Università degli Studi di Padova

DIPARTIMENTO DI INGEGNERIA

Uso di COLLADA nella rappresentazione CAD

Relazione finale di Tirocinio

Laureando: Alessio Michele
Relatore: Prof. G.Clemente

Dipartimento di Ingegneria dell'Informazione
Anno Accademico 2012-2013

Indice

Introduzione

La seguente relazione nasce dall'esperienza maturata durante le 500 ore di tirocinio svoltosi presso BrainSoftware srl, che da anni sviluppa un'applicazione CAD evoluta, denominata Spazio3D.

Il progetto realizzato si colloca all'interno di un più ampio processo di modernizzazione ed evoluzione delle potenzialità offerte dal loro prodotto. Nello specifico l'obiettivo prefissato richiedeva l'utilizzo di uno standard Open Source per la rappresentazione e l'interscambio di qualità grafiche tra architetture CAD differenti. A questo scopo sono state individuate in COLLADA le principali caratteristiche richieste. Questo formato offre la possibilità di usufruire delle tecniche più comuni della computer grafica 3D in generale ed inoltre è in grado di adattarsi con facilità alla struttura interna propria di Spazio3D.

L'oggetto di questa relazione consiste dunque nella presentazione delle varie fasi di studio e di realizzazione del progetto, affrontando nei capitoli seguenti l'analisi del lavoro svolto proponendo una panoramica quanto più esaustiva di tutti gli aspetti essenziali per la sua comprensione.

Nel primo capitolo verrà esposto un quadro generale delle motivazioni e degli obiettivi di questo progetto presentando inoltre il contesto in cui si è svolto, attraverso la descrizione dell'azienda e del CAD da essa sviluppato.

All'interno del secondo capitolo saranno descritte le principali caratteristiche delle tecnologie utilizzate mentre il terzo capitolo sarà dedicato alle specifiche dello standard COLLADA, la cui comprensione è essenziale in quanto costituisce il cuore del progetto e permette di chiarire certe scelte fatte durante l'implementazione.

Infine, nel quarto capitolo verrà presentato il lavoro svolto in tutte le sue fasi, dall'approccio iniziale alle considerazioni finali fatte nei test, concludendo con un resoconto globale di ciò che è stato fatto e di quanto ancora sarà possibile realizzare.

1 Progetto e contesto

Questo primo capitolo ha come scopo quello di presentare il progetto contestualizzandolo all'interno dell'azienda e del settore in cui essa opera attraverso il loro prodotto Spazio3D.

Inizialmente sarà presentata l'azienda attraverso una panoramica delle esigenze proprie del contesto in cui è nata, fino ad arrivare agli obiettivi attualmente prefissati.

Successivamente, approfondendo il mondo del CAD/CAM e delle macchine CNC, si vogliono fornire i concetti di base necessari per una miglior comprensione dell'ambito lavorativo in cui la BrainSoftware gravita.

Infine, dopo aver descritto cos'è Spazio3D assieme alle sue potenzialità, saranno chiariti gli obiettivi del progetto realizzato.

1.1 BrainSoftware srl

Negli anni ottanta il Veneto godeva di grande prosperità e settori come quello mobiliero vivevano un periodo di forte sviluppo. La necessità d'innovazione si scontrava con l'assenza di validi strumenti di lavoro in grado di supportare in modo adeguato la produzione.

E' in questo contesto che BrainSoftware Srl cominciò a muovere i sui primi passi. L'idea di poter soddisfare i bisogni dell'industria del mobile attraverso la creazione di un software CAD/CAM capace d'ottimizzare le fasi di progettazione e realizzazione dell'intero iter produttivo, portò alla nascita di Spazio3D.

Nei primi anni di vita dell'azienda, la distribuzione dei suoi prodotti era ristretta al solo alto vicentino e alto trevigiano. A partire dal 2001, BrainSoftware iniziò un processo di espansione commerciale per ampliare il proprio mercato. Con l'impiego di campagne pubblicitarie e la partecipazione alle principali fiere del settore, riuscì a farsi conoscere all'intero territorio nazionale.

Negli ultimi dieci anni però, l'intera economia europea risente di una grave recessione ed in Italia molte industrie dell'arredamento e dell'edilizia sono messe in ginocchio. Questa situazione porta come diretta conseguenza la necessità di aprire i propri confini verso nuovi mercati esteri (come quello indiano o cinese) .

BrainSoftware la concretizza attraverso un processo d'internazionalizzazione di Spazio3D, uno strumento in grado di rendere l'azienda competitiva nel proprio settore e che assicura sempre ai clienti servizi adeguati alle loro esigenze. L'assistenza analizza i problemi proponendo soluzioni mirate al miglioramento della produttività interna ed organizza corsi di formazione periodici che permettono di sfruttare al meglio tutte le potenzialità di Spazio3D.

Tutto questo è reso possibile dallo staff di BrainSoftware che vanta, nel suo organico, la presenza di ingegneri e tecnici con esperienza ventennale sempre al passo con l'evoluzione del mondo informatico e della produzione industriale.

1.2 CAD, CAM e CNC

Prima dell'avvento dei computer, nel mondo industriale, l'intero onere progettuale e sperimentale gravava sulle spalle di ingegneri e disegnatori. Le moli di lavoro richieste crescevano in modo esponenziale rispetto alla complessità dei progetti a loro affidati, ponendo grossi limiti realizzativi. Operazioni come la rappresentazione grafica e la creazione fisica dei modelli essendo realizzate quasi interamente "a mano" o con l'ausilio di macchinari rudimentali risentivano della scarsa precisione intrinseca nella natura umana.

Mantenere in vita un'azienda, all'interno di un mercato libero, richiede il perseguimento di un obiettivo: accrescere la propria redditività aumentando la competitività. Nell'economia aziendale questo fine è raggiungibile solo attraverso uno sforzo continuo di ottimizzazione ed innovazione. Molti manager ritengono che le strade da seguire siano due: diminuire i costi di produzione ed aumentare la qualità della merce. Questi concetti, diffici da conciliare, furono a suo tempo e sono a tutt'oggi ottenuti grazie all'introduzione dei computer.

E' grazie all'ingegneria, capace di tradurre problemi tecnici in calcoli numerici attraverso equazioni in grado di descriverli, che si ha la possibilità di poter sfruttare la grande precisione e la rapidità dei calcolatori elettronici all'interno dell'iter produttivo.

Industriali e militari colsero immediatamente l'enorme potenzialità che questo nuovo strumento garantiva e ne finanziarono lo sviluppo dell'hardware unitamente alla creazione di nuovi software in grado di sfruttarne appieno le capacità, in relazione alle necessità d'utilizzo. Nacquero così potenti strumenti di lavoro che cambiarono radicalmente il mondo industriale; tra questi vi sono il CAD, il CAM e le macchine CNC.

CAD - Computer Aided Design e Computer Aided Drafting

L'acronimo CAD non corrisponde ad una definizione precisa ma rappresenta principalmente due concetti strettamente correlati tra loro: Computer-Aided Design e Computer-Aided Drafting, rispettivamente la progettazione e il disegno tecnico assistito dall'elaboratore. L'idea di base consiste nell'unire il disegno tecnico per la descrizione di manufatti all'attività progettuale di modelli in due o tre dimensioni attraverso una virtualizzazione gestita dall'elaboratore.

A seconda dell'ambito in cui è utilizzato e da chi l'ha sviluppato, esistono molti CAD differenti. Ognuno di loro è costituito da un sistema automatizzato per il disegno che si presenta all'utente per mezzo di un'interfaccia grafica ricca di comandi per realizzare in rapidità elementi geometrici di qualsiasi complessità. La rappresentazione a video in due dimensioni (paragonabile all'immagine catturata dall'obiettivo di una telecamera) può essere realizzata con l'utilizzo di diverse viste secondo assi bidimensionali fissi oppure capaci di cambiare il proprio orientamento in un sistema di coordinate tridimensionali.

Lavorando in un ambiente virtuale si ha il vantaggio di poter combinare diversi elementi grafici per formare geometrie più articolate alle quali assegnare un significato, delle proprietà e dei componenti ausiliari utili alla loro completa descrizione. Si realizzano così dei veri e propri prototipi che possono essere testati attraverso simulazioni con modelli fisico-matematici accurati.

Sostituendo i prototipi con queste rapide e poco costose simulazioni si rende agevole la possibile correzione degli errori di progettazione, aumentando l'affidabilità dei dati e rendendo conseguentemente più sicuro il prodotto che verrà poi realizzato e commercializzato.

Un ulteriore punto di forza della progettazione assistita dal computer consiste nella capacità di archiviare in una base di dati tutte le informazioni che descrivono le componenti di un progetto. In tal modo si dispone di una documentazione esaustiva in grado di far fronte a futuri accessi ed aggiornamenti. La composizione di nuove strutture viene semplificata attraverso un approccio di tipo modulare, in cui si sfrutta il riutilizzo in tempi pressochè immediati degli elementi già collaudati salvati in memoria.

Con questi strumenti informatici anche l'organizzazione del lavoro all'interno dell'azienda ha potuto evolversi. Ad esempio sfruttando una semplice rete ethernet si agevola lo scambio d'informazioni, permettendo una differenziazione in reparti specializzati dedicati ad aspetti particolari del progetto.

Parallelamente a questi aspetti tecnici, si nota un chiaro incentivo alla creatività dei progettisti i quali possono cimentarsi, attraverso uno sviluppo per tentativi, nella ricerca di soluzioni innovative capaci di conquistare nuove fette di mercato.

L'impatto che il CAD ha avuto nell'industria si evince osservando il suo utilizzo nella quasi totalità delle aziende dove è richiesta una progettazione grafica di qualsiasi tipologia.

CNC - Computer Numerical Control

Uno dei concetti chiave che ha rivoluzionato il mondo della produzione industriale riguarda il suo processo di automazione. A partire dagli anni ottanta far eseguire interi cicli di lavorazioni a macchine ad altissima precisione in modo completamente automatizzato, rapido ed economico è diventato possibile grazie all'introduzione delle Computer Numerical Control.

Le macchine CNC sono delle macchine utensili comandate da un elaboratore elettronico, il quale gestisce un programma di lavoro comprendente tutte le istruzioni per creare tagli, incisioni, sculture, ecc. Le istruzioni impiegate sono generalmente un'estensione del cosiddetto codice ISO 6983 che è un linguaggio standardizzato nel quale si stabilisce una base comune per i comandi di tutti i tipi di CNC.

Solitamente una macchina CNC è comandata da un computer esterno attraverso delle linee di comunicazione. Esso svolge l'attività di controllo gestendo le istruzioni da eseguire in funzione dei dati forniti da encoder che fungono da feedback sul movimento e sulla posizione dei componenti su cui sono montati. Il movimento dei vari corpi meccanici è controllato da motori elettrici lungo degli assi; in questo contesto parlare di quanti assi ha una determinata macchina significa indicare qual è il numero dei gradi di libertà a sua disposizione.

Oggigiorno queste macchine vengono impiegate in quasi tutti i campi della meccanica. Le più comuni sono torni, presse piegatrici, saldatrici, fresatrici e macchine da taglio. Generalmente sono capaci di lavorare materiali diversi come legno, metalli o marmo. La qualità delle lavorazioni raggiunge livelli impensabili manualmente, con una precisione che può arrivare a 100 nanometri.

