazie alla sua ubicazione e al suo polo culturale, è uno dei centri economici di riferimento per il pianeta. L'architettura che si presenta è mista, distribuite radialmente all'interno della cupola: gli anelli più esterni ricordano una metropoli asiatica dei giorni nostri, mentre avvicinandosi al centro si trovano i grattacieli più futuristici e irregolari. L'ambiente esterno circostante è simile a una tundra fatta di basse colline erbose e con poche conifere sparse così come lo sono i laghi. La vegetazione è di svariati colori porporini e muschiosi, e ci sono dei laghi acidi che ricordano molto quelli del parco di Yellowstone". Tutti questi elementi sono posti in modo da portare sempre lo spettatore a chiedersi se si stia trovando effettivamente sul nostro pianeta e quali siano le cause che l'hanno portato a mostrarsi come lo vede dipinto.

Figura 5.4: Enviroment design di Dome City vista dall'esterno.

7. Teaser Trailer

L'obbiettivo proposto dall'azienda *Robin S.r.l.s.* è la realizzazione di una seconda versione del teaser trailer della serie animata *Reverie Dawnfall;* la prima versione del teaser fu realizzata tra il 2017 e il 2018, come progetto di tesi dei miei precedenti colleghi *Melissa Coarezza* e *Michele Cannata*.

Questa seconda versione del teaser avrebbe la finalità di presentare e caratterizzare meglio la serie animata, studiando nuove moderne possibili soluzioni per uno stile che si avvicini molto al risultato finale ricercato dall'azienda. Con questo scopo, in fase di preproduzione, si è dovuti partire quasi da zero per riproporre dei concept nuovi, più in linea con l'idea definitiva del prodotto e allo stesso tempo più unici per il mercato concorrente nazionale e internazionale.

7.1 Teaser Trailer Versione 1

Per la prima versione del trailer fu scelta una narrazione in stile **v-log**, in cui la protagonista, *Nadya*, raccontasse tutto ciò che accade nella sua vita e all'interno di *Dome City*; in questo teaser l'unico ambiente presentato è la stanza del dormitorio universitario di *Nadya* e *Jameela*, location appunto del monologo di *Nadya*.

Di seguito vengono presentati in sintesi gli strumenti ed i software che vennero studiati ed utilizzati per la realizzazione di questa prima versione di teaser, in parte mantenuti nella produzione della seconda da noi realizzata:

- *Smartsuit Pro* per la motion capture del corpo;
- *Autodesk Maya*, come ambiente di lavoro durante la pipeline 3D;
- *Blender Cycles* come motore di render;
- *Advanced Skeleton*, uno script MEL usato per il rigging del viso in Maya;
- *Faceware Analyzer e Retargeter* per il tracking e il retargeting facciale;
- *Adobe After Effects* per il compositing.

Figura 7.1: Frame preso dalla prima versione del teaser trailer di *Reverie Dawnfall.*

7.2 Teaser Trailer Versione 2

La seconda versione del teaser, non ha soltanto lo scopo di migliorare i contenuti già esistenti raffinandone lo stile e le tecniche, rigenerando e modernizzando la resa della prima versione, ma ha anche e soprattutto uno scopo narrativo nuovo e diverso: se prima si era scelto di far parlare la protagonista in prima persona per raccontarsi e presentare al pubblico lo scenario in cui è ambientata la serie, ora invece a parlare sono solo le immagini. L'idea nasce dalla necessità di sperimentare nuovi stili per trovare quello che più si avvicina all'immaginario dell'ideatore, cercando di introdurre il progetto nel mercato nazionale e internazionale: si è quindi deciso di creare nuovi contenuti, oltre a ritoccare quelli precedenti, dando vita a nuovi scenari e ambienti suscitando maggiore interesse allo spettatore.

Il teaser mostra la protagonista che, a partire dalla stanza del dormitorio universitario che condivide con la sua amica Jameela, è intenta in una corsa agitata per raggiungere la serra universitaria: durante il tragitto attraverso le strette strade cittadine di *Dome City,* Nadya è soggetta a delle allucinazioni dovute alla sinestesia e iperestesia di cui soffre che raggiungono il loro apice nel momento in cui la ragazza fa ingresso alla

serra: il luogo in cui viene a trovarsi, non è un ampio ambiente umido popolato da piante di ogni specie e colore, bensì lo spazio cosmico come noi lo conosciamo; davanti a lei, un uomo con un'ingombrante tuta da astronauta che fluttua nel vuoto.

7.3 Teaser 1 VS Teaser 2: differenze sostanziali

A primo impatto risulta subito evidente la prima divergenza fondamentale tra le due versioni del teaser, divergenza dettata da una scelta di narrativa: il numero di ambienti 3D presentati, più numerosi e vari all'interno di questa nuova versione.
Se nella prima versione veniva mostrata soltanto la stanza del dormitorio universitario della protagonista e dell'amica, ora nel secondo teaser vediamo esposti 4 ambienti: la stanza di Nadya, una strada cittadina di *Dome City,* l'esterno della serra e un ultimo ambiente che raffigura il nostro spazio cosmico, il *void*, il vuoto celestiale.

In quanto responsabile dei modelli 3D degli enviroments, nei prossimi capitoli verranno analizzati nel dettaglio i singoli ambienti che mi sono stati commissionati dai produttori, con le relative problematiche che ne sono seguite e le scelte adottate invece per risolverle.

La seconda differenza sostanziale sta nella scelta del motore di render grafico adottato per il teaser: si è deciso di abbandonare momentaneamente *Cycles*, per sperimentare e testare, come già detto e spiegato nei capitoli precedenti, il nuovo motore di render Real Time *Eevee*, sempre proprio di Blender. Anche in questo caso verranno esaminati i vari problemi avuti durante la produzione del teaser con questo innovativo motore di render, e le soluzioni trovate e adottate per superarli.

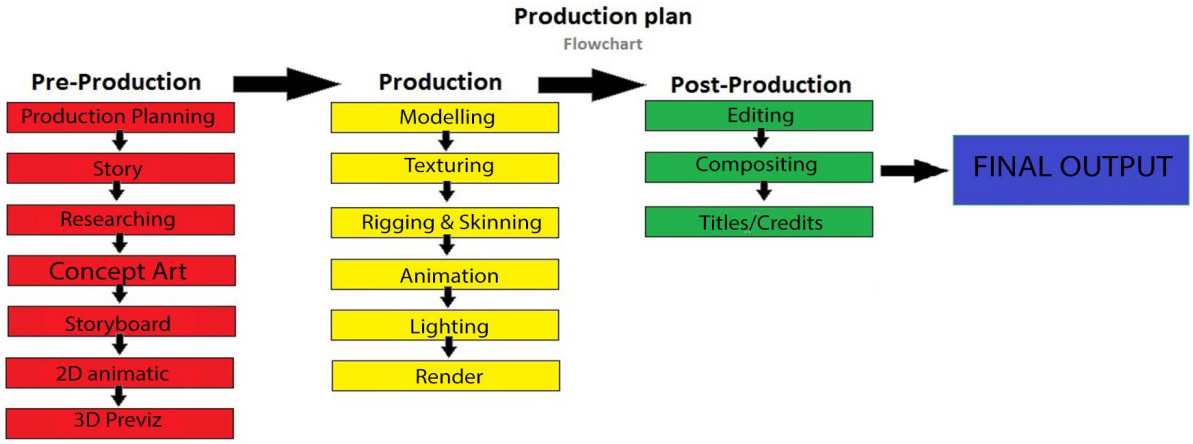
8. Pre-produzione e produzione

Reverie Dawnfall, come si è già più volte dichiarato, è un progetto la cui idea nasce nel 2017, dunque vi era già un background di materiale di partenza su cui poter lavorare per raggiungere gli obbiettvi richiesti, essendo stato il primo teaser trailer soggetto di tesi di due miei colleghi. In particolare, i concepts dei personaggi protagonisti, creati dal nostro grafico *Edoardo Audino,* non sono stati più modificati in quanto ritenuti soddisfacenti dalla produzione. Per quanto riguarda invece gli environments, lo stile e lo shading invece, si è dovuti partire nuovamente da zero e compiere quindi un nuovo lavoro di ricerca e di sviluppo, caratterizzante di solito la fase di *pre-produzione*.

8.1 Il caso particolare di Reverie Dawnfall

Generalmente, un prodotto di animazione 3D passa attraverso diversi specifici processi prima di poter essere definito concluso e vendibile, processi che vengono raggruppati e divisi in tre fasi principali: la *pre-produzione,* fase in cui si esplicita al meglio quello che si vuole ottenere come risultato finale, che rappresenta quindi la fase concettuale del prodotto finito in cui dunque viene concepita l'idea e resa solida; la *produzione,* che rappresenta la vera e propria fase esecutiva del prodotto in cui il lavoro entra nel vivo, fase in cui vengono realizzati i contenuti, possiamo dire la fase del 'point of no return', in cui tutto ciò che è stato definito precedentemente in fase di pre-produzione deve essere rispettato e svolto restando fedele agli scadenziari definiti anteriormente; e infine la fase di *post-produzione,* composta da una serie di differenti processi riguardanti sia la parte grafica che quella sonora al fine di migliorarne nel complesso la resa; essa rappresenta, in sostanza, l'ultima fase prima della distribuzione al pubblico del prodotto finale.

Di seguito viene mostrato uno schema generico dei diversi steps che compongono la *production plan* per la realizzazione di un film di animazione 3D:



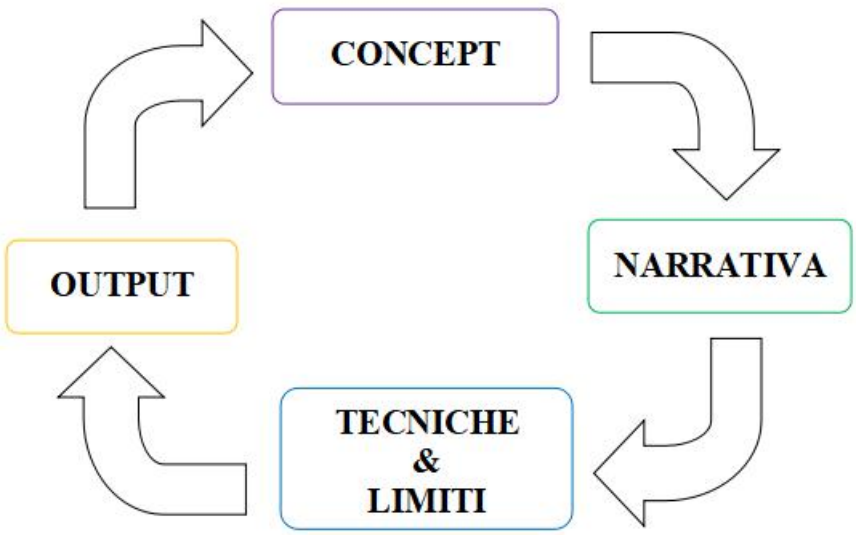
Come si può evincere dallo schema riportato sopra, se all'interno di ognuna delle fasi produttive la pipeline che ne organizza il flusso di lavoro può non essere per forza di cose lineare, pre-produzione, produzione e post-produzione rappresentano invece tre fasi distinte raramente miscibili tra loro: alla prima segue la seconda, conclusa quest'ultima si passa alla terza.

Nel nostro caso però, si può affermare non esista una vera e propria separazione tra le tre principali fasi che caratterizzano la buona riuscita del prodotto animato: essendo Reverie Dawnfall, come più volte già dichiarato, una serie animata che ha nella sperimentazione il suo ingrediente principale e caratteristico, è stato praticamente impossibile sancire con precisione e accuratezza lo stile desiderato del prodotto finale, elemento fondamentale per concludere la normale fase di pre-produzione, ma piuttosto si è proseguiti a tentativi di gusto grafico, specialmente per quanto riguarda lo shading e il lighting e alcuni elementi della modellazione degli environments: dato uno specifico concept art degli elementi che compongono la narrativa, ci si è impegnati a realizzarne il corrispettivo modello 3D con prove sperimentali di materiali e shading, cercando di risolvere in maniera più opportuna i problemi dovuti ai limiti e alle tecniche utilizzate, producendo come output diverse versioni di render

fino alla versione definitiva, che racchiude la sintesi di tutte le modifiche e gli esperimenti apportati dal team di produzione.
Si può dire quindi che nel caso studiato, *pre-produzione* e *produzione* fanno parte di un *unico processo*, poichè le due fasi si influenzano in modo sostanziale a tal punto da non poter essere propriamente considerate distinte; questo è inoltre dovuto al fatto che il team di produzione non è formato da professionisti specifici in un unico ambito, e dalla natura *indie* dell'azienda, che in questo determinato progetto lavora a zero budget.

Ad esempio, in corso d'opera della modellazione della stanza del dormitorio universitario di Nadya, ci si è accorti che una planimetria quadrata dell'ambiente come era stata pensata e quindi inizialmente realizzata, non soddisfava appieno il gusto personale della produzione, poichè rappresentava qualcosa di 'già visto' all'interno della concorrenza nazionale ed internazionale: è stato quindi necessario fare passi indietro e ripensare ad un concept della stanza più funzionale ed originale che permettesse di caratterizzare e rendere unica la serie animata.

Se si vuole quindi sintetizzare in maniera generica uno schema del **workflow** adottato per la realizzazione degli ambienti e dello stile di questo teaser, il diagramma di seguito riportato descrive in modo schematico e sintetico come ci siamo dovuti muovere per ottenere un risultato finale:



Questo continuo iterare fino ad ottenere il risultato voluto, è soprattutto dovuto alle *tecniche e ai limiti* dei software che sono stati utilizzati nella realizzazione del teaser: infatti questi ultimi influenzano molto la narrativa, e allo stesso tempo la narrativa va ad influenzare le tecniche che si scelgono di adoperare.
La natura ciclica di questo processo organizzativo di produzione è nata anche dalla necessità di ottimizzazione dei tempi: dovendo infatti sperimentare il più possibile, è risultato obbligatorio e fondamentale cercare di produrre output diversi, a seconda delle richieste dell'azienda, nel minor tempo supponibile.

8.2 Concept e Reference per lo stile grafico

Si è già espresso nei paragrafi precedenti quanto non fosse esattamente chiara l'idea di stile che si voleva ottenere per la serie: la cosa certa, è che si desiderava una resa grafica nuova agli occhi dello spettatore, qualcosa di non ancora visto all'interno della concorrenza nazionale.
Fin dalle prime fasi del progetto, quelle della produzione della prima versione del teaser, si è compiuto uno studio stilistico, volgendo lo sguardo ai lavori di artisti come l'irlandese *David OReilly* e al suo videoclip animato per il gruppo musicale *U2* sulla canzone *'I'll go crazy if you don't go crazy tonight',* ritenuta un'interessante reference per uno stile poligonale non troppo complesso.

