ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

FACOLTÀ DI SCIENZE MATEMATICHE FISICHE E NATURALI

Corso di Laurea in Scienze e Tecnologie Informatiche

Integrazione del motore grafico Unreal Engine e di Autodesk Maya nella realizzazione di una realtà grafica interattiva.

Relazione Finale in Metodi Numerici per la Grafica

Relatore:

Damiana Lazzaro

Presentata da:

Lorenzo Marinelli

Sessione I

Anno Accademico 2011 / 2012

INDICE

INDICE DELLE IMMAGINI

INTRODUZIONE

Nati come semplici esperimenti di carattere universitario verso l'inizio degli anni '50, i videogiochi hanno subito una rapida e quanto mai radicale evoluzione nel tempo. Nonostante sia possibile trovare i primi esempi di videogioco già nel 1947 (*Cathode-ray tube amusement device)* questi sono diventati un fenomeno di massa solamente all'inizio degli anni '70, con la creazione di *Galaxy Game*, un videogioco basato su *Spacewar!* il quale era stato a sua volta creato nel 1961 da un gruppo di studenti del MIT. *Galaxy Game* fu realizzato alla Stanford University e rappresenta il primo esempio di videogioco arcade giocabile con l'inserimento di una monetina. Il gioco fu distribuito in 1500 copie, ma non fu un successo clamoroso, data l'elevata difficoltà. L'anno successivo fu realizzata invece la prima console casalinga, la *Magnavox Odissey*. Nel 1972 venne fondata l'Atari, che produsse quello che viene considerato il primo vero grande successo dell'industria dei videogiochi: *PONG.* Nel finire degli anni '70 era ormai chiaro che questo genere di prodotti avrebbe inesorabilmente conquistato il mercato mondiale, con la realizzazione di capolavori del calibro di *Space Invaders, Pac Man e Asteroids*. [WIKIa]

L'introduzione di nuove e sempre più potenti console porterà i videogiochi in casa di tutte le persone che potranno permetterselo, rendendo effettivamente questo un fenomeno di massa in continua crescita.

Oggi possiamo già utilizzare quelle che vengono considerate le console di settima generazione e le prime console di ottava generazione sono già nelle case di molti. Passando da una generazione di console all'altra e analizzandone alcuni videogiochi di esempio possiamo notare che l'evoluzione dei prodotti è evidentissima. La giocabilità di alcuni titoli di vecchio stampo risulta ad un nuovo giocatore qualcosa di noioso e complicato; chi è cresciuto con i primi videogiochi non potrà non aver notato come la difficoltà dei titoli sia calata notevolmente grazie all'introduzione di nuovi metodi di gioco che rendono decisamente più semplice e meno frustrante la vita di ogni genere di giocatore. Questa particolarità però viene notata solo con un accurato esame del prodotto che si è andati a testare. La prima cosa che salta all'occhio quando si prende un videogioco come *Pac Man* e lo si confronta a uno uscito negli ultimi anni è certamente la grafica di gioco. [WIKIa]

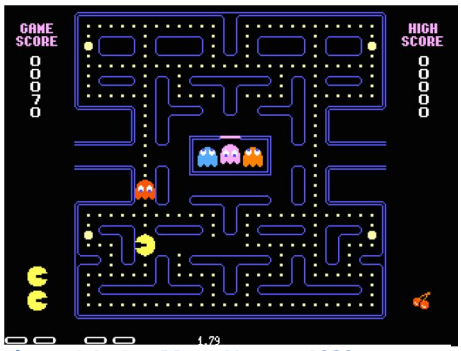


Figura 1.1 - Pac Man - Namco, 1980


Figura 1.2 - Final Fantasy XIII - Square Enix, 2010

Ciò che ha reso possibile un simile miglioramento è la rapida evoluzione che ha interessato il mondo dell'informatica. Infatti è ovvio notare come l'evoluzione dei videogiochi sia andati di pari passo con quella dei computer. Inizialmente non si avevano mezzi potenti e soprattutto la memoria disponibile in un computer non era molto elevata. La crescita esponenziale dei mezzi a disposizione ha fatto sì che i responsabili del mercato dei videogiochi riuscissero a sviluppare prodotti sempre più gradevoli a livello di dettaglio grafico per il giocatore. Ciò ha però portato in alcuni casi a curare più l'aspetto grafico che il resto, facendo sì che alcuni titoli di nuova generazione fossero piuttosto scarsi a livello di giocabilità.

La computer grafica viene utilizzata anche in altri ambiti (da quello cinematografico a quello di progettazione architettonica [WIKIb]) e al giorno d'oggi è possibile, per chiunque possieda un computer sufficientemente potente e il software adatto, realizzare modelli tridimensionali che con le dovute cure potrebbero essere utilizzati in un videogioco.

In ogni software house videoludica la creazione di un nuovo prodotto è affidata ad un gruppo piuttosto numeroso di dipendenti, ognuno dei quali avrà un compito ben preciso nella realizzazione del titolo. A livello grafico, ci sarà una fase di modellazione degli oggetti di gioco e una fase di creazione di mappe complesse che utilizzano questi modelli. Queste sono strettamente collegate fra loro.

L'obiettivo di questa tesi è quello di illustrare come avvengono queste importanti fasi nella creazione di una realtà grafica interattiva, soffermandosi in particolare sull'uso di un potente tool di sviluppo utilizzato a livello lavorativo, UDK integrandolo con un software per la modellazione grafica 3D.

La tesi è così organizzata:

Nel capitolo 1 andremo ad esaminare i principali strumenti utilizzati nella realizzazione del progetto, i quali corrispondono a due potenti software, Autodesk Maya (versione 2011) e UDK (versione 3), descrivendone le funzionalità e gli utilizzi che ne possono essere fatti. In particolare porremo attenzione sull'interfaccia grafica di UDK, software sul quale utilizzo è posta la maggiore attenzione.

Nel capitolo 2 studieremo come è possibile integrare i due software, ovvero su come importare su UDK ciò che si è creato in Autodesk Maya, ovvero i modelli 3D che andranno a comporre la nostra realtà interattiva.

Nel capitolo 3 entreremo per la prima volta nella descrizione di una particolare fase del Level Design, in questo caso quella che va a costruire ciò è chiamato "Terreno di gioco" (o semplicemente Terrain), descrivendo inoltre come sia possibile integrare a questo uno strumento per la realizzazione di superfici fluide, per simulare la superficie di laghi, fiumi o mari.

Nel capitolo 4 porremo la nostra attenzione sulla creazione dei materiali che vanno a definire infine l'aspetto degli oggetti di scena. Questa fase incide molto sulla qualità finale della grafica di un progetto.

Nel capitolo 5 esamineremo i particolari dell'illuminazione in UDK, studiando tutte le possibilità messe a disposizione del software, visualizzando tramite immagini le differenze e gli effetti generati dalle diverse scelte. La relazione fra l'illuminazione e fra come i materiali sono costruiti (capitolo 4) è importante, in quanto un materiale per cui non ci si cura particolarmente della reazione che deve avere alla luce, per quanto ben realizzato, potrebbe vanificare tutti gli sforzi fatti nella sua costruzione.

Nel capitolo 6 studieremo come è possibile creare un particolare tipo di oggetto, il Cloth, che altro non è che una Mesh costruita in modo che reagisca al motore fisico di UDK simulando l'effetto di un tessuto.

Nel capitolo 7 studieremo la gestione degli eventi e delle animazioni dell'ambiente, punto focale della creazione di un videogioco, tramite i tool Kismet e Matinee di UDK. Questa fase è molto importante in quanto si va a creare la *giocabilità* del titolo, fattore spesso determinante per gli utenti per giudicare un titolo buono o pessimo.

Nel capitolo 8 vedremo come UDK possa gestire oggetti che seguano le leggi del motore fisico (*Rigid Body*) reagendo alle interazioni con il giocatore. Vedremo inoltre come è possibile modificare delle semplici Mesh generate in Maya in modo da

renderle distruttibili all'interazione del giocatore tramite lo studio del *Fracture Tool*, uno strumento apposito per questo obiettivo.

Nel capitolo 9 infine vedremo come è stato sviluppato il progetto di tesi che è servito anche come terreno di prova per lo studio del software UDK (con l'ausilio della ricca documentazione online del sito ufficiale). In particolare vedremo come tutte le fasi descritte nei precedenti capitoli si mescoleranno fra loro andando a definire l'ambiente virtuale finale.

Capitolo 1 : Strumenti Utilizzati

In questo capitolo andremo ad esaminare i due principali strumenti che sono stati utilizzati per la realizzazione del progetto di tesi: Autodesk Maya 2011 e UDK 3. Porremo particolare attenzione sull'interfaccia grafica di UDK, il quale è lo strumento che effettivamente studieremo approfonditamente nei capitoli successivi.

1.1 - AUTODESK MAYA

Autodesk Maya è un software professionale di modellazione, animazione e rendering 3D [MAYA]. Si tratta di un programma molto profondo, in grado di fornire all'utilizzatore un numero davvero vasto di funzionalità. Permette anche di creare procedure personalizzate e l'aggiunta di plugin, per migliorare o specializzare alcune funzionalità non presenti nel prodotto base, che rende comunque possibile numerose operazioni di modellazione. Questo software, per le potenzialità offerte risulta essere uno dei più utilizzati per la realizzazione di film di animazione in computer grafica e per creare effetti speciali per film, dalle semplici produzioni casalinghe alle più dispendiose produzioni Hollywoodiane. Gli strumenti resi disponibili da Maya permettono la creazione di modelli tridimensionali di altissima qualità e la *Character Animation* ,risulta davvero ottimale. Anche per questo motivo Maya si presta per la realizzazione di modelli tridimensionali da integrare per il Level Design di un videogioco e per la realizzazione di tutti i personaggi e le relative animazioni.

Autodesk Maya però da solo non rende possibile la realizzazione di un ambiente interattivo o di un videogioco. Infatti mette a disposizione alcuni motori di rendering (Maya Software, Maya Hardware, Maya Vector, Mental Ray,… [MAYA]) in grado però di realizzare solamente immagini della scena realizzata. Questo significa che Maya non permette la creazione di una scena veramente dinamica, ma ciò che avviene all'interno di essa è tutto precalcolato tramite l'introduzione di attributi e variabili per i vari componenti. Per realizzare ciò è necessario l'utilizzo di un tipo di software diverso, ovvero che permetta di realizzare scene dinamiche tramite motori grafici in grado di calcolare tutti i valori degli elementi della scena in base agli spostamenti e alle interazioni dell'utente.

Quindi per la realizzazioni di un ambiente interattivo non basta Autodesk Maya, ma è necessario integrare quest'ultimo con software specializzati per fornire questa possibilità.

1.2 - UNREAL DEVELOPMENT KIT

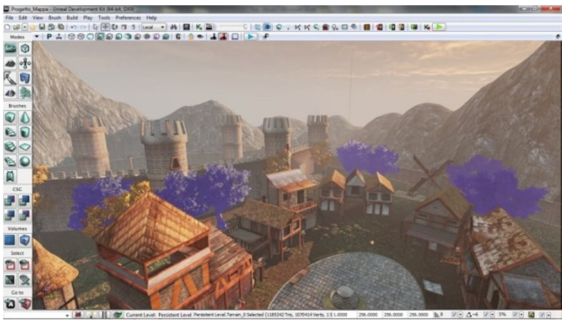
UDK è un software che utilizza il motore grafico Unreal Engine 3, il terzo della famiglia dei motori grafici creati da Epic Games [UDK]. Si tratta di un programma gratuito (se utilizzato senza scopo di lucro) scaricabile dal sito ufficiale di UDK. Tramite questo è possibile realizzare videogiochi, realtà interattive e programmi di simulazione. Il motore grafico del software rende possibile ciò che invece non lo era in Autodesk Maya: la creazione di ambienti dove la realtà è interattiva e non precalcolata, permettendo quindi all'utente (o giocatore) di interagire con gli oggetti che compongono il "mondo di gioco". Il rendering di quest'ultimo avviene runtime, differenza sostanziale con i motori di rendering di Maya.

UDK mette a disposizione tutti gli strumenti utili alla creazione di videogiochi. Principalmente rende disponibili le seguenti funzionalità [UDKF]:

- Strumenti di Level Design e Environment Editing;
- Rendering Runtime;
- Assoluta flessibilità con le animazioni degli oggetti di scena;
- Scripting;
- Motore fisico realistico;
- Sistema di illuminazione e gestione delle ombre realistico;
- Creazione di cut scenes tramite appositi tool;
- Creazione di terrain (terreni, montagne, pianure, vegetazione,…);
- Creazione di applicazioni con funzionalità online;
- Creazione di materiali realistici e shaders real-time;
- Supporto audio;
- Creazione di effetti tramite particelle (particle system);
- Gestione intelligenza artificiale;
- Gestione automatica delle performance del computer per garantire fluidità;
- Creazione di oggetti realisticamente "distruttibili";
- Creazione di vegetazione realistica tramite Foliage;
- Gestione di animazioni facciali di personaggi tramite audio.

Si può notare come effettivamente UDK sia un software molto elaborato e decisamente all'avanguardia. È stato infatti utilizzato a livello aziendale per la realizzazione di numerosi videogiochi di successo (*Batman: Arkham City*,*Bioshock 2*, *Borderlands* [EPUE]).

1.3 – UDK: interfaccia grafica



1.3.1 - UDK, Interfaccia

Descriviamo innanzitutto l'interfaccia grafica di UDK, così da capire meglio come muoverci all'interno di questo software e come raggiungere rapidamente le sezioni o i comandi più utilizzati durante la progettazione e soprattutto all'interno di questo documento, nel quale spesso si farà riferimento a comandi che dovranno poi essere utilizzati per eseguire anche i compiti più semplici. UDK si presenta come nella figura 1.3.1, mostrando un menù in alto con due barre degli strumenti subito sotto. A sinistra un'altra barra degli strumenti, utili a selezionare i principali strumenti di modellazione dell'ambiente. In basso troviamo un'ulteriore sezione che permette di cambiare alcune opzioni inerenti gli strumenti utilizzati per navigare in UDK e la visualizzazione del piano di lavoro, che è ovviamente rappresentato dalla grande finestra centrale (nell'immagine 1.3.1 si vede un particolare del progetto di tesi).

Per un novizio è necessario innanzitutto comprendere come navigare all'interno della scena: tramite la pressione del tasto destro del mouse sarà possibile cambiare la visuale, mentre con la pressione dei tasti ***W, A, S,*** e ***D*** è possibile muoversi rispettivamente in alto, a sinistra , in basso e a destra nel piano di lavoro. Con la rotella del mouse si può invece zoomare. Una volta compresi questi comandi di base si può iniziare a descrivere l'interfaccia grafica. (Informazioni tratte da [GSUDK]).

Barra del menù : 

Dalla barra del menù è possibile raggiungere ogni sezione, ogni Tool e ogni comando disponibile in UDK. Vediamo in dettaglio cosa ci viene messo a disposizione:

- ***File:*** tramite questo menù è possibile aprire un progetto o inizializzarne uno, salvare e importare e esportare oggetti da progetti diversi.
- ***Edit:*** è possibile da qui selezionare i comandi di *Rotate*, *Translate* o *Move*, anche se questi possono essere selezionati con la semplice pressione della barra spaziatrice. Inoltre è possibile trovare qui il menù *Find Actors* che facilita la ricerca di oggetti nelle scene più complesse.

- **View:** questo menù permette di raggiungere il Content Browser e dunque i package contenenti le Mesh e tutti gli altri componenti del progetto. Consente inoltre di impostare le proprietà del mondo di gioco, degli actors e delle superfici.
- ***Brush*:** consente di effettuare alcune operazioni di modellazione sulle Brush costruite tramite UDK come addizione, sottrazione e intersezione.
- ***Build***: menù per la costruzione della scena. Una volta creato un livello infatti è necessario selezionare generalmente il comando *Build All* da questo menù, in modo da proiettare tutte le luci e generare tutti gli oggetti di scena e le loro collisioni, così da poter poi effettivamente visitarlo e giocare. Si tratta in poche parole della compilazione del progetto.
- ***Play:*** selezione della modalità di gioco (disponibile anche per dispositivi Mobile).
- ***Tools:*** si può da qui effettuare il controllo della mappa per trovare errori, creare un nuovo terrain o selezionare alcuni comandi utili alla correzione della scena.
- ***Preferences*:** imposta le preferenze dell'utente riguardo l'interfaccia grafica di UDK.
- ***Help***: mostra aiuti e suggerimenti per lavorare in UDK.

Barra degli strumenti 1 :

Questa barra degli strumenti mostra alcune scorciatoie per raggiungere strumenti di sviluppo e comandi che possiamo trovare anche nel menù precedentemente descritto. Oltre al salvataggio rapido permette di raggiungere Content Browser, Kismet, Matinee e di far partire il gioco.

Barra degli strumenti 2 :

Tramite i comandi qui disponibili è possibile cambiare il tipo di illuminazione della scena in fase di editing. Cambiare tra le diverse modalità può essere utile a vedere in anteprima gli effetti della luce sugli oggetti del livello. Nell'immagine 1.3.1 ad esempio è attiva l'illuminazione *Lit*, che mostra l'illuminazione della fase di gioco.

Barra degli strumenti laterale :

Nel lato sinistro della finestra di lavoro troviamo la barra degli strumenti che verrà maggiormente utilizzata in fase di Level Design. Le svariate icone vengono suddivise in 6 categorie:

Modes: le otto icone qui disponibili sono utilizzate per avviare i principali Tool per la creazione di una scena.