Il poter realizzare diversi tipi di lavorazioni si deve alla possibilità di cambiare l'utensile ancorato al mandrino, il quale è un dispositivo meccanico capace di serrare e tener fermo qualsiasi pezzo di forma circolare, quadrata o esagonale. Ad ogni lavorazione corrispondono particolari utensili appositamente montati su basi compatibili con il funzionamento del mandrino. Un esempio è rappresentato dagli autocentranti usati normalmente sui trapani, montati su basi coniche per essere installati nei torni.

Esempio di lavorazione per mezzo di una macchina CNC

Per sfruttare appieno le potenzialità di queste macchine si deve tener conto di un aspetto fondamentale, quello dei tempi morti. Questi ritardi sono dovuti alla procedura necessaria al cambio degli utensili e ai percorsi che questi devono eseguire in base alle lavorazioni richieste.

Anche se molte macchine non richiedono più l'intervento manuale per la loro sostituzione, questi tempi comunque non sono nulli e devono essere gestiti in modo ottimale per ridurli al minimo indispensabile.

Se si vuole avere il massimo rendimento da una macchina CNC bisogna prestare particolare attenzione anche alla fase di traduzione delle lavorazioni necessarie per realizzare le forme desiderate in sequenze d'istruzioni adeguate.

CAM - Computer-Aided Manufacturing

L'anello di congiunzione tra il sistema CAD e le macchine CNC è rappresentato dal CAM. Questo strumento, serve ad assistere la fabbricazione del prodotto per mezzo del computer. Pertanto il suo compito principale consiste nel ricevere modelli di geometrie bidimensionali o tridimensionali (normalmente creati con un sistema CAD) e generare il programma di lavoro per una macchina a controllo numerico computerizzato.

Il criterio per determinare la qualità di questi sistemi risiede nella loro capacità d'ottimizzare le lavorazioni. Disporre di un programma CAM di qualità superiore permette di ridurre i tempi di lavorazione e di conseguenza si ha un aumento della produttività e un guadagno in termini di denaro.

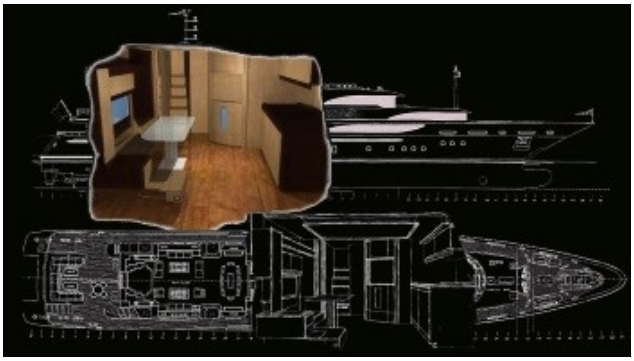
I CAM sono nati come sistemi stand alone, ossia avevano il solo compito d'importare le forme da realizzare e tradurle in istruzioni a seconda dei parametri di lavorazione settati. Attualmente, si trovano spesso sistemi CAD integrati con quelli CAM che consentono all'utente, attraverso un solo software, sia la creazione dei prototipi sia la generazione del codice per la loro realizzazione. Tali sistemi vengono chiamati CAD/CAM. Spazio3D è esattamente uno di questi.

1.3 Prodotto CAD

Spazio3D è il prodotto sviluppato dalla BrainSoftware e consiste in un CAD tridimensionale costituito da moduli software compatibili che consentono di avere una configurazione personalizzata adatta ad ogni necessità e budget.

Rivolto in modo particolare al settore del mobile, si presta per qualsiasi attività in cui venga richiesta la progettazione grafica, trovando largo impiego anche in altre industrie come quella del marmo o aeronavale e presso studi tecnici per l'architettura

d'interni per i quali la sua grande libertà realizzativa permette di comporre ambienti secondo qualunque stile desiderato, dal moderno al classico, dal salotto al bagno.

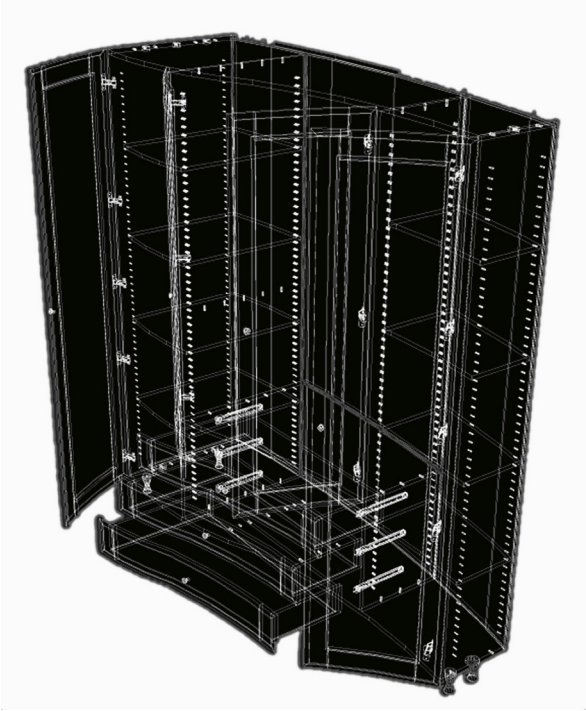


Attualmente disponibile in cinque lingue: italiano, inglese, tedesco, francese e spagnolo, Spazio3D verrà presto tradotto anche in altre lingue per essere proposto ai nuovi mercati esteri emergenti.

Realizzato come un ambiente di lavoro "user friendly", questo CAD assiste l'utente passo dopo passo nella costruzione di qualsiasi tipologia di modelli tridimensionali, potendo inoltre utilizzare innumerevoli funzioni integrate, capaci di generare in automatico tavole tecniche, preventivi, rappresentazioni fotorealistiche e la gestione dell'intero iter produttivo con validi strumenti quali rapporti di produzione, di fase e molti altri ancora.

Distribuito in molte versioni, permette al cliente di scegliere quella che più si adegua alle proprie necessità. A seconda della versione scelta, è possibile integrarla con moduli aggiuntivi in grado di ampliarne enormemente le funzionalità, quali:

- il modulo CAM, che permette con un unico applicativo, di gestire le varie fasi realizzative del prodotto, dalla progettazione alla produzione.
  Con procedure automatiche o manuali, questo modulo facilita sia il lavoro di caricamento delle lavorazioni di giuntura e ferramenta (migliorandone la visione d'insieme), sia la gestione di qualsiasi macchina a controllo numerico a 3 assi + n, supportando varie tipologie di lavorazioni come forature, sinature, fresature e tagli con lama.

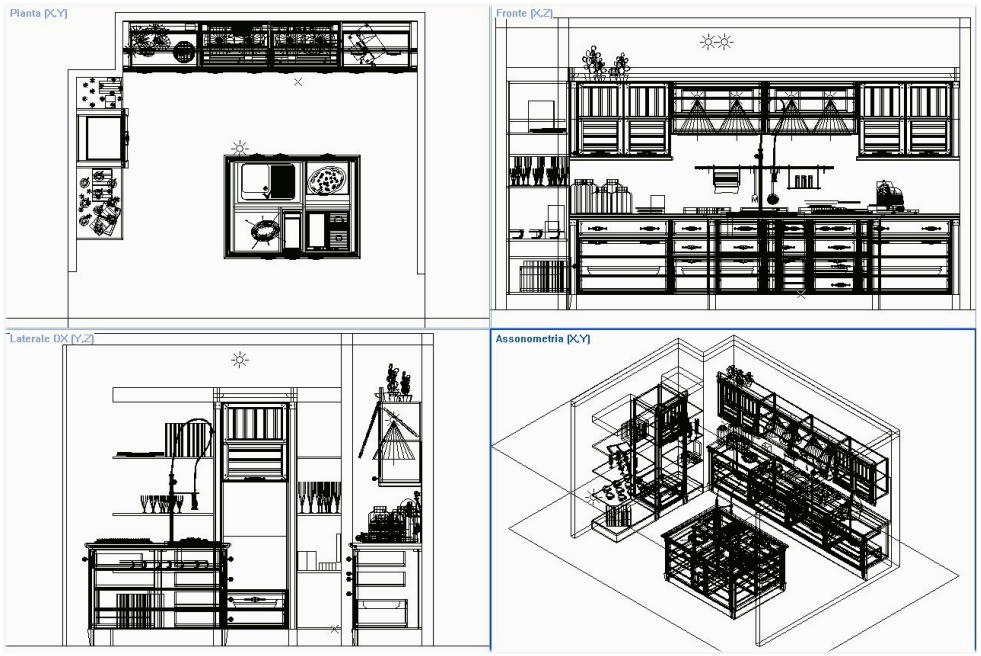


- il modulo OTTIMIZZAZIONE PANNELLI, consente l'ottimizzazione del nesting delle lavorazioni. Nell'ambito dei centri di lavoro delle linee di produzione è di fondamentale importanza ridurre gli scarti di lavorazione assieme ai tempi per il set up e riavvio delle macchine predisposte all'esecuzione dei piani di taglio. Dunque, gestendo al meglio il posizionamento delle sagome, questo modulo rappresenta uno strumento estremamente utile e soprattutto redditizio.
- il modulo FABBISOGNO, permette la gestione e la quantificazione dei materiali utilizzati nel progetto, aiutando l'azienda nell'ammistrazione delle necessità e nell'organizzazione delle giacenze di magazzino.

Oltre ai moduli elencati, ne sono disponibili molti altri, in grado di coprire quanti più aspetti possibili dell'iter produttivo.

La possibilità di attingere ad un ricco archivio espandibile, con ben oltre 300 elementi, permette al cliente di adattarli ed assemblarli a piacere secondo le proprie esigenze. La creazione di nuove strutture più complesse ed articolate viene così enormemente semplificata, lasciando più tempo al progettista per la cura di altri lavori.

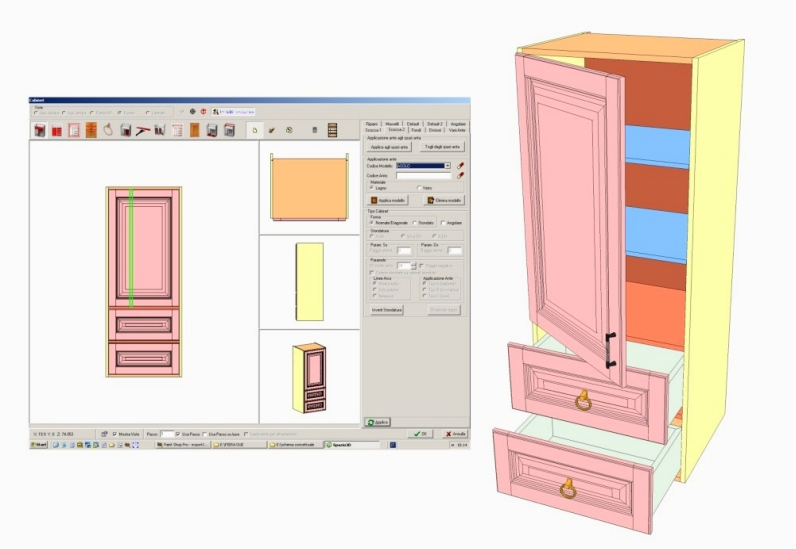
L'interfaccia grafica con cui viene visualizzata la scena è costituita da quattro differenti prospettive, dove le prime tre sono rappresentazioni dei piani (X, Y), (X, Z) e (Y, Z), mentre la quarta è costituita da un'assonometria del piano (X, Y).