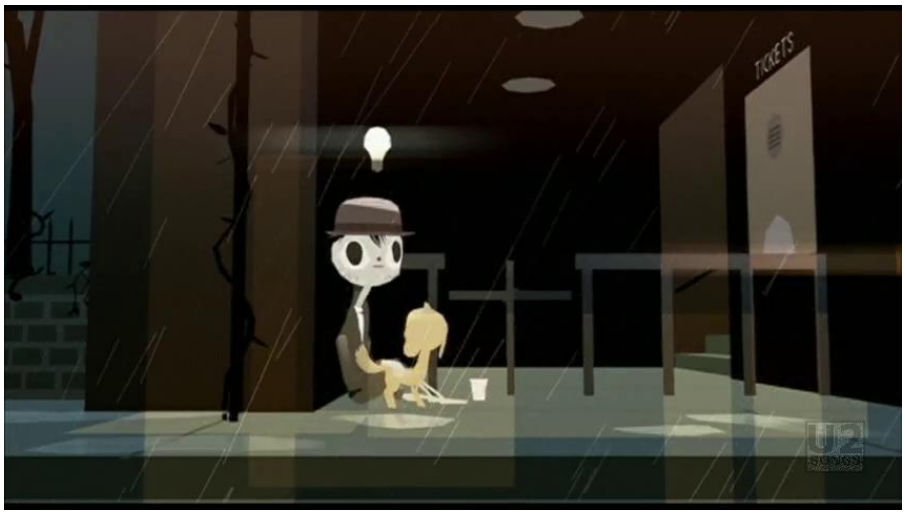


Figura 8.1: Frame preso dal videoclip degli U2, '*I'll go crazy if I don't go crazy tonight'*

Un’altro importante riferimento riguardo lo stile grafico è rappresentato sicuramente dal successo tutto italiano di *‘Gatta Cenerentola’*, film di animazione datato 2017 della casa di produzione *MAD Entertainment* e soggetto a 7 nomination ai David di Donatello del 2018.

Di *‘Gatta Cenerentola’* si è cercato di imitare lo stile dell’illuminazione, principalmente volumetrica, e soprattutto il suo particolare stile grafico che sembra quasi un cartonato, con disegni colorati ad acquarello: il primo compito svolto da chi si è ufficialmente occupato di realizzare uno shader personalizzato per la nostra serie, il mio compagno e collega *Giacomo Balma*, è stato infatti quello di cercare di riprodurre sulle superfici poligonali dei personaggi e degli ambienti, un materiale che simulasse delle pennellate di colore, tipiche dei quadri e dei disegni 2D.

È significante anche notare e puntualizzare che tale film d’animazione è stato realizzato principalmente su *Blender*, software di modellazione ed animazione 3D scelto anche dalla nostra produzione.

Figura 8.2: Alcune immagini tratte dal film animato *Gatta Cenerentola, MAD Entertainment.*

Ma è con l'uscita del film d'animazione statunitense *'Spider-man: into the spider verse*', che l'idea sullo stile grafico da apportare al progetto incomincia via via a farsi più chiara: vincitore nel 2019 dell'*Oscar* e del *Golden Globe per il miglior film d'animazione* , questo lungometraggio animato racchiude un miscuglio unico di stili di animazione e stili grafici molto intriganti, che lo hanno reso sicuramente uno dei prodotti animati più interessanti degli ultimi anni. Uno degli stili artistici utilizzato che si può notare in ogni scena del film è il *Ben-Day Dots,* una tecnica di stampa degli anni '60 che utilizza un pattern di punti per conferire colore all'immagine, utilizzata con voga nei fumetti, che ha ispirato tra le varie cose i lavori del famoso artista pop art *Roy Lichtenstein*.

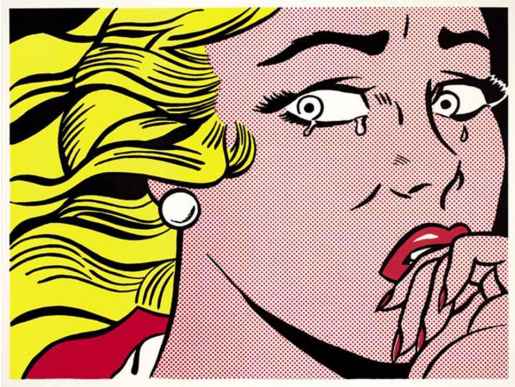


Figura 8.3: Lavoro di Roy Lichtenstein

Il risultato finale è infatti un film d'animazione che ricorda molto nello stile grafico un fumetto, frutto degli artisti che hanno contribuito alla realizzazione del film, artisti diversi con stili e tecniche altrettanto diverse. Di seguito alcune immagini prese dal film *'Spider-man: into the spider verse'* in cui si può notare l'uso dei dots accompagnato da quello di linee oblique sulle ombre.

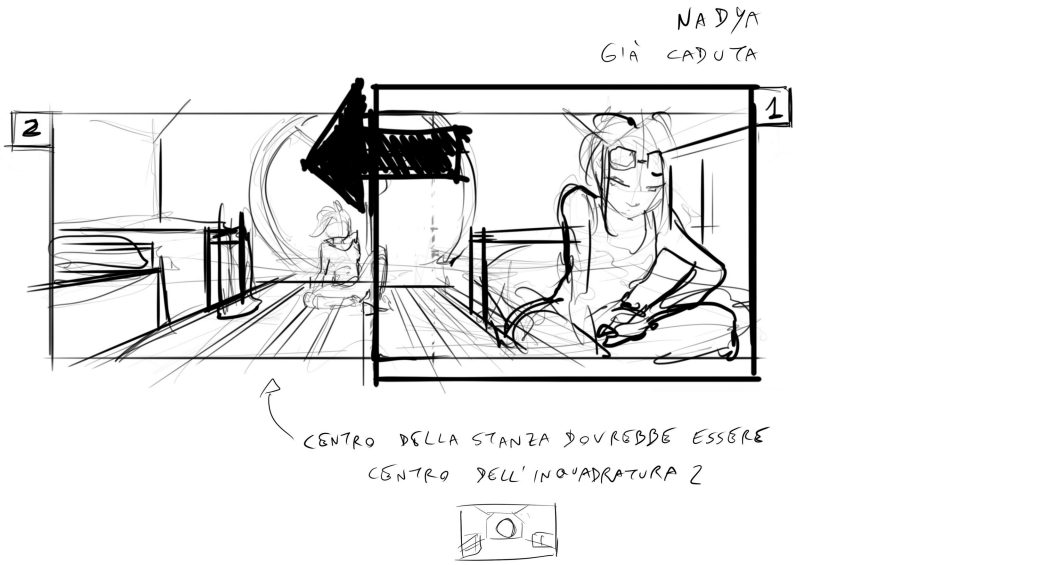
Lo stile grafico che si è cercato di realizzare ed ottenere è quindi una fusione dei due stili sopra indicati: si è tentato di trovare un compromesso tra i due creando uno shader che simulasse, laddove richiesto, pennellate di colore simile ad acquarelli, uniti a pattern di linee, punti ed esagoni per rappresentare luci ed ombre.

Figura 8.4: *Spider-man: into the spider verse*, maggiore reference per lo stile grafico di *Reverie Dawnfall.*

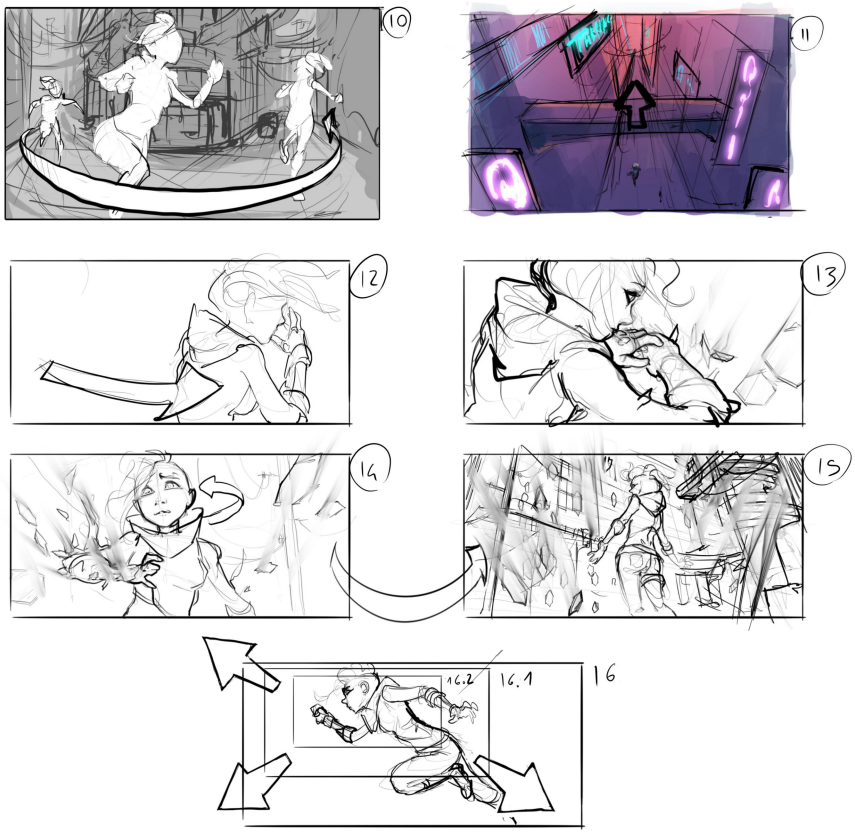
8.3 La narrativa

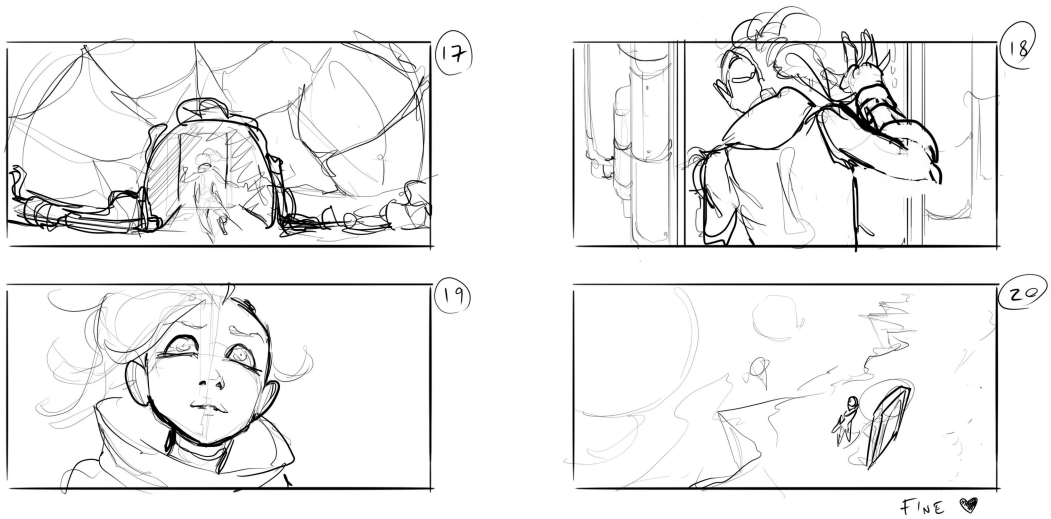
Come già anticipato nei paragrafi precedenti, l'intento di questo teaser è in primo luogo la sperimentazione, sia essa stilistica o tecnica; sin da subito, si è deciso quindi di abbandonare l'idea di fondo che vi era dietro alla prima versione del teaser del 2018, dove veniva mostrata la protagonista davanti alla webcam del suo computer parlare e raccontarsi, introducendo lo spettatore nello strano mondo di Reverie Dawnfall; per questa prima versione del teaser, si era dovuta creare una sceneggiatura ad hoc, traducibile poi con il solo monologo di Nadya. Ora l'intento della produzione con questo secondo teaser trailer a livello narrativo è sicuramente quello di immergere maggiormente il pubblico nella storia di Dome City, dandogli un primo assaggio di quello che potrebbe essere il risultato finale del progetto, scegliendo quindi di far parlare le immagini piuttosto che i personaggi protagonisti.
Tenendo a mente questi propositi, si è deciso di creare l'idea del nuovo teaser partendo dalla sceneggiatura dell'episodio pilota, scritto dallo sceneggiatore della serie *Mark Gore:* la possibilità di realizzare e mettere in scena proprio l'intero pilot si è mostrata da subito un'impresa impossibile per molte ragioni, come per esempio la mancanza di fondi e la natura low budget del prodotto stesso, l'insufficienza del personale artistico, di numero limitato e in secondo luogo non professionista essendo noi del team tutti studenti universitari, e ultimo ma non meno importante è sicuramente la variabile temporale, essendo infatti nostro volere e nostra necessità far uscire un contenuto presentabile al pubblico nel minor tempo plausibile. Si è decretato quindi di scremare il più possibile la sceneggiatura dell'episodio pilota, rappresentandone soltanto i contenuti più salienti e quelli che meglio riuscivano, a nostro parere, a caratterizzare la serie animata: ecco quindi che il video mostra la protagonista che, nel tragitto dal dormitorio universitario in cui vive fino alla serra botanica della città, è soggetta a delle visioni ed allucinazioni tipiche della sua condizione di sinestesia e iperestesia.

Di seguito verrà riportata una bozza dello storyboard con realativi appunti utili alla produzione realizzato come linea guida per la realizzazione degli shot che compongono il teaser video.


NADYA
GIÀ CADUTA
2
1
CENTRO DELLA STANZA DOVREBBE ESSERE
CENTRO DELL'INQUADRATURA 2


INQ 1-2
Nadya inquadrata che si sta rialzando
(FIG. INTERA)
MEZZA
FIGURA su Nadya (3)
CONTRO
CAMPO, PP su Nadya da dietro (4)
SI TOCCA
DISPOSITIVO IN
TESTA
3
4
MEZZA
FIGURA
5
6
WTF?!
7
8
9
CONT
CONT
TAGLIO SU JAMEELA (NADIA DA
SX VERSO
DX)


10
11
12
13
14
15
16.2
16.1
16


17
18
19
20
FINE

9. Analisi degli ambienti 3D

Il mio compito e la mia responsabilità in questo potenziale progetto ha riguardato la creazione degli ambienti 3D rappresentati nel teaser trailer avvalendomi del software Blender per la modellazione; l'ultima versione di questo software open source, la 2.80, mi ha permesso di risolvere problematiche tecniche altrimenti irrisolubili con la sua non più recente versione 2.79.

Si è già parlato nella prima parte di questo documento riguardo le principali caratteristiche che differenziano la versione 2.79 con la 2.80 di Blender e del motivo fondamentale per cui è stato scelto Eevee come motore di render per la realizzazione di questo progetto; di seguito verrano esaminate altre caratteristiche della 2.80 che hanno permesso di agevolare il lavoro sulla modellazione di alcuni elementi degli enviroments: verranno quindi esaminati tutti gli ambienti protagonisti del teaser con relative problematiche e risoluzioni trovate. Gli enviroments verranno presentati in ordine di apparizione all'interno del teaser trailer: STANZA UNIVERSITARIA, STRADA DI DOME CITY, GREENHOUSE, VOID COSMICO.