- **Camera Mode,** per navigare all'interno della scena;
- ***Geometry Mode***, per avviare il tool di modellazione di UDK, che però non è ai livelli di Autodesk Maya e generalmente non viene utilizzato per modellare oggetti complessi;
- ***Terrain Editing Mode***, avvia il Tool di modellazione terreni (vedi cap. 3);
- ***Texture Alignment Mode***, utilizzabile per modificare l'aspetto delle Texture degli oggetti creati in UDK;
- ***Mesh Paint Mode***, avvia il Tool di colorazione delle Mesh senza Texture;
- ***Static Mesh Mode***, avvia il Tool per la modifica di alcune impostazioni per le Static Mesh della scena;
- ***Landscape Mode***, tool di terrain editing ultimamente introdotto da UDK, in grado di fornire qualche strumento in più del Terrain Editing Tool;
- ***Foliage Mode***, tool per decorazione ambientale;

Nota: gli ultimi due elementi descritti sono un'espansione del Terrain Editing, che permette di svolgere gli stessi compiti ma in maniera meno elaborata.

Brushes: fornisce delle Brush di forme uguali a quelle rappresentate dalle icone che possono porsi come punti di partenza per la modellazione di alcuni oggetti di scena direttamente in UDK.

CSG: icone scorciatoia per le operazioni di addizione, sottrazione e intersezione delle Brush.

Select: cambia il tipo di selezione degli oggetti della scena.

Go to: visualizza per intero gli oggetti selezionati.

Capitolo 2 – Integrazione degli strumenti

In questo capitolo andremo a parlare di come è possibile integrare due strumenti quali Autodesk Maya, software di modellazione 3D, ed UDK, software utilizzato per la creazione di veri e propri videogiochi che sfruttano oggetti 3D. Affronteremo il discorso in maniera diretta, vedendo subito con un semplice esempio come sia possibile questa "collaborazione" tra software.

2.1 – Maya to UDK

Una volta creato un modello 3D di un oggetto qualsiasi tramite Autodesk Maya, è possibile modificarlo aggiungendo materiali e texture di alto livello, fino a poter effettuare dei rendering dell'oggetto (o di una scena), creando immagini davvero fotorealistiche, in grado persino di ingannare un occhio non allenato a notare le differenze tra una foto reale e un'immagine renderizzata. Tramite questo software però, come detto in precedenza, non risulterà possibile la creazione di una scena con la quale un qualsiasi utente potrà interagire. È possibile fare ciò con UDK, che però non permette effettivamente di modellare oggetti di ottima fattura e soprattutto lo strumento di modellazione di UDK non risulta affatto comodo e sviluppato rispetto a quello fornito da Maya.

Esiste un modo per integrare i due strumenti in maniera da ottenere realtà virtuali in UDK costruite con mesh create in Maya?

La risposta a questa domanda si può trovare nel plugin *Actor X [ACTX]*, fornito dal sito ufficiale di UDK. Infatti è previsto che gli sviluppatori utilizzino altri programmi per la realizzazione degli oggetti che andranno a comporre le scene di un videogioco creato con UDK. Oltre alla versione di *Actor X* da integrare con Maya, infatti, esiste una versione dello stesso plugin per il software di modellazione *3DS Studio Max*. Quest'ultimo, insieme ad Autodesk Maya, permette lo sviluppo di mesh di alto livello, il massimo che si può trovare nel mercato dei software del genere.

Actor X è un plugin e come tale è integrabile con Maya, tramite appositi comandi per l'inserimento di plugin esterni, che sono reperibili in grande quantità in rete. In particolare Actor X permette di esportare mesh create in Maya, traducendo tutte le informazioni di queste (vertici, archi, facce, materiali,…) e creando file in formato .ASE importabili all'interno dei package di progetto in UDK (questo nel caso di semplici mesh statiche). Permette anche di esportare da Maya le animazioni create e le Skeletal Mesh, le quali non sono altro che mesh dotate di uno *scheletro* e quindi, come si può immaginare, in grado di muoversi a seconda di come è stato deciso dal

modellatore. Tramite l'importazione di Skeletal Mesh è persino possibile creare oggetti distruttibili, in grado di andare in frantumi. Tutto ciò, con un'attenta costruzione in Maya, risulterà molto realistico poi in UDK. Sempre tramite esportazioni di Skeletal Mesh è possibile creare oggetti di tipo Cloth per creare poi oggetti che simulano la stoffa (bandiere, teli, vestiti). Cosa molto importante è la capacità di Actor X di importare la suddivisione dei livelli di dettaglio delle mesh: una volta creato un oggetto e aggiunti materiali e texture alle generalmente numerose facce che lo compongono, andando ad esportare tramite Actor X e poi importando in UDK, noteremo che le informazioni della suddivisione delle facce in materiali differenti sarà memorizzata. Unica pecca, se così la si vuole chiamare, la si trova nel fatto che non vengono importati direttamente anche i materiali e le texture utilizzate in Maya, ma questa operazione andrà fatta in un secondo momento; infatti UDK utilizza un tool molto particolare per la creazione dei materiali, che vedremo in seguito. Comunque le informazioni sul texturing vengono mantenute, quindi una volta creati i materiali adatti, sarà possibile aggiungerli ai LOD (level of detail) degli oggetti inseriti in UDK, ultimando la fase di importazione.

Una volta importata una Mesh in UDK è possibile operare in svariati modi su questa, modificando un numero davvero notevole di impostazioni dell'oggetto, tra le quali la possibilità di definire le collisioni, molto importanti per gli oggetti che compongono una scena: nessuno vorrebbe che nel proprio videogioco gli utenti possano attraversare un cancello o le mura di una casa. La creazione dei confini di collisione può essere fatta in automatico selezionando il livello di profondità di questa, rappresentabile da un semplice cubo che definisca questi confini o da un poliedro ben più complesso che ricalchi esattamente l'oggetto interessato. In questo ultimo caso bisogna però considerare che le collisioni vengono generate tramite la lettura dei vertici delle Mesh e quindi è necessario creare in Maya degli oggetti "semplici" [ACTX]. Vanno anche evitati oggetti con facce sovrapposte che risultano pessimi anche per quanto riguarda il rendering, che potrebbe mostrare degli effetti sgradevoli alla vista quali sfarfallio o simili. Quando si è intenzionati a costruire grandi strutture (ad esempio un castello completamente esplorabile) è assolutamente consigliato creare diverse Mesh che una volta unite nella scena andranno a costituire l'intero oggetto che si aveva in mente. In linea di massima è dunque consigliato non creare oggetti troppo complessi in blocco unico, ma suddividerli in fase di modellazione e poi ricomporli nella fase di Level Design, così da evitare anche rallentamento delle prestazioni del computer utilizzato. È importante anche porre attenzione durante la fase di texturing, in quanto un errato mapping delle texture potrebbe fornire degli errori durante la fase di proiezione delle luci di scena, fornendo pessimi risultati.

2.2 - Actor X

Vedremo di seguito come utilizzare il Plugin Actor X, in modo da poter combinare le potenzialità di Maya e UDK.

2.2.1 – Installazione del Plugin su Maya

Una volta scaricato il Plugin Actor X dal sito ufficiale di UDK [ACTX], questo dovrà essere inserito nella lista dei plugin utilizzati da Maya. Actor X può operare fino alla versione 2011 di Autodesk Maya e ancora non è possibile scaricarne la versione 2012.

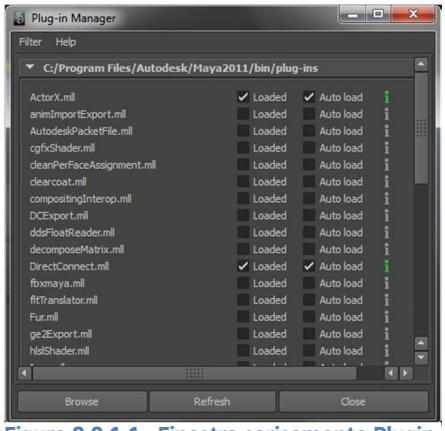


Figura 2.2.1.1 - Finestra caricamento Plugin

Window ➡ *Setting/Preferences* ➡ *Plug-in Manager*

Seguendo le istruzioni qui sopra descritte apriremo la finestra di selezione dei Plug-In. Una volta selezionato Actor X tramite pulsante Browse, dovremo spuntare le caselle Loaded e Auto Load, per indicare a Maya che il plug-in deve essere utilizzabile in questa sessione e dovrà essere caricato automaticamente nelle successive, per evitare di dover rieseguire l'operazione.

Ora Actor X potrà essere utilizzato in tutte le sue funzioni e sarà richiamabile tramite la Barra MEL di Maya. MEL sta per Maya Embedded Language ed è il linguaggio di scripting utilizzato in Maya, spesso utilizzato per velocizzare alcune operazioni e per richiamare i plugin installati.

2.2.2 – Funzionalità di Actor X

Una volta installato il plugin e creata una mesh di qualsiasi tipo che raggiunga possibilmente gli standard descritti in precedenza (una certa "semplicità" e un buon texturing), potremo richiamare Actor X, inserendo un comando su MEL.

I comandi principali per l'utilizzo di Actor X sono:

- *Axmesh* : utilizzato per l'esportazione di mesh statiche;
- *Axmain* : per l'esportazione di animazioni o skeletal mesh;
- *Axoption* : utilizzabile per modificare le opzioni di Actor-X.

Inserendo le sopra citate stringhe nella barra MEL si apriranno tre differenti finestre tramite le quali sarà possibile svolgere differenti operazioni che principalmente

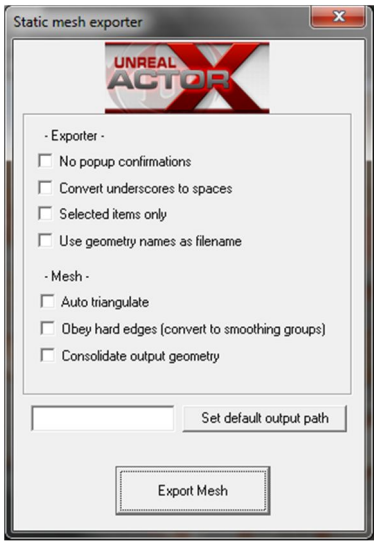


Figura 2.2.2.1 - Static Mesh Exporter

serviranno a definire il metodo di esportazione degli oggetti realizzati. Digitando axmesh visualizzeremo questa particolare finestra, tramite la quale è possibile definire alcune opzioni di esportazione, ovvero:

- Non ricevere conferme dell'esportazione;
- Convertire "_" in spazi (non utilizzato);
- Esportare solo l'oggetto selezionato;
- Usare il nome dell'oggetto come nome file;
- Triangolazione automatica;
- Arrotondare gli spigoli;
- Consolidare la geometria di output.

Una volta premuto il pulsante Export Mesh, sarà possibile salvare il nostro file in formato ASE, rendendo possibile l'importazione di questo in UDK.

Tramite l'inserimento di *axmain* invece apriremo lo Skeletal Exporter, tramite il quale potremo esportare delle Skeletal Mesh. Ovviamente l'utilizzo di questo comando di Actor X non avrà alcun effetto su Mesh statiche. La finestra dà la possibilità di selezionare la cartella di output dell'operazione di esportazione e il nome della Mesh creata. In caso sia necessario si potranno anche esportare delle animazioni definendo nome del file, della sequenza di animazione e il range, ovvero la quantità di frame da considerare nell'esportazione. Tramite l'Animation Manager (sempre raggiungibile tramite lo Skeletal Exporter) sarà possibile gestire al meglio l'animazione che si desidera esportare.

Tramite *axoptions* sarà invece possibile gestire tutte le opzioni di esportazione generali che sono comunque modificabili anche tramite *axmain* e *axmesh* al momento effettivo dell'esportazione dell'oggetto considerato.

Vedremo nei prossimi capitoli un esempio di creazione di mesh statiche in Autodesk Maya 2011 e come importare in UDK un file in formato ASE.

2.3 – Regole di Modellazione e Texturing in Maya

Quando si intende modellare un qualsiasi oggetto in Maya, pensato in maniera da dover essere poi esportato in software di sviluppo quali UDK, è necessario prestare molta attenzione alle operazioni che si svolgono sulla nostra mesh [GSWM10].

In particolare bisogna porre attenzione durante le operazioni di estrusione e simili le quali, se eseguite in maniera sbagliata, porteranno alla creazione di oggetti con errori che causeranno poi pessime interpretazioni da parte del motore di rendering di UDK.

Principalmente bisogna dedicare la propria attenzione sui seguenti punti:

- Non sovrapporre o avvicinare troppo facce diverse della stessa Mesh.
  Questo potrebbe portare infatti a errori in fase di proiezioni delle luci, i quali sono in alcuni casi ignorabili, ma in altri casi portano a spiacevoli errori di resa grafica e sfarfallio delle facce delle mesh;
- Non creare oggetti complessi (con geometrie complicate) e con errori di modellazione: in questo caso potrebbe avvenire persino che lo Static Mesh Exporter non funzioni o restituisca errori in fase di Export;
- Eseguire un corretto UV Mapping in fase di texturing, per evitare un'errata elaborazione delle luci e delle ombre sull'oggetto in questione.

CORRETTO UV MAPPING

Con UV *Mapping* si indica la fase nella quale il modellista, tramite determinate operazioni, crea un'immagine 2D a partire dalla mesh 3D, in maniera da indicare poi in quale modo le Texture dovranno essere impresse nel nostro modello. Per effettuare un corretto UV Mapping è necessario che nessuna faccia della mappa sia sovrapposta, neanche per una piccola parte [GSWM10]. Ciò porterà a delle incongruenze in un ambiente virtuale nel quale vengono proiettate delle luci dinamiche. Per fare un esempio, basti considerare un oggetto quale un cubo; questo ha sei facce. Se durante la fase di UV Mapping si sovrapponessero tutte e sei le facce, in fase di proiezione dei fotoni in UDK, innanzitutto saranno rilevati errori di sovrapposizione delle UV; inoltre il nostro cubo avrà probabilmente un'illuminazione uniforme su tutte le facce, nonostante le ombre proiettate dall'ambiente. Oppure, in caso di sovrapposizione non completa delle facce, vi sarà un'illuminazione troppo differente ,non graduale o scollegata. Per evitare simili errori è necessario collegare correttamente le facce fra loro tramite i loro *edge*, così come risultano collegate nella Mesh 3D.

2.4 Import di Static Mesh in UDK

Creata la Static Mesh desiderata tramite Autodesk Maya, comprensiva di UV Mapping, potremo finalmente richiamare il Plugin Actor X tramite barra MEL [ACTX] per l'effettiva esportazione. Digitando *axmesh* apparirà in Maya la finestra per la gestione delle opzioni di Static Mesh Exporting. Ovviamente non esistono opzioni che vanno sempre utilizzate (come la triangolazione automatica), la scelta del loro utilizzo sta al modellatore. Una volta premuto il pulsante *Export Mesh* verrà creato nella cartella da noi scelta un file in formato .ASE, supportato da UDK.

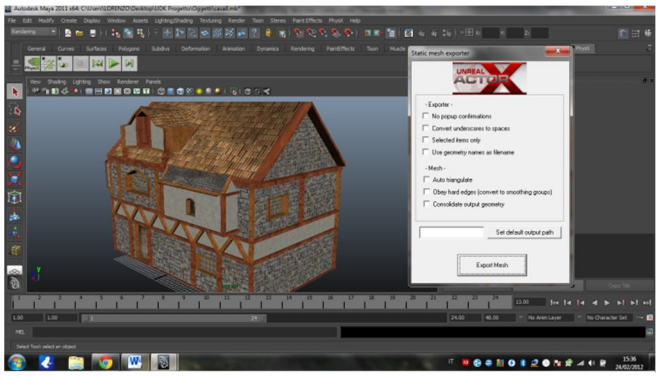
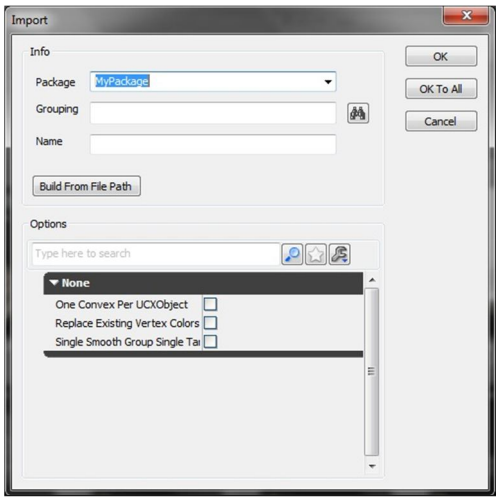


Figura 2.4.1 - Maya e Actor X, esportazione

Potremo ora avviare UDK. Per importare una Mesh da un file esterno sarà necessario aprire il Content Browser (Ctrl+Shift+F), che non è altro che un'interfaccia che dà allo sviluppatore la possibilità di selezionare un qualsiasi elemento del Package ufficiale di UDK per poi utilizzarlo nella scena che si sta creando. Il Content Browser contiene inizialmente una buona quantità di materiale (dalle Mesh create da Epic Games a svariati effetti sonori) che però risulta monotematico. Ciò è dovuto al fatto che UDK è predisposto per la creazione di giochi FPS (sparatutto in prima persona) di ambientazione fantascientifica futuristica. Anche gli effetti sonori disponibili sono di questo genere, così come alcuni materiali. Qualche Static Mesh richiama invece il tema medievale. Resta però il fatto che UDK può essere poi personalizzato e utilizzato per la creazione di applicazioni completamente differenti, quindi è possibile aggiungere ai contenuti iniziali dei contenuti esterni, che potranno essere inseriti in un Package di progetto per distinguerli da quelli standard. È possibile anche inserirli nelle cartelle dei contenuti ufficiali, ma per una distinzione e per avere una più rapida accessibilità è consigliato distinguerli da questi, in quanto inizialmente il Content Browser mette a disposizione davvero molto materiale.