Viste della scena secondo le proiezioni di Spazio3D

L'insieme di queste viste permette di lavorare agevolmente all'interno della scena che sarà composta da entità aventi delle specifiche proprietà in relazione alla tipologia a cui appartengono.

Essendo rivolto all'industria del mobile questo CAD fornisce, oltre alla costruzione di entità geometriche generiche, la possibilità di creare oggetti specifici per l'arredo attraverso la definizione assistita di cabinet in grado di gestire in automatico molte lavorazioni di modellazione semplicemente settando alcuni parametri di costruzione come lo spessore di una mensola o la distanza tra gli elementi.



Creazione assistita di un cabinet

Per dare realismo agli elementi si possono attribuire delle proprietà fisiche alle superfici con l'assegnazione del valore di trasparenza, il grado di rifrazione, la specularità, etc. In questo modo, unitamente all'applicazione di texture, si possono creare diversi effetti visivi che variano in base alle caratteristiche e al posizionamento delle fonti luminose presenti nella scena.

La visualizzazione di questi effetti avviene per mezzo di un processo render, capace di generare l'accesso virtualmente all'interno degli spazi progettati e dei mobili. Spazio3D offre, per una visione fotorealitica ed estremamente dettagliata dell'ambiente, la possibilità di usufruire della tecnica del Ray Tracing particolarmente utile per tutte le aziende che necessitano della cura del design per la creazione ad esempio di brochure pubblicitarie.

Immagine fotorealistica realizzata con la tecnica Ray Tracing

Il core di Spazio3D è costituito da un motore grafico realizzato in modo tale da utilizzare in modo nativo le specifiche definite da OpenGL, questo perchè le moderne schede grafiche hanno già al loro interno implementate le funzioni dello standard, sfruttandone ove possibile l'accelerazione dell'hardware, con un conseguente miglioramento globale delle prestazioni.

L'insieme di tutti questi aspetti rendono Spazio3D uno strumento di lavoro estremamente valido, in grado di fare la differenza nell'economia interna di qualsiasi azienda lo utilizzi.

1.4 Obiettivo del progetto

Per offrire un prodotto sempre all'avanguardia ed aderente alle richieste avanzate dai propri utenti, l'azienda si impegna costantemente nell'individuazione di nuove funzionalità da poter implementare.

Spesso, alcuni nuovi clienti e altri che invece già utilizzano Spazio3D ma che collaborano con terzi, hanno la necessità di utilizzare archivi creati con altri sistemi CAD composti da svariati modelli che risultano essere di vitale importanza per il proseguo del loro lavoro. Ogni CAD, avendo una propria metodologia di rappresentazione, memorizza i dati in modo differente, rendendoli a volte non interscambiabili.

L'obiettivo del tirocinio è stato dunque quello d'individuare e sviluppare un meccanismo capace di permettere la collaborazione tra programmi CAD con architetture interne differenti.

La soluzione che più risponde alle indicazioni date, si basa sulla realizzazione di una comunicazione asincrona attraverso un sistema d'esportazione ed importazione delle informazioni, memorizzate in file secondo uno standard open source che non sia vincolato ad altri formati proprietari.

Questo standard è stato individuato in COLLADA, un formato creato appositamente per l'interscambio tra applicazioni di grafica 3D, senza la perdita di informazioni. Realizzato in codice XML, il contenuto dei file viene salvato in formato testuale, per garantire così la massima compatibilità con i linguaggi di programmazione e una maggiore facilità nella lettura e modifica dei dati. Essendo definito in modo assolutamente generale, è in grado di rappresentare tutti i principali elementi delle numerose applicazioni 3D sul mercato.

Utilizzando costrutti propri di OpenGL questo standard è, inoltre, particolarmente adatto per essere utilizzato dal motore grafico di Spazio3D.

2 Tecnologie utilizzate

Di seguito verrà presentata una panoramica generale delle tecnologie software inerenti al contesto tecnologico nel dominio d'interesse del prodotto Spazio3D e del progetto svolto.

L'esposizione degli argomenti sarà volta a presentarne gli aspetti più significativi, tralasciando quelli più tecnici e specifici, che esulano dall'obiettivo di questa relazione.

Nella prima parte del capitolo sarà presentato l'ambiente di programmazione Borlan C++ Builder, con cui è stato realizzato Spazio3D e l'implementazione del progetto.

La seconda parte verrà dedicata a XML ed in particolare a XML Schema, essendo il linguaggio schema quello utilizzato per definire lo standard COLLADA.

Saranno presentati anche i due CAD utilizzati durante il progetto, per la verifica e comparazione del lavoro svolto e, infine, seguirà una breve descrizione di OpenGL e del suo impiego.

2.1 Borland C++ Builder

BCB (Borland C++ Builder) è un ambiente di sviluppo RAD (Rapid Application Development) per la programmazione visuale orientata agli oggetti, con il quale è possibile creare in modo semplice e rapido applicazioni distribuibili in ambiente Windows, tra cui lo stesso Spazio3D.

Creato da un progetto della Borland, che nel 1997 ne distribuì la prima versione, trae le proprie origini da un altro ambiente di programmazione sviluppato in linguaggio Object Pascal di nome Delphi. L'obiettivo di questo progetto era creare un compilatore per il linguaggio C++ in grado appunto di utilizzare la struttura già collaudata del Delphi.

Il risultato fu la creazione di un compilatore capace di tradurre il codice C++ comprensivo d'istruzioni scritte in Object Pascal che permise il riutilizzo dell'intera libreria VCL e dell'IDE. La loro comprensione risulta necessaria nella comprensione di BCB ed è questo il motivo per cui verranno di seguito esaminate.

Linguaggio C++

Sviluppato nel 1983 da Bjarne Stroustrup nei laboratori della Bell, il linguaggio C++ è l'evoluzione del linguaggio C.

Pur essendo un linguaggio molto potente e flessibile, il linguaggio C non riusciva a far fronte ad appicazioni sempre più articolate a causa della mancanza di una base di tipo modulare. La stesura di grandi programmi con le poche strutture standard messe a disposizione, costringeva il programmatore ad una continua ed inutile complicazione del codice, fino a farlo diventare ingestibile.

L'introduzione di un paradigma di programmazione gerarchico orientato agli oggetti, insieme alle nuove strutture in grado di realizzare funzioni virtuali, l'overloading degli operatori, i template e la gestione delle eccezioni, resero il C++ un linguaggio "all purpose". Con esso, infatti, è possibile sviluppare qualsiasi tipo d'applicazione potendo gestire direttamente dettagli di programmazione a livelli molto specifici, che generalmente vengono controllati e svolti dal compilatore. Tutti questi aspetti hanno reso il linguaggio C++ particolarmente adatto ad essere impiegato in contesti anche molto diversi tra loro, come quello dei sistemi real time, dei data base o anche di sistemi operativi.

Nonostante gli enormi pregi di questo linguaggio, è doveroso sottolineare che la grande libertà lasciata al programmatore spesso rappresenta anche la fonte principale dei suoi errori.

VCL

Quando si sviluppano delle applicazioni, ci si ritrova spesso a dover riutilizzare frammenti di codice già scritti in altre classi, le quali vengono racchiuse in librerie per ottimizzarne l'organizzazione.

In un contesto visuale, la necessità di riutilizzo di componenti grafici è molto frequente nell'interfaccia con l'utente attraverso finestre di dialogo che usano soprattutto elementi propri dell'ambiente Windows chiamati controlli, come pulsanti, etichette di testo o le finestre stesse. Questi controlli per loro natura non hanno proprietà o metodi, ma vengono utilizzati attraverso un sistema di messaggi.

La libreria VCL funge da framework ed ha il compito di incapsulare tutti quei controlli Windows maggiormente utilizzati nascondendo la loro gestione, portandoli ad un livello di programmazione più alto. In questo modo possono essere trattati come normali oggetti chiamati componenti, aventi delle proprietà, dei metodi ed eventi.

Componenti grafici

Un metodo è semplicemente una funzione che è membro di una classe. Definisce il comportamento dell'oggetto a cui appartiene e può accedere a tutte le proprietà pubbliche, protette e private della classe.

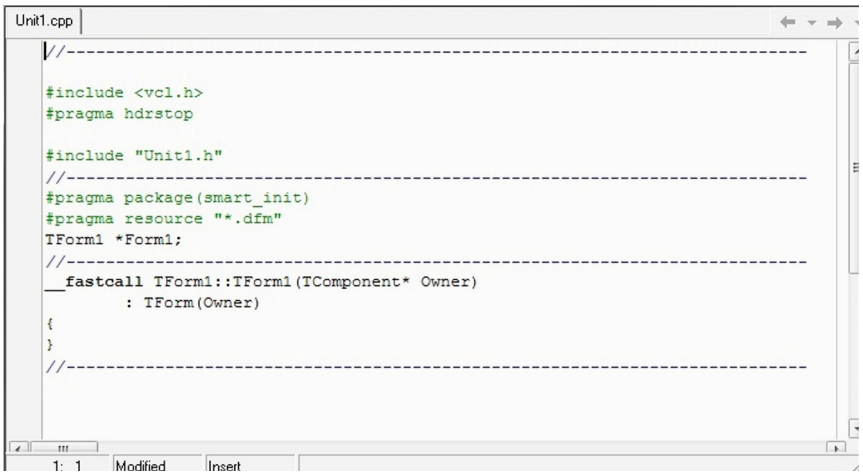
Le proprietà di un componente rappresentano l'insieme delle caratteristiche tecniche e visuali che ne determinano il funzionamento e l'aspetto. Con un loro corretto utilizzo è possibile gestire in modo intelligente l'interfaccia dell'applicazione. Potendo scegliere ad esempio di visualizzare determinati componenti o abilitarne alcuni piuttosto che altri, in funzione ad alcuni parametri, si può rendere più chiaro il significato stesso dei componenti e di conseguenza facilitarne l'interazione.

Quando l'utente interagisce con un componente (come nella sua selezione mediante un clic) viene attivato un evento che non è altro che la chiamata ad una determinata funzione con il compito di eseguire delle specifiche azioni di risposta. Un programmatore, quando inserisce un componente all'interno della finestra di dialogo che sta definendo, sceglie tra gli eventi propri di quel dato componente quelli che sono importanti e quindi li andrà ad implementare. Con questa tecnica degli eventi la sequenza esatta di istruzioni eseguite cessa di esistere, divenendo una variabile dipendente dal futuro utilizzatore dell'applicazione.