9.1 La stanza universitaria

*"La **STANZA DI NADYA** nel dormitorio universitario ha una conformazione e una disposizione degli arredi come nella figura di reference, con un secondo letto sul lato sinistro dell'immagine. Sul soffitto vi è una botola per accedere ai condotti d'aerazione (quella da cui cade Nadya nell'episodio 01). La stanza è condivisa da due ragazze, una appassionata di robotica e una di entomologia: gli oggetti sparpagliati riflettono questo, quindi saranno parti meccaniche e elettroniche, dodecaedri trasparenti (servono a contenere gli insetti da gara), un paio di capi di vestiario. Come in figura sul bancone dovrebbero esserci un "microscopio" e un computer portatile" Mark Gore*

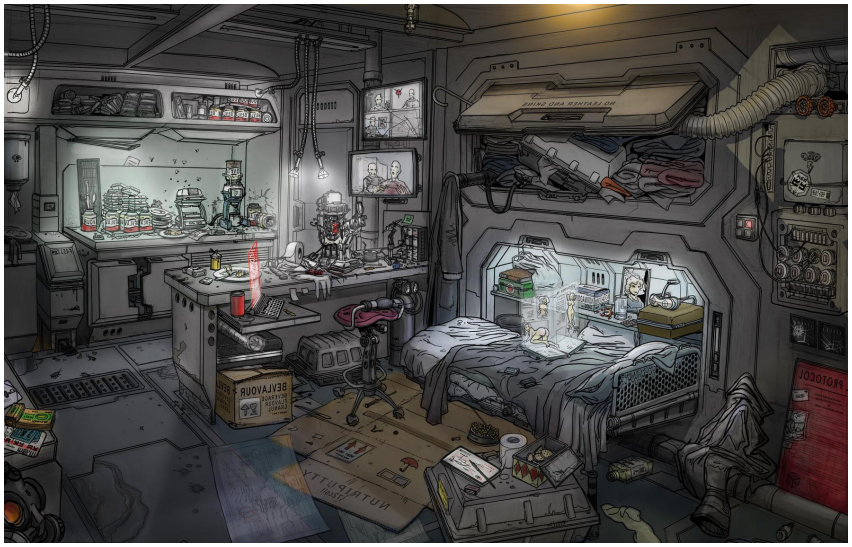


Figura 9.1: *Prima reference usata per la stanza di Nadya e Jameela.*

La stanza del dormitorio universitario di Nadya e Jameela è stato il primo environment commissionatomi dalla produzione; la scelta di iniziare con la modellazione di questo ambiente nasce soprattutto dal fatto che, come già anticipato, questo era l'unico environment di quelli che ci servivano, già presente in corso d'opera poichè scenario protagonista della prima versione del teaser, quindi ritenuto il più veloce da realizzare in tempi di produzione.

Partendo dal materiale già esistente dunque, è stato chiesto di migliorare la stanza stilisticamente e tecnicamente, andando cioè a modificare le strutture già esistenti

dell'abitazione rendendole più fluide di forma e più leggere in termini di vertici, aggiungendo oggetti caratteristici senza però eccedere nei numeri di poligoni. Cercare di creare una scena complessa ma con il minor numero possibile di vertici all'interno è l'obbiettivo che ci si pone ogni volta che si vuole realizzare un'environment che dev'essere poi usato in un prodotto animato: i tempi di render infatti, se la scena è eccessivamente ricca di poligoni risultano essere molto lunghi, andando a compromettere i ritmi di produzione, senza contare il fatto che l'avere in scena un personaggio animato è già di per sé un alto costo computazionale.
Per la stanza, ma in generale per tutti e quattro gli ambienti, mi è stato chiesto di non superare il *milione di vertici*.

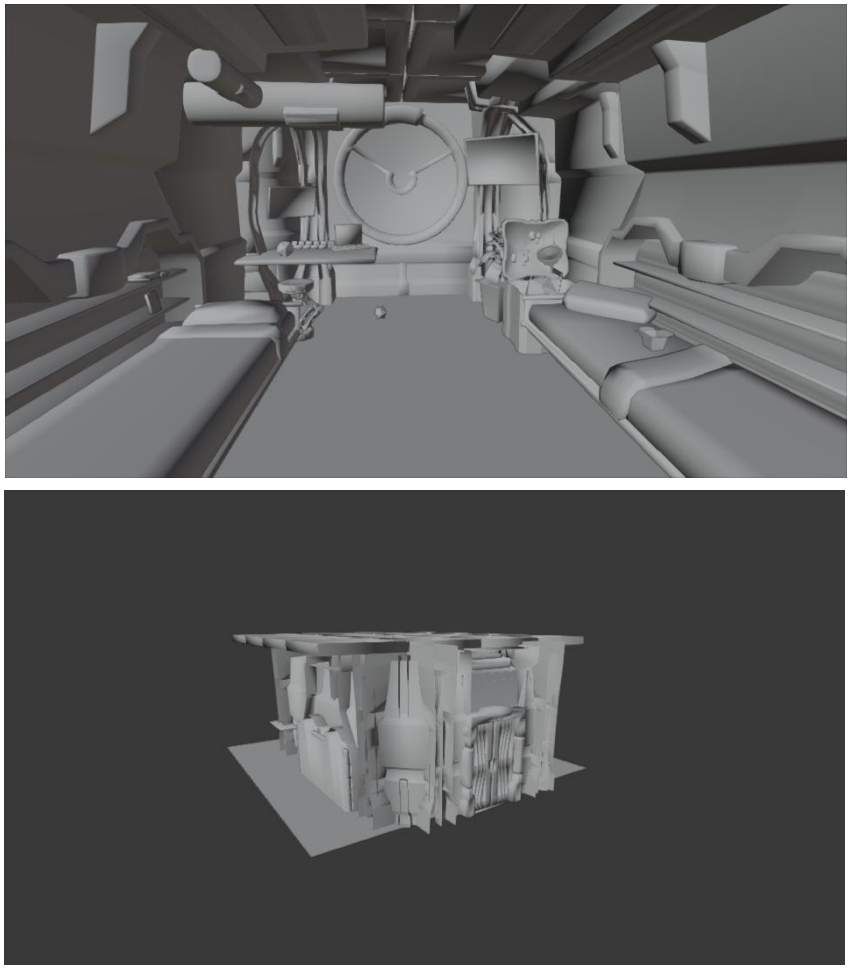


Figura 9.2: Render dell'interno e dell'esterno in *solid mode* della stanza preesistente.

In figura , è rappresentata la versione della stanza già esistente ed utilizzata per il primo teaser: come si può notare, la struttura è a semplice base rettangolare, e la conformazione esterna inoltre non era stata pensata a priori per essere parte di un unico edificio rappresentante il dormitorio universitario; per prima cosa, dopo aver reso l'intera abitazione più fluida e semplice nelle forme, ci si è subito accorti infatti che bisognasse ripensare ad una forma della struttura più caratteristica, che potesse essere replicata per rappresentare una composizione più grande quale l'intero dormitorio.

La soluzione a questo problema è risultata evidente pensando alla caratteristica propria della serie animata nel dare particolare attenzione agli insetti ed al loro mondo: proprio gli insetti infatti saranno un'importante indizio nella risoluzione del mistero che si cela dietro alle vicissitudini di Dome City, inoltre Nadya è una studentesa di entomologia, e tra le varie cose lo sport maggiormente seguito dagli abitanti del pianeta è una versione entomologica del sumo, ma tra insetti-sauri eleganti che combattono in piccoli ring; con queste considerazioni, è trasparsa subito l'idea di ripensare la maggior parte degli oggetti e degli ambienti propri di Reverie in un'*ottica 'entomologica'*, cercando ovvero di replicare e copiare il più possibile particolari caratteristiche e forme del mondo entomologico applicandole al mondo umano.

Si è quindi in definitiva optato per creare la struttura del dormitorio universitario ad **alveare**, in cui ogni cella esagonale rappresenta un'unità con all'interno due stanze, una superiore con due letti, come quella delle nostre protagoniste, ed una inferiore più povera per quattro persone: la figura riportata di seguito mostra una bozza del concept 3D dell'edificio; non è stata realizzata la modellazione completa e definitiva dell'esterno del dormitorio poichè non presente nel teaser e dunque non urgente ed essenziale ai nostri fini.
L'alveare, e in particolare l'esagono, sarà inoltre un elemento ricorrente della serie, tant'è che verrà usato addirittura come uno dei pattern per lo shader finale.

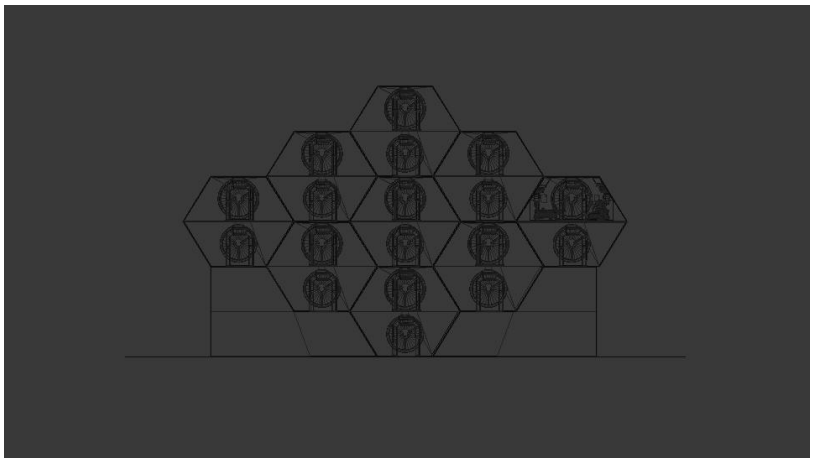


Figura 9.3: Wireframe del concept 3D per la struttura esterna del dormitorio universitario. Non essendo necessario per il teaser, non è stato ancora sviluppato il modello relativo a questo ambiente.

Una volta ripensata la conformazione dell'intero edificio, e di conseguenza quella della stanza, il nostro disegnatore nonchè concept artist *Edoardo Audino* ha subito provveduto a realizzare il nuovo concept art 2D dell'ambiente, che dopo essere stato approvato dai produttori, è passato subito nelle mie mani per essere concretizzato in un modello 3D.

Ogni elemento della scena, come le luci, i muri della stanza e gli oggetti che la popolano, sono stati raggruppati e ordinati in varie *collezioni*, novità di Blender 2.8, passaggio utile per rendere più fluido ed intuitivo il lavoro di chi, dopo di me, doveva occuparsi dell'illuminazione globale della scena e dello shading.

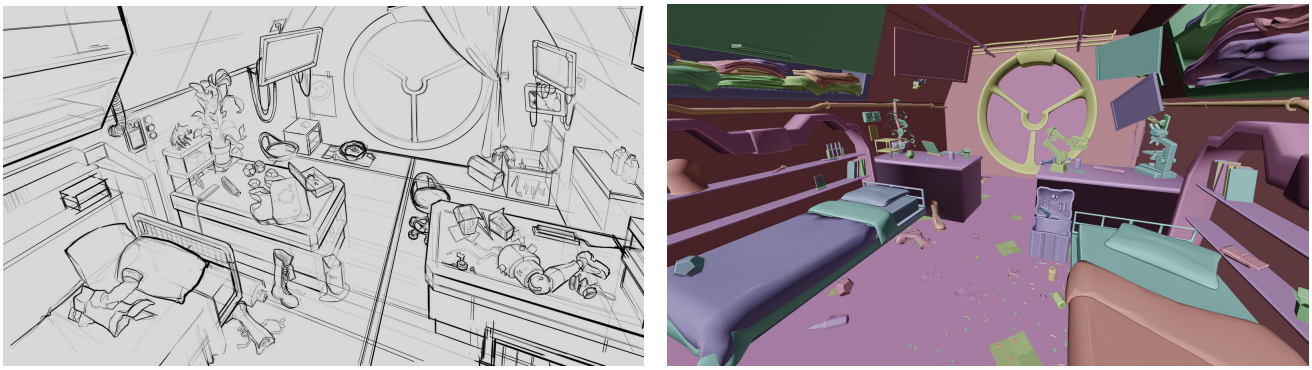


Figura 9.4: Concept art definitivo della stanza disegnato da Edoardo Audino a sinistra, realizzazione del rispettivo modello 3D in object mode a destra.

La stanza doveva essere popolata da oggetti che riflettessero al meglio il carattere e le abitudini delle due ragazze: la parte sinistra della camera è vissuta da Nadya e lo si può evincere per esempio dai tanti dodecaedri sparsi, che servono a contenere gli insetti per le gare, dall'unica pianta presente nell'ambiente che rappresenta la fonte di sostentamento di Alep, l'insetto inseparabile della protagonista, e da altri elementi che sottolineano un interesse ed un amore per il mondo entomologico; l'altra metà di stanza invece è vissuta da Jameela, come lo si può facilmente intuire grazie alla presenza di parti meccaniche e robotiche sulla scrivania e ai tanti bulloni e viti sparpagliati a terra.

La maggior parte degli oggetti è stata sottoposta ad un processo artistico di riadattamento della forma al mondo immaginario di Reverie Dawnfall: essendo la serie ambientata in un pianeta che non è la nostra Terra, e in un tempo cronologico che non coincide con il nostro, molti elementi della vita delle persone di Dome City e dell'interno pianeta, sono stati ripensati per poter essere introdotte nell'universo di Reverie e per dare ancor più originalità alla serie; un esempio è rappresentato dal personal computer presente sopra le scrivanie delle ragazze: nulla a che vedere con i tradizionali pc del nostro mondo, quelli di Reverie hanno una forma cilindrica ettagonale quando sono spenti per facilitarne il trasporto all'interno di particolari zaini, mentre in accensione desktop e keyboard si 'srotolano' dall'interno del cilindro come fossero due pergamene, rivelandone il materiale trasparente su cui viene poi proiettata la grafica.

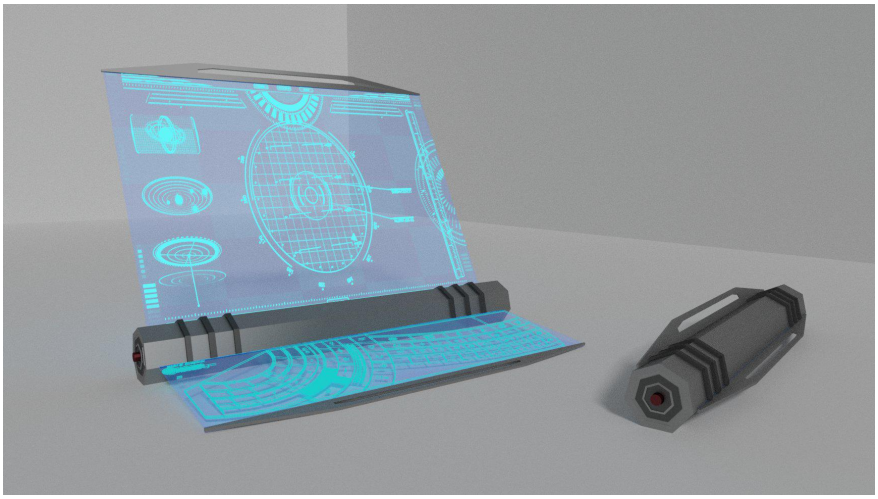


Figura 9.5: Concept 3D del pc a pergamena proprio dell'universo immaginario di Reverie.