Per effettuare l'importazione basterà cliccare con il tasto destro su uno spazio vuoto del nostro Content Browser e selezionare *Import*. Selezionando il file desiderato si aprirà una finestra tramite la quale potremo selezionare alcune opzioni di importazione, similmente a come accadeva in fase di esportazione con Actor X in Maya. Le opzioni si differenziano a seconda del tipo di importazione che si intende fare. Come vedremo più avanti infatti, quando si intende, ad esempio, importare una

nuova texture da utilizzare per la creazione di un materiale, avremo a disposizione opzioni completamente diverse da quelle fornite nell'importazione di una Static Mesh. Comunque UDK riconosce in automatico il tipo di file che si intende importare, dunque fornirà le giuste opzioni e funzionalità a seconda del tipo di formato. In generale avremo:



2.4.2 - Static Mesh Import, UDK

Package: utilizzato per indicare in quale package andremo a inserire il file da importare. Si può selezionare un package già esistente oppure scrivere il nome di uno non esistente. Nel secondo caso, in automatico, verrà creato il nuovo package.

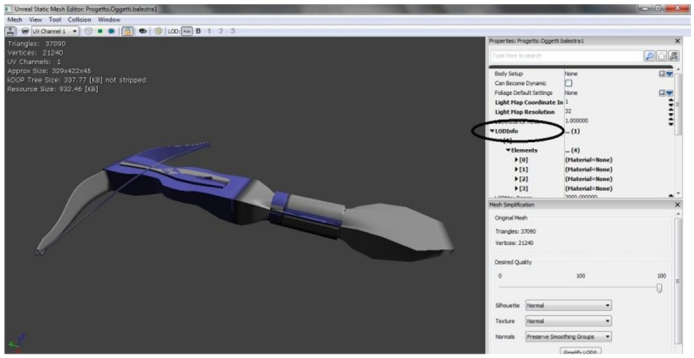
Grouping: utile se si vogliono creare dei sottogruppi all'interno del package.

Name: indica il nome che avrà l'oggetto una volta importato.

Questi tre campi sono comuni per ogni genere di importazione. Nel caso specifico, nell'immagine 2.4.2 vediamo le opzioni mostrate nell'importazione di una Static Mesh; queste servono ad indicare principalmente se abbiamo già generato le collisioni dell'oggetto in Maya e se vogliamo mantenerle in UDK. Una volta selezionato le opzioni di nostro gradimento, cliccando il tasto OK, completeremo la nostra importazione. Da questo momento in poi sarà possibile trovare nel Content Browser del nostro progetto l'oggetto in questione (quando si va a creare un progetto personalizzato conviene sempre creare anche un Package personalizzato all'interno del Content Browser).

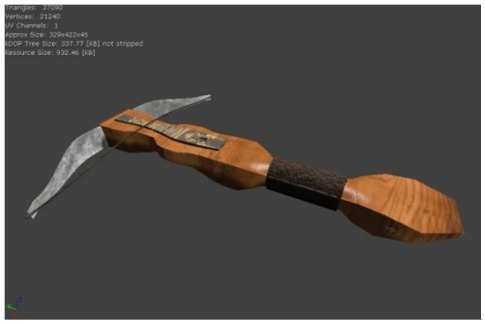
Come già detto però la Static Mesh non conserverà le texture inserite in Maya. Infatti, come già accennato, l'esportazione manterrà solo le informazioni sul posizionamento e sulla suddivisione delle texture, ma non manterrà i collegamenti ai file esterni utilizzati. Quindi ciò non significa dover ricreare da capo il texturing dell'oggetto, ma ci si dovrà applicare per la creazione dei materiali desiderati oppure si potranno utilizzare i materiali già compresi nell'originario Content Browser. Infatti materiali con stessa definizione e risoluzione possono essere scambiati se il texture mapping è stato fatto in maniera corretta (questa affermazione non sempre però risulta veritiera) o in generale se lo si desidera.

2.5 Gestione dei LOD e creazione di Collisioni



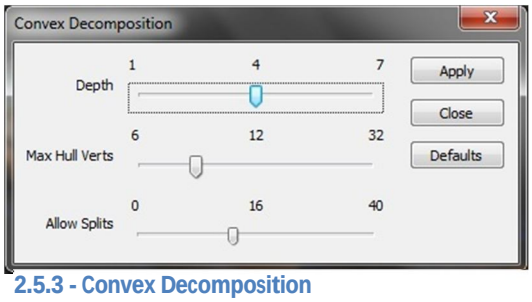
2.5.1 - Static Mesh senza Texture

Una volta importato un oggetto creato in Maya, questo apparirà come nella figura 2.5.1 ovvero senza materiali applicati. Nella sezione di destra dell'Unreal Static Mesh Editor, che si apre con un doppio click sulla Mesh desiderata nel Content Browser, troveremo diverse opzioni, utili per la gestione delle collisioni, della fisica e dei LOD. LOD sta per Level Of Detail e rappresenta i vari livelli nei quali è suddivisa una Mesh [UDKDOC]. Un livello rappresenta un diverso materiale applicabile alla nostra Mesh. Selezionando *LODInfo* vedremo in quanti livelli è divisa la mappatura dell'oggetto. Selezionando uno dei nuovi elementi nella voce *Material* potremo inserire il materiale voluto. Per fare ciò è necessario aprire il Content Browser e selezionarne uno, poi, tornando nella finestra dell'Editor, premere il pulsante  di fianco alla finestra di inserimento del *Material*. A questo punto ripetendo l'operazione con i diversi livelli (e con diversi materiali, se necessario) otterremo l'oggetto completo con Texture applicate (per la creazione di materiali da texture 2D esportate vedere il capitolo 4). L'oggetto può ora definirsi completo in quanto le modifiche inerenti l'aspetto sono state completate.



2.5.2 - Static Mesh Completa

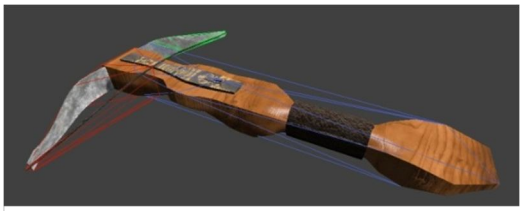
Ovviamente se la Static Mesh in questione deve presentare anche delle collisioni, a meno che queste non siano già state importate da Maya, andranno aggiunte selezionando una delle opzioni dal pulsante *Collision* che si può trovare nella barra del menù in alto nella finestra di Editing. Procedendo dall'alto verso in basso nella finestra di queste opzioni raggiungeremo un più dettagliato e complicato livello di gestione delle collisioni. Per la creazione di oggetti con collisioni realistiche i quali magari si desidera utilizzare come componenti della scena in grado di seguire le regole del motore fisico e di risultare credibili all'utente, è necessario generalmente fornire un livello di collisioni profondo. Si può fare ciò selezionando *Auto Convex Collision* e selezionando i valori desiderati.



2.5.3 - Convex Decomposition

Apparirà la finestra di Convex Decomposition nella quale potremo selezionare tre opzioni, che definiscono i valori per i quali sarà creato il poliedro convesso rappresentante i confini e quindi le collisioni dell'oggetto. In particolare potremo definire:

- *Depth:* la profondità della decomposizione della Mesh e delle collisioni. Depth rappresenta effettivamente la complessità del poliedro risultante;
- *Max Hull Verts:* numero massimo di vertici per il poliedro convesso rappresentante le collisioni;
- *Allow Splits*: suddivisioni del poliedro risultante consentite.


2.5.4 - Poliedro di collisione

Selezionati i valori desiderati e confermati con *Apply*, la nostra Mesh avrà ora confini di collisione rappresentati da un poliedro generato tramite lettura dei vertici della stessa, applicando poi le opzioni da noi richieste. Tramite la pressione del tasto nella finestra dell'Editor potremo visualizzare questo poliedro, rappresentato con diversi colori. Nell'immagine 2.5.4 viene mostrato un caso nel quale non si è desiderato ricreare delle collisioni molto realistiche, infatti il poliedro non segue esattamente la Mesh. Si può fare ciò nel caso un oggetto risulti piuttosto piccolo: in questo caso nel momento dell'esplorazione del nostro ambiente virtuale, sarà molto difficile notare che l'oggetto in questione non collide in maniera corretta con gli altri oggetti. Si può utilizzare un livello di collisioni non complesso anche nei casi in cui non si intenda fornire a determinati oggetti una certa importanza nell'ambiente, così da non appesantire troppo il carico di lavoro della scheda grafica dell'utente che ha il gravoso compito di calcolare la reazioni di tutti i Rigid Body (Static Meshes per le quali vanno calcolate le collisioni in fase di gioco [UDKDOC]), oltre a gestire luce, animazioni, effetti grafici, rendering dei materiali e numerose altre funzionalità dell'Unreal Engine 3.

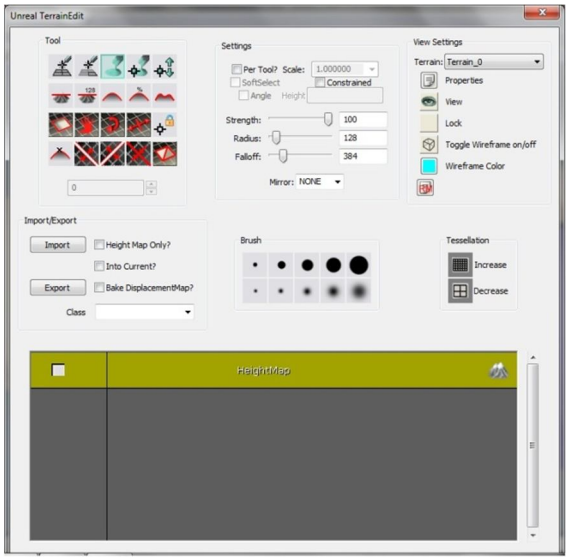
Come ultima nota è utile ricordare che una volta create delle collisioni per una Mesh, queste possono poi essere disattivate o attivate a piacimento una volta inserito l'oggetto in questione nella nostra scena. Ciò risulta utile nel caso un oggetto debba comparire più volte in scena e solo in certi casi si debbano gestirne le collisioni.

Capitolo 3 – Terrain Editing e Fluid Surface

Una fase molto importante che riguarda la creazione di una scena in UDK è rappresentata dalla fase di generazione del terreno che sarà poi la base dell'intero scenario. Il terreno, una volta completato potrà essere riempito con tutti le Mesh create o già disponibili. Dovrà poi essere correttamente illuminato e potranno essere inseriti gli eventi con i quali l'utente finale avrà a che fare. La Terrain Surface sarà inizialmente rappresentata da un oggetto di tipo Plane (un piano) senza alcuna texture. Tramite lo strumento di Terrain Editing messo a disposizione da UDK, questo piano potrà poi essere diviso in più sezioni per una maggiore definizione dell'ambiente, decorato con Materiali e Mesh, deformato per rappresentare montagne, colline, fosse e tutto ciò che è possibile trovare nel mondo reale in natura. L'inserimento in scena di superfici liquide invece è affidato alla gestione delle Fluid Surface, che non sono altro che dei veri e propri Plane i quali però sono stati studiati per simulare una superficie liquida. Tramite alcuni particolari oggetti ed eventi darà poi possibile gestirne le proprietà, a seconda delle necessità, come l'increspatura, la forza delle onde, la reazione alle forze esterne o al passaggio dell'utente.
Pertanto, quando si è intenzionati a cominciare a creare la propria scena è sempre necessario partire in particolare dalla fase di Terrain Editing. Il terreno di norma va completato prima dell'inserimento delle mesh che vanno a decorarlo e a completarlo, in quanto modifiche successive (creazione di nuove deformazioni) potrebbero costringere a rimuovere o spostare le Mesh inserite. Ovviamente questo non è un problema nel caso si cambi idea sull'aspetto che deve avere il nostro Terrain in quanto sarà possibile modificarlo a piacimento in qualsiasi momento, ma conviene sempre studiarsi da subito il futuro posizionamento degli oggetti di scena, in maniera da evitare di dover apportare troppe fastidiose modifiche. Questo non conta per la fase di "colorazione" del terreno, in quanto l'applicazione di diverse texture su di esso non implicherà alcuna modifica a livello di deformazioni. Inoltre, quando si vuole creare un'ambientazione chiusa e quindi non un paesaggio naturale, la creazione del terreno risulta strettamente collegata alle mesh che andranno inserite nella scena, quindi si può procedere parallelamente in modellazione del terreno e applicazione delle Mesh, oppure si può far si che il terreno non sia necessario ma che sia rappresentato dalle Mesh stesse.
Andremo ora a studiare i due elementi qui velocemente presentati e a vedere come sarà possibile crearli e gestirne le proprietà.

3.1 Terrain Editing

Come già detto, un terreno è un oggetto simile a un piano che è però predisposto per essere trattato con L'Unreal TerrainEdit, lo strumento per il Terrain Editing messo a disposizione da UDK. L'oggetto di tipo Terrain, come tutti gli oggetti che possono essere inseriti in una mappa, può essere trovato nella Actor Classes, che contiene la lista suddivisa in categorie di tutti gli Actor disponibili. Un Actor (attore) non è altro che un componente della scena. Ce ne sono di diversi tipi e ognuno ha la sua utilità. Infatti sono attori sia gli elementi di tipo *lights* (luci) che gli elementi di tipo *trigger* (innesco o grilletto); la diversità tra questi tipi di attore è piuttosto evidente.
Selezionando *Actor Classes* da *View – Browser Window* e cercando la categoria *Uncategorized* è possibile trovare l'attore *Terrain*. Selezionandolo e trascinandolo nella nostra scena inizialmente vuota creeremo un oggetto di questo genere.
Apparirà inizialmente come un piano, di piccole dimensioni e non texturizzato. Per modificarlo dovremo aprire l'Unreal TerrainEdit premendo il tasto  sulla sinistra della finestra di lavoro [UDKDOC].



3.1.1 - Unreal TerrainEdit

Come prima cosa da fare si consiglia di regolare le dimensioni del Terrain a seconda della scena che si ha in mente di creare. Per fare ciò basta selezionare *Properties* il quale aprirà una finestra dalla quale impostare tutte le proprietà disponibili per l'oggetto. Per regolare le dimensioni è necessario aumentare il valore dei campi *Num Patches X* e *Num Patches Y* oltre che il campo *Max Component Size* che determina la dimensione delle singole componenti del terreno, che viene infatti diviso in un numero di sezioni date dai primi due campi citati.
Una volta scelti i valori desiderati potremo cominciare a lavorare per creare il nostro terreno. Nella figura 3.1.1 viene mostrato l'Unreal TerrainEdit, il quale comprende un certo numero di opzioni. Tramite queste è possibile modellare il nostro terreno e decorarlo [UDKDOC].

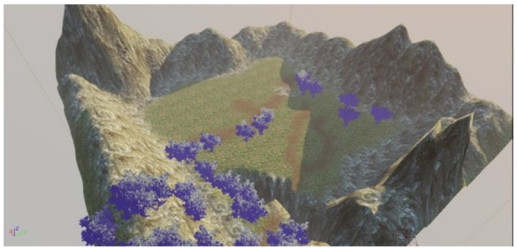
***Tool*:** è qui presente una serie di pulsanti che serve a definire in che modo si agirà nel modificare il Terrain. Tutti i pulsanti definiscono un diverso tipo di

deformazione che si andrà ad attuare con la pressione di *Ctrl* e del tasto sinistro del mouse sul terreno. Vi sono opzioni per arrotondare le deformazioni o per creare delle irregolarità volute. Si può inoltre gestire la visibilità di certe zone del terreno, potendo così cancellarne alcune non utili in quanto magari totalmente coperte da oggetti di scena;

- ***Settings*:** definiscono in che modo gli strumenti selezionati in *Tool* agiscano sul Terrain, in particolare con quale intensità (*Strenght)* e in quanto spazio debbano fare effetto. L'area di azione della pressione del mouse e relativa modifica è determinata dal *Radius* che rappresenta il raggio di una circonferenza. *Falloff* invece rappresenta in quale area l'effetto dovrà procedere attenuandosi rispetto alla zona delimitata da Radius. Questo può essere utilizzato ad esempio per definire delle leggere salite, ponendo un raggio molto piccolo e un falloff molto grande;
- ***View Settings*:** si può qui modificare la modalità di visualizzazione del terreno in fase di modellazione, mostrandone il wireframe o decidendo di nasconderlo. Inoltre da questo menù è possibile raggiungere le proprietà del Terrain, come visto in precedenza;
- ***Import / Export:*** permette di importare o esportare terreni;
- ***Brush:*** permette di selezionare delle dimensioni preimpostate per l'area di azione del tipo di deformazione selezionata sul Terrain;
- ***Tassellation:*** tramite i due pulsanti qui presenti è possibile aumentare o diminuire il livello di suddivisione del Terrain. Un'alta suddivisione permette un'ottima qualità e migliori possibilità di lavoro sul Terrain, permettendo dunque di creare un terreno più realistico e più particolareggiato;
- ***Finestra di selezione DecoLayer e Terrain Material:*** la sezione in basso della finestra di Unreal TerrainEdit permette di selezionare i materiali o le mesh di decorazione del terreno. Inizialmente è possibile trovare solo l'*Height Map* che va selezionato quando si intende modellare e non decorare.

Per aggiungere nuovi elementi per la decorazione del Terrain è necessario selezionare un Materiale o una Static Mesh dal content Browser e poi, cliccando con il tasto destro nella sezione, selezionare *New Terrain Material* o *New DecoLayer*. Un Terrain Material verrà utilizzato per "colorare" il terreno, mentre il DecoLayer servirà ad aggiungere Mesh alla scena che sono però strettamente collegate al terreno e lo decorano. Generalmente tra i DecoLayer si inseriscono alberi, piante, rocce o comunque oggetti presenti in grande quantità sulla scena e che il modellatore considera utile inserire in questo modo. Nelle proprietà del Terrain, alla voce

DecoLayers è possibile determinare con quale intensità le Mesh del DecoLayer selezionato debbano apparire nel momento dell'utilizzo del Brush. Si può anche indicare di seguire una certa casualità nel posizionamento delle decorazioni.