IDE

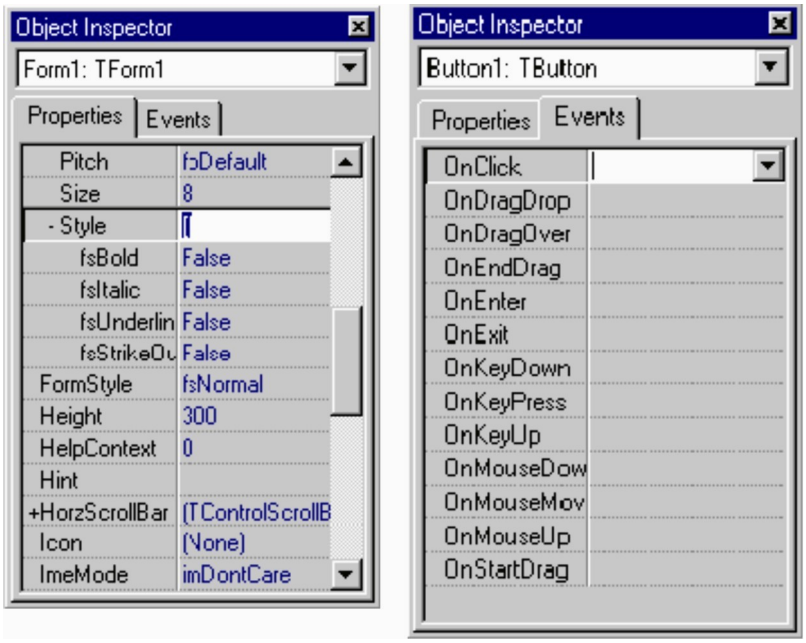
Per aiutare un programmatore nel suo lavoro, BCB mette a disposizione un IDE (Integrated Development Environment), cioè un ambiente di sviluppo integrato che si presenta attraverso un'interfaccia grafica comprensiva di svariati strumenti atti a facilitare e soprattutto velocizzare la progettazione di nuove applicazioni. Tra questi, quelli di maggior interesse sono: la Component palette, l'Object Inspector, il Debugger integrato e il Code Editor.

Il Code Editor non è altro che un normale editor di testo a cui sono state aggiunte alcune funzioni di supporto per il programmatore. Alcune di queste sono semplici come la colorazione diversificata, rispetto al resto del codice, delle parole riservate proprie della sintassi del linguaggio mentre altre, più complesse, sono molto utili per la progettazione. Un esempio è rappresentato dalla possibilità di visualizzare, attraverso una ricerca automatica, la definizione di una classe semplicemente selezionandone il nome scritto all'interno del codice.



Code Editor

L'utilizzo della Component palette e dell'Object Inspector, per quanto riguarda la gestione dei componenti della libreria VCL, è di vitale importanza. La prima consiste nel raggruppamento (per insiemi d'appartenenza logica) dei componenti, i quali possono essere selezionati e trascinati all'interno della finestra di dialogo che si sta allestendo. In questo modo avviene l'effettivo inserimento nel progetto delle relative istanze, ognuna avente le proprie specifiche proprietà ed eventi, gestibili attraverso l'Object Inspector. Questo secondo strumento è formato da due distinte schede, una con la lista delle proprietà con i relativi valori e l'altra contenente la lista degli eventi gestibili con i metodi che saranno chiamati in corrispondenza alla loro attivazione.



Object Inspector

L'ultimo strumento qui trattato è il Debugger, supporto fondamentale per la verifica della correttezza logica del codice scritto. La sua importanza consiste nel permettere, durante l'esecuzione del programma, la visualizzazione delle proprietà appartenenti a qualsiasi tipo d'istanza in modalità step-by-step, potendo così cogliere eventuali errori logici altresì difficili da identificare. Questo strumento si è rivelato di vitale importanza, senza il quale sarebbe stato difficile portare a compimento lo sviluppo del progetto.

2.2 XML

Per comprendere esattamente cos'è COLLADA, è necessario prima introdurre il mondo dei linguaggi markup con XML ed in particolare di XML Schema. Esso, infatti, costituisce la base dalla quale sono stati definiti migliaia di linguaggi markup, atti alla memorizzazione di dati nei più svariati domini d'applicazione, compreso appunto quello della grafica 3D.

Linguaggi Markup

Il linguaggio XML (Extensible Markup Language) è un framework per la definizione di linguaggi markup. In verità XML non è un linguaggio vero e proprio ma piuttosto una notazione comune per poterne costruire altri.

L'origine dei linguaggi markup risale al 1970 ad opera di Charles Goldfarb che sviluppò, per l'IBM, l'idea di creare un linguaggio che permettesse lo scambio di documenti da un computer ad un altro. Questo linguaggio però doveva poter "resistere" per molti anni, indipendentemente dall'evoluzione dei computer stessi e delle applicazioni software che l'avrebbero utilizzato. Questo progetto culminò con la creazione di SGML (Standard Generalized Markup Language conosciuta anche come ISO 8879) che fu standardizzato nel 1986.

Questo standard è un metalinguaggio con lo scopo di definire linguaggi di marcatura generici per la stesura di documenti di testo destinati ad essere trasmessi ed archiviati con strumenti informatici. Questi linguaggi perciò servono ad aggiungere una struttura formale al contenuto che ne descrive la modalità con cui essere interpretati.

Con l'introduzione di SGML vennero utilizzate per la prima volta sia le parentesi angolari per esprimere una struttura che il linguaggio DTD (Document Type Definition); quest'ultimo aveva il compito di definire in maniera rigorosa la sintassi e la struttura logica dei documenti in relazione allo specifico linguaggio utilizzato.

SGML diventò molto importante perché rappresentava il modello per lo sviluppo di molti altri linguaggi, che successivamente rivoluzionarono il mondo dell'informatica. Tra i più celebri si ricorda HTML (HyperText Markup Language) che assieme al protocollo HTTP (HyperText Transfert Protocol), usato per la trasmissione di ipertesti (documenti messi in relazione tra loro per mezzo di parole chiave), e il concetto di URL (Uniform Resource Locator), cioè una sequenza di caratteri che identificano univocamente l'indirizzo di una risorsa, furono impiegati da Tim

Berners-Lee nel 1990 nello sviluppo dell'infrastruttura ben nota come WWW (World Wide Web).

Derivato da due concetti indipenenti, cioè quello degli ipertesti e quello dei linguaggi markup, HTML ha il compito di definire degli ipertesti che comprendano semplici meccanismi di formattazione e la possibilità di poterli collegare ad altri documenti o file.

La rapida diffusione che questo linguaggio ebbe ne fece emergere, oltre alle enormi potenzialità (anche economiche e commerciali), il suo grande limite che consiste nel fatto che i suoi elementi sono predefiniti e non sufficienti a soddisfare le crescenti nuove richieste degli utenti.

Per ovviare a queste limitazioni, verso la metà degli anni '90, ebbe inizio lo sviluppo di una nuova tecnologia che aveva l'obiettivo di portare nel web una versione semplificata di SGML in grado di definire in modo semplice nuovi linguaggi markup.

Nacque così nel Novembre del 1996, come puro sottoinsieme di SGML, la prima bozza di XML. Nel febbraio del 1998 questo nuovo linguaggio divenne uno standard del W3C (World Wide Web Consortium), l'organizzazione internazionale non governativa diretta da Tim Berners-Lee, che ha come scopo quello di sviluppare tutte le potenzialità del WWW stabilendo standard tecnici inerenti sia ai linguaggi markup sia ai protocolli di comunicazione.

Ben presto ci si accorse che le possibilità di XML andavano oltre il solo contesto web ma potevano essere estese alla rappresentazione di qualsiasi tipologia d'informazione strutturata. Questa opportunità è resa possibile potendo creare nuovi elementi a cui associare uno specifico significato in base alle proprie necessità rappresentative.

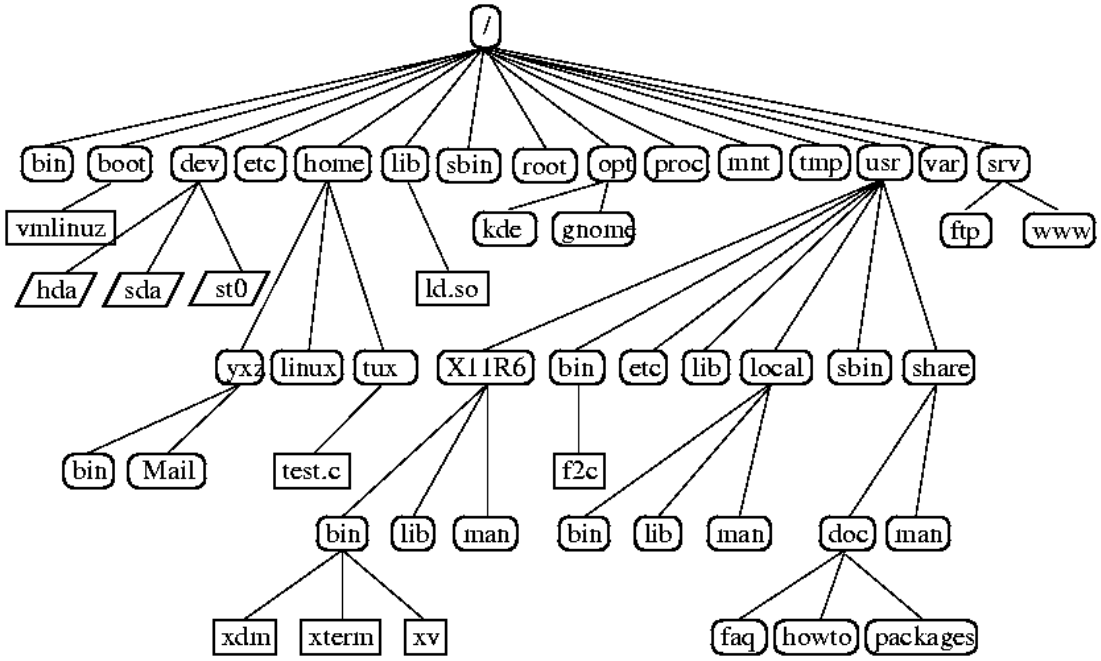
Attualmente i linguaggi creati con XML (che prendono il nome di applicazioni XML) sono migliaia e sono impiegati nei più svariati campi che vanno dalla semplice formattazione di testi alla rappresentazione della struttura del DNA o ai dati contenuti nei DBMS (Database Management System).

Documenti XML

La rappresentazione concreta di un documento XML non varia rispetto al modello concettuale utilizzato. Il contenuto consiste semplicemente in un testo Unicode con tag di marcatura e meta-informazione per la descrizione di elementi, attributi e nodi.

La sua semplicità di memorizzazione ne consente un'estrema facilità di lettura e modifica. Infatti, con l'ausilio di un normale editor di testo, è possibile caricare il documento e riconoscerne agevolmente non solo gli elementi che lo compongono ma anche il loro significato perché ad essi è associato un nome che li descrive.

L'insieme di tutti gli elementi va a formare una struttura di tipo gerarchica, chiamata albero XML; in analogia può essere paragonato alla struttura delle directory nel file system di un computer.



Struttura ad albero di un file system

La terminologia usata per la descrizione di tali alberi non è omogenea e generalmente viene utilizzata quella del linguaggio XDM (XPath Data Model). Pur essendo stato creato per la localizzazione dei nodi e non per la definizione della struttura degli elementi all'interno di un documento XML, ne è stata adottata la semantica per la sua semplicità e completezza. Probabilmente però il fattore determinante della sua diffusione risiede nell'ampio uso del linguaggio stesso da parte dei programmatori XML.

In XDM un albero XML è associato ad un caso particolare di albero ordinato in cui i nodi possono appartenere a sei diverse categorie:

- **Nodo radice**: da non confondere con l'elemento radice, esso rappresenta l'inizio del documento. I suoi figli possono comprendere un numero arbitrario di commenti e istruzioni per l'elaborazione ma un solo elemento radice.
- **Nodo di testo**: contiene una stringa mai vuota che corrisponde ad un frammento d'informazione. Non può avere figli e perciò è sempre una foglia dell'albero.
- **Nodo elemento**: definisce un raggruppamento logico dell'informazione, rappresentata dai suoi discendenti. Ad esso è associato un nome che lo descrive.
- **Nodo attributo**: costituito da una coppia nome/valore, ha lo scopo di definire ulteriori proprietà di raggruppamento del nodo elemento a cui è associato.
- **Nodo commento**: è una foglia speciale, etichettato con una stringa di testo. Esso esprime in modo informale metainformazioni ed è spesso ignorato dalla maggioranza degli strumenti che elaborano il documento.