Un'altra importante modifica che mi è stata richiesta di apportare a questo ambiente, riguarda il soffitto di quest'ultimo: l'azienda Robin desiderava a riguardo qualcosa di particolare, che non fosse comune nell'immaginario collettivo umano, che introducesse una caratteristica intrinseca e identificativa agli ambienti della serie; ancora una volta quindi, si è deciso di soddisfare questa richesta prendendo ispirazione dal mondo degli insetti, e sfruttando ancora una volta il pattern esagonale degli alveari, per creare un sistema di lampade a led ognuna avente un sensore di movimento che fa accendere nello specifico le celle esagonali soprastanti i personaggi all'interno della stanza.
Il problema principale di questo sistema di illuminazione è stato capire in che modo si potessero gestire indipendentemente tra di loro un numero indefinito di oggetti separati, rappresentati dagli esagoni del soffitto ad alveare, dovendo ogni cella esagonale accendersi nel caso in cui un personaggio entrasse nel raggio d'azione del suo sensore. Una volta modellata la shape del nuovo soffitto, creando semplicemente un esagono e servendosi dell'*array modifier* per creare una matrice NxN, ho provveduto a creare la sua intelligenza artificiale tramite script in *Python* e servendomi dell'addon di Blender *Animation Node*.

9.1.1 Animation Node

L'addon Animation Node è un sistema di scripting visivo nodale progettato per la motion graphics in Blender. L'obbiettivo principale di questo plugin è avere un framework di nodi molto personalizzabile ed estensibile: Animation Node viene usato per esempio come potente alternativa ai drivers, come manipolatore di testi, viene addirittura utilizzato come alternativa alla modellazione tradizionale, insomma i suoi utilizzi sono svariati e i risultati che si possono ottenere con questo addon sono di grande impatto visivo. Nella figura a lato ne viene mostrata l'interfaccia con l'elenco delle

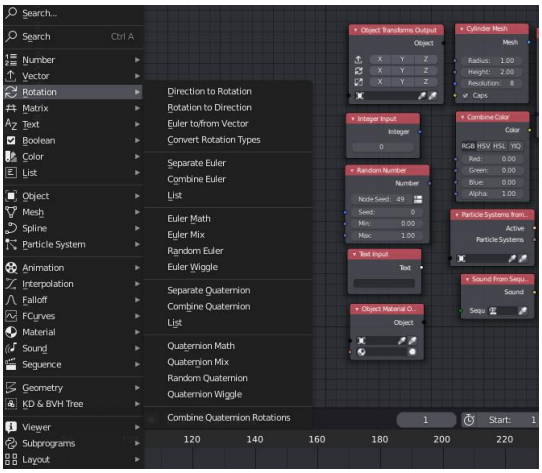


Figura 9.6

tipologie di nodi che l'addon mette a disposizione.

L'utilizzo di questo addon è risultato utile per creare l'illuminazione del soffitto della stanza, essendo questa un sistema di luci a sensore di movimento. Per prima cosa, per poter accedere alle informazioni di ogni singola cella esagonale, volendo modificarne i parametri occorrenti, è neccessario trattare ogni elemento della matrice singolarmente, come oggetto separato da tutti gli altri, quindi viene applicata la matrice ottenuta con l'array modifier e successivamente, ad ogni esagono viene impostata la propria origine come coincidente con il suo centro di massa.

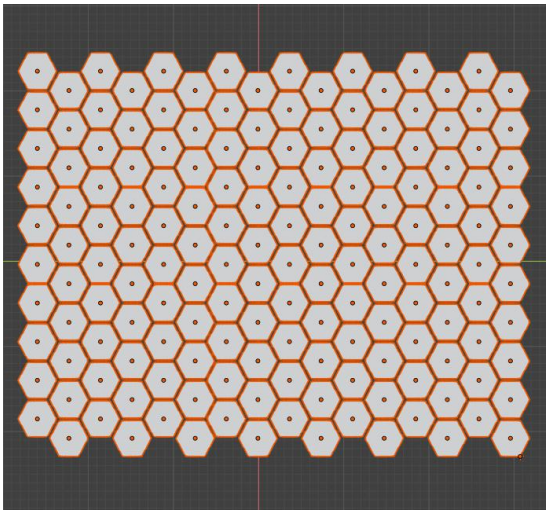


Figura 9.7

Tutti gli oggetti risultanti dovranno poi essere raggruppati in un'unica *collezione,* che denomineremo *hivecell.* Una volta creata tale collezione si può passare alla creazione dello script che andrà a gestire l'accensione di ogni singola cella esagonale: lo script in questione dovrà sostanzialmente controllare, ad ogni iterazione di tempo, la distanza che intercorre tra il personaggio presente nella stanza e ogni specifico esagono presente all'interno di una lista, *hiveCellList*, coincidente con la collezione *hivecell*; se questa non supera un certo valore di soglia, impostato a piacere secondo le necessità e l'effetto che si vuole andare a creare, tramite un ciclo *if,* lo script associa all'esagono individuato un materiale emissivo, *mat_on*, che ne indica l'accensione; in caso contrario, verrà assegnato all'oggetto un materiale *mat_off* per rappresentare il suo stato da spento.

```
import bpy
from math import sqrt

size= len(hiveCellList)
mat_on=bpy.data.materials.get("ON")
mat_off=bpy.data.materials.get("OFF")
mat_flicker=bpy.data.materials.get("Flicker")
flck = bpy.data.objects['flickerLight']
flck.hide_render = True
flck.hide_viewport = True

for i in range(0, size):
    hiveCell = hiveCellList[i]
    distance = sqrt((empty.location.x-hiveCell.location.x)**2 + (empty.location.y-hiveCell.location.y)**2)
    if distance <= 0.35:
        if hiveCell in hiveCellFlickerList:
            hiveCell.data.materials[0] = mat_flicker
            flck.location.x= hiveCell.location.x
            flck.location.y= hiveCell.location.y
            flck.hide_render= False
            flck.hide_viewport = False

        else:
            hiveCell.data.materials[0] = mat_on
    else:
        hiveCell.data.materials[0] = mat_off
```

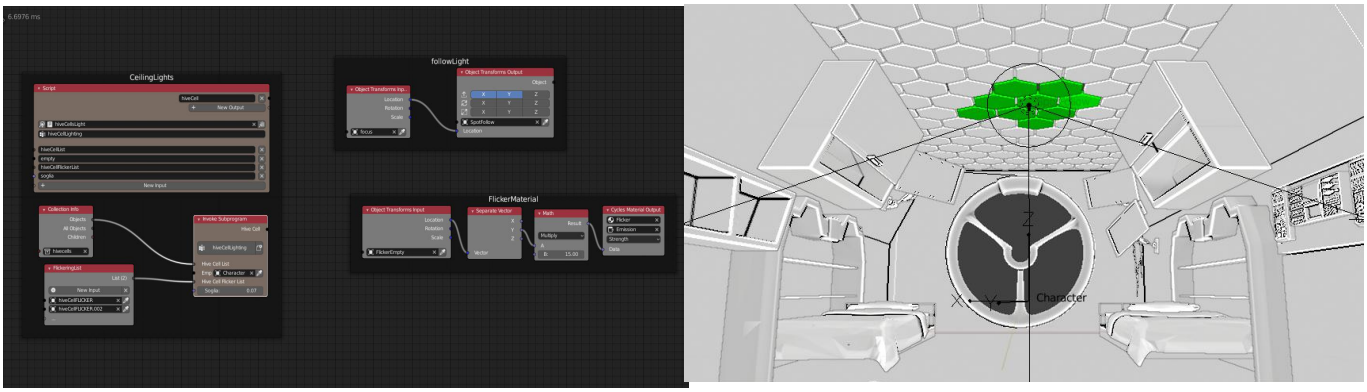
Figura 9.8

Dopo aver realizzato lo script, mi è stato chiesto di implementare un metodo per poter controllare il flicker di

alcuni led, per rendere l'effetto più suggestivo e realistico; è bastato quindi creare tramite animation node una seconda lista, denominata *hiveCellFlickerList,* popolata dai led esagonali scelti per simulare il tremolio della luce, e introdurre all'interno del già esistente ciclo if per il controllo del materiale, un ulteriore medesimo ciclo che verifichi per ogni esagono la propria presenza all'interno di *hiveCellFlickerList*, e che associ in caso positivo un terzo nuovo materiale denominato *mat_flicker.* Quest'ultimo materiale viene creato proprio con animation node andando a modificare il valore *Strength* dell'emission tramite un oggetto empty a cui viene applicato un noise sull'asse y del suo movimento: questo rappresenta a tutti gli effetti un metodo alternativo per sostituire l'uso dei driver in animazione.

Il materiale emissivo del led però, non basta per simulare l'illuminazione del soffitto della stanza: uno shader emission infatti, nel motore di render Eevee, non influenza l'illuminazione sugli altri oggetti adiacenti, e non può essere utilizzato quindi come sostitutivo ad un oggetto *Lamp;* è necessario dunque introdurre nella scena anche una *Spot Light* posta in corrispondenza agli esagoni accesi, a cui è stato assegnato cioè il materiale *mat_on*, che segua i movimenti del personaggio presente in scena tramite un constraint o usufruendo ulteriormente di animation node. Anche per l'effetto flicker è necessario introdurre un nuovo oggetto Light, che verrà attivato soltanto nel caso in cui una delle celle sovrastanti il personaggio è presente in *hiveCellFlicker* e che verrà posizionato in corrispondenza della cella esagonale; l'effetto a intermittenza di questa Spot Light infine, è stato creato tramite un driver.

Le figure sottostanti mostrano a sinistra l'albero nodale costruito per l'illuminazione del soffitto e a destra il risultato ottenuto sulla griglia di celle esagonali che lo costituiscono (le celle rappresentate in verde sono quelle accese poichè in



corrispondenza del personaggio, rappresentato idealmente da un oggetto *Empty Arrows*).

9.1.2 Limiti e problematiche: shader come soluzione

Nonostante la prima soluzione trovata tramite l'uso dell'addon Animation Node abbia soddisfatto i requisiti richiesti dall'azienda per l'illuminazione della stanza, ci si accorse subito però che il metodo trovato non fosse del tutto ottimale, in quanto richiede un alto costo computazionale per essere elaborato.
Alla base di Animation Node, infatti, vi è una conversione dell'albero nodale in uno script, il quale viene eseguito in base alle opzioni di esecuzione scelte dall'utente; di default, il plugin esegue il node tree quante più volte possibili al fine di avere un aggiornamento migliore e regolare della scena. Tuttavia questo rallenta di gran lunga la CPU e altre aree attive di Blender. Animation Node, per questo motivo, offre un sistema di esecuzione automatico e manuale in base alle esigenze necessarie del progetto. L'esecuzione automatica del node tree può essere impostata secondo un set di opzioni offerte dall'addon:

- *Always:* tale opzione è settata di default e compila l'albero nodale tante più volte possibile; per questo motivo, al fine di evitare grossi rallentamenti, non deve essere utilizzata se non è indispensabile; l'uso di questa opzione può essere necessaria laddove vi è bisogno di accedere alle informazioni di locazione di un oggetto che si muove in scena, oppure quando si eseguono simulazioni discretizzate basate sul tempo che devono essere calcolate più velocemente;
- *Tree Changed:* se attivata queata opzione, il nodetree viene eseguito tutte le volte che si apportano modifiche ad esso, per esempio aggiungendo o rimuovendo un node all'albero;
- *Frame Changed:* l'albero nodale viene eseguito ogni qualvolta cambi il frame corrente della scena, quindi quando per esempio viene avviata l'animazione;
- *Property Changed:* questa opzione fa eseguire il nodetree ogni volta che viene cambiata una proprietà (un input per esempio) dell'albero stesso;

Per i nostri scopi, volendo visualizzare in real time ciò che succede nell'albero nodale, e dovendo avere a che fare con dei vettori di posizione che mutano poichè gli oggetti associati sono in movimento, è necessario scegliere l'opzione *Always* di esecuzione.
Ma tale opzione rallenta di gran lunga le prestazioni del software poichè l'addon impiega dai **3 ai 10 ms** per elaborare ed eseguire lo script associato al node tree creato per il soffitto della stanza, senza contare che in modalità render, utilizzando Eevee come motore di render real time, non viene mostrato l'effetto del nodetree sulla matrice di esagoni, essendo probabilmente troppo pesante per essere elaborato in render mode. Questo rappresenta un tempo troppo alto, poichè rende la scena sovraccaricata e troppo pesante in tempi di render; per questo motivo ci si è impegnati a trovare un'altra soluzione più consona per risolvere il problema, cercando di ottenere lo stesso risultato grafico ma riducendo di gran lunga i tempi di esecuzione, dunque la latenza.
La soluzione a questo problema è arrivata cercando di sostituire e tradurre il nodetree di Animation Node, in un nodetree dello *shader material* molto più semplice e leggero dell primo.

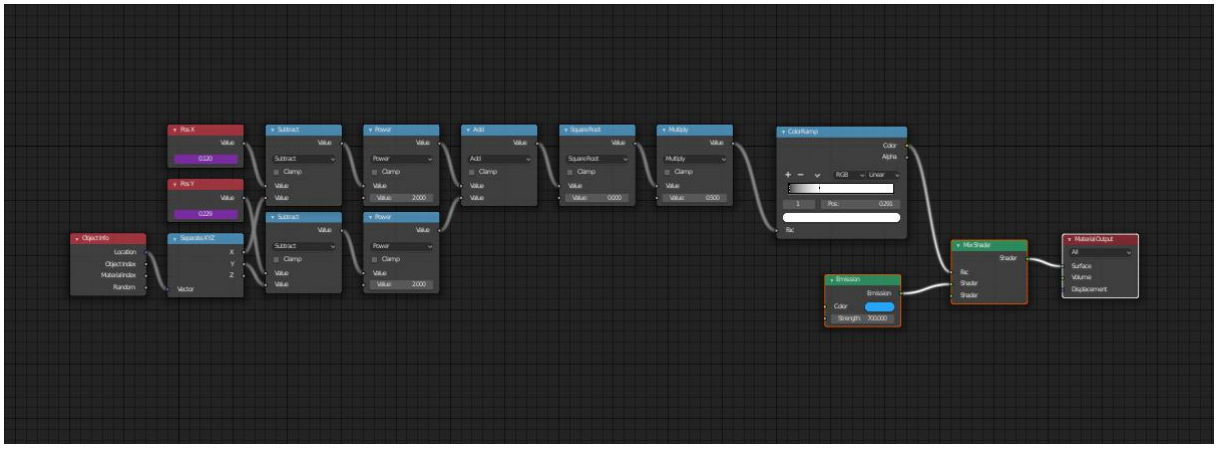


Figura 9.9: Shader Nodetree per il soffitto della stanza

Il materiale associato al nodetree rappresentato in figura, viene applicato a tutti gli esagoni componenti la matrice rappresentante il soffitto della stanza del dormitorio; il nodo *Object Info* prende le informazioni sulla posizione, espressa in vettore, di ogni esagono; tale vettore viene poi scomposto prendendo i valori delle sue componenti

sulla x e sulla y (la componente z del vettore locazione non è considerata in quanto la distanza in verticale con il character non deve influenzare l'illuminazione degli esagoni); il blocco centrale di nodi è responsabile del controllo della distanza tra i singoli esagoni e il personaggio, le cui componenti x e y del proprio vettore posizione sono passate in input grazie all'utilizzo di due drivers. L'output dell'operazione matematica risultante entra in ingresso al nodo *ColorRamp* che ne regola l'area di influenza dell'accensione dei led, quindi l'effetto grafico risultante. Si è scelto infine un colore azzurro per i led illuminati.
L'introduzione di questo nodetree e l'utilizzo quindi di uno shader per rappresentare l'intelligenza artificiale del soffitto, ha diminuito di gran lunga i tempi di elaborazione del file, e cosa più importante, l'effetto risultante è visibile in real time anche in modalità render; questo poichè si è riusciti ad evitare l'utilizzo di script esterni che, in aggiunta poi all'addon Animation Node che tramuta le operazioni nodali in ulteriori scripts, avrebbero rallentato ed appesantito a dismisura la scena.