3.1.2 - Terrain del progetto di tesi

È molto importante ricordare che le Mesh inserite tramite DecoLayer nel terreno sono totalmente diverse dalle Static Mesh inserite tramite Content Browser nella scena. Infatti l'utente finale non potrà interagire con le decorazioni, mentre potrà farlo, se desiderato dai programmatori, con le Static Mesh (una volta trasformatele in Actors corretti per il tipo di interazione desiderata). Nella figura 3.1.2 è possibile notare degli alberi. Questi sono decorazioni e non Static Mesh, dunque riempiranno la scena ma l'utente non potrà fare altro che osservarli, senza interagire. Per le decorazioni è comunque possibile definire delle collisioni, per evitare che il giocatore (o altri oggetti di scena) ci passi attraverso in caso ciò non sia desiderato.

3.2 Fluid Surface

Nella creazione di un terreno non sono presenti funzionalità per la creazione di superfici acquatiche o in generale superfici fluide. Questo si può fare tramite l'inserimento in scena di un altro tipo di attore, chiamato *FluidSurfaceActor* [UDKDOC].
Un attore di questo tipo è una superficie che possiede la proprietà di comportarsi come un fluido. L'oggetto in questione è da subito disponibile in UDK e il progettista non dovrà far altro che inserirlo nella sua scena dove desidera.
Come per i Terrain, inizialmente una Fluid Surface avrà dimensioni ridotte e nessuna texture applicata. Ovviamente ciò può essere cambiato visualizzando e modificando le proprietà dell'oggetto. Tra queste possiamo trovare l'intensità delle increspature della superficie, la lunghezza d'onda, la forza.
Una Fluid Surface da sola non calcola le interazioni con l'attore guidato dall'utente finale all'interno dello scenario, ma ciò va fatto tramite l'aggiunta nel livello di un Actor chiamato *FluidInfluenceActor*.
Questo oggetto, definisce in che modo la Fluid Surface dovrà comportarsi alla vicinanza o al contatto con il giocatore. Si possono impostare quattro differenti generi di influenza:

- ***Fluid Flow:*** genera increspature casuali nella superficie, ottimo per simulare la presenza di un vento leggero nell'ambiente;
- ***Fluid Raindrops:*** simula la caduta di gocce nella superficie, utilizzato per simulare la pioggia e il suo effetto sull'acqua;
- ***Fluid Wave:*** gestisce le onde che si generano all'avvicinarsi del giocatore;
- ***Fluid Sphere:*** deforma la superficie in maniera simile al Fluid Raindrops, in dimensioni però maggiori. Utile per simulare la caduta di oggetti di modesta grandezza in acqua o per gestire fluidi di altro genere come ad esempio lava vulcanica, in quanto effettivamente la superficie si comporta più come un vero e proprio fluido più che come un liquido.

Per completare la creazione di una Fluid Surface è necessario aggiungere un materiale di qualche tipo alla stessa. Il materiale può essere di ogni genere, ma ovviamente se si intende simulare una superficie liquida è necessario un materiale semitrasparente.
Per aggiungere un materiale ad una Fluid Surface basta selezionarlo nel Content Browser, aprire la finestra delle proprietà della Fluid Surface e in *Fluid Surface Component* premere il tasto  .

In conclusione le Fluid Surface possono essere sfruttate per la creazione di superfici acquatiche, ma non rappresentano un vero e proprio liquido, in quanto non sono composte da particelle d'acqua, ma da un'unica superficie. Inoltre bisogna obbligatoriamente inserire le Fluid Surface in un contenitore con quattro pareti se si desidera vederlo in fase di rendering e non è possibile creare una di queste superfici di forma diversa da quella quadrata.
Queste ultime sono le maggiori pecche di questo altrimenti ottimo strumento fornito da UDK.

3.3 – Landscape Mode e Foliage Mode

Il Terrain Editing da solo permette di creare degli ambienti di un certo livello piuttosto facilmente, ma UDK ha introdotto recentemente degli strumenti chiamati Landscape Mode e Foliage Mode [UDKDOC] che espandono il Terrain Editing, permettendo di creare degli ambienti di maggiore qualità e con più precisione. Ovviamente questo comporta il dover svolgere anche un lavoro più lungo e paziente, quindi la scelta di una o dell'altra modalità di creazione di un terreno dipende strettamente da cosa si vuole realizzare e da quanta attenzione di vuole porgere all'ambiente di contorno piuttosto che agli oggetti di scena.

Landscape Mode:

Questo Tool permette di creare un Landscape e poi di modificarlo, come avveniva nel Terrain Editing descritto nel capitolo 3.1. Differentemente da quest'ultimo però permette lavorare su terreni più vasti e creare effetti di erosione ambientale con degli appositi comandi. Se si vuole passare da un Terrain precedentemente creato ad un Landscape, è possibile tramite il comando di conversione della Landscape Mode. I comandi disponibili sono molto simili al Terrain Editing per quanto riguarda la loro disposizione. Ciò che cambia sono gli effetti che si ottengono e la maggiore precisione ottenibile da questo Tool. La Landscape Mode non permette di decorare il Landscape, ma ciò può essere fatto nella Foliage Mode.

3.3.1 - Foliage Mode

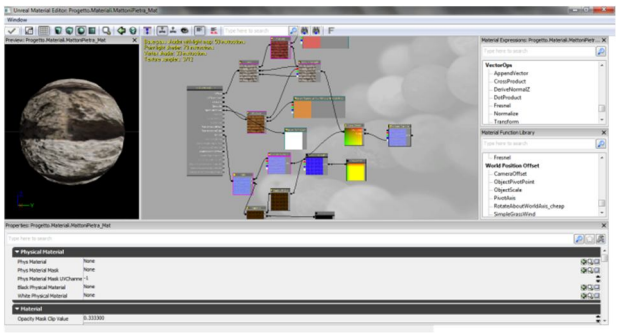
Foliage Mode:

Questa modalità permette di decorare l'ambiente, che sia un Terrain o un Landscape, utilizzando delle Static Mesh. Trascinando dal Content Browser alla sezione in basso della Foliage Mode (vedi figura 3.3.1) una qualsiasi Static Mesh, è possibile "disegnarla" nel nostro terreno. Se avessimo trascinato nella Foliage Mode una Mesh di un albero qualsiasi, selezionando il Tool *Paint* (in figura 3.3.1 è l'icona in alto sulla sinistra) potremo inserirne la quantità desiderata semplicemente trascinando il puntatore del mouse e contemporaneamente tenendo premuto ***Ctrl*** e il pulsante sinistro.

La Foliage Mode fa dunque quello che si facevo nel Terrain Editing con i *DecoLayer*, ma in maniera più rapida e decisamente più precisa.

Capitolo 4 – Creazione di materiali in UDK

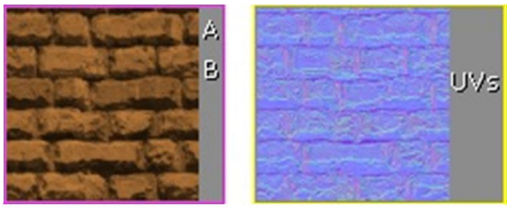
Parte fondamentale della modellazione è quella dell'applicazione delle texture all'oggetto creato [UDKMAT]. Ciò, come detto precedentemente, avviene in una prima fase in Maya, dove viene effettuato l'UV Mapping, tramite il quale si definisce in che modo dovranno e con quale distribuzione dovranno essere applicati i materiali nelle facce che compongono la Mesh. Una volta effettuata l'esportazione tramite Actor X, saranno conservate le informazione di mapping ma non le informazioni inerenti i materiali. UDK mette a disposizione un elaborato tool per la creazione di questi.



4.1 - Unreal Material Editor

L'*Unreal Material Editor* permette di generare dei materiali personalizzati (ce ne sono molti disponibili già da subito nel Content Browser, anche questi personalizzabili) tramite impostazione dei canali di diffusione, emissione, specularità, maschere e tutto ciò che definisce effettivamente un materiale realistico. Questo editor mette e disposizione un numero davvero elevato di comandi e opzioni, ma queste non vanno tutte conosciute obbligatoriamente per creare un materiali realistico o quasi. Tramite alcune opzioni è possibile impostare un materiale personalizzato in maniera da animarlo. Questo può essere sfruttato ad esempio per creare uno specchio d'acqua visibile magari in lontananza che non necessita dunque di Fluid Surface in quanto non vi saranno interazioni con l'utente. Si possono inoltre creare materiali trasparenti o semitrasparenti utilizzabili per creare foschia.

La creazione di un materiale di semplice aspetto risulta comunque molto intuitivo da comprendere e attuare, in quanto, quando si vanno a sfruttare delle Texture che rappresentano ad esempio mura, roccia, legno e comunque dei materiali del genere, non si lavora generalmente su animazioni ma soltanto sull'aspetto finale statico; si andranno dunque generalmente a sfruttare un numero limitato di comandi, che approfondiremo presto, mostrando un esempio pratico per comprendere il funzionamento di questo tool. Prima di tutto però, quando si intende creare un materiale personalizzato è necessario importare le texture 2D (un materiale può essere creato anche unendo più texture) nel package del progetto o in generale nel Content Browser.

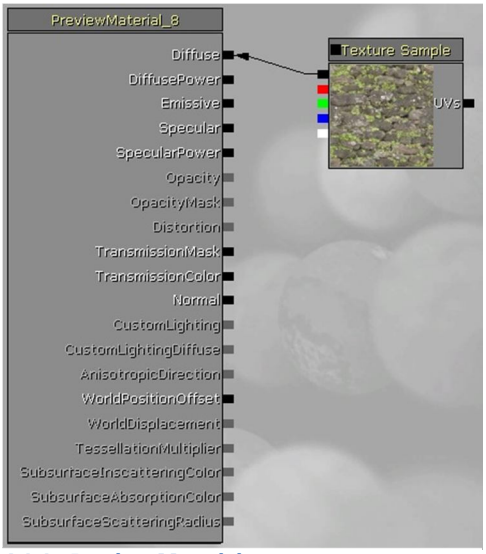


4.2 - Texture Base e Normal Map

Come per le Mesh è necessario cliccare con il destro in uno spazio vuoto del CB e selezionare *import*. Potremo dunque scegliere un file di tipo immagine e importarlo direttamente come Texture 2D.
È fondamentale che le dimensioni dell'immagine siano rappresentate da potenze di 2 e che siano uguali in altezza e larghezza (es: 512x512, 1024x1024); ciò è dovuto semplicemente al fatto che l'*Unreal Material Editor* supporta solo texture di questo formato. Nella fase di importazione potremo anche far sì che UDK crei già un materiale che sfrutti la stessa Texture, semplicemente mettendo una spunta nella casella con la dicitura *Auto-Create Material?*. Questo materiale sarà composto semplicemente dalla Texture importata, collegata al canale di diffusione. In questo modo avremo già creato qualcosa di utilizzabile su una Static Mesh, anche se sarà estremamente semplice e piatto. L'Unreal Material Editor permette infatti di fornire una certa profondità ai materiali creati, tramite l'unione di *Normal Map*, che possono essere generate a partire dalla texture d'origine tramite programmi come *Photoshop*.
Il Normal Mapping è utile per simulare una migliore qualità del materiale, fornendo la possibilità di rappresentare difetti realistici o ammaccature di questo senza l'utilizzo di Mesh appositamente deformate, semplificando dunque la modellazione e la complessità delle Mesh. Anche le normal map possono essere importate nello stesso modo delle texture classiche e vengono viste come tali (Texture 2D), ma verranno poi utilizzate nella creazione del materiale in combinazione alla texture da cui sono state generate. Inoltre in fase di importazione andrebbe generalmente cambiata la compressione della Texture da *TC_Default* a *TC_Normalmap* (nella sezione *CompressionSettings* dell'Import).
Va ricordato inoltre che all'interno della scena finale, il materiale potrà generare, se richiesto, specularità o riflessione e ciò dipende da come viene creato ma soprattutto dalle luci ambientali, dalla forza e dal colore di queste e anche dagli oggetti di scena. In generale è consigliato creare materiali simili per la stessa scena, ovvero evitare di creare alcuni materiali molto semplici e altri molto complessi, poiché una volta accostati potrebbero fornire alla scena un aspetto poco gradevole e una diffusione dei materiali per nulla uniforme.

4.1 Unreal Material Editor

Una volta importata una texture e creato un materiale (non ancora completo e utilizzabile ma editabile) partendo dalla stessa, potremo aprire l'Unreal Material Editor (che da ora chiameremo UME) semplicemente con un doppio click del mouse sul materiale che intendiamo modificare.
Inizialmente vedremo una *Preview* come nella figura 4.1.1 alla quale sarà collegata la semplice componente di tipo *Texture* Sample collegata nel canale *Diffuse*. Tramite questa tabella si va a definire l'aspetto finale del materiale, combinando svariate componenti fra loro e collegandole infine in uno dei canali di impostazione. Il collegamento viene fatto tramite delle frecce. L'UME è dunque un Tool ibrido: abbiamo infatti la parte visuale (appena descritta, molto semplice e immediata) e una parte testuale, dove principalmente andremo a impostare dei valori numerici o a selezionare delle opzioni tramite checkbox. Quest'ultima sezione varia a seconda della componente le cui proprietà vogliamo modificare e appare nella parte bassa dell'Editor una volta selezionatane una. Nel caso di un *texture sample,* non si può impostare altro che la locazione della texture alla quale vogliamo fare riferimento nel content browser.

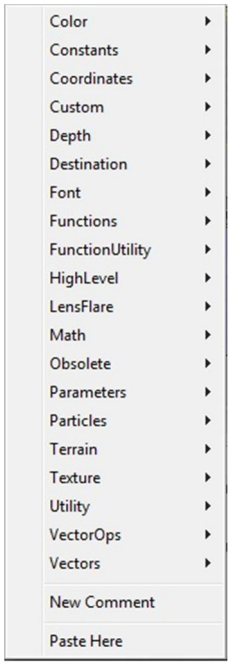


4.1.1 - Preview Material

Vediamo ora i comandi principali nella *preview material:*

- ***Diffuse:*** ciò che viene collegato a questo canale è ciò che il materiale diffonde nell'ambiente. Generalmente a Diffuse è collegata la Texture Sample principale collegata se desiderato ad altre componenti per modificarne alcune impostazioni;
- ***Diffuse Power:*** vanno qui collegata componenti di tipo *scalar* o in generale numeriche, che andranno a definire la forza del canale diffusivo;
- ***Emissive:*** impostazione del canale emissivo;
- ***Specular:*** impostazione del canale speculare;
- ***Specular Power:*** come il Diffuse Power, ma per l'attributo Specular;
- ***Normal:*** va qui collegata la Normal Map, modificata come si desidera.

Tramite la combinazione e la gestione degli attributi appena citati è possibile creare un materiale personalizzato che mantenga un certo realismo.



4.1.2 - Selezione Componenti UME

Per aggiungere componenti al nostro materiale basta cliccare con il tasto destro in uno spazio vuoto della finestra di lavoro e apparirà la finestra della figura 4.1.2. Le componenti sono numerose e suddivise in queste categorie. Si possono creare costanti, variabili, operazioni matematiche, parametri, vettori, tutte utili per modificare e gestire i materiali.

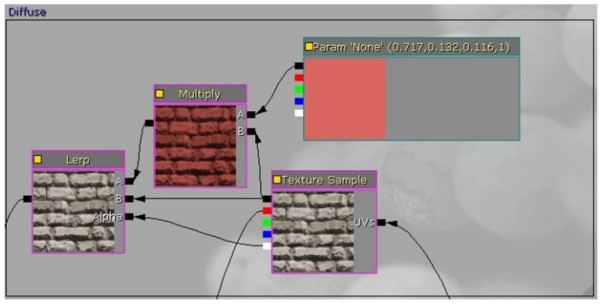
Dato che si tratta di un numero davvero vasto di funzionalità, studieremo ora la creazione di un materiale d'esempio dove vengono utilizzate alcune delle principali componenti utili alla creazione di qualcosa di realistico.

Il materiale che andremo a rappresentare sarà quello della figura 4.1.3. Si tratta di un muro di mattoni, utilizzato in seguito nel progetto per il texturing di mura e torri.

4.1.3 - Materiale Esempio

Creando questo materiale in particolare si è voluto raggiungere un certo realismo rappresentando al meglio la profondità e i difetti della pietra, unendo sapientemente la normal map generata in *Photoshop* ad alcune componenti disponibili in UME e sfruttando anche il canale speculare per tentare di rappresentare l'umidità della roccia di questo genere.

Per quanto riguarda il Diffuse del materiale, la *Texture Sample* è collegata come si vede nella figura 4.1.4 ad altre componenti.
Param, Multiply e Lerp [UDKMAT].

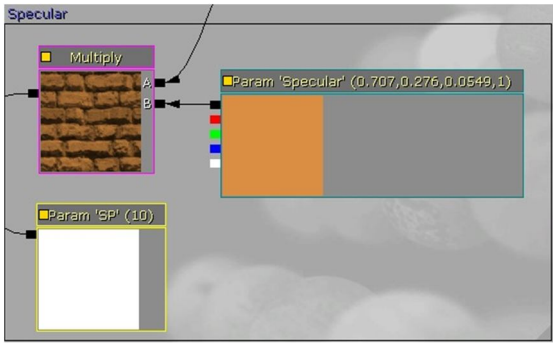


4.1.4 - Diffuse, impostazione

- ***Param***: rappresenta un parametro, in questo caso utilizzato per definire il colore come in figura;
- ***Multiply:*** moltiplicazione matematica;
- ***Lerp***: Interpolazione Lineare.