- **Nodo d'istruzione per l'elaborazione**: ha lo scopo di trasmettere metainformazioni specializzate agli strumenti che utilizzano il codice XML. La tramissione avviene per mezzo di una coppia target/valore. L'obiettivo rappresentato dal target è una parola che specifica il tipo di strumento a cui si rivolge mentre il valore è una stringa contenente l'informazione.

Un nodo attributo non figura mai nell'insieme dei figli di un nodo elemento ma allo stesso tempo ne ha sempre uno come padre. Un nodo elemento dunque può avere, come figli, solamente altri elementi, commenti oppure istruzioni per l'elaborazione. Gli attributi perciò sono da considerarsi come un insieme di proprietà associato all'elemento.

Con questa terminologia, un documento può essere considerato sia per la sua rappresentazione testuale sia per la sua struttura ad albero. Sfruttando queste regole semantiche si è in grado di dare un significato ai dati contenuti in un elemento, sapendone solamente il nome e la sua collocazione nella gerarchia degli elementi. In questo modo si è potuto dare ai dati strutturati all'interno di un documento XML la proprietà di essere autoesplicativi.

Quando la nidificazione degli elementi nella struttura rispetta le regole sintattiche stabilite dal linguaggio XML si dice che il documento è ben formato ed è la condizione necessaria affinchè abbia un significato mentre, quando si crea un nuovo linguaggio, si vuole definire una particolare famiglia di documenti che rispetti una specifica sintassi. La sua definizione formale prende il nome di schema. Se sono rispettate le regole sintattiche di un determinato schema dato, si dice che il documento è valido.

Il concetto di corretta formazione per un documento XML esprime perciò sia l'essere o meno ben formato, sia l'essere valido.

XML Schema

Come già detto XML è una notazione per la definizione di altri linguaggi markup. Si è visto che per definire un nuovo linguaggio è necessario delineare il suo corrispondente schema. A questo scopo sono stati creati molti linguaggi di descrizione in grado di specificare il contenuto di un documento XML, tra questi ci sono DDML, SOX e DTD.

Il più importante fra questi è DTD (Document Type Definition), progettato, come molti altri, come un sottoinsieme di SGML. Largamente utilizzato per la validazione di un documento istanza, presenta però delle carenze molto gravi. Le principali sono sicuramente l'impossibilità di supportare i namespace (poiché è precedente rispetto alla loro adozione) e il non utilizzare una notazione XML.

Per risolvere questi problemi lo stesso consorzio W3C annunciò, attraverso una nota contenente alcune linee guida, lo sviluppo di un linguaggio schema di nuova generazione. Questo progetto terminò nel 2001 quando l'XML Schema Working Group pubblicò la direttiva XML Schema.

La nuova specifica creata, diversamente da quanto sperato, non soddisfa interamente i requisiti originariamente prefissati. Infatti, nonostante la sua complessità e potenza espressiva, non riesce a rappresentare completamente particolari tipi di schemi e risulta inoltre, contrariamente all'iniziale intento di semplificazione, molto difficile da comprendere anche per gli esperti di XML.

XML Schema, che rappresenta comunque uno dei migliori strumenti per la definizione di schemi XML, è l'unico ad aver raggiunto la validazione ufficiale del W3C.

Un'ultima considerazione degna di nota riguarda il suo sofisticato sistema di tipi di dato, ispirato ai linguaggi di programmazione orientati agli oggetti che fornisce ben 19 tipi di dati primitivi (come boolean, string, float), ai quali si aggiunge la possibilità di crearne altri, derivati da questi ultimi, attraverso tre possibili meccanismi: la restrizione, la lista e l'unione.

Questa struttura di tipi di dato rende agevole l'accostamento in ambienti di sviluppo come il BCB, che ha una quasi completa compatibilità con quelli definiti nello standard COLLADA.

2.3 SketchUp e Blender

Durante la realizzazione del progetto, sono stati utilizzati diversi sistemi CAD, per avere un confronto diretto che permettesse di valutare la correttezza del lavoro svolto. In particolare sono stati utilizzati due CAD con licenza freeware: SketchUp e Blender.

Il primo è un modellatore 3D molto semplice ed intuitivo, realizzato per la progettazione architettonica, l'urbanistica, l'ingegneria civile e la creazione di video giochi. La creazione di modelli viene assistita con strumenti molto flessibili ed efficati, come il push/pull, in grado di realizzare figure tridimensionali partendo dall'estrusione di forme bidimensionali.

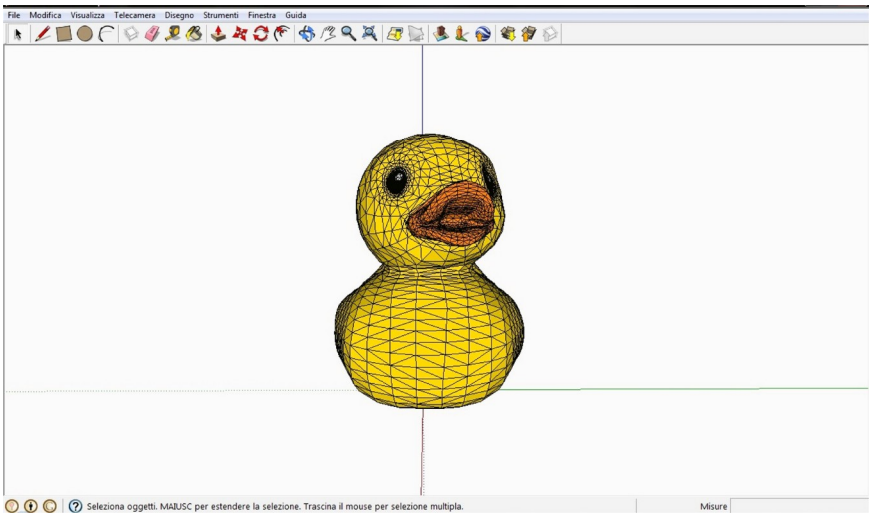
Una particolarità di SketchUp consiste nel suo rendering non fotorealistico basato sulla tecnologia OpenGL. Questa tecnica permette al disegnatore di lavorare senza preoccuparsi della precisione nei dettagli ma al contempo non consente l'inserimento nella scena di elementi come lampade e luci localizzate, se non attraverso dei plugin prodotti da terzi.

Anche Blender è un modellatore 3D ma rappresenta uno strumento molto più completo, potendo realizzare una gran varietà di funzionalità come rigging, animazioni, simulazione di fluidi, fotorealismo, etc.

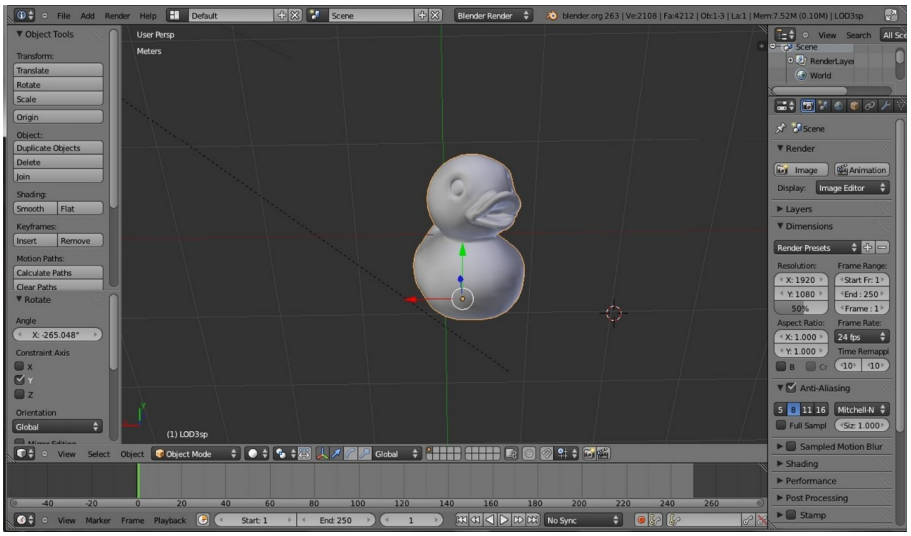
Pur essendo CAD molto potenti ed utilizzati, entrambi gestiscono l'import e l'export di file COLLADA in modi differenti. SketchUp ad esempio non importa tutte le entità, come le luci e le telecamere ma riesce a gestire correttamente tutte le primitive. Blender invece ha dimostrato di avere molte carenze nei confronti di

questo formato, non riuscendo a rappresentare correttamente le trasformazioni all'interno della gerarchia dei nodi che compongono la scena.

Ai fini del progetto, questi due CAD sono stati molto utili e per il loro impiego si è mirato a sfruttarne i rispettivi punti di forza.



Rappresentazione di una papera con SketchUp



Rappresentazione della stessa papera con Blender

2.4 OpenGL

Una delle librerie più importanti ed utilizzate nella computer grafica è l'Open Graphics Library. Essa raccoglie un insieme di funzioni di basso livello per disegnare complesse scene tridimensionali a partire da semplici primitive che saranno poi convertite in pixel attraverso un processo di rasterizzazione. Il suo compito è di definire il comportamento preciso che queste funzioni devono avere, fornendo un

API unica ed uniforme adatta per essere implementata con qualsiasi linguaggio a prescindere dalla piattaforma utilizzata.

Il suo sviluppo inizia nel 1992 a cura della Silicon Graphics Inc ed è stata inizialmente supervisionata dall'ARB (OpenGL Architecture Review Board), un'associazione i cui membri fondatori sono per lo più industrie leader del settore. Tra i più noti vi sono NVIDIA, ATI Technologies, Intel, Dell, Apple Computer, Sun, Microsoft e moltri altri ancora. Molti tra questi produttori realizzano le procedure direttamente nell'hardware per sfruttarne l'accelerazione ed averne un vantaggio in termini di prestazioni. La maggior parte delle schede grafiche moderne hanno la possibilità di eseguire direttamente la maggior parte delle funzioni di OpenGL, sfruttando così le prestazioni della GPU in modo quanto più completo.

Nel 2007 il controllo di OpenGL è passato nelle mani della Khronos Group, che aveva l'intento di poterne migliorare la diffusione e le capacità. Questo consorzio è lo stesso che cura lo sviluppo di COLLADA ed è questo il motivo per cui nello standard si ritrovano molti costrutti propri di OpenGL.

3 COLLADA

In questo capitolo, dopo una breve introduzione di COLLADA, ne verranno presentate le caratteristiche principali assieme ai concetti di base della grafica computerizzata. Le informazioni rappresentabili con questo standard sono molteplici; alcune di queste, come la gestione delle animazioni, non sono state prese in considerazione perché non utilizzabili in Spazio3D. L'analisi seguente sarà quindi rivolta ai soli aspetti realmente trattati nel progetto.

La prima parte del capitolo sarà dedicata alla descrizione generale delle basi della computer grafica 3D, mentre nella seconda verranno approfondite le specifiche d'interesse del formato COLLADA.