9.1.3 Materiali e Textures

Per la scelta dei materiali da applicare ai modelli della stanza, ed in generali a tutti gli altri ambienti, si è scelto di limitare l'utilizzo di texture cercando di valorizzare lo shader creato ad hoc per la serie dal mio collega *Giacomo Balma,* e per avere un effetto meno fotorealistico e quanto più possibile cartoon.
Le uniche textures presenti in questa scena sono state applicate al pavimento della stanza ed ai muri, questo per creare più dettagli sulle superficie senza introdurre ulteriore lavoro sulla modellazione; altre texture sono state applicate naturalmente per rappresentare le immagini sui display a muro e sul desktop e keyboard dei computers portatili delle ragazze; il resto dei materiali è stato creato con un settaggio molto semplice e basilare sui nodi degli shaders.

Di seguito verranno riportate due immagini rappresentanti il risultato finale dell'ambiente con shader ed illuminazione definitivi.

Figura 9.10: Render della stanza con illuminazione e shader finale

9.2 Strada di Dome City

*"L'**ARCHITETTURA** degli edifici cittadini ricorda,nei settori più esterni alla cupola, quella di una metropoli asiatica del nostro 2019 (Seoul ad esempio), solo più avanzata. Il centro cittadino invece, è più nuovo e futuristico, con palazzi più alti e dalla forma irregolare." Mark Gore*

Figura 9.11: Prima reference usata per modellare la strada di Dome City

La strada di Dome City in cui vediamo correre Nadya all'interno del teaser trailer è stato il secondo ambiente commissionatomi dall'azienda Robin Studio. In tempi di produzione è stato l'environment più lungo da realizzare, data la complessità dei modelli dei palazzi in scena e l'assenza di un progetto preesistente da cui poter iniziare a lavorare, a differenza di come già visto per la stanza del dormitorio.
Non essendo presente alcun concept art dell'ambiente, mi è stato chiesto di realizzare i modelli degli edifici partendo da alcune reference trovate sul web, come quella in Figura 9.11, tenendo ben presente delle caratteristiche della città protagonista della serie: i palazzi di Dome City infatti, si sviluppano in altezza poichè il pianeta su cui è ambientata la serie è privo di rotazione, dunque vi è la stessa illuminazione, di alba o tramonto, in ogni ora del giorno; per questo motivo, idealmente gli abitanti della città hanno edificato le costruzioni urbane estentendole in altitudine, cercando di usufruire di quanta più luce possibile; è naturale quindi, che i piani più alti dei palazzi siano

abitati da cittadini più benestanti, in quanto possono godere di un'illuminazione naturale più forte ed estesa, mentre i piani più bassi saranno occupati da residenti più poveri.
Tale sviluppo verticale degli edifici introduce inoltre la necessità di costruire ponti o altre strutture che servano come passaggio da un palazzo all'altro, per rendere più veloce lo spostamento degli abitanti all'interno della città: ogni costruzione sarà quindi collegata a quelle adiacenti tramite passerelle o impalcature: il risultato che si vuole ottenere è una sorta di 'formicaio' umano, ricordando ancora una volta infatti come il mondo degli insetti influisca sulla vita e sulle caratteristiche di Dome City.

9.2.1 Collection Instance e Sistemi Particellari: modellazione procedurale

Essendo la scena molto complicata a livello di modellazione e di numero di poligoni, si è cercato di rendere nel complesso l'ambientazione quanto più leggera possibile al fine di rendere il lavoro più fluido e veloce: è importante ricordare infatti che più la scena è pesante, più lento sarà il software ad elaborare anche la più piccola modifica apportata. A questo scopo è risultata d'aiuto una delle novità della versione 2.8 di Blender: le *Collection Instances.* Nel menù *add,* infatti ora compare tra le opzione di aggiunta un nuovo tipo di oggetto denominato appunto Collection Instance; se selezionata, quest'opzione ti permette di scegliere la collezione da istanziare tra quelle presente nell'outline della scena. A questo punto il software raggrupperà tutti gli oggetti presenti nella collezione selezionata creando un'unica istanza e, in scena oltre agli oggetti duplicati apparirà anche un oggetto *empty*: ora è possibile duplicare quest'ultimo e tutte le impostazioni della Collection istanziata verranno conservate per ogni oggetto *empty* creato; in questo modo è possibile ottenere molto facilmente un numero idealmente illimitato di copie di dati linkati, alleggerendo di gran lunga la memoria. Questo rappresenta uno dei modi per duplicare proceduralmente oggetti direttamente da altri oggetti o, come in questo caso, da collezioni.
È importante per questo scopo cercare di ordinare al meglio le collezioni all'interno

dell'outline, oltre ad essere un passo fondamentale per rendere più facile e fluido il lavoro di illuminazione e shading della scena successivamente apportato dai miei colleghi. Basteranno dunque pochi palazzi poichè verranno istanziati in scena le loro copie proceduralmente create tramite l'utilizzo di Collection Instance, per popolare la città.

Un altro modo per popolare la scena di oggetti senza aumentarne la complessità a livello di poligoni è l'utilizzo di *sistemi particellari*.
In questa determinata ambientazione sono stati creati due interessanti sistemi particellari, entrambi di tipo *hair,* uno con lo scopo di aggiungere quanti più oggetti possibili sull'asfalto e sui marciapiedi per rappresentare il disordine e la sporcizia urbana, e l'altro per popolare la scena di insegne luminose, in stile cyberpunk con lo scopo di rendere viva la strada di Dome City.

Per quanto riguarda il primo sistema particellare, è bastato creare una collezione con all'interno gli oggetti modellati per riempire la strada, come per esempio lattine, fogli di carta, bottiglie e quant'altro; una volta esistente tale collezione, è stato sufficiente posizionare un piano lungo tutta la strada, settarlo come emitter di un Particle Hair System e, nel menù corrispondente, scegliere tra le varie opzioni di modalità render per le particelle quella *Collection:* in questo modo sul piano emitter verrano generati automaticamente tutti gli oggetti presenti nella collezione indicata, e si potrà procedere al settaggio del resto delle caratteristiche del sistema particellare, come il numero di particelle emesse, da dove emmetterle (*faces, verts, volume*), la rotazione degli oggetti ed altro ancora; si noti come ancora una volta si è sfruttata la novità delle collezioni di Blender 2.8.
Nella figura 9.12 è riportato il risultato in solid mode del particellare per il popolamento della strada con oggetti randomici applicato all'ambiente modellato.

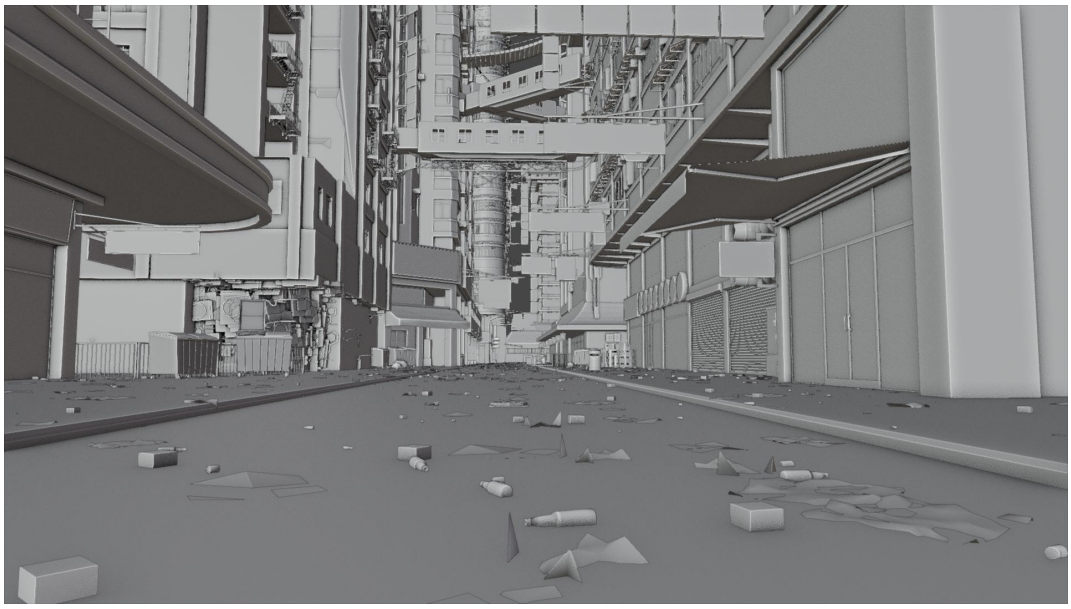


Figura 9.12: Strada di Dome City in solid mode.

Il secondo interessante uso del sistema particellare di Blender è stato realizzato per creare le insegne luminose della città ed ottenere un risultato grafico in stile cyberpunk.

Questa volta l'oggetto utilizzato come emitter per il particellare di tipo hair è un parallelepipedo contenente l'intera strada in lunghezza e in altezza; all'emitter viene impostata una sorgente di tipo volumetrico, e come oggetto renderizzato viene impostato un semplice piano la cui rotazione è settata come randomica.

Il risultato grafico ottenuto, rappresentante una strada viva e caotica, lo si deve ad un settaggio interessante dei nodi dello shader applicato al materiale del piano utilizzato come particella: su di esso viene generata in modo randomico una texture scelta tra 4 predefinite, a cui viene assegnata un colore anch'esso randomico in base al settaggio dei colori nel nodo *ColorRamp* secondo una palette prestabilita. Al tutto viene poi applicato un valore di *Emission* settato a piacere per ottenere l'effetto luminoso dei neon. In Figura 9.14 si può notare come la presenza di questo sistema particellare renda l'atmosfera più caotica e graficamente piacevole, nonostante le insegne luminose generate proceduralmente non siano esattamente in posizioni corrispondenti alla superficie dei palazzi. Importante anche specificare che nel render in questione non è presente alcun oggetto di tipo *Light*; la Figura 9.13 invece, riporta l'albero nodale creato per il materiale dei billboards.

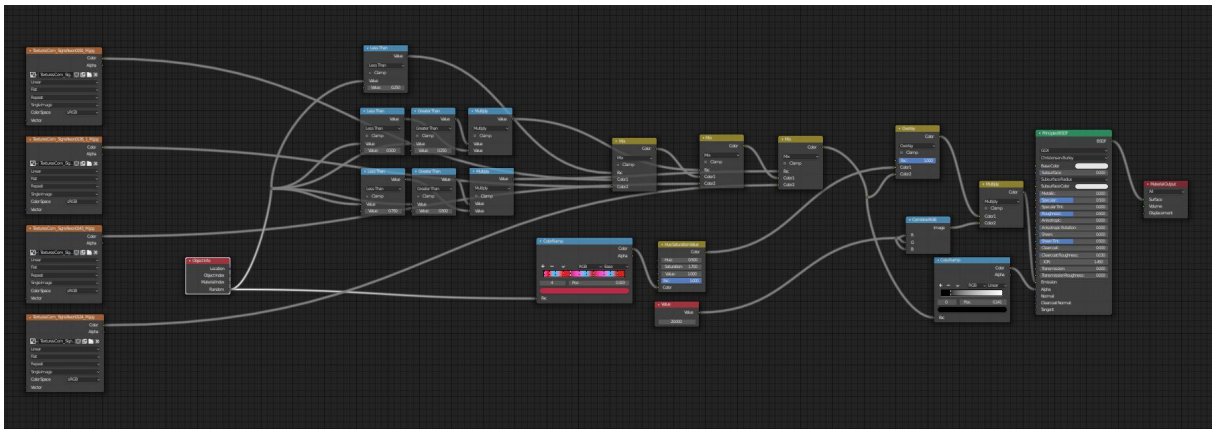


Figura 9.13: Node tree dello shader per i billboards.

Figura 9.14: Strada di Dome City in render mode con billboards procedurali; nessun settaggio dell'illuminazione.

9.2.2 Simulazione degli interni dei palazzi

L'ultimo problema rilevante di questo environment è stato trovare un modo per rendere 'vivi' i palazzi, per simulare presenza di persone al loro interno.

A questo scopo la reference grafica del film di animazione "*Spider-man into the spider verse"* è stata ancora una volta di molto aiuto: si è infatti cercato di riprodurre la tecnica usata per rappresentare gli interni dei palazzi di New York City all'interno del lungometraggio, in cui proceduralmente vengono prodotte delle pennellate di colore e altri pattern desiderati, lungo la superficie di una serie di cubi concentrici.

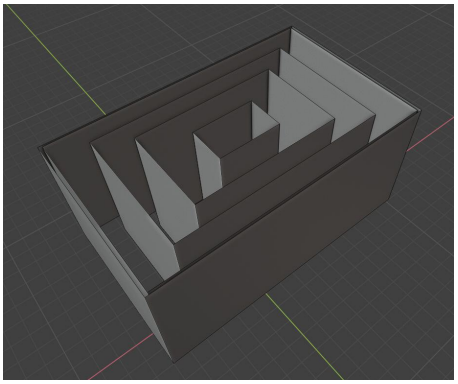


Figura 9.15

Grazie all'aiuto di Giacomo Balma è stato possibile riprodurre sinteticamente la tecnica utilizzata dal film di animazione grazie, ancora una volta, ad un ingegnoso settaggio dei nodi dello shader, ed ottenere il risultato desiderato.