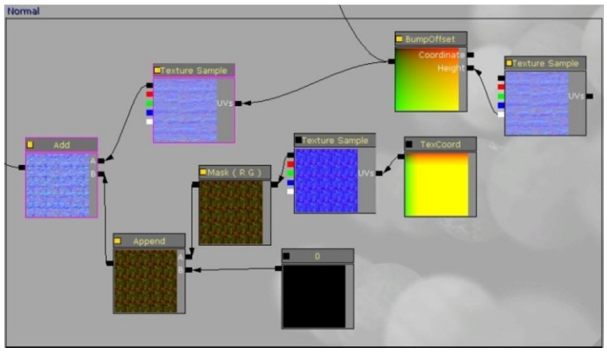
Nel caso di *Param*, si vanno a definire dei valori per i canali RGB per i colori rosso, verde e blu, e Alpha per gestire le trasparenze. In uscita si possono prendere i parametri separatamente o in un unico blocco tramite *Black Channel* (facendo partire i collegamenti dal quadrato nero). Dal componente *Param* è stato fatto partire un

collegamento al canale di entrata A di *Multiply*, che in B ha in ingresso il *Black Channel* del *Texture Sample*. *Multiply* fa esattamente ciò che il suo nome dice, ovvero moltiplica due valori in ingresso restituendo un risultato che viene in questo caso inserito in ingresso del *Lerp* in A. Come si può vedere, moltiplicando un parametro per una texture otterremo una modifica del colore di questa. *Multiply* comprende automaticamente che genere di componenti riceve in ingresso e il risultato dipende dal loro tipo. Collegando nel canale B di *Lerp* il *Black Channel* di *Texture Sample* e in alpha *l'Alpha Channel*, otterremo infine il nostro risultato, che consiste nella nostra Texture 2D iniziale con delle lievi sfumature di rosso, che non si notano però nel preview fornito da Lerp, ma solo in fase di gioco.



4.1.5 – Specular, impostazione

Per quanto riguarda il canale speculare del materiale, si è sempre sfruttato un *Multiply* che ottiene in ingresso un *Param* "*Specular*" di color arancione e la *Texture Sample* iniziale del materiale.
Il *Param* "*SP*" di colore bianco è collegato allo *Specular Power* e indica il valore di forze speculare che il materiale deve avere. In questo modo otterremo delle sfumature colorate nella riflessione. Il valore di "*SP*" non è molto elevato e fornisce un effetto "viscido" alla pietra. Ciò che viene riflesso nel materiale dipende ovviamente dall'ambiente circonstante e il valore 10 di "SP" non definirà una specularità evidente, quindi il materiale finale non avrà un effetto specchio, ma assorbirà e riemetterà poca luce fornita dall'ambiente con una leggera sfumatura arancione. Dunque questo effetto non è visibile nella preview dell'UME, ma andrà testato in fase di gioco [UDKMAT].



4.1.6 - Normal, impostazione

Una volta definiti i canali di diffusioni e specularità sarà il turno dell'impostazione del canale *Normal*. Tramite questo sarà possibile fornire profondità e realismo a un materiale che altrimenti risulterebbe piatto come un semplice disegno su foglio di carta.

Intervengono in questo caso parecchie componenti nuove e come *Texture Sample* non vengono più utilizzate le Texture principali del materiale ma le Normal Map Texture che devono essere create tramite appositi programmi quali *Photoshop.* Generalmente è necessario utilizzare un plugin fornito dalla *Nvidia* e di fa tutto con pochi semplici passaggi che però qui non andremo a descrivere.
Le nuove componenti utilizzate nella sezione del *Normal* sono (a partire da sinistra):

- ***Add:*** i due valori in ingresso vengono sommati. Come per tutte le operazioni matematiche di UME, l'operazione darà risultati differenti a seconda dei valori in ingresso;
- ***Append:*** permette di combinare dei canali insieme per creare un vettore contenente più canali rispetto all'originale;
- ***Mask:*** impostati i canali che si vogliono visualizzare (R G e B), maschera quelli selezionati, nascondendoli nel risultato finale;
- ***Constant:*** definisce una costante;
- ***TexCoord:*** definisce le coordinate delle Texture;
- ***Bump Offset:*** tramite questa particolare componente è possibile fornire l'illusione di profondità del materiale.

Tramite impostazione dei collegamenti come in figura, riusciremo ad ottenere la profondità del materiale e sfruttando la *TexCoord* con sempre la *Normal Texture* del nostro materiale sarà possibile aggiungere nuovi dettagli e difetti alla *Texture Sample* base collegata al *Diffuse.* Tramite *Add* si crea infatti una "nuova" texture che ne sovrappone due differenti.
Non esiste comunque un metodo unico per ottenere certi risultati. UME permette di combinare qualsiasi componente e provando questo Editor si possono sempre trovare nuovi metodi per la creazione di materiali che raggiungano la qualità che ci si attende, che dipende anche questa dal creatore dei materiali.
Quello che è stato mostrato è dunque solo un esempio di come creare un materiale verosimile ed è quindi giusto ricordare che esistono metodi diversi e spesso personalizzati, che variano a seconda dell'obbiettivo che si desidera raggiungere.

Capitolo 5 – Illuminazione

Nelle pagine che seguono andremo a vedere in che modo in UDK possono essere utilizzate le luci di scena e come queste influenzano gli oggetti che la compongono.
Tutto ciò sempre tramite degli esempi chiarificativi che mostrano il genere di luci utilizzate poi nella realizzazione del progetto di tesi.

5.1 – Le luci in UDK

L'illuminazione ambientale è parte fondamentale del level design: questa serve infatti a fornire realismo alla scena, introducendo fonti di luce in grado di proiettare le ombre degli oggetti, sia statici che dinamici. La luce contribuisce inoltre a creare atmosfera: capita spesso di vedere un videogioco nel quale i giochi di luce e ombra possono creare situazioni di tensione o fornire effetti molto gradevoli all'occhio dell'utente.

5.1.1 - Particolare del Progetto – Illuminazione e proiezione delle ombre

Quando si crea un nuovo progetto in UDK è possibile creare una scena vuota oppure decidere di utilizzare un scena con una luce ambientale preimpostata.
Vengono fornite luci impostate per simulare la luminosità mattutina, notturna, dell'alba e del tramonto. Nel caso dell'immagine 5.1.1 si può vedere un particolare del progetto di tesi, per il quale è stata utilizzata una luce preimpostata per un tramonto, poi modificata a livello di colorazione per renderla un po' più chiara. Inoltre ne è stata modificata l'inclinazione dei raggi proiettati. La selezione di una luce dominante preimpostata fornisce in automatico anche la creazione delle nuvole che si possono notare sempre nella figura 5.1.1.

In generale UDK fornisce quattro tipi di luci [UDKLS]:

- ***Directional Lights***: genere dei raggi di luce paralleli fra loro con raggio d'azione "infinito": non hanno un punto d'origine ma solo un'inclinazione che definisce in che modo questi raggi proietteranno poi le ombre. Può essere di tipo *Dominant* ovvero dominante (in questo caso ce ne sarà solamente una in scena) o *Toggleable (movibile).*

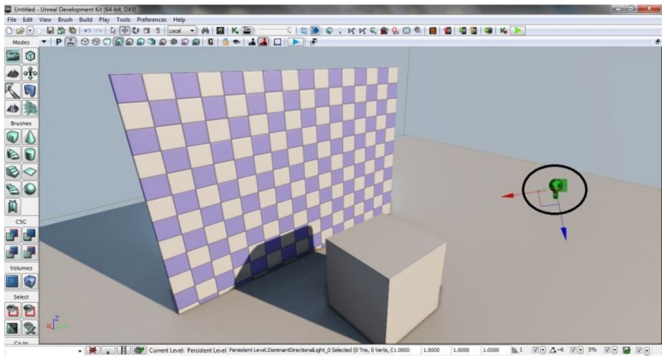
Point Lights: traducibile con "Luce puntuale", definisce una fonte di luce a partire da un punto. Come la *Directional Light* può proiettare ombre ma non viene utilizzata per generare raggi di azione di ampiezza potenzialmente infinita.

- ***Sky Lights***: definisce la luce ambientale generica che influisce su un certo volume spaziale, che va impostato dal progettista. Una Sky Light non proietta ombre.
- ***Spot Lights***: genera una fonte di luce che viene proiettata seguendo un volume di forma conica.

Tutte le tipologie di luce fornite da UDK possono essere personalizzate in ogni attributo che le costituisce, dal colore della luce che va proiettata al raggio d'azione, dalla generazione delle ombre alla qualità delle stesse. È sempre necessario fornire alla scena un *Lightmass Importance Volume* che indica a UDK in che area della scena si deve spendere più tempo a calcolare gli effetti delle luci. Questo può essere utilizzato per ottimizzare il calcolo da parte dell'Unreal Engine 3, così da evitare gravose operazioni e quindi rallentamenti in fase di gaming; infatti, limitando il volume di illuminazione alle sole zone utili della scena, si evitano calcoli inutili e si ottiene maggiore fluidità dal motore grafico.

5.2 – Aspetto e gestione delle luci in UDK

5.2.1 – *Directional Lights*



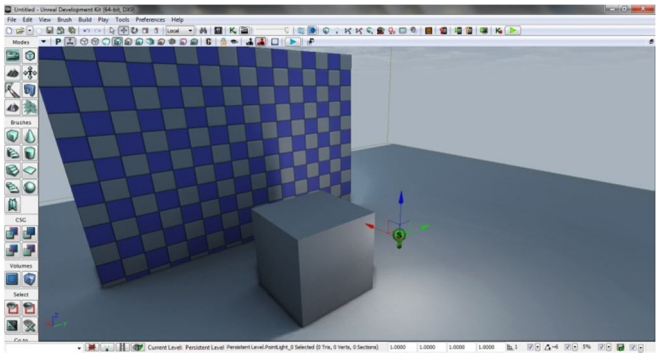
5.2.1.1 - Directional Light

La luce direzionale può essere utilizzata per simulare la luce del sole e generalmente viene utilizzata proprio per questo motivo. Ovviamente può essere utilizzata in qualsiasi altro modo venga in mente al Level Designer.

Come si può vedere dall'immagine 5.2.1.1 le ombre che vengono proiettate dalla luce (l'immagine mostra le luci di qualità *Production*, che sono effettivamente quelle che vedremo nella scena finale) sono realistiche e hanno l'aspetto che si attendiamo di ottenere. L'*actor DirectionalLight* cerchiato in nero in figura, può essere posizionato in qualsiasi zona della scena, ma non cambierà l'effetto che otterremo, in quanto possiamo considerare che le luci vengano proiettate da un piano di dimensione infinita

posizionato a una distanza infinita dalla scena. L'unica differenza che si ottiene nel modificare l'*actor* in questione è nell'inclinarlo: a diversa inclinazione corrisponde un diverso aspetto delle ombre proiettate.

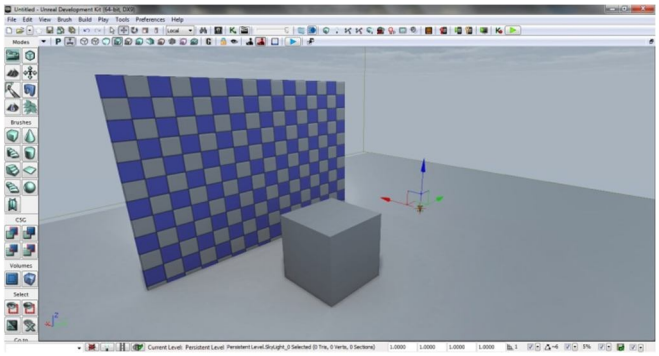
5.2.2 – *Point Lights*



5.2.2.1 - Point Light

Una Point Light, a differenza della Directional Light, rappresenta una fonte di luce che ha origine esattamente da dove viene posizionato il *Point Light Actor*. La luce parte in tutte le direzioni e generalmente ha un raggio d'azione piuttosto limitato. Luci di questo genere vengono utilizzate di solito per rappresentare l'illuminazione fornita da un fuoco o in generale fonti di luce che non hanno una grande portata.
Se lo si desidera possono comunque essere modificate in maniera da risultare fonti di luce molto forti e di ampio raggio, ma questo dipende solo dal tipo di scena che si sta creando o comunque dalla particolarità che intende creare un Level Designer.
Come si nota in figura, le ombre proiettate da una *Point Light* non sono ben definite come quelle di una *Directional Light,* ma sono più approssimative, come effettivamente ci si aspetterebbe dalle ombre generate dalla luce di un fuoco.

5.2.3 – *Sky Lights*



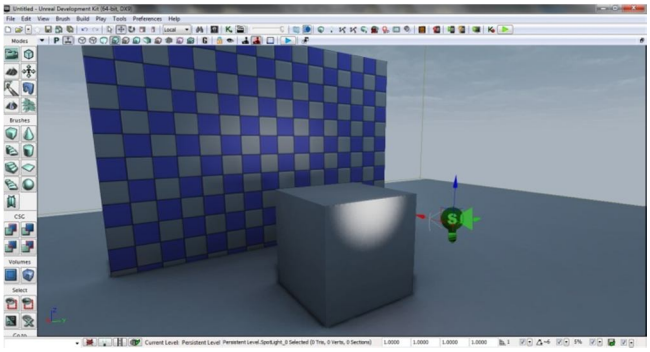
5.2.3.1 - Sky Light

Come si può notare istantaneamente dalla figura 5.2.4.1, non vi sono ombre nella scena. Questo avviene perché la Sky Light è una luce ambientale che non proietta ombre ma fornisce un illuminazione generale.

Una Sky Light va utilizzata in combinazione con gli altri generi di

luce, e possiamo considerarla una luce non fondamentale nella creazione di un ambiente. Infatti, utilizzando *Directional Light, Point Light e Spot Light* dovremmo essere in grado di creare un livello adeguatamente illuminato.

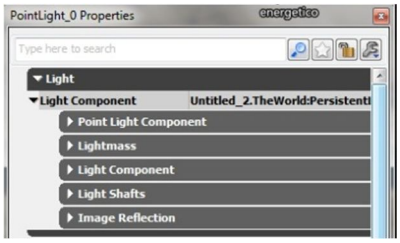
5.2.4 – Spot Lights



5.2.4.1 – Spot Light

Dallo *Spot Light Actor* viene proiettata una luce che occupa l'interno di un volume di forma conica. Una luce del genere viene spesso utilizzata per simulare l'illuminazione fornita dai tipici lampioni che possiamo trovare per le nostre strade, in quanto queste proiettano a terra un vero e proprio cono di luce. Sono simili alle *Point Light*, in quanto la posizione dell'actor indica la fonte della luce. La dimensione del cono può essere impostata a piacere e allo stesso modo la luminosità. La proiezione delle ombre non è neanche qui netta come nel caso della *Directional Light*, ma è decisamente realistica.

5.2.5 – Proprietà generiche delle luci



5.2.5.1 – Proprietà delle Luci in UDK

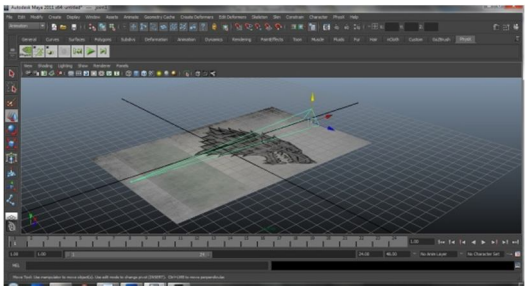
Ogni tipo di luce in UDK può essere impostato a seconda delle necessità [UDKLS]. A seconda del tipo di luce si avranno le opzioni per modificare le componenti dedicate (nella figura 5.2.5.1 possiamo notare *Point Light Component* dal quale deduciamo che stiamo analizzando le proprietà di una *Point Light*. Si gestiscono qui le particolarità del tipo di luce: in questo caso definiremo ad esempio il raggio d'azione della luce, impostando il valore *Radius*.
In *Lightmass* possiamo definire l'illuminazione indiretta, la saturazione, la penombra e la dimensione della superficie emissiva della luce.
In *Light Component* gestiremo varie opzioni che vanno a definire la qualità delle ombre prodotte e la forza luminosa dell'attore.
In *Light Shafts* possiamo gestire i raggi proiettati dalla luce.
Con *Image Reflection* si può attivare la capacità della luce di sfruttare la riflessione generata dai materiali che sono stati predisposti per farlo.

Capitolo 6 – Cloth: simulazione di tessuti

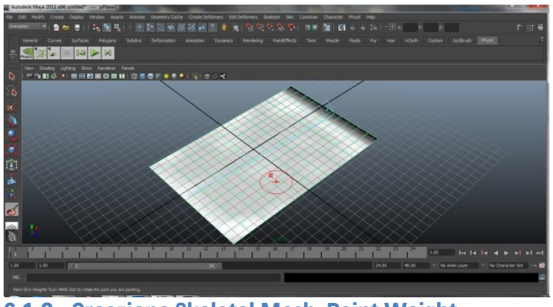
Quando si intende creare un ambiente virtuale può capitare di voler inserire degli oggetti capaci di comportarsi come dei veri e propri tessuti come ad esempio bandiere, teli, vele o comunque i vestiti indossati dai personaggi in scena.
UDK non dispone di per sé di strumenti predisposti per la simulazione dei tessuti, ma ovviamente permette di gestire eventuali Mesh importate da Maya che vadano a svolgere il lavoro richiesto. Una Mesh adatta ad assolvere a questo compito dovrà essere una Skeletal Mesh adeguatamente costruita e importata tramite il già descritto Plugin Actor X.

6.1 – Creazione di una Skeletal Mesh per simulare tessuti in Maya

In questo capitolo verrà mostrato come creare una Skeletal Mesh che servirà poi a simulare una semplice bandiera rettangolare in UDK.
Una volta avviato Autodesk Maya sarà necessario creare un oggetto di tipo Plane.
Definita la forma del nostro poligono si potrebbe già procedere con il Texturing, così da potersi poi completamente dedicare alla creazione della struttura scheletrica della Mesh [GSWM10].