Pur essendo disponibile la versione 1.5, quella presentata sarà la 1.4.1 che è stata scelta in quanto risulta essere attualmente quella maggiormente supportata nei programmi di grafica disponibili.

3.1 Introduzione

COLLADA è un COLLAborative Design Activity nato dalla collaborazione tra la Sony Computer Entertainment e molte altre compagnie come Alias SystemCorporation, Criterion Software, Autodesk, Avid Technology, etc. Lo presentarono per la prima volta nel 2004 a Los Angeles in occasione del SIGGRAPH (abbreviazione di Special Interest Group on GRAPHics and Interactive Techniques una conferenza annuale organizzata negli Stati Uniti sulla grafica computerizzata) e ne misero in risalto le grandi potenzialità per lo sviluppo di videogames (in particolare per console come Xbox e Play Station) poiché in grado di supportare tutti i principali elementi delle numerose applicazioni di grafica 3D sul mercato. Nel 2005 la Khronos Group Inc. acquistò il formato che dal 2006 è stato reso pubblico.

L'obiettivo del progetto COLLADA era quello di fornire alle applicazioni uno strumento comune in grado di memorizzare e scambiare attraverso file quante più qualità grafiche digitali possibili, secondo uno schema di tipo generale. Non essendo formulato come un meccanismo ottimizzato per lo sviluppo, doveva essere definito in modo tale da poter essere facilmente interpretato ed esteso.

Gli sviluppatori optarono così per la realizzazione di un'applicazione XML, definendo lo schema del linguaggio, utilizzando XML Schema. Così facendo, essendo di tipo testuale, i file creati possono essere gestiti con facilità, eliminando qualsiasi incompatibilità con i linguaggi di programmazione utilizzati per l'elaborazione dei dati in essi contenuti.

3.2 Computer grafica 3D

Prima di addentrarsi nelle specifiche, è bene fare una panoramica dei concetti chiave della computer grafica 3D. Esistono molte tecniche diverse per realizzare lo stesso concetto, perciò di seguito saranno trattate solo quelle effettivamente impiegate da COLLADA che spesso corrispondono a quelle definite nelle specifiche di OpenGL.

Scene

Per comprendere meglio questo concetto, si può paragonare una scena di un ambiente CAD al palcoscenico di un teatro allestito con i vari componenti per riprodurre uno specifico scenario. Una scena perciò consiste nella descrizione di ciò che si intende visualizzare, ossia la composizione di modelli tridimensionali in un ambiente virtuale.

Figure geometriche

Una delle tecniche utilizzabili per la definizione di figure geometriche complesse consiste in una composizione di geometrie semplici, definite da equazioni, su un sistema di riferimento cartesiano tridimensionale. Purtroppo, in questo modo l'insieme delle figure rappresentabili è limitato e richiede un dispendio enorme di calcoli per la loro rappresentazione.

Una soluzione migliore e maggiormente impiegata, consiste nell'utilizzo della modellazione poligonale. Questa tecnica prevede l'approssimazione di una superficie curva attraverso un processo di "sfaccettamento". In questo modo, la superficie viene rappresentata come un insieme di poligoni e/o linee chiamate primitive, ognuno dei quali è identificato dall'insieme dei suoi vertici. Nella modellazione solida una

collezione di vertici e facce, che definiscono la forma di un oggetto poliedrico, prende il nome di mesh.

Utilizzando dei poligoni ci si deve assicurare che i rispettivi vertici giacciano tutti sullo stesso piano. Molti software grafici per garantire questa proprietà utilizzano solo facce triangolari, in considerazione del fatto che tre punti sullo spazio possono identificare solamente un piano.

Se il modello così rappresentato risulta essere troppo grezzo, si possono applicare algoritmi capaci di aumentare la densità dei poligoni, arrivando a creare un effetto ottico in grado di simulare una curva perfetta.

Un'alternativa per la descrizione di superfici curve consiste nell'utilizzo delle spline, cioè l'interpolazione di curve polinomiali composte da archi di curve in successione, dette curve spline. La trattazione delle spline non viene approfondita perché non sono state implementate durante il progetto per mancanza di tempo.

Illuminazione

Per rendere l'immagine dell'ambiente più reale, è necessario ricostruisce l'interazione tra gli oggetti e le sorgenti di luce, ricorrendo ad un processo che simula la fisica della luce.

Per il calcolo dell'effetto visivo in un determinato punto bisogna disporre delle proprietà della luce, delle proprietà di riflessione e della normale alla superficie colpita. I modelli di illuminazione più semplici considerano solo la luce che viaggia direttamente da una sorgente luminosa ad un oggetto, valutandone così solo l'illuminazione diretta. Per avere invece un modello più globale e completo si può valutare anche l'illuminazione indiretta, cioè considerando anche la luce riflessa da un oggetto ad un altro finchè non perde interamente la sua energia.

Principalmente sono utilizzate quattro tipologie di sorgenti luminose: luce d'ambiente, luce direzionale, punto luce e spot.

Una luce d'ambiente o una direzionale non sono soggette ad attenuazione e mentre la prima illumina costantemente ogni oggetto presente nella scena da tutte le angolature, l'altra diffonde la luce secondo una precisa direzione.

L'illuminazione prodotta da un punto luce o uno spot è soggetta ad attenuazione e può essere paragonata alla luce emessa rispettivamente da una lampadina e da un faretto.

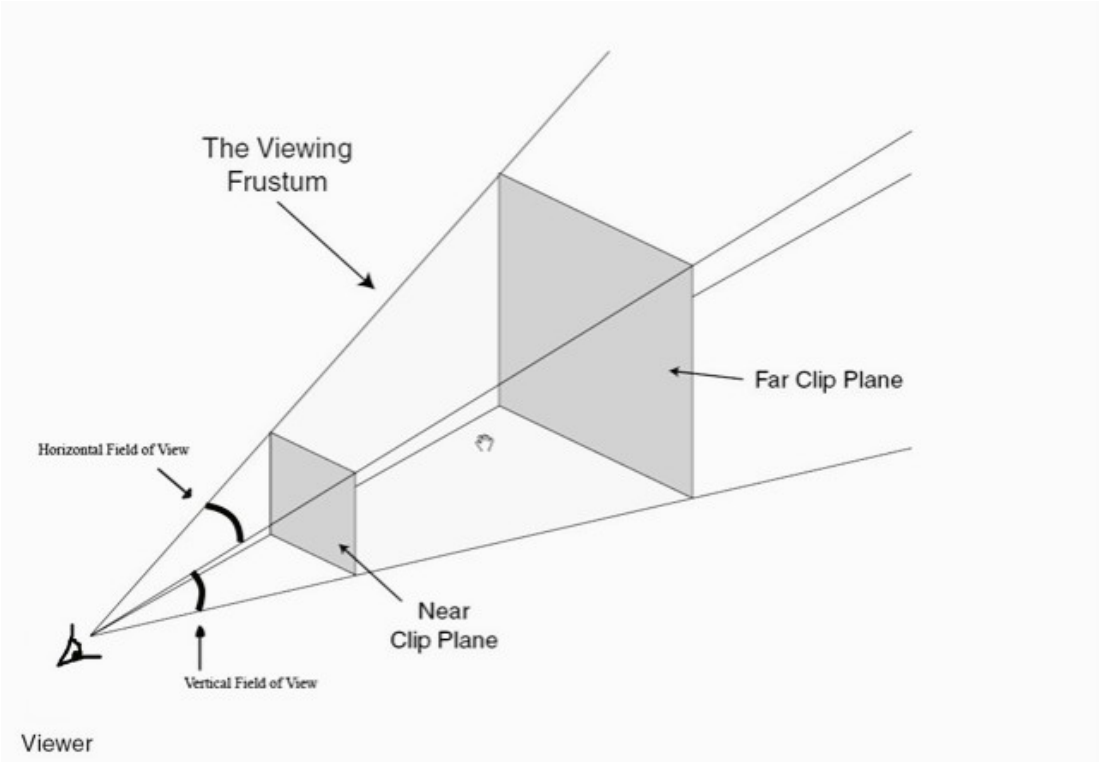
Telecamere

Spesso si vogliono realizzare delle proiezioni dell'ambiente creato rispetto a determinati punti di vista e per farlo si utilizzano delle telecamere virtuali in grado di "fotografare" la scena.

Nella simulazione di una telecamera reale vengono adottate due differenti strutture del volume di vista: ortogonali e prospettiche, nelle quali per la prima viene definito un parellelepipedo, invece per la seconda un tronco di piramide. Qualsiasi oggetto o

porzione di esso sia compreso in questo spazio verrà rappresentato nella proiezione mentre tutto il resto sarà semplicemente non considerato.

I parametri richiesti per la definizione dei volumi sono la distanza e la dimensione della faccia più vicina rispetto al punto d'osservazione unitamente alla distanza della faccia più lontana.



Colori e texture

Per dare maggior dettaglio all'effetto visivo della scena, possono essere assegnati dei colori alle facce dei modelli che la compongono. La rappresentazione digitale di un colore avviene principalmente secondo il modello RGB (o RGBA nel caso si voglia utilizzare anche il canale alpha per la descrizione della trasparenza/opacità), il quale definisce per ogni pixel la sintesi additiva dei colori rosso, verde e blu, con valori che variano in un range d'interi da 0 a 255 o di razionali da 0 a 1 nel caso siano normalizzati. Pur non riuscendo a rappresentare tutte le cromaticità, questo modello è attualmente quello più utilizzato.

Spesso, come nel caso dei videogiochi, si vuole dare un senso realistico alla scena e di conseguenza ai suoi componenti. Per creare questo effetto, si usano le texture. Questo metodo consiste nel riprodurre un'immagine bidimensionale (sprite), su una o più facce del modello. La sovrapposizione di più texture è utilizzata ad esempio per la creazione di effetti speciali come nebbia o luci particolari.

Rendering, Shader e Ray Tracing

Il termine rendering, nell'ambito della computer grafica, significa generare l'immagine di una scena tridimensionale a partire dalla sua descrizione matematica, interpretata da algoritmi che definiscono il colore di ogni punto dell'immagine digitale.

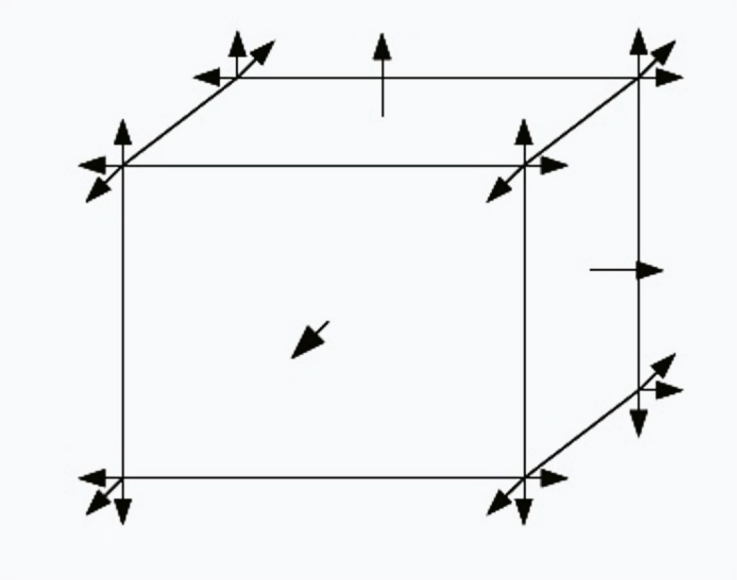
L'insieme delle istruzioni (cioè il programma), utilizzato per potenziare l'effetto fotorealistico ed aumentare nel contempo le performance, prende il nome di shader e riproduce il modello fisico del materiale dell'oggetto a cui è applicato.