Il primo step è stato appunto quello di creare un oggetto composto da una serie di cubi concentrici la cui dimensione dipende dal palazzo su cui si sta costruendo tale oggetto di simulazione di interni, su cui poi si è applicato un *Array Modifier* sull'asse z per raggiungere l'altezza della costruzione; ora è tutto un uso strategico dei materiali: al cubo più esterno viene applicato un materiale di tipo glass con un semplice settaggio del nodo *PrincipleBSDF,* questo per ottenere anche ulteriori effetti di rifrazione e riflessione; al secondo cubo invece, è stato attribuito un materiale creato con una texture di tipo *Voronoi* e con opportuni settaggi dei nodi in modo da ottenere un pattern disordinato di cerchi di vari colori, utili a simulare gli ambienti interni con le relative illuminazioni. Agli ultimi due cubi interni, infine, è stata applicata una semplice texture realizzata ad

Figura 9.16

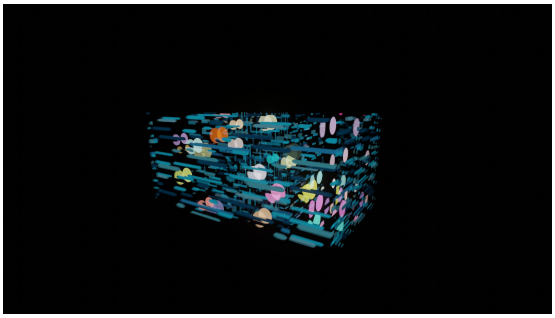


Figura 9.17

hoc sul software *Adobe Illustrator* per riprodurre l'illuminazione degli interni e aumentare il numero dei dettagli dell'effetto; per evitare di ottenere un risultato troppo statico, si è fatta dipendere la texture in questione dalla posizione dell'oggetto. In figura 9.17 è possibile vedere il risultato dei diversi materiali applicati ai cubi concentrici, mentre in figura è riportato in render il risultato complessivo della scena, ancora privo di un settaggio luci, ma comunque già molto soddisfacente anche a livello di illuminazione. Si noti quanto viva e caotica risulta essere la strada, nonostante non siano presenti in scena nemmeno un personaggio; si può dire dunque di aver ottenuto, con queste varie tecniche il risultato desiderato dall'azienda Robin.

Figura 9.18: Strada di Dome City in render mode con billboards procedurali e cubi concentrici per la simulazione degli interni; nessun oggetto di tipo *Lamp* è presente ancora in scena.

9.2.3 Animazione del palazzo frantumato

All'interno del teaser vi è una scena in cui Nadya, mentre corre tra le vie di Dome City, è soggetta ad una delle sue visioni dovute al suo difetto genetico che la porta a soffrire di sinestesia e iperestesi: in questa visione la protagonista vede uno dei palazzi della città sgretolarsi letteralmente davanti ai suoi occhi.
La buona riuscita di questo effetto è senza dubbio stata garantita con l'utilizzo dell'addon per Blender 2.8 *Cell Fracture* unito all'uso del già discusso plugin *Animation Node.*

9.2.4 Cell fracture add-on

Cell Fracture è un add-on gratuito messo a disposizione da Blender che, in modo facile ed intuitivo, permette di creare, data una certa mesh, la sua corrispondente versione 'fratturata', a seconda dei settaggi impostati a piacere nel menù proprio del plugin; una volta applicato Cell Fracture ad una mesh è possibile realizzare interessanti animazioni sfruttando la fisica dei corpi rigidi di Blender.
Alla base di questo plugin vi è un algoritmo che decompone la geometria su cui viene applicato secondo il **Diagramma di Voronoi**: in matematica un *Diagramma di Voronoi* è una partizione di un piano in un numero determinato di oggetti. Nel caso più semplice questi oggetti sono solo punti finiti nel piano chiamati *semi* o *generatori*.

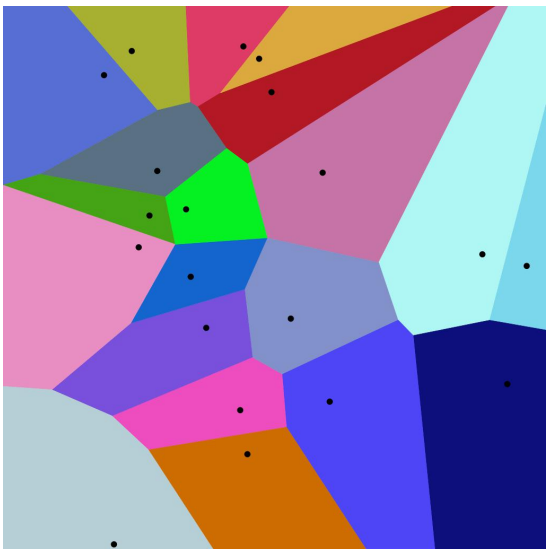


Figura 9.19: Voronoi Cells.

Per ogni seme c'è una regione del piano più vicino ad esso che a qualsiasi altro: tale cellula è chiamata *cellula di Voronoi*.
Nel caso più semplice, mostrato in Figura 9.19, viene data una serie finiti di punti {p1,...,pn} sul piano euclideo. In questo caso ogni seme pk è semplicemente un punto e la sua corrispondente cella Voronoi Rk è costituita da ogni puno del piano euclideo la

cui distanza da pk è uguale o inferiore alla sua distanza rispetto a qualsiasi altro pk.
Ciascune di queste celle è ottenuta dall'intersezione di semispazi, dunque è un poligono convesso. I segmenti di linea del Diagramma di Voronoi sono tutti i punti nel piano equidistanti dai due semi più vicini. I *vertici (nodi) di Voronoi* sono i punti equidistanti da tre o più generatori. Ogni cella Voronoi è matematicamente descritta come segue:

$$R_k = \{x \in X \mid d(x, P_k) \leq d(x, P_j) \text{ for all } j \neq k\}$$

Il seguente ragionamento può essere esteso ed applicato al 3D considerando come insieme uno spazio anzichè un piano. L'algoritmo alla base di Cell Fracture genera un insieme di punti posizionati randomicamente all'interno del volume della mesh; successivamente crea le celle, che andranno a comporre il nuovo oggetto fratturato, secondo la formula del diagramma di Voronoi.
In seguito verranno analizzati i processi e gli steps compiuti da me per ottenere il risultato desiderato sul take della strada per la simulazione della visione di Nadya.
Per prima cosa è stato necessario modellare un palazzo avente una mesh molto semplice ed ordinata, destinato ad essere protagonista della simulazione: i palazzi già presenti in scena infatti, sono risultati poco adatti per l'applicazione del plugin in quanto aventi una mesh decisamente complessa. Una volta soddisfatti della geometria desiderata per il palazzo, ed installato l'add-on *Cell Fracture*, selezionando l'oggetto e andando nel menù *Object* del 3D viewport e cliccando *QuickEffect* > *Cell Fracture* si può procedere con il settaggio dei parametri che regolano l'effetto presenti nel menù che appare.
*Source Limi*t rappresenta il numero massimo di poligoni generabili dalla mesh sorgente; *Noise* invece rappresenta un fattore randomico sulle posizioni dei baricentri di ogni cella quindi sulla loro geometria; *Material* assegna ad ogni cella il materiale associato alla mesh originaria avente indice quello settato; un importante parametro da andare a settare sempre prima di procede con l'applicazione dell'effetto è il *Collection*: è utile infatti, per comodità e per necessità, avere una collezione a sé stante contente solo ed esclusivamente gli oggetti celle generati dall'algoritmo.

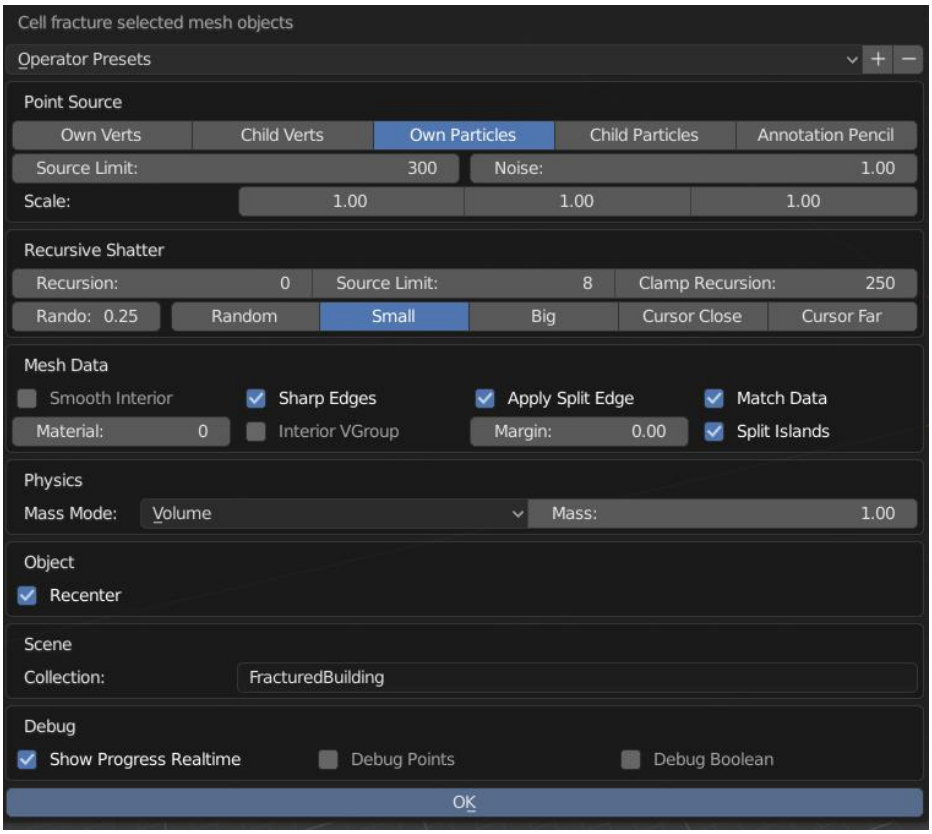


Figura 9.20: Menù Cell Fracture impostato per il palazzo.

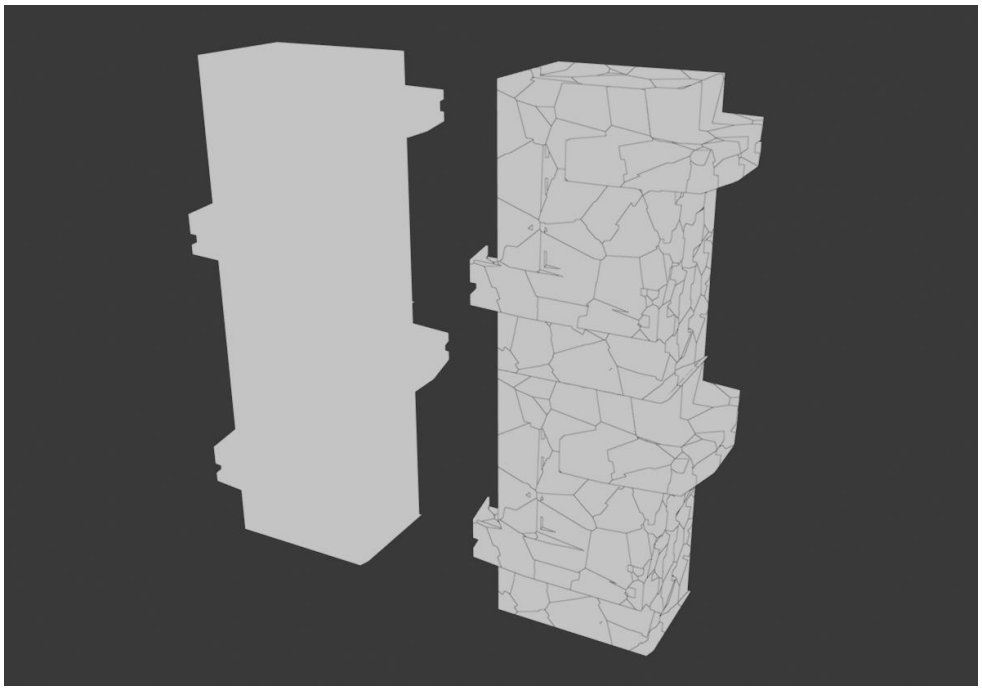


Figura 9.21: Risultato di Cell Fracture sulla mesh del palazzo.

Dopo aver impostato il settaggio dei parametri ed ottenuto la nuova collezione di geometrie che compongono il nuovo palazzo fratturato, si può procedere nel realizzare l'effetto animato usufruendo del plug-in *Animation Node*.

L'idea di base è quella di poter controllare la 'disintegrazione' del palazzo con un oggetto *Empty*, utilizzato come controller anche per attivare la fisica del corpo rigido ad ogni oggetto cella che sia influenzato dalla presenza di tale Empty

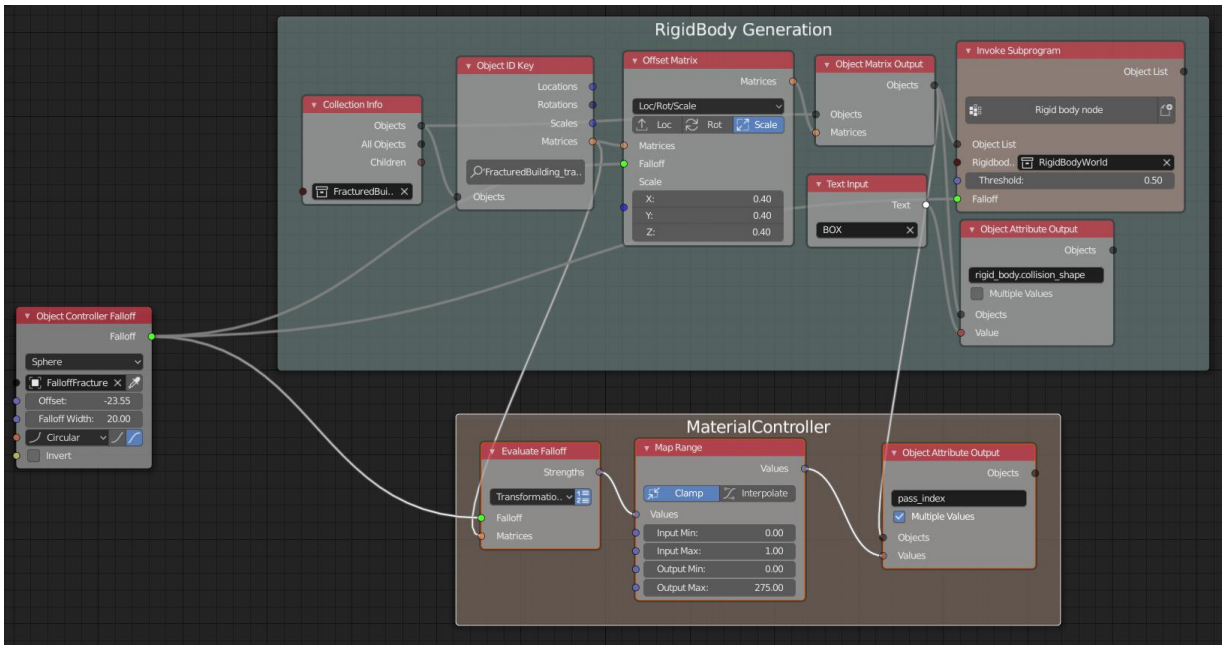


Figura 9.22: Nodetree di Animation Node per la simulazione di disintegrazione

In Figura 9.22 è rappresentato l'albero nodale costruito in Animation Node; all'interno del frame denominato *RigidBodyCollection*, vengono applicate, ad ogni oggetto presente nella collezione *FracturedBuilding* le forze fisiche presenti in scena, in altre parole tali oggetti vengono considerati dei corpi rigidi nella simulazione. Per la realizzazione di questo effetto si è impostata la forza gravitazionale della scena a $+9{,}81 ms^2$ per far si che le celle, invece di cadere verso il basso, si muovessero verso l'alto. L'applicazione di questa operazione su un oggetto dipende poi da un valore di *Falloff,* a sua volta controllato da un *Object Controller* di tipo *Sphere* a cui abbiamo associato un oggetto Empty sferico. Scalando l'Empty Sphere si controlla l'influenza di quest'ultimo sulle geometrie degli oggetti, secondo i valori impostati sui parametri

Width e *Offset*. Prima di entrare nel nodo di generazione della fisica del corpo rigido, alla lista di oggetti viene applicata una trasformazione di scalamento, che dipende sempre dal valore di Falloff generato dalla sfera, per una migliore resa della simulazione.
Il secondo Frame contiene invece i nodi che controllano il materiale associato ad ogni cella, che dipenderà anch'esso dall'influenza dell'Empty: infatti il valore di Falloff va a modificare e settare come diversa da zero, la componente *pass_index* degli oggetti, componente che si può sfruttare per creare un gradiente di colori al palazzo quando si decompone.
L'effetto è stato successivamente arricchito e migliorato tramite l'uso di un sistema particellare applicato ad un piano, posizionato sotto l'edificio, che emette come oggetto una mesh casuale della collezione FracturedBuilding.