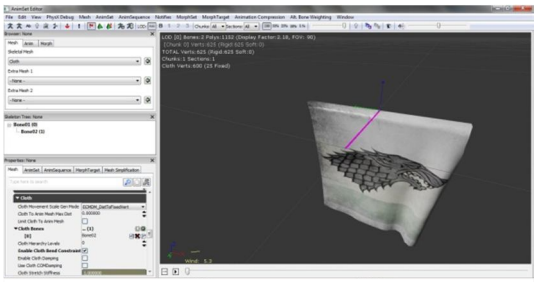

6.1.1 - Creazione Skeletal Mesh

Per creare i vari *joint* dello scheletro, è necessario selezionare, dopo aver scelto di lavorare sul menù *Animation*, il *Joint Tool* dal menù Skeletal Mesh. Nel caso di una bandiera di forma triangolare, basterà creare un solo e unico Joint, come nella figura 6.1.1. Ora, selezionando contemporaneamente il Joint creato e la superficie, si dovrà selezionare il comando *Skin* ➡ *Bind Skin* ➡ *Smooth Bind.*
In questo modo avremo creato la *Skin* della nostra Mesh, dunque potremo agire con il *Paint Skin Weight Tools*. Sarà necessario distribuire il peso dei vertici come nella figura 6.1.2, impostando un peso uguale a 0 per i vertici che rappresentano la zona dove la bandiera sarà congiunta con una possibile asta. Ricordiamo che nel Paint Weight, è possibile dare ai vertici dei poligoni un peso con un valore variabile tra 0 e 1.


6.1.2 - Creazione Skeletal Mesh, Paint Weight

Con valore 0, si intende dire che lo dal movimento dei *joint*, quella particolare zona non avrà alcuna influenza. Con valore 1 l'influenza del movimento sarà massima. Distribuita come in figura, l'influenza del *joint* sui vertici dovrebbe essere poi resa realisticamente in UDK. Vediamo però ora come effettuare l'esportazione tramite Actor X e in seguito l'importazione nel Content Browser di UDK.

6.2 – Esportazione e importazione della Skeletal Mesh



6.2.1 - Cloth in UDK 1

Controllando che il Plugin Actor X sia stato adeguatamente caricato da Maya, digitiamo *axmain* [ACTX]. A questo punto scegliamo una destinazione per il file, un nome e completiamo l'esportazione. Aprendo UDK, come già spiegato nel capitolo 2, importiamo il file in UDK (attenzione, il formato del file non sarà più ASE, ma PSK). Aperto su Content Browser il file importato dovremo agire come segue:

a. Nel menù delle proprietà sulla sinistra, alla voce *material* inserire il materiale con in quale vogliamo sia texturizzata la nostra mesh;
b. Aggiungere una spunta alla voce *Force CPUSkinning* per forzare lo *Skinning* delle *bones* sulla mesh*;*
c. Nel sottomenù *Cloth* alla voce *Cloth Bones*, aggiungiamo uno dei *bones* (in maya *joint*) che vogliamo utilizzare come principale per gli effetti sulla skeletal mesh;
d. Aggiungere una spunta alla voce *Enable Cloth Bend Constraint*, per attivare un miglior controllo per le piegature del tessuto;
e. Modificare a piacimento le restanti opzioni (per vederne gli effetti direttamente da qui, basta premere il pulsante  in alto nella finestra di editing).

In particolare nel punto "*e*" sarà possibile applicare un effetto di *Tearness* al tessuto, in maniera da indicare se questo può strapparsi e quale forza deve essere applicata per far sì che questo avvenga. Inoltre si possono applicare altri svariati valori per gestire le collisioni fra le varie parti del tessuto in maniera più realistica e aumentare in generale la qualità della resa finale, sempre ricordando che maggiore operazioni si richiedono, più calcoli la GPU dovrà effettuare e più sarà possibile avere dei rallentamenti.

Ora l'oggetto sarà pronto per essere utilizzato e una volta inserito in una scena sarà necessario selezionare l'opzione *Wake On Level Start* tra le proprietà della Skeletal Mesh in scena, per far sì che l'animazione della nostra bandiera si avvii in automatico all'avvio del livello.

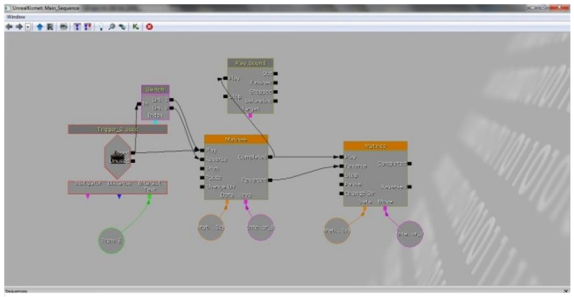
In generale, quando si vuole creare un oggetto simile a quello che siamo andati a descrivere bisognerà tenere conto di creare delle strutture scheletriche che seguano la forma dell'oggetto e effettuare un Paint Weight tale da poter permettere poi a UDK di generare in automatico un'animazione realistica o quanto meno l'animazione desiderata. Nel caso della nostra bandiera infatti avevamo previsto l'applicazione di un'asta in uno dei lati del poligono. Nel caso di un vestito non avremo aste da applicare all'oggetto ma un corpo da circondare al quale l'oggetto dovrà aderire correttamente. Si tratta sempre di operare tramite Paint Weight, ma la costruzione dell'oggetto risulterà decisamente più complessa e lunga.

Capitolo 7 – Kismet e Matinee: gestione eventi e animazioni

Parte fondamentale di un videogioco è costituita da eventi con il quale l'utente avrà a che fare; ogni interazione dell'utente con gli oggetti di scena può essere definita come un evento: ad esempio l'apertura di una porta tramite la pressione di un tasto predefinito, è un evento semplice che associa un *trigger* (in italiano grilletto o innesco) alla pressione del tasto in questione in una zona delimitata alla vicinanza dalla porta. In seguito avverrà l'apertura della stessa, tramite animazione della Mesh che verrà riposizionata a seconda delle intenzioni del progettista. Un evento del genere e eventi ben più complessi possono essere creati tramite la gestione di *Unreal Kismet* e *Matinee*.

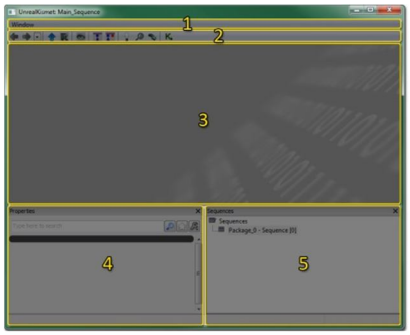
7.1 - Kismet, una breve introduzione

UnrealKismet [KIS] è uno strumento flessibile e potente che permette anche a un non programmatore la creazione di script di elevata profondità per gestire il gameplay di un livello di gioco. Il suo funzionamento è relativamente semplice: consente di creare dei grafici formati da semplici oggetti funzionali che collegati nelle maniere desiderate sono in grado di costruire eventi complessi. Nella figura 7.1.1 si può vedere il grafico che gestisce l'evento di apertura e chiusura di un cancello nella scena del progetto di tesi tramite la pressione del tasto ***E*** alla vicinanza di una leva.



7.1.1 - UnrealKismet, evento d'esempio

Per avviare Kismet nella nostra scena bisogna cliccare nel pulsante  che si può trovare nella barra degli strumenti di UDK. Si aprirà una finestra come nella figura 7.1.2 divisa in questa maniera:



7.1.2 - UnrealKismet, pagina principale

1 – Barra del menù: permette di visualizzare o meno il pannello delle proprietà e quello delle sequenze;

2 – Barra degli strumenti: mette a disposizione dei comandi per navigare e modificare l'aspetto del pannello di lavoro;

3 – Pannello di lavoro: permette di creare e visualizzare i grafici degli eventi di gioco;

4 – Pannello delle proprietà: mostra le proprietà per gli oggetti selezionati;
5 – Pannello delle sequenze: mostra le gerarchie di tutte le sequenze e sottosequenze del livello.

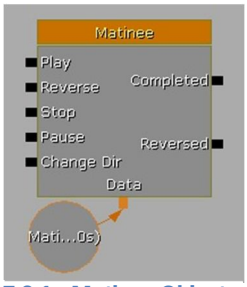
Un evento viene generalmente avviato quando l'utente entra in contatto o interagisce con un certo *actor* chiamato *trigger*. Questi oggetti possono essere impostati in maniera da attivarsi al semplice contatto con l'utente o tramite pressione di un qualche tasto (come per ogni *actor*, è possibile trovare i *trigger* nell'*actor classes*). È anche possibile fare in modo che un evento parta in automatico senza che l'utente faccia alcunché, semplicemente nell'istante in cui comincia il livello.
Una sequenza è formata dai seguenti elementi [KIS]:

- Evento: rappresenta il punto di partenza di tutta la sequenza, ovvero l'*input* che attiva l'esecuzione dello script;
- Azione: oggetti che eseguono determinate azioni (animazioni, visualizzazione di messaggi, cambio delle impostazioni delle luci di scena, …);
- Condizioni: è possibile inserire dei controlli condizionali nell'esecuzione di un grafico. Ad esempio l'utente cerca di aprire una porta ma non ne possiede la chiave; la condizione per l'apertura non è rispettata e sarà restituito al giocatore un messaggio su schermo che lo avverte di cercare l'oggetto per proseguire;
- Variabili: semplici oggetti ai quali possono essere assegnati valori o riferimenti a oggetti; possono essere ad esempio dei Bool, delle stringhe o dei riferimenti a qualsiasi componente di scena;

Per creare nel pannello di lavoro uno dei qualsiasi oggetti per la creazione di grafici, basterà cliccare con il tasto destro e semplicemente selezionarlo della lunga lista a disposizione. Gli oggetti possono essere poi collegati fra loro tramite frecce, similmente al tool per la creazione dei materiali visto in precedenza.

7.2 – Matinee, creazione di animazioni

Quando si lavora in UnrealKismet e si intende introdurre l'animazione di una Mesh nell'esecuzione di certi eventi, è necessario creare un oggetto di tipo *Matinee* [MAT].
Tramite questo tipo di oggetto sarà possibile associare ad un componente di scena di tipo *InterpActor* una determinata serie di movimenti da compiere in un intervallo di tempo stabilito. Può anche trattarsi di animazioni di lunga durata ed è inoltre possibile creare dei cicli nei movimenti degli oggetti quando si intende ad esempio far ripetere per un tempo indefinito le animazioni desiderate a un oggetto.



7.2.1 - Matinee Object

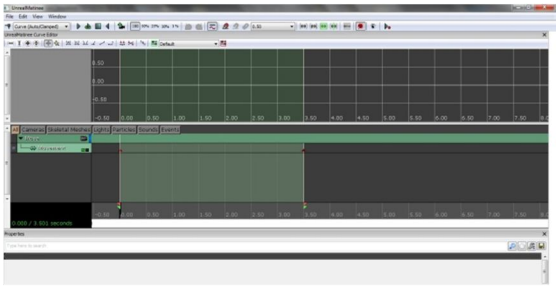
Nella finestra di lavoro di *UnrealKismet* basta selezionare *New Matinee* nel menù a tendina che apparirà cliccando con il tasto destro su uno spazio vuoto e apparirà un oggetto come quello in figura 7.2.1.
A questo vanno associati un oggetto di scena e un set di animazioni e movimenti. Nella parte sinistra del *Matinee* (e in generale di tutti gli elementi che andranno a comporre un grafico in Kismet) abbiamo la lista delle interfacce di input, mentre a destra quelle di output. In genere alle prime sono collegate delle frecce uscenti da un qualche *trigger* o comunque da oggetti di tipo condizionale [MAT].

- **Play**: la freccia collegata in entrata causa l'avvio dell'animazione;
- **Reverse**: mostra l'animazione invertita;
- **Stop**: ferma l'animazione e la riporta allo stato iniziale (al primo frame che la compone);
- **Pause**: ferma l'animazione nel momento in cui avviene l'input in questa interfaccia;
- **Change Dir**: cambia direzione

Le interfacce di output sono invece di solo due tipi e vengono attivate al momento della fine dell'animazione:

- ***Completed:*** attiva un evento o una condizione collegata nel momento in cui l'animazione viene completata;
- ***Reversed:*** attiva un evento o una condizione collegata nel momento in cui l'animazione viene invertita;

Ad un oggetto di tipo Matinee va infine collegato almeno un *InterpActor* sul quale poi si andrà a costruire l'animazione.


7.2.2 - Matinee, gestione di animazioni

Con un doppio click sull'oggetto *Matinee* per il quale siamo intenzionati creare e gestire delle animazioni, apriremo la finestra che si può vedere in 7.2.2.
Nella sezione in basso si va a gestire innanzitutto l'intervallo di tempo entro il quale avviene la nostra animazione e impostare tutte le *key* che indicano

determinati momenti del nostro intervallo e quale posizione assumerà l'oggetto in quel momento.
In parole povere e impostando un classico esempio, se si vuole far sì che un oggetto si muova seguendo una linea retta da un punto A ad un punto B in un intervallo di tempo di 5 secondi, basterà impostare due *key*: una all'inizio (a 0 secondi) e una alla fine (a 5 secondi). In scena imposteremo come posizione dell'oggetto al primo *keyframe* quella che desideriamo come posizione iniziale. In seguito sposteremo l'oggetto in B e imposteremo a 5 secondi la seconda *key*. Se andremo ad osservare l'animazione noteremo che in 5 secondi l'oggetto passerà da A a B anche se abbiamo definito solo la partenza e l'arrivo. Matinee calcola infatti in automatico il cammino più breve in casi come questo. Quando si intende invece creare animazioni più complesse con movimenti più imprevedibili e non calcolabili in automatico sarà ovviamente necessario settare più *key* nell'intervallo.

7.3 – Creare un evento con Kismet e Matinee

Per entrare meglio nel dettaglio della gestione di un evento conviene mostrare un esempio. Vedremo come creare l'evento mostrato nella figura 7.1.1 che è stato utilizzato nel progetto di tesi proprio per testare le potenzialità del tool a disposizione. Si tratta di un evento molto semplice: con la pressione del tasto ***E*** nei pressi una leva vicina ad un cancello, si dovrà abbassare la leva e sollevare il cancello. Una volta in questo stato, ad una seconda pressione del tasto dovremo ottenere invece l'effetto opposto.
Quando si intende andare a creare un evento qualsiasi conviene sempre crearsi prima uno schema su carta o in mente di cosa si vuole fare, o perlomeno ragionare bene su come si potrebbe farlo e quali oggetti andranno ad interessare le animazioni.
In questo caso, come abbiamo detto, avremo due oggetti da animare, una leva e un cancello. Posizionate sulla scena le *static mesh* dovremo convertirle in *InterpActor* (si può fare premendo il tasto destro del mouse e selezionando *Replace With* e in seguito *Add InterpActor*). Ora questi oggetti possono essere collegati a Matinee.


7.3.1 - Trigger, raggio d'influenza

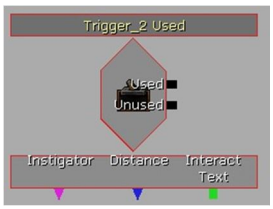
Analizzando bene il tipo di evento che andremo a creare possiamo renderci subito conto che si tratta di un qualcosa che viene attivato direttamente e solo dall'utente, quindi, selezionando un *Trigger* dall'*Actor Classes*, lo

posizioneremo nei pressi della nostra leva, definendo uno spazio entro il quale le interazioni dell'utente saranno considerate (in figura 7.3.1 vediamo questo spazio delimitato da un cilindro verde).

Sarà possibile indicare le dimensioni di questo cilindro dalle proprietà del Trigger.

Dopo aver disposto questi elementi della scena, potremo andare a costruire il nostro evento.

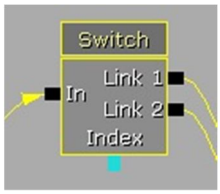
Avviamo UnrealKismet e, selezionando in scena il Trigger che ci interessa, premiamo il tasto destro sul pannello di lavoro e selezioniamo *New Event Using Trigger* e in seguito *Used*; in questo modo indicheremo al Kismet che siamo intenzionati a creare un evento che sarà attivabile dall'utente su utilizzo del nostro Trigger (nel nostro caso, con la pressione del tasto E in prossimità). Se avessimo ad esempio utilizzato l'opzione *Touch* avremmo fatto in modo che l'evento si attivasse all'ingresso dell'utente nella zona delimitata dal Trigger.



7.3.2 - Trigger

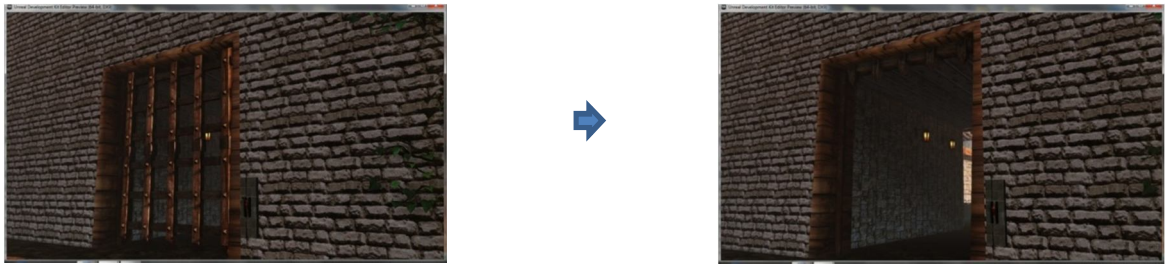
Un Trigger del genere da noi utilizzato nell'esempio viene mostrato nel pannello di lavoro di UnrealKismet come in figura 7.3.2. Ha due interfacce di output, che indicheranno cosa fare quando viene utilizzato (*Used)* o quando non lo è (*Unused)*.