Per la realizzazione del rendering viene spesso utilizzato l'algoritmo del Ray Tracing, che è una tecnica generale di geometria ottica che si basa sul calcolo del percorso fatto dalla luce, seguendone i raggi attraverso l'interazione con le superfici.

Ray Tracing di una cucina

Normali

In generale, quando si descrive una superficie geometrica, oltre ai vertici che la compongono si vuole riportare anche il valore della rispettiva normale, la quale consiste in un vettore tridimensionale perpendicolare alla superficie, che ne identifica il verso.



Associazione delle normali ai vertici delle facce

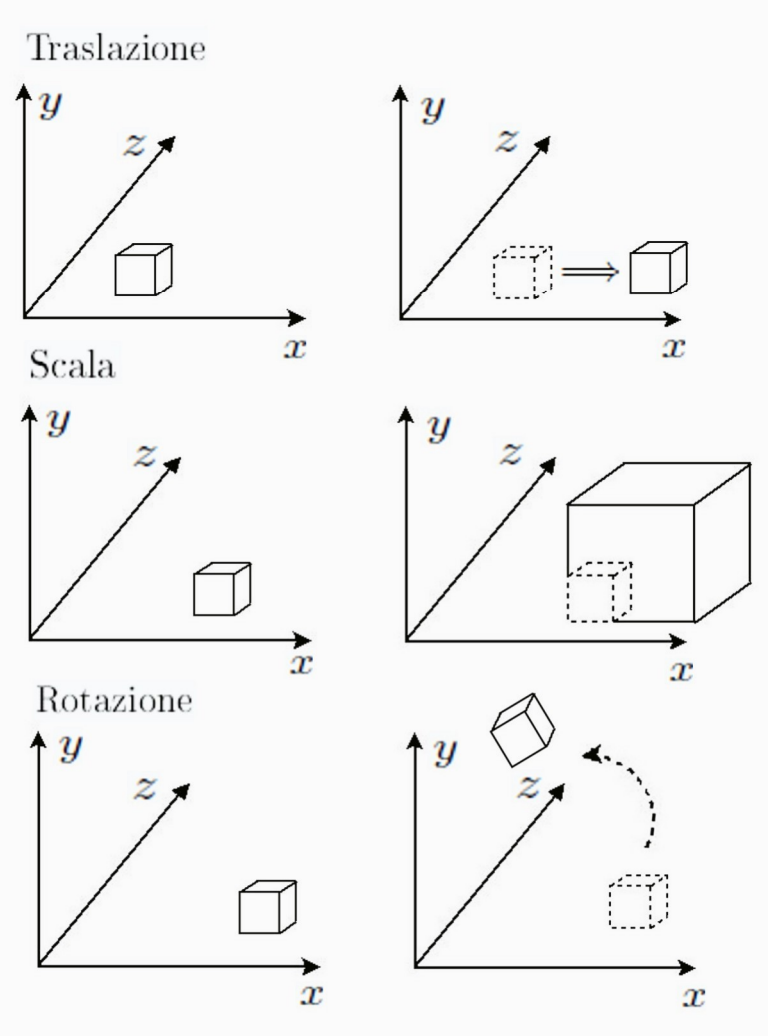
L'uso delle normali nella computer grafica è molto comune per il calcolo dell'illuminazione. Una sorgente luminosa crea un illuminamento della superficie che dipende anche dalla normale della stessa, come accade secondo la legge di Lambert.

Associando una normale ad ogni vertice della superficie, si assegnano versi differenti ad ognuno di essi, producendo un effetto visivo, in grado di far apparire una sagoma curva mentre nella realtà è piana.

Trasformazioni

In uno spazio tridimensionale è possibile applicare ad una faccia, e di conseguenza ai relativi vertici, delle trasformazioni in modo da poterla traslare, ruotare e cambiare di scala; generalmente vengono utilizzate delle matrici 4 x 4 per la loro rappresentazione.

L'immagine seguente mostrerà come vengono modificati i valori in base alla trasformazione.



3.3 Schema

La struttura di un documento istanza di COLLADA ha una semantica e una sintassi ben precisa. Presentare questo formato attraverso lo schema sarebbe una semplice ripetizione delle specifiche presenti nel manuale ufficiale scaricabile dal sito http://www.khronos.org/collada, perciò la trattazione che segue sarà discorsiva con lo scopo di presentare gli elementi e i particolari utilizzati nel progetto.

Librerie e istanziamento degli elementi

Il contenuto di un file creato con COLLADA ha sempre inizio con il tag radice <COLLADA>. Tra i suoi figli, possono essere definite alcune librerie, appositamente ideate per contenere specifiche informazioni come animazioni, geometrie, luci, materiali, effetti e scene. COLLADA permette di gestire ben quindici librerie per altrettante tipologie d'informazione, ma per compatibilità con le funzionalità di Spazio3D solo sette di esse sono state utilizzate.

Per semplificare la manipolazione di strutture complesse, COLLADA prevede un sistema d'identificazione all'interno delle librerie attraverso l'aggiunta dell'attributo id ai rispettivi elementi chiave. Il valore di questo attributo segue uno schema d'indirizzamento URI che ne permette l'istanziamento univoco all'interno del file. Grazie a questo meccanismo, elementi molto pesanti che richiedono un grande dispendio in termini di spazio in memoria, possono comparire più volte all'interno di una o più scene, semplicemente richiamandone la definizione. Così facendo, si può mantenere un'unica copia dell'istanza (secondo un sistema di coordinate di riferimento), mentre nel momento in cui viene inserita nella scena se ne memorizzano solamente le rispettive trasformazioni.

Library_visual_scenes e library_nodes

La library_visual_scenes ha la funzione di raccogliere tutte le scene gestite dall'applicazione, nelle quali vengono inseriti tutti i componenti che non prevedono l'applicazione di proprietà fisiche. Ad ogni scena viene associato un id univoco necessario per poterle istanziare e l'organizzazione dei rispettivi componenti avviene all'interno di elementi node, che fungono da insiemi per il loro raggruppamento logico.

Un node inoltre può includere delle trasformazioni, come rotazioni, scale o traslazioni, le quali vengono applicate, oltre a tutte le istanze, in cascata anche ai possibili altri node della propria gerarchia di figli. Per definire una rotazione viene utilizzato l'elemento rotate, in cui viene specificato l'angolo di rotazione assieme al vettore di riferimento mediante una quaterna di valori float. Le scale e le traslazioni sono invece definite da terne di valori float, i quali sono rispettivamente associati agli assi X, Y e Z. Alternativamente, si possono definere queste trasformazioni o loro combinazioni, attraverso l'elemento matrix che definisce una matrice di trasformazione 4 x 4.

```
<library_visual_scenes>
    <visual_scene id="Scene">
        <node id="Cubo-node"
            <matrix>
                1 0 0 0
                0 1 0 0
                0 0 1 0
                0 0 0 1
            </matrix>
            <translate>0 0 0</translate>
            <rotate>0 0 1 0</rotate>
            <rotate>0 1 0 0</rotate>
            <rotate>1 0 0 0</rotate>
            <scale>1 1 1</scale>
            <instance_geometry url="#Cubo-mesh"/>
        </node>
    </visual_scene>
</library_visual_scenes>
```

Come per le scene, anche gli node hanno un id che ne consente l'istanza. Prevedendo l'uso ripetuto di determinati node, essi possono essere direttamente definiti all'interno della library_node e richiamati quando necessario. Un errore da evitare

nel caricamento di una scena è la possibile creazione di cicli ricorsivi nelle chiamate di questi node, che possono mandare in stallo l'applicazione.

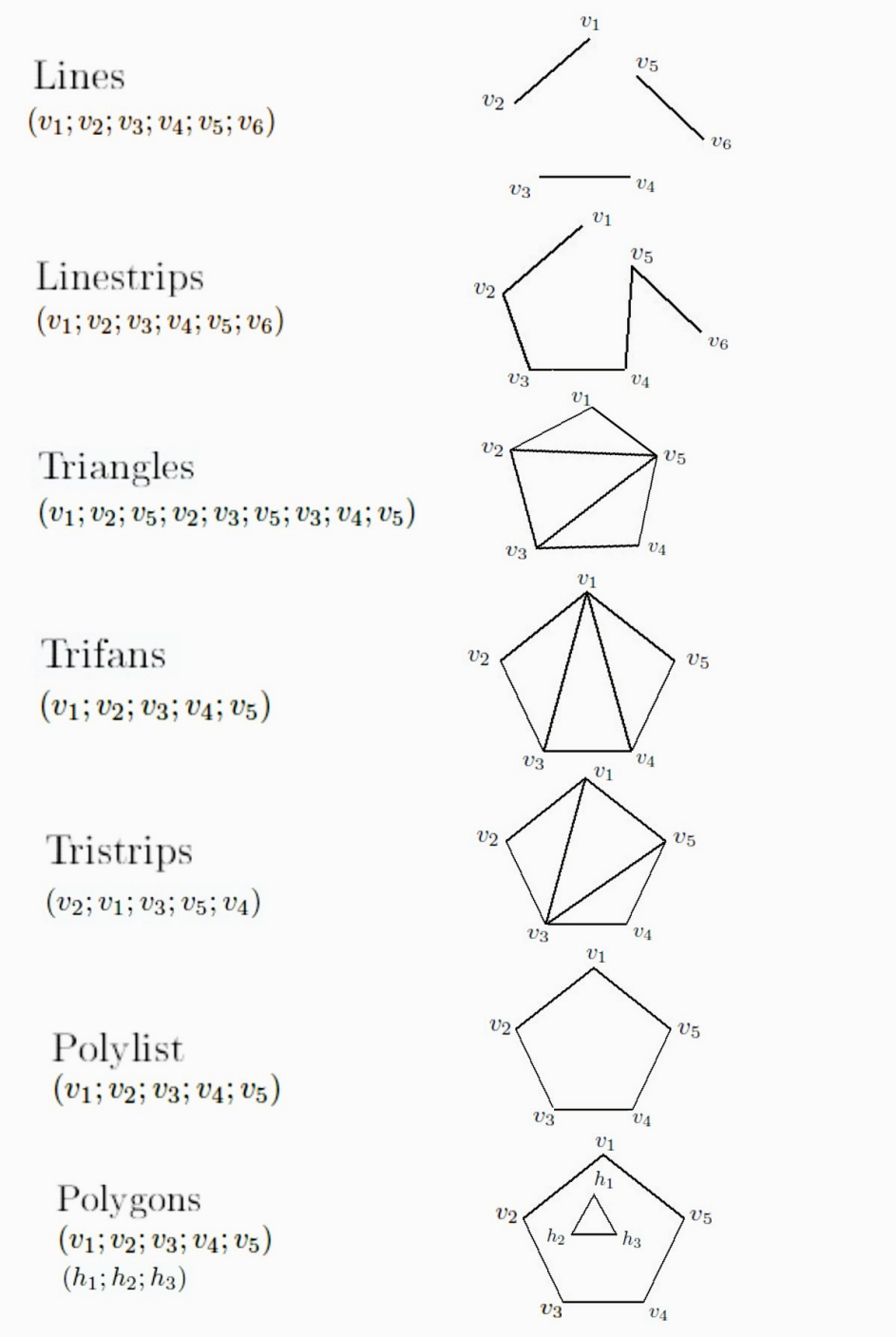
```
<library_nodes>
    <node id="Cubo-node">
        <node>
            <instance_geometry url="#Cubo-mesh">
        </node>
    </node>
</library_nodes>
<library_visual_scenes>
    <visual_scene id="Scene">
        <node>
            <matrix>
                1 0 0 0
                0 1 0 0
                0 0 1 0
                0 0 0 1
            </matrix>
            <instance_geometry url="#Cubo-node"/>
        </node>
    </visual_scene>
</library_visual_scenes>
```

Library_geometries

Questa libreria è forse quella più importante perchè in essa sono contenuti tutti i modelli delle geometrie. Attualmente COLLADA ne gestisce tre tipologie: le mesh, le convex_mesh e le spline.