Tutta la simulazione del palazzo frantumato è stata sottoposta all'operazione di baking per ottenerne i keyframe; questo ha ridotto notevolmente il peso della scena, e ha permesso di creare dei proxy dello stesso palazzo simulato da poter inserire all'interno dell'ambiente della scena di cui è protagonista.

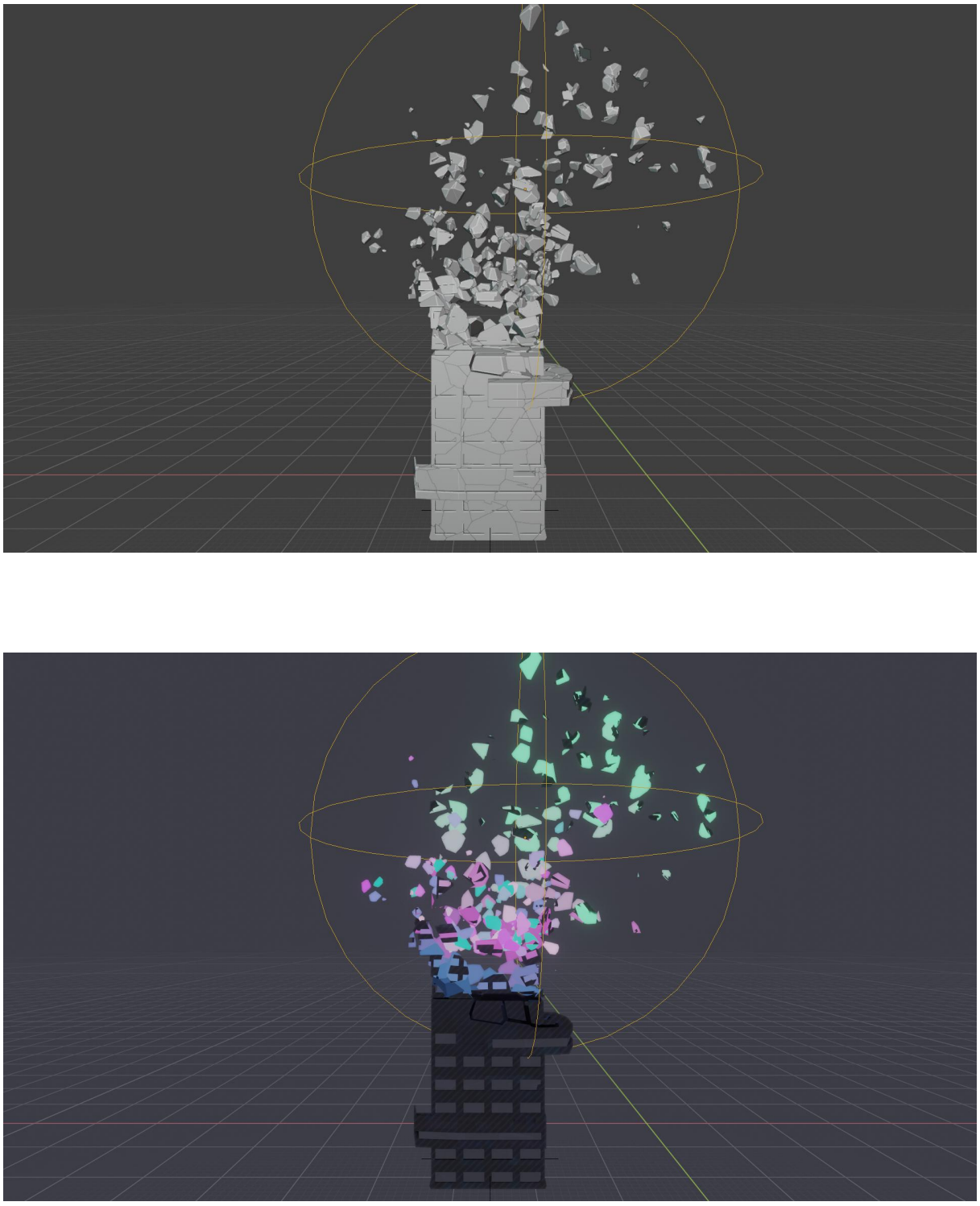


Figura 9.23: Risultato dell'animazione in solid mode e in render viewport.

9.3 Serra Universitaria

Gli autori della serie per questo ambiente si sono fatti ispirare dalla nuova sede Amazon di Seattle, del 2017. Questa struttura in effetti ricorda molto una serra di enormi dimensioni.

Figura 9.24: Reference utilizzata per la serra, sede Amazon.

Questo luogo rappresenta per Dome City, e probabilmente per l'intero pianeta, una delle poche 'aree verdi' ancora rimaste, poichè l'aria tossica ed inquinata e le condizioni climatiche a cui è sottoposto l'exoplanet hanno fatto scomparire la gran maggioranza della vegetazione prima esistente. La serra ha dunque la funzionalità di contenere e preservare ogni specie vegetale esistente e preesistente del pianeta, da quelle terrestri a quelle marine; proprio per questo motivo le strutture sferiche saranno numerose e di diverse dimensioni, dovendo ospitare al loro interno ognuna un tipo di vegetazione diversa dalle altre.

Ancora una volta l'idea è quella di rappresentare un edificio quanto più organico possibile, richiamando caratteristiche e forme presenti in natura: si basti pensare al pattern di bolle della schiuma che si forma sulla superficie dell'acqua per esempio; l'idea è proprio quella di creare la serra come formata da un insieme di cupole di

diverse dimensioni (la più grande è quella centrale che ne ospita l'entrata, da cui si diramano tutte le altre), ognuna collegata a quella adiacente.
Un altro elemento della reference a cui ci si è ispirati è il materiale dell'edificio: una struttura trasparente sorretta da un'impalcatura metallica che funge da portante, da sostegno per l'intera serra.

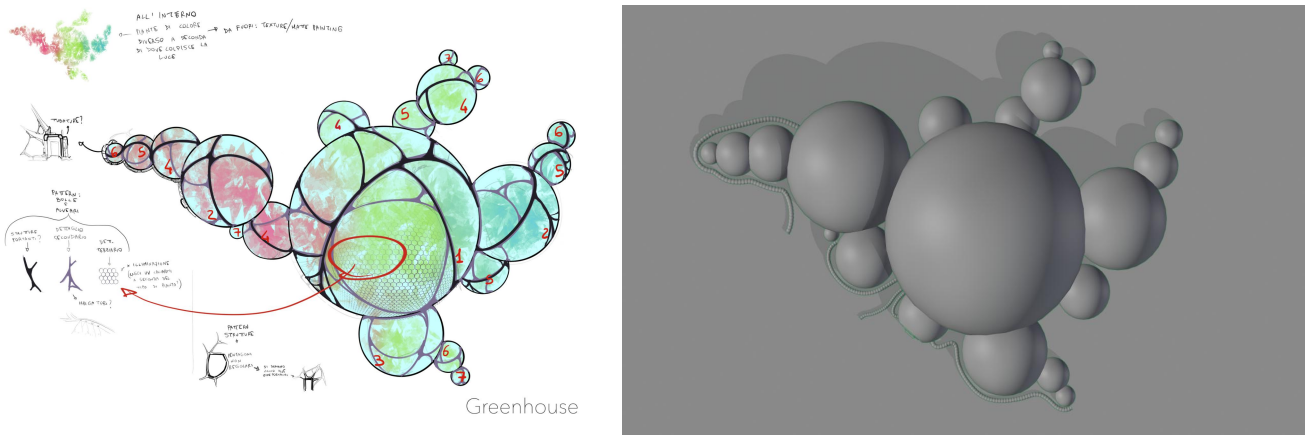


Figura 9.25: Concept art della serra e il modello 3D associato.

Dopo aver realizzato il modello dell'ambiente, secondo il concept art disegnato dal nostro concept artist Edoardo Audino, ci si è posti il quesito di come poter modellare le strutture portanti in modo da renderle il più organiche possibile, come rappresentate dal disegno in 2D in Figura 9.25.
Rappresentare i portanti della serra tramite il modifier di tipo G*enerate Wireframe* avrebbe portato ad un risultato non soddisfacente, poichè le diramazioni, componenti tale struttura, devono necessariamente essere come richiesto curve morbide e non una semplice riproduzione del wireframe della costruzione principale. Ci si è impegnati quindi a trovare un'altra tecnica che permettesse di portare a termine il requisito.

La realizzazione di questo environment non sarebbe forse stato possibile senza una delle più grosse novità di Blender 2.8: lo strumento *Grease Pencil.* Le modifiche apportate nella nuova versione del software per questo specifico tool sono molteplici e meritano quindi una breve descrizione prima di proseguire l'analisi di esecuzione della modellazione della serra.

9.3.1 Grease Pencil

Il Grease Pencil nelle versioni di Blender precedenti la 2.8, era soltanto uno strumento utilizzato per disegnare delle piccole note nella 3D view, con lo scopo di fornire spiegazioni sul progetto che si stava realizzando; in seguito però, fu fornito di altre funzionalità riguardanti sia il disegno che l'animazione che lo trasformarono in un vero e proprio strumento per il disegno 2D all'interno di un software di modellazione 3D. In Blender 2.8 il Grease Pencil diventa un oggetto indipendente, che può essere gestito facilmente tramite un'interfaccia semplificata e tutti i modifier e le funzionalità ad esso associate, per poter realizzare, tra le altre cose, dei veri e propri cartoni animati.
Lo strumento è stato quindi diviso in due parti distinte dal punto di vista logico, una più semplice per realizzare annotazioni sul file di lavoro, e un'altra che lo rappresenta come oggetto, che potrà inoltre essere convertito in curva tridimensionale su cui poter applicare un volume ottenendo quindi una mesh solida.

9.3.2 Grease Pencil come strumento di annotazione

Il menù *Annotate* per invocare lo strumento di annotazione è situato nella toolbar sinistra dell'interfaccia di Blender: tra le opzioni è possibile scegliere tra *annotazione semplice* che permette di disegnare o scrivere a mano libera, *line,* utile per tracciare linee di riferimento, *polygon* per creare più agevolmente dei poligoni o delle linee spezzate, *erase* invece servirà a correggere quanto scritto. Dopo aver utilizzato lo strumento, nel pannello con le proprietà dei tool apparirà un menù a discesa per la gestione dei layers, del colore e dello spessore del tratto della penna. Blender fornisce l'opzione *Placement* per questo strumento che permette di decidere il tipo di visualizzazione delle annotazioni, se cioè posizionarle ortogonalmente alla vista tutto sullo stesso piano (*View),* se abilitare il disegno direttamente sulle facce dell'oggetto (*Surface*), oppure utilizzare la funzione *3D Cursor* che è simile a View ma permette di posizionare il piano perpendicolare alla vista nella posizione definita dal cursore.

9.3.3 Grease Pencil come Oggetto

La maggior parte delle nuove caratteristiche del Grease Pencil, comprese quelle ereditate dalle versioni precedenti la 2.8 di Blender, si possono trovare nel nuovo Oggetto Grease Pencil, richiamabile nel menù *Add* in Object Mode.
Esistono tre tipi di oggetti Grease Pencil: *Stroke* crea un tratto di penna definito, *Monkey* disegna Suzanne ma in formato 2D, *Blank,* è simile all'Empty, crea infatti un oggetto vuoto nel quale poter inserire il disegno vero e proprio. Con l'oggetto Grease Pencil potremo compiere tutte le operazioni che si potevano ottenere con la semplice annotazione con la differenza che ora si possono avere a disposizione una serie di strumenti necessari per il disegno e l'animazione 2D su Blender. Ci saranno a disposizione anche tutte le features per questo oggetto, come i suoi specifici modifier, material, interfaccia per l'animazione e i Grease Pencil Visual Effects.
Anche l'oggetto Grease Pencil, come l'annotazione, permette di scegliere la modalità di disegno su View o su Surface; in particolare, se viene scelta quest'ultima è possibile creare effetti grafici sulla mesh in pochissimo tempo, oppure trasformare quegli strokes in curve o geometrie che possono essere integrate al modello di partenza. Quest'ultima operazione è stata proprio quella utilizzata per la realizzazione delle strutture portanti della Serra Universitaria Di Reverie Dawnfall.

Una volta finita la modellazione della struttura a cupole, con l'oggetto Grease Pencil sono state disegnate in modalità *Surface* sulla superficie della serra le linee rappresentanti le strutture portanti principali e quelle sui giunti, aventi una funzione di sostegno dell'architettura, oltre che essere anche dei motivi ornamentali.
In Figura 9.26 è riportato il risultato dell'environment dopo aver utilizzano l'oggetto Stroke Grease Pencil. I due tipi di portanti sono stati creati a partire da due oggetti Strokes separati, evidenziati uno in rosso e l'altro in nero, per rendere più comoda la gestione delle curve da loro derivanti.

Una volta soddisfatti della disposizione delle curve sulla superficie principale dell'ambiente ottenute con il nuovo tool di Grease Pencil, è possibile procede nel

dare forma concreta a queste linee disegnate, trasformandole quindi da un oggetto *Grease Pencil* a uno *Curve*. In questo modo è possibile in primo luogo aggiustare gli angoli delle curve risultanti non precise con Grease Pencil, o regolare il numero di punti che compongono la curva, probabilmente troppo numerosi in alcuni tratti.
Dopo averla pulita, si procede nell'assegnare alla curva come bevel un altro oggetto Curve, un semplice cerchio per i portanti sui giunti, ed un rettangolo per i portanti principali. A questo punto si apportano le modifiche opportune alla rotazione della curva in relazione alla forma data al bevel, specialmente per quanto riguarda le curve con shape rettangolare.
L'ultimo step è la trasformazione dell'oggetto da Curve a mesh.
In seguito è riportato il risultato in solid mode del modello ottenuto.