Consideriamo ora di nuovo il nostro evento: vogliamo fare in modo che la leva che l'utente utilizzerà funzioni in due direzioni. Conviene introdurre dunque uno *Switch* (figura 7.3.3)*:* tramite un oggetto di questo tipo possiamo indicare quali azioni compiere dopo l'attivazione di un trigger a seconda di quante volte lo abbiamo già utilizzato. Nel nostro caso imposteremo dalle proprietà dello Switch il *Link Count* a 2, ovvero diremo che l'evento che otterrà in input potrà avere due conseguenze diverse. A *Link 1* faremo corrispondere l'evento di apertura del cancello e quando l'utente andrà a premere per la seconda volta il tasto ***E,*** Kismet capirà di dover avviare *Link 2* e gli eventi che ne conseguono, ovvero la chiusura. Se l'utente vorrà riaprire il cancello, se attivata l'opzione *looping* dello *switch*, si ripartirà da attivare gli eventi di *Link 1*. Avremo creato quindi un meccanismo di apertura e chiusura. Senza Looping l'utente avrebbe potuto aprire il cancello, richiuderlo e poi il trigger avrebbe smesso di funzionare. Ora, a livello di Kismet, collegheremo a Link 1 il Matinee che andrà ad animare la leva su *Play*. Collegheremo poi il *Completed* al *Play* del Matinee che animerà il cancello. Link 2 sarà invece collegato al *Reverse* del primo Matinee e dal *Reversed* di questo si passerà al *Reverse* del secondo Matinee (per chiarezza vedere la figura 7.1.1.



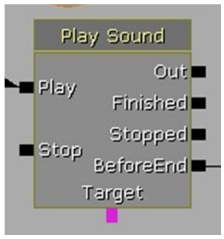
7.3.3 - Switch

Il grafico che rappresenta il nostro evento è dunque pronto, anche se è ancora necessario creare le animazioni dei nostri oggetti.
Vediamo ad esempio come creare il movimento del nostro cancello.
Selezioneremo *l'InterpActor* del cancello in scena e apriremo il relativo Matinee con un doppio click sull'oggetto nell'*UnrealKismet*. Nella parte sinistra in basso della finestra di *UnrealMatinee,* cliccando con il tasto destro, si selezioni Add *New Empty Group*. Cliccando poi sul nuovo gruppo creato al quale avremo data un nome a piacere, selezioneremo *Add New Movement Track*. Ora avremo creato una nuova traccia (di default con una durata di 5 secondi) con la quale definire i movimenti del cancello. Selezionando di nuovo il cancello in scena e aprendo ancora Matinee, premiamo il tasto ***S*** con il quale imposteremo la prima *key*. In questo modo diremo a Matinee che il cancello nel momento 0 del tracciato di movimento deve trovarsi in quella posizione. Ora selezioniamo nella traccia su Matinee la sua fine (nel nostro caso il terzo secondo). Nella scena spostiamo il cancello nella posizione desiderata e tornando su Matinee settiamo la *key* sempre premendo il tasto ***S***. Ora avremo creato la nostra animazione nella quale in tre secondi il cancello si solleverà.



Nel momento in cui vi sarà la seconda pressione del tasto ***E*** in prossimità del Trigger, si avvierà la chiusura del cancello, ovvero l'animazione nel verso opposto, grazie all'inserimento dello Switch precedentemente descritto.

Infine, per rendere il tutto più realistico si potrebbe aggiungere un effetto sonoro, per accompagnare l'animazione. Trovatone uno che potrebbe essere adatto lo si dovrà importare nel Content Browser (il procedimento è sempre lo stesso usato per importare qualsiasi altra cosa). Aggiunto poi un nodo Play Sound al grafico in UnrealKismet e indicato il suono che si è interessati utilizzare, si potrà collegare in *Play* il nodo del Matinee che anima il cancello, per indicare di far partire l'effetto sonoro nel momento in cui parte l'animazione.



7.3.4 – Play Sound

Capitolo 8 – Rigid Body e Fracture Tool

In questo capitolo mostreremo due esempi di oggetti con i quali un utente può effettuare interazione senza l'ausilio dei precedentemente citati Kismet e Matinee.
In particolare vedremo come far sì che un oggetto sottostia alle leggi del motore fisico proposto da Unreal e come far sì che un oggetto sia distruttibile in maniera sufficientemente realistica sfruttando uno dei tool forniti da UDK, il Fracture Tool.

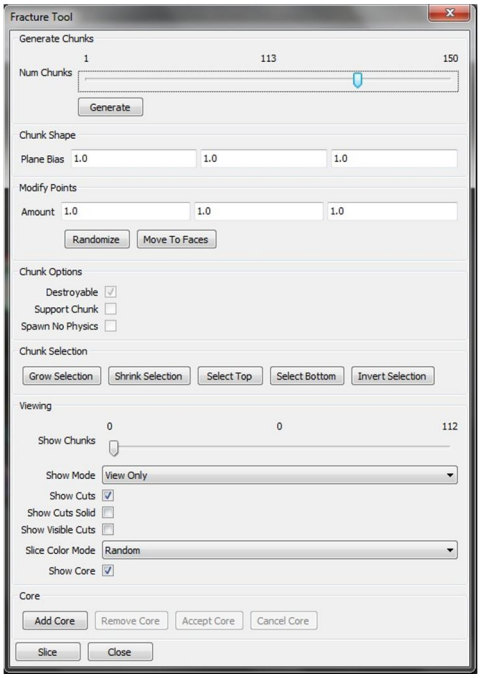
8.1 – Trasformare una Static Mesh in un Rigid Body

Nella creazione di una scena dinamica nel quale si intende far interagire l'utente con alcuni oggetti o far sì che l'ambiente non risulti immobile con oggetti totalmente statici, occorre introdurre il concetto di *Rigid Body* [UDKDOC].
Un *Rigid Body* è un oggetto che è soggetto alle leggi di gravità dell'Unreal Engine 3 e può collidere con gli altri componenti della scena seguendo il poliedro di collisione su di lui costruito. In UDK è possibile convertire una qualsiasi Static Mesh in un Rigid Body e in seguito far sì che sia utilizzabile in scena anche dall'utente.
Un esempio: creata la Mesh di un pallone da calcio, definitele le collisioni e inserito l'oggetto in scena, se l'utente si avvicinerà non potrà attraversarlo in quanto le collisioni lo impediscono, ma non potrà neanche spostarlo. Se trasformato in Rigid Body, al contatto con l'utente, il pallone si muoverà in una certa direzione con una certa velocità a seconda del tipo di interazione e a seconda del tipo di collisioni costruite per l'oggetto: se venisse creato un cubo per rappresentare le collisioni di un pallone di forma sferica, nel movimento dell'oggetto non noteremo nulla di naturale, in quanto la palla effettivamente non rotolerà, ma si comporterà come ci si aspetta si comporti un cubo. Dunque se si intende inserire in scena dei rigid body che seguano dei movimenti realistici è necessario generare delle collisioni complesse e precise, cosa che UDK può fare in automatico, come visto nel capitolo 2.
Per aggiungere un Rigid Body da una Static Mesh in una scena è necessario selezionare l'oggetto nel Content Browser e cliccare col tasto destro nella scena, selezionando poi *Add UT Rigid Body* : *"nome oggetto"*. A questo punto, sarà necessario mettere una spunta nell'opzione *Wake on level start* tra le proprietà dell'oggetto inserito in scena sotto la voce *KActor*, in modo che il Rigid Body sia attivo da subito.
Sfruttando le potenzialità di più Rigid Body è possibile ingegnarsi per creare eventi, magari attivati dall'utente, nei quali degli oggetti si muoveranno senza seguire però animazioni generate con UnrealMatinee ma automaticamente tramite il motore fisico messo a disposizione.

Come esempio consideriamo l'evento della caduta di un albero. Senza perdere tempo a creare l'intera animazione ci si potrebbe affidare al motore fisico semplicemente avviando la caduta tramite evento e poi lasciando che il resto venga fatto "naturalmente".

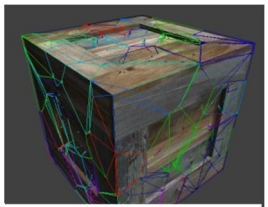
8.2 – Fracture Tool: rendere una Static Mesh distruttibile

Se si intende inserire in scena un oggetto di qualsiasi tipo che possa essere distrutto dall'utente è possibile affidarsi ad un comodo strumento offerto da UDK: il *Fracture Tool*. Aprendo una qualsiasi Static Mesh con l'Unreal Static Mesh Editor (con un semplice doppio click sull'oggetto nel Content Browser), nella sezione della barra degli strumenti è possibile notare il pulsante  . Cliccandoci, apriremo il Fracture Tool, visualizzando la finestra della figura 8.2.1. Nel campo *Num Chunks* possiamo selezionare un valore da 1 a 150, andando a scegliere in quante parti vorremmo far sì che si possa distruggere la Mesh. Cliccando su *Generate*, nel giro di qualche secondo avremo predisposto la Mesh ad essere suddivisa in varie parti.



8.2.1 - Fracture Tool

È possibile selezionare quali parti della Mesh non devono subire comunque le influenze dell'utente e quindi non rompersi. Lo si può fare semplicemente cliccando nelle zone interessate della Mesh ed è inoltre possibile selezionare rapidamente la base o la cima tramite *Select Bottom* e *Select Top*. Una volta selezionate le regioni della Mesh che ci interessa non rompere sarà necessario aggiungere una spunta in *Support Chunks.* La suddivisione delle Mesh avviene in maniera uniforme, ma è comunque possibile generare casualmente la forma dei frammenti tramite i comandi nella sezione *Chunk Shape*.



8.2.2 – Fractured Static Mesh

Una volta eseguite tutte le operazioni desiderate sul nostro oggetto, selezioneremo *Slice*: in questo modo andremo effettivamente a generare una *Fractured Static Mesh*.

I segmenti colorati, come si vede in figura 8.2.2, rappresentano i punti in cui avverranno le fratture e delimitano le forme dei frammenti. Prima di inserire l'oggetto in scena è necessario


8.2.3 - Scatola distrutta

togliere la spunta *Use Simple Box Collisione* e *Use Simple Line Collision* poiché si tratta di un genere di collisioni che non ci interessano. Fatto questo è necessario salvare il lavoro e si potrà infine utilizzarlo nella scena. Per vederne i risultati si può ad esempio avviare il gioco in modalità di prova, selezionando la modalità di gioco di Unreal Tournament, uno sparatutto in prima persona realizzato da Epic Games sfruttando ovviamente UDK. Avremo a disposizione un fucile con il quale sparare. L'oggetto al quale abbiamo applicato il Fracture Tool, se colpito, si romperà proprio come ci aspettiamo, dividendosi nei frammenti che abbiamo creato in precedenza.

9 – Sviluppo del progetto di tesi

In questo ultimo capitolo viene descritto il processo di Level Design e in particolare lo sviluppo del progetto di tesi.

Non esiste una scienza esatta da cui prendere esempio per destreggiarsi in questa specifica fase, ma vi sono alcune regole che andrebbero seguite per creare qualcosa di stilisticamente soddisfacente per i possibili futuri utenti; come già detto in precedenza bisogna porre particolare attenzione soprattutto nella fase di modellazione delle Mesh che poi andranno tutte insieme a comporre un'unica scena, cercando di non sovrapporre facce, archi e vertici e, nella fase di UV Mapping, non sovrapporre gli UV, dato che ciò viene segnalato giustamente da UDK come problema da risolvere per evitare spiacevoli risultati grafici.
In UDK bisognerà poi porre la propria attenzione nella creazione del terreno che conterrà il nostro livello, facendo in modo che non vi siano settori troppo innaturali o mal texturizzati, oltre a dover cercare di decorare al meglio l'ambiente, per non farlo risultare vuoto o dispersivo. Anche i materiali e le luci giocano un ruolo fondamentale nella resa finale ed è necessario prefissarsi un modello di materiale che si vuole utilizzare, in modo da far sì che tutti i materiali di tutti gli oggetti alla fine risultino omogenei fra loro (non si dovrebbe ad esempio utilizzare una texture realistica insieme ad una texture in Cell Shading, dato che stonerebbero totalmente fra loro, rendendo la scena piuttosto ambigua).

Vediamo ora come è stato sviluppato il progetto della tesi, a partire dall'ideazione, fino alla creazione di alcuni piccoli eventi nel livello.

9.1 - Ideazione

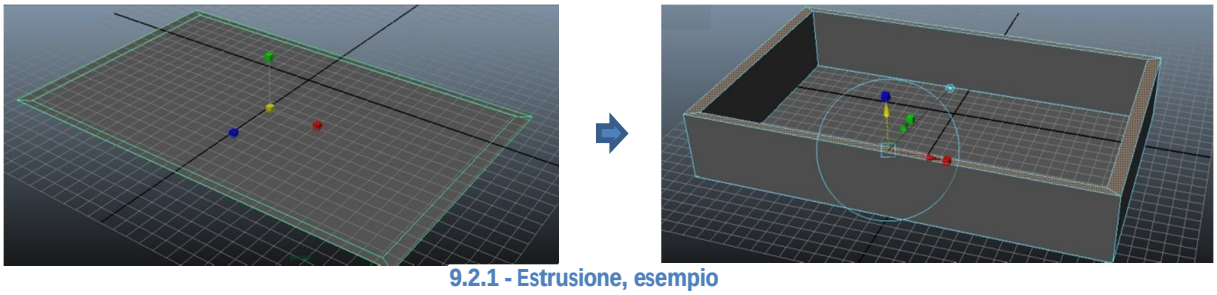
Non ha alcun senso cominciare a modellare oggetti da inserire in un livello senza prima lavorare su un'idea e creare un qualche schema (anche mentale) di ciò che si vuole andare a comporre. Nel nostro caso si è pensato di creare un piccolo villaggio di stampo medievale, circondato da mura fortificate e con all'interno una piccola fortezza. Partendo da questo presupposto si è pensato di creare una decina di modelli di case differenti fra loro, così da non dare un effetto di monotonia dell'ambiente e di creare poi modelli di alcune torri di pietra e di mura, oltre ad un cancello di ingresso del villaggio. In corso di posizionamento delle Mesh principali nel livello, è stato anche pensato a qualche oggetto di contorno (casse, botti, una fontana, bandiere) per

ravvivare un po' l'ambiente, che altrimenti sarebbe stato composto solo da case e alberi (che sono stati inseriti non come Mesh, ma come DecoLayer nella fase di Terrain Editing, vedi 3.1). Gli eventi invece riguarderanno principalmente l'apertura di cancelli e la possibilità di interagire con alcuni oggetti di scena.

9.2 - Modellazione

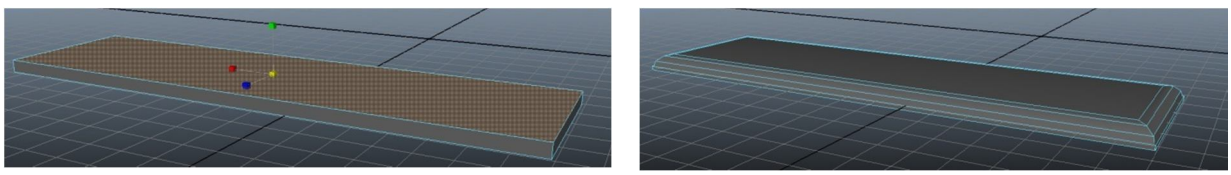
Nella fase di modellazione è stato seguito sempre un certo schema nella creazione delle case che compongono la scena, partendo sempre da un oggetto di tipo Plane a rappresentarne il pavimento e lavorando poi con gli strumenti Extrude, Bevel e Merge.
Andiamo a descrivere brevemente questi tre strumenti, che in fase di modellazione in Maya vengono utilizzati spesso, in quanto decisamente comodi e utili [GSWM10].

- **Extrude:** operazione tramite la quale è possibile agire su una o più facce (ma anche su archi e vertici) di una Mesh, estrudendo le parti selezionate formando nuove parti della Mesh che seguono il profilo di queste. Metodo utilizzato di frequente nella modellazione di Mesh del progetto. Nella figura 9.2.1 si può notare un esempio di come si potrebbe sfruttare questo strumento per la creazione di mura di una casa.



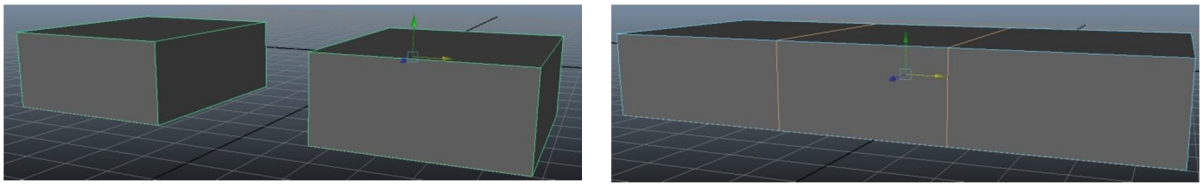
9.2.1 - Estrusione, esempio

- **Bevel:** comando utilizzato per la realizzazione di dislivelli e scanalature nelle facce di una Mesh. Utile per "arrotondare" rapidamente degli spigoli, creando degli oggetti più realistici, ma nell'evenienza può essere sfruttato per qualsiasi idea venga in mente.



9.2.2 - Bevel, esempio

- **Merge:** Operazione che permette di collegare due facce (e non di più alla volta) fra loro, anche di Mesh differenti (in questo caso bisogna prima unire i due oggetti tramite comando *Combine*).