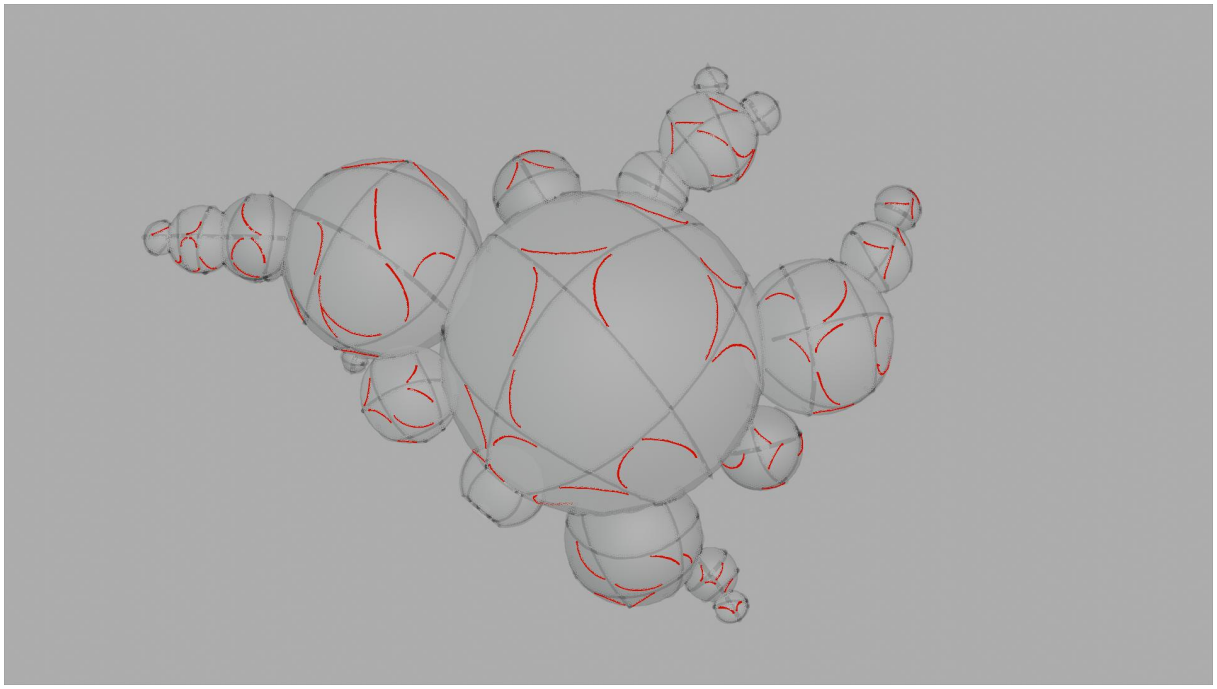


Figura 9.26: Linee disegnate con Grease Pencil sulla superficie della serra.

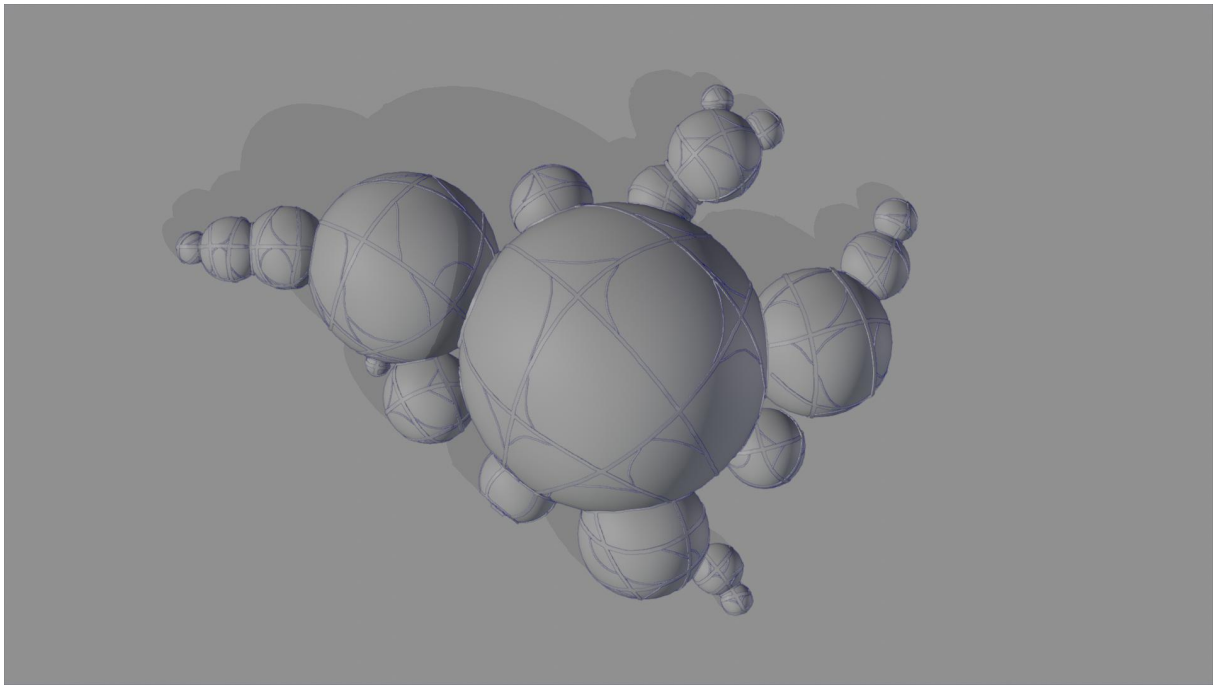


Figura 9.27: Risultato in solid mode della trasformazione e successivo editing degli strokes in mesh.

L’ambiente è stato poi arricchito di dettagli necessari a contestualizzarlo nella narrativa del teaser; in quanto edificio situato nelle strade di Dome City, l’environment complessivo deve infatti risultare coerente con la strada cittadina, la cui modellazione e il risultato grafico sono stati spiegati ed illustrati nel paragrafo precedente.

Figura 9.28: Render della serra.

9.4 Il Vuoto cosmico

L'ultimo ambiente che ho dovuto realizzare per il teaser di Reverie Dawnfall è diverso da tutti gli altri in quanto ambiente estraneo al mondo in cui vivono i protagonisti della serie: esso è il soggetto di una visione di Nadya, una visione illusoria che però lei intuirà essere molto più concreta di quello che sembra. In realtà, infatti, fluttuazioni quantiche le permettono di percepire brevi visioni di dimensioni adiacenti le tre percepibili dall'essere umano. Durante le sue crisi più violente, Nadya può infatti, per una manciata di secondi, traslare nello spazio tra le realtà e vedere la nostra, od altre contigue, da fuori, oppure scorgere tasselli di passato, futuro, luoghi distanti. Le immagini psichedeliche che lei inizialmente reputa uno scherzo del suo cervello malato, saranno una finestra su una chance di salvezza per il morente pianeta. In questo caso la visione che la colpisce le mostra l'universo come noi lo conosciamo, la trasporta nel vuoto cosmico, in mezzo a miliardi di stelle che mai prima aveva visto, sopra ad una dirupo di un suolo lunare che pare fluttuare nell'aria, davanti a pianeti sconosciuti.

Questo è stato l'ambiente che ha richiesto il minor tempo di realizzazione, in quanto l'obbiettivo alla base era il colpire lo spettatore con immagini di una realtà a noi ben nota, quale l'universo e la nostra galassia: sono quindi bastate semplicemente delle references fotografiche prese dal sito della *Nasa* per riprodurre un ambiente in 3D che ne fosse quanto più fedele possibile, mantenendo però lo stesso stile grafico, quindi lo stesso shader non fotorealistico, di tutta la serie.

Per quanto riguarda la modellazione, gli unici elementi modellati all'interno della scena sono il dirupo su cui è situata Nadya, a forma di iceberg rovesciato, ed i pianeti, semplici sfere su cui è stata poi applicata una texture. Per la creazione del dirupo invece, ci si è avvalsi di un altro plugin gratuito di Blender il cui uso ed interfaccia sono stati migliorati nella nuova versione 2.8: *A.T.N. Landscape.*

9.4.1 A.T.N. Landscape

Questo add-on serve per creare paesaggi e pianeti, ed agisce utilizzando ed applicando alla mesh vari tipi di 'rumori': A.T.N. è infatti l'acronimo di *Another Noise Tool*. L'algoritmo alla base applica ad una mesh di un piano multi-suddivisa, vari displacements regolati da parametri scelti dall'utente all'interno del menù che appare dopo aver aggiunto in scena una *Mesh* di tipo *A.N.T. Landscape:* nel pannello delle operazioni di progettazione della mesh sono presenti tre aree principali, il *Main Settings* dove troviamo le impostazioni relative a oggetti o mesh come la dimensione e il numero delle suddivisioni; *Noise Settings* in cui si determinano le impostazioni del noise che darà forma al nostro terreno; infine il *Displace Settings* dove vengono settate le impostazioni relative al valore di altezza (*High*) del terreno e dell'*edge falloff*.

La superficie del terreno così realizzato, è stato successivamente popolato, tramite un particellare di tipo hair, di massi e rocce lunari. Il restante lavoro è passato poi nelle mani del mio collega Andrea Lorusso, che si è occupato, in fase di illuminazione, di creare l'effetto grafico desiderato grazie ad un ottimo ed ingegnoso uso delle luci volumetriche offerte da Blender 2.8.

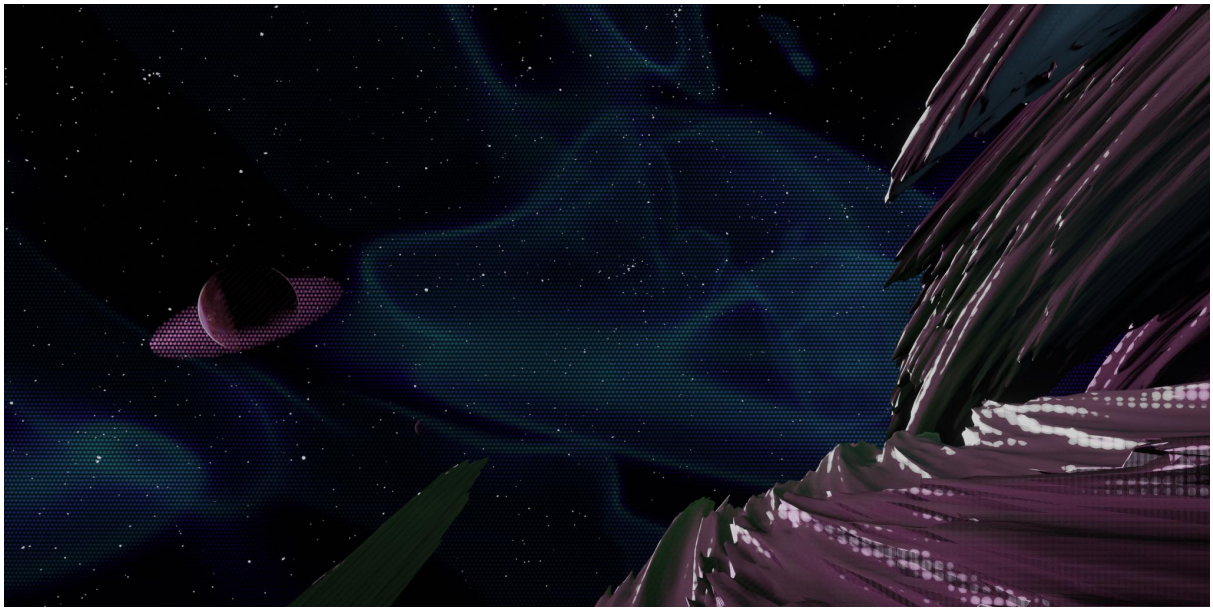


Figura 9.29: Render di una delle camere poste nella scena del Void.

Conclusioni

La scelta di utilizzare come software per la modellazione tridimensionale degli ambienti e per tutte le altre fasi di sviluppo del teaser trailer Belnder, è risultata nella maggior parte dei casi favorevole al team di produzione; la versione 2.80 del programma, ha infatti reso possibile il compimento del principale obiettivo posto dall'azienda, ovvero quello di sperimentare stili grafici e tecniche quanto più innovative possibili. Grazie ad alcuni nuovi tools come il Grease Pencil e al nuovo motore di render real time Eevee che tra le altre cose ha permesso di osservare il risultato finale in tempo reale di ogni minima modifica apportata ad una scena, Blender ha favorito l'accelerazione di alcuni processi a livello produttivo.
Il risultato dell'intero lavoro svolto dal gruppo di produzione è racchiuso all'interno della seconda versione del teaser trailer di Reverie Dawnfall, il cui contenuto visivo è stato approvato dall'azienda per cui si è svolta la mansione.
L'obiettivo seguente il conseguimento della buona riuscita del teaser, è quello di poter ottenere dei fondi da investire nella produzione dell'intera serie animata, dunque proporre lo stesso trailer a bandi e festival italiani ed europei per il cinema e per prodotti audiovisivi.

Lavorare ad un progetto indipendente e dall'alto potenziale come Reverie Dawnfall, ha permesso al team di produzione di ampliare la conoscenza del software Blender, e ad ogni singolo componente del gruppo di testare le proprie capacità anche su tutti gli altri settori diversi dal proprio argomento di tesi. Principalmente in fase di animazione infatti, tutto il team ha lavorato ai vari take di ogni scena del teaser, cercando di ottendere il risultato voluto pur in assenza di un animatore; a questo scopo sono stati creati inoltre svariati scripts in Python per facilitarne la buona riuscita, dimostrando ancora una volta la natura sperimentale del progetto.

Sitografia Software

Modellazione 3D

https://www.pluralsight.com/blog/film-games/setting-the-stage-for-a-3d-world-with-prop-and-environment-modeling

https://www.cgmasteracademy.com/programs/5-3d-environment-arts-program

https://it.wikipedia.org/wiki/Modellazione_3D#Modellazione_nella_pratica_operativa_dalla_grafica_3D

Addon Blender

https://www.graphicsandprogramming.net/ita/tutorial/blender/il-grease-pencil/grease-pencil-in-blender-2-8-introduzione

https://docs.blender.org/manual/en/latest/grease_pencil/introduction.html

https://www.ipiratigrafici.it/disegnare-3d-grease-pencil-blender-2-8/

https://www.blender.it/news/blender-2-8-grease-pencil/

https://blender-addons.org/cell-fracture-addon/

https://docs.blender.org/manual/fr/dev/addons/add_mesh/ant_landscape.html

Motori di render

https://www.blender.it/motori-di-rendering/

https://www.treddi.com/cms/news/eevee-il-nuovo-motore-di-render-realtime-per-blender-28/3517/

https://www.blender.org/download/releases/2-80/

https://cgcookie.com/articles/blender-cycles-vs-eevee-15-limitations-of-real-time-rendering

https://www.blendernation.com/2019/07/20/spring-comparing-eevee-to-cycles/

Reference grafiche

https://it.wikipedia.org/wiki/Gatta_Cenerentola

https://www.quora.com/What-do-you-call-the-animation-style-in-Spiderman-Into-the-Spiderverse

Software di modellazione

https://it.wikipedia.org/wiki/AutoCAD

https://it.wikipedia.org/wiki/3ds_Max

https://it.wikipedia.org/wiki/Autodesk_Maya

https://www.prisma-tech.it/scheda-prodotto/maya/

http://pixologic.com/zbrush/features/overview/

https://it.wikipedia.org/wiki/ZBrush