9.2.3 – Merge, esempio

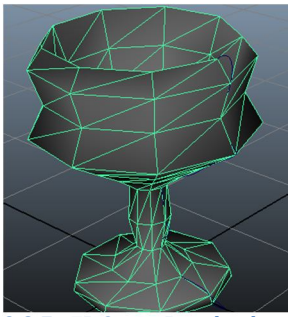
Sfruttando principalmente queste operazioni e in casi particolari alcune delle altre innumerevoli operazioni di Maya (come nel caso dei Cloth nel capitolo 6) sono stati creati tutti gli oggetti di scena che sono stati utilizzati come Static Mesh o Rigid Body o InterpActor.
In generale è possibile generare un oggetto sempre a partire da oggetti più semplici messi a disposizione da Maya come cubi, piani, piramidi, ecc. In seguito, applicando le operazioni desiderate si potrà arrivare al proprio obbiettivo.
Per la creazione di oggetti che invece non seguono di base dei solidi geometrici adeguatamente modificati, come ad esempio statue rappresentanti persone o animali, calici, fontane o comunque oggetti di forma particolare, esistono degli strumenti appositamente pensati: le curve.

Come sfruttare le curve per la modellazione di oggetti di forma irregolare?

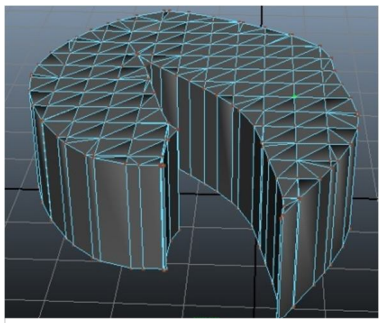
Tramite lo strumento delle curve EP è possibile costruire degli oggetti di tipo NURBS, convertibili poi ad oggetti poligonali esportabili tramite ActorX. Nel menù di Maya Polygons è possibile selezionare la voce Create -> EP Curves Tool, tramite il quale si andrà a disegnare la curva desiderata. Seguendo le immagini vediamo come si possono sfruttare queste curve per la creazione di un calice. Creiamo una curva che segua il profilo dell'oggetto desiderato (come in figura 9.2.4). Ora selezioniamo il menù Surfaces di Maya e in seguito Surfaces -> Revolve, selezionandolo però cliccando sul quadrato alla sua destra, così da aprire il menù di selezione della rotazione. Selezioniamo come asse di rotazione quello che ci interessa (in questo caso l'asse Y) e indichiamo una rotazione di 360 gradi. Otterremo ciò che si vede in figura 9.2.5, ovvero un oggetto che ricorda un calice, anche se di qualità piuttosto scarsa. La qualità dipende direttamente dal numero di punti di interpolazione che si sono utilizzati nella

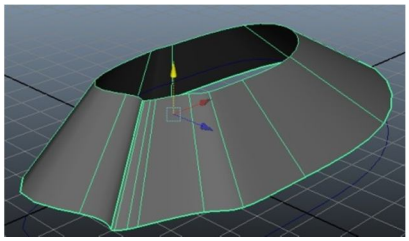
9.2.4 - EP Curve

creazione della curva ed è comunque migliorabile tramite gli strumenti di Smooth o Proxy di Maya.
Una volta creato, questo oggetto sarà di tipo NURBS e si potrà convertire in Poligonale tramite *Modify -> Convert -> NURBS to Polygon.* Una volta applicate le texture e fatte le ultime modifiche dell'oggetto sarà possibile utilizzare Actor X per l'esportazione.


9.2.5 - EP Curve Revolved

Lo strumento EP Curves Tool può tornare utile anche per altri generi di operazioni: si può ad esempio creare una curva chiusa e tramite il comando *Planar* (sempre nel menù *Surfaces*) creare in automatico una superficie che segua come bordi la curva stessa (figura 9.2.6). Infine, tramite il comando Loft è possibile, selezionando contemporaneamente almeno due curve EP, creare in automatico una superficie di collegamento tra loro. Questo può essere fatto anche tra curve di forma completamente diverse che comunque verranno collegate tramite la superficie più plausibile (che non sembra risulta essere quella desiderata). Il collegamento può essere fatto anche fra più curve, e il Tool le genererà sempre in modo automatico. Una volta selezionato il comando Loft e ruotata, ridimensionata o spostata una delle Curve EP, avverrà in automatico il riposizionamento della superficie di collegamento, dato che le curve risultano degli oggetti separati dall'oggetto che ne rappresenta l'unione Loft.


9.2.6 – Oggetto di forma irregolare estruso da una superficie creata tramite Planar


9.2.7 – Loft tra due curve EP

Come "scolpire" oggetti?

In Autodesk Maya risulta possibile creare degli oggetti a partire da dei solidi semplici "scolpendoli" direttamente. Il Tool, chiamato *Sculpt Geometry Tool*, selezionabile dal menù Edit NURBS di Surfaces, permette di agire su oggetti di tipo NURBS e di modificarli come se si agisse su dell'argilla. Risulta uno strumento molto simile a quello di Terrain Editing di UDK per quanto riguarda le funzionalità, ma viene utilizzato per la creazione di ogni genere di Mesh, da statue a modelli di persone.
Si può sfruttare anche solo per applicare delle semplici deformazioni localizzate.

9.3 – Level Design: Terrain

La fase di creazione del terreno, se si intende creare un'ambientazione che lo comprende, è certamente la prima a cui si va incontro nel momento in cui si comincia con il Level Design. Va tenuto conto però di alcune cose che in realtà saranno fatte in precedenza, almeno teoricamente:

- Le Mesh che comporranno la scena;
- Quali eventi si intende creare e gestire;
- Che genere di gioco si andrà a creare.

Nel progetto di tesi non è stato considerato l'ultimo punto in quanto non si è inteso creare nessun gioco completo ma un esempio di livello per la sola componente visiva ed alcuni esempi di eventistica semplice.

9.3.1 - Livello con Terrain e Mesh

Nella figura 9.3.1 si può vedere l'intero progetto (la parte grafica ovviamente). Era stato calcolato di creare un villaggio di genere medievale in una valle circondata da montagne. Il Terrain, inizialmente un Plane di forma quadrata, è stato modellato nei contorni tramite apposito strumento per la creazione delle montagne e nella zona in basso a sinistra è stata aggiunta una depressione nella quale sarà aggiunto in seguito un oggetto di tipo Fluid Surface (vedi capitolo 3) per simulare la superficie di un fiume. Il punto di partenza del giocatore è stato previsto appositamente nell'angolo in basso a sinistra così da fargli attraversare un ponte (Mesh appositamente creata) per condurlo dunque all'ingresso del villaggio che è costituito da un cancello che deve aprirsi automaticamente all'avvicinarsi del giocatore. Le mura che delimitano la città corrispondono circa al profilo delimitato dalla base delle montagne e l'unico ingresso corrisponde a quello appena citato. Il villaggio è composto da sette case tutte differenti fra loro per forma e dimensioni e da una serie di torri che non sono raggiungibili dall'utente ma solo visibili come contorno. Le case si trovano intorno ad una piazza di forma ovale con una fontana nel mezzo. Al di fuori della piazza si trova un mulino. Si può inoltre raggiungere un patibolo con ghigliottina. Nella zona alta della mappa c'è un cancello che consente di accedere al cortile interno della fortezza del progetto. Il cancello può essere aperto e chiuso a piacimento tramite pressione del tasto E in corrispondenza della leva vicina. Tutti gli oggetti citati sono stati realizzati in Maya e inseriti in seguito tutti insieme. Alcuni di questi invece sono stati

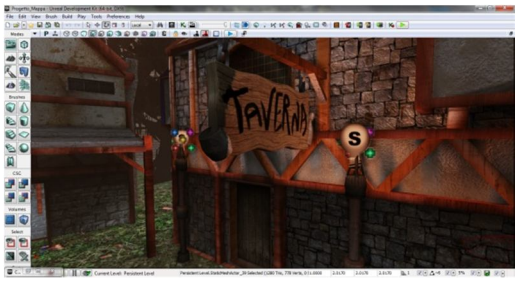
realizzati quando il Level Design era già a buon punto, per aggiungere all'ambiente dei particolari. Sono stati aggiunte inoltre decorazioni naturali (alberi e piante).

9.4 – Importazione e posizionamento Mesh.

Tutte le Mesh sono state realizzate tramite Autodesk Maya. Solamente il Terrain e la Fluid Surface rappresentante il fiume sono state realizzate tramite l'ausilio di UDK. Una volta creato il Terrain progettato come descritto nel precedente capitolo, si è proceduto al posizionamento delle varie Mesh, inizialmente quelle di tipo Static. Solo al termine del posizionamento di queste si è proceduto a posizionare gli eventuali Rigid Body e le Skeletal Mesh, dato che lo scheletro del livello di gioco è stato definitivamente stabilito. Infatti è un dato di fatto che le Static Mesh fungono da contorno della scena e non è possibile nessuna interazione attiva con queste, conviene dunque iniziare dal loro posizionamento. In seguito andrebbero posizionate le Mesh che saranno poi convertite in Rigid Body o Interp Actor utilizzabili negli eventi tramite Kismet e Matinee.
Questo tipo di oggetti vanno inseriti nella scena come Static Mesh e poi convertite con un semplici click destro del mouse e selezionando dal menù che apparirà il tipo di conversione desiderata.

9.5 – Level Design : Diffusione delle Luci

Una volta creata la mappa del livello è possibile cominciare a posizionare le luci. Questo può essere anche fatto contemporaneamente alle altre fasi, ma conviene considerarla un'unica fase così da vedere l'idea di insieme dell'illuminazione. Se all'inizio della creazione del progetto viene scelto un ambiente con luce direzionale preinserita, avremo già un qualche tipo di illuminazione, uniforme in tutta la mappa e senza particolari. Conviene a questo punto effettuare un *Build* dell'intero progetto, per vedere in che modo il livello è illuminato dalla luce che corrisponderebbe effettivamente a quella solare. A questo punto possiamo contribuire all'illuminazione migliore di zone d'ombra, con inserimento di luci ambientali del genere desiderato. Nel progetto, nei luoghi interessati sono state aggiunte delle Mesh rappresentanti torce nella cui cima è stato inserito un oggetto di tipo Particle System a rappresentare un fuoco. Di per sé



9.5.1 – Esempio di posizionamento luci e effetto sulle Mesh vicine

questo oggetto non proietta luci quindi sono state aggiunte delle Point Light con zona d'origine la cima di queste torce. La luce proiettata è debole e di colore rosso-arancione e raggiunge una distanza molto ravvicinata, tant'è che la si nota solamente nella pareti delle Mesh vicine. Nella figura 9.5.1 si può notare una colorazione differente nelle zone degli oggetti vicini alle fonte di luce. La qualità di gioco in questo caso è molto simile a quella mostrata nell'immagine, dato che la modalità d'illuminazione dell'ambiente di sviluppo può permettere anche di vedere la scena già in modalità di produzione o in una maniera molto simile a questa.
In altre zone sono state inserite delle luci non provenienti da Mesh, ma solo per aumentare la luminosità delle zone d'ombra.
Le ombre vengono proiettate in automatico spuntando l'opzione negli oggetti Light (in genere l'opzione è automaticamente attiva) ed è possibile deciderne la qualità.

9.6 – Level Design : creazione di eventi

Sono stati creati per questo 5 eventi differenti. Non era scopo della tesi creare un videogioco vero e proprio con obbiettivi e una giocabilità ben delineata, ma nella sperimentazione delle strumentazioni si è voluto anche testare la creazione di semplici eventi di gioco, soprattutto sfruttando Matinee, per lo spostamento di oggetti nella scena, automaticamente o in maniera indotta dall'utente.

Tramite gli eventi creati è possibile vedere:

- L'apertura automatica dei cancelli d'ingresso del villaggio all'approcciarsi del giocatore in una zona delimitata dal Trigger (vedi capitolo 7);
- Rotazione delle pale del mulino nella zona est del villaggio e musica di sottofondo: questi due eventi vengono attivati all'avvio del livello in automatico, tramite un trigger che ricopre la zona di spawn del giocatore, ovvero la zona dalla quale l'utente inizierà a giocare;
- Apertura e chiusura del cancello di ingresso al cortile della fortezza, tramite la pressione del tasto E davanti all'apposita leva: questo evento è una combinazione di due Matinee che gestiscono gli spostamenti del cancello e della leva e inoltre l'utente può in questo casa chiudere e aprire il cancello senza limiti, a differenza del primo evento descritto che è automatico;
- Attivazione della ghigliottina;
- Comparsa di messaggi e suggerimenti all'approcciarsi delle zone d'evento.

Sono stati inoltre inseriti alcuni oggetti di tipo Rigid Body e Fractured che reagiscono alle interazioni dell'utente senza seguire però Kismet o Matinee ma le leggi imposte dal motore fisico di UDK e le opzioni inserite dal progettista.

GLOSSARIO

Actor X: plugin per programmi di modellazione 3D quali Autodesk Maya e 3D Studio Max per l'esportazione di Mesh in formati compatibili con UDK.

Autodesk Maya: programma di modellazione 3D.

Character Animation: fase di creazione delle animazioni di una Skeletal Mesh, generalmente rappresentante una creatura di qualche tipo, che sia umana o animale, di fantasia o meccanica.

Cell Shading: tecnica di modellazione 3D non fotorealistica finalizzata a far apparire l'oggetto creato come se fosse disegnato a mano, in stile cartoon.

Cloth: oggetto creato per la simulazione di tessuti; mentre in Maya è possibile crearne direttamente tramite appositi comandi, in UDK è possibile attivare invece la funzione Cloth su Skeletal Mesh appositamente pensate e create su Maya.

Content Browser: interfaccia di UDK che permette di visualizzare e selezionare tutti i contenuti dei package interni o esterni al programma (static mesh, skeletal mesh, suoni, materiali, ecc..).

Fluid Surface: oggetto di scena reso disponibile da UDK capace di simulare una superficie fluida.

Foliage Mode: editor di UDK che permette di inserire in scena delle Mesh come decorazioni; si utilizza generalmente per Mesh quali alberi e piante che vengono spesso incluse più volte nella stessa scena.

Fracture Tool: strumenti per la creazione automatica di oggetti distruttibili a partire da semplici Static Mesh.

Kismet: tool per la creazione e gestione di eventi in UDK.

Landscape Mode: editor di UDK che permette di modellare ambienti naturali in maniera realistica.

Level Design: fase di sviluppo di un videogioco nella quale si costruisce l'architettura di un livello di gioco sfruttando gli oggetti messi precedentemente a disposizione all'interno del Content Browser.

LOD: level of details; rappresentano zone precise di una Mesh le quali probabilmente presentano materiali differenti.

Matinee: tool per la creazione di animazioni, spesso utilizzato tramite Kismet.

Mesh: insieme di vertici, archi e facce che vanno a formare un oggetto tridimensionale.

Normal Map: tecnica usata per simulare l'illuminazione di urti e ammaccature. È usata per aggiungere dettagli senza utilizzare altri modelli poligonali.

Rendering: ultima fase di sviluppo di un livello di gioco, nella quale vengono proiettati svariati fasci di luce nella scena e si registrano le posizioni dei vari oggetti e la loro reazione alla luce, realizzando dunque una scena che sia visitabile in tempo reale; il rendering può essere un'operazione dispendiosa, a seconda di quanto materiale si dovrà analizzare.

Rigid Body: oggetto a cui sono applicate in gioco le leggi fisiche fornite dal motore di gioco.

Skeletal Mesh: Mesh con struttura scheletrica tramite la quale si possono creare le articolazioni dell'oggetto. Uno scheletro viene sempre fornito a Mesh che rappresentano persone o animali, ma molto spesso anche ad oggetti.

Static Mesh: Mesh di tipo statico, che dunque non è utilizzata per fornire direttamente interazioni con il motore fisico di gioco o con l'utente.

Terrain: Oggetto di tipo Terreno. Un oggetto del genere può essere modellato tramite Terrain Editing Tool, per la creazione di ambienti naturali e non.

Texture: immagine di qualsiasi tipo utilizzata per rivestire la superficie di un oggetto virtuale, tridimensionale, o bidimensionale, con un apposito programma di grafica.

UDK: Unreal Development Kit, software per lo sviluppo di videogiochi.

Unreal Material Editor (UME): strumento per la creazione di materiali applicabili poi a qualsiasi Mesh dei Package del Content Browser.

UV Mapping: processo tramite il quale si produce un'immagine 2D rappresentante un modello 3D.

BIBLIOGRAFIA

[GSWM10] Autodesk Maya, Getting Started With Maya 2011, Autodesk, *Summer 2010, pp. 13–150*; *pp. 359–435.*

[WIKIa] *Wikipedia*: http://it.wikipedia.org/wiki/Storia_dei_videogiochi

[WIKIb] *Wikipedia*: http://it.wikipedia.org/wiki/Computer_grafica

[MAYA] *Autodesk*: http://www.autodesk.it/adsk/servlet/pc/index?siteID=457036&id=14646075

[UDK] *UDK Official Site* : http://udk.com/

[UDKF] *UDK, Features*: http://udk.com/features

[EPUE] *Wikipedia*: http://it.wikipedia.org/wiki/Unreal_Engine

[GSUDK] *Getting Started with UDK*: http://udn.epicgames.com/Three/WebHome.html

[ACTX] *Actor X plugin*: http://udn.epicgames.com/Three/ActorX.html

[UDKDOC] *UDK, Online Documentation*: http://udk.com/documentation

[UDKMAT] *UME*: http://udn.epicgames.com/Three/MaterialsAndTexturesHome.html

[UDKLS] *UDK Light*: http://udn.epicgames.com/Three/LightingAndShadowsHome.html

[KIS] *UDK, Kismet*: http://udn.epicgames.com/Three/KismetHome.html

[MAT] *UDK, Matinee*: http://udn.epicgames.com/Three/MatineeAndCinematicsHome.html