Per la creazione delle mesh sono definite sette tipologie di primitive:

- LINES: questa primitiva è la più semplice e permette la definizione di singole linee, indicando, per ognuna di esse, il vertice iniziale e quello finale.
- LINESTRIPS: rappresentazione leggermente evoluta rispetto alla singola linea, ne definisce una serie attraverso una sequenza di vertici. Ogni vertice, tranne il primo e l'ultimo, costituisce il punto finale e iniziale tra due diverse linee.
- TRIANGLES: contiene una serie di vertici che saranno presi a tre a tre per formare un insieme di facce di forma triangolare. Il vantaggio nell'utilizzare solo tre vertici per faccia garantisce che essi giacciano tutti sullo stesso piano.
- TRISTRIPS: ha lo stesso funzionamento della primitiva triangles. Dopo la prima terna di vertici, le successive facce vengono create utilizzando l'ultimo esaminato con i due successivi.
- TRIFANS: in questo caso la creazione delle facce triangolari viene generata tenendo fisso il primo vertice e aggiungendo a quello corrente il successivo.
- POLYLIST: primitiva che permette la rappresentazione di un insieme di poligoni con un numero arbitrario di vertici.
- POLYGONS: è la primitiva generalizzata della Polylist. Oltre a permettere la creazione di poligoni con qualsiasi tipo di forma, consente la definizione di fori al suo interno.



Tipologie di primitive e loro raffigurazione

Le tipologie elencate seguono tutte lo stesso schema di definizione, cioè contengono tutte una sequenza di vertici e normali, che saranno poi istanziati attraverso il loro rispettivo indice. In generale, ogni vertice appartiene a più facce e nel caso di trasformazioni che modifichino unicamente la sagoma della figura, ma non la sua

struttura, si dovranno aggiornare solamente i valori dei vertici, lasciando invariata la struttura indicizzata delle facce.

Per chiarire tutti questi concetti si rappresenterà un semplice esempio in cui verrà definita una geometria mediante una primitiva di tipo triangles:

```
<library_geometries>
    <geometry id="ID2">
        <mesh>
            <source id="ID5">
                <float_array id="ID8" count="12">
                    163.13 201.49 0
                    70.61 54.25 0
                    70.61 201.49 0
                    163.13 54.25 0
                </float_array>
                <technique_common>
                    <accessor count="4" source="#ID8" stride="3">
                        <param name="X" type="float"/>
                        <param name="Y" type="float"/>
                        <param name="Z" type="float"/>
                    </accessor>
                </technique_common>
            </source>
            <source id="ID15">
                <float_array id="ID18" count="12">
                    1 0 0
                    0 1 0
                    0 0 1
                    1 0 0
                </float_array>
                <technique_common>
                    <accessor count="4" source="#ID18" stride="3">
                        <param name="X" type="float"/>
                        <param name="Y" type="float"/>
                        <param name="Z" type="float"/>
                    </accessor>
                </technique_common>
            </source>
            <vertices id="ID7">
                <input semantic="POSITION" source="#ID5"/>
                <input semantic="NORMAL" source="#ID15"/>>
            </vertices>
            <triangles count="2" material="Material2">
                <input offset="0" semantic="VERTEX" source="#ID7"/>
                <p>0 1 2 1 0 3</p>
            </triangles>
        </mesh>
    </geometry>
</library_geometries>
```

L'analisi di questo esempio esamina gli elementi source, i quali vengono impiegati per definire degli array di valori float, successivamente interpretati come terne di coordinate XYZ per la rappresentazione di punti sullo spazio e vettori per le normali. L'elemento vertices attraverso la specificazione dell'attributo semantic, richiama una sorgente di punti da utilizzare come posizioni per i vertici, nella creazione delle mesh, un secondo source viene invece utilizzato per le rispettive normali. Il vertice

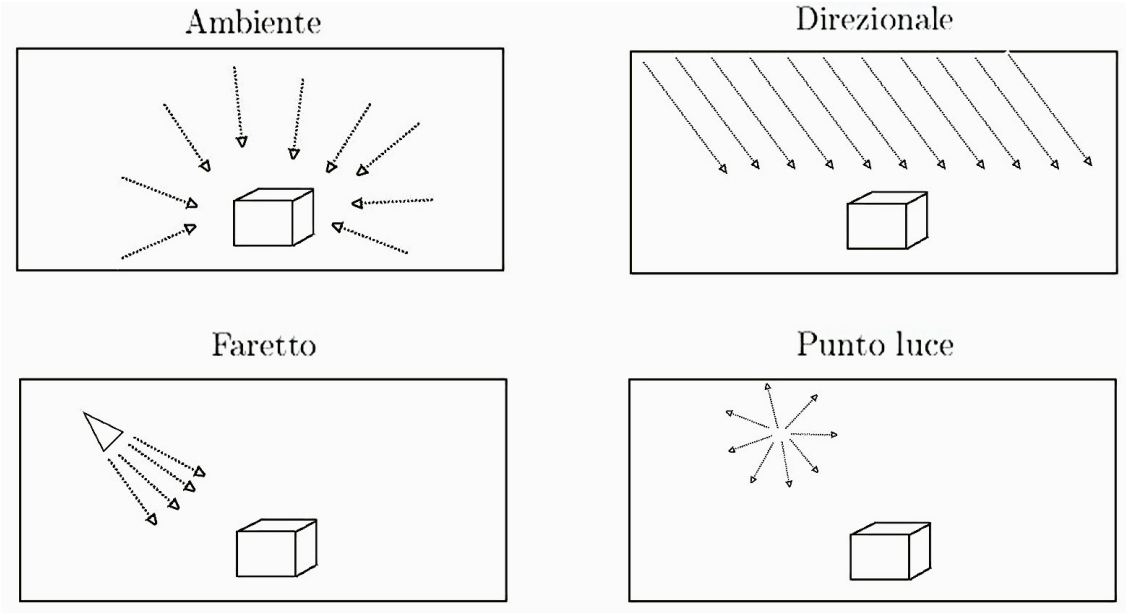
con indice 0 corrisponderà perciò al punto (163.13; 201.49; 0) e la normale sempre d'indice 0 sarà (1; 0; 0). Nel triangles viene infine definita all'interno dell'elemento p la sequenza di primitive che contiene gli indici corrispondenti ai vertici e alle normali. Nel caso specifico, gli indici sono 0 1 2 1 0 3 e vengono utilizzati per creare i due triangoli 0 1 2 e 1 0 3. Il primo triangolo sarà dunque composto dai punti (163.13; 201.49; 0) (70.61; 54.25; 0) (70.61; 201.49; 0), aventi per normali i rispettivi vettori (1; 0; 0) (0; 1; 0) (0; 0; 1).

Le convex_mesh sono definite esattamente come le mesh ma si differenziano da quest'ultime perché definiscono solo geometrie convesse. Alternativamente alla definizione esplicita di una convex_mesh, è possibile richiamare la geometria contenuta in una mesh e ricavarne il relativo guscio convesso.

Per quanto riguarda le spline, esse sono utilizzate per rappresentare l'interpolazione di curve formate da segmenti definiti da funzioni spline come quelle di Bézier. Questa tecnica non è stata implementata durante il progetto per motivi di tempo e perciò non verrà approfondita.

Library_lights

In questa libreria sono memorizzate le informazioni riguardanti le luci all'interno delle scene. Sono possibili quattro diverse fonti luminose: ambient, directional, point e spot.



Raffigurazione delle tipologie di luci gestite da COLLADA

La prima è una semplice luce d'ambiente mentre la seconda rappresenta una luce direzionale fissa secondo il vettore (0; 0; -1), la quale verrà direzionata secondo le trasformazioni applicate al nodo in cui verrà istanziata. Entrambe queste tipologie non sono soggette ad attenuazione.

Il tipo di luce definito da point è quello di un punto luce soggetto a tre diversi parametri attenuazione: costante, lineare e quadratica, calcolati in relazione alla distanza.

Lo spot ha le stesse caratteristiche del point ma rappresentando un faretto può contenere informazioni aggiuntive atte a descriverne la forma, come ad esempio l'angolo d'uscita della luce.

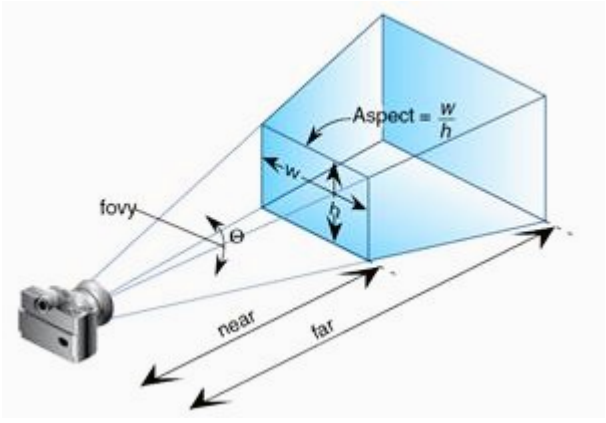
Un esempio di definizione di una luce può essere la seguente:

```
<library_lights>
    <light id="Punto luce">
        <technique_common>
            <point>
                <color>1 0 0</color>
                <constant_attenuation>1</constant_attenuation>
                <linear_attenuation>0</linear_attenuation>
                <quadratic_attenuation>0</quadratic_attenuation>
            </point>
        </technique_common>
    </light>
</library_lights>
```

In questo esempio viene descritto un punto luce di colore rosso, con una costante d'attenuazione pari a uno.

Library_cameras

Le descrizioni delle telecamere sono inserite in questa libreria. Ogni telecamera può avere un ottica di tipo ortogonale o prospettica. Per entrambe è richiesta la distanza, rispetto al punto di vista, del primo e dell'ultimo piano del volume di vista (rispettivamente il near e il far clipping plane), congiuntamente alla dimensione del primo piano espressa come larghezza e altezza oppure, in alternativa una sola tra le due e il valore del loro rapporto.



L'esempio sottostante, riporta la definizione di una telecamera con un'ottica di tipo prospettica avente i rispettivi clipping plane a una distanza di 0.1 e 100 dal punto di vista.

L'unità di misura è specificata all'interno dell'elemento asset (successivamente presentato) e viene utilizzata anche per la dichiarazione obbligatoria della dimensione del near clipping plane, che in questo caso è di larghezza 50 e altezza 25, espressa mediante il loro rapporto.

```
<library_cameras>
    <camera id="Camera-camera" name="Camera">
        <optics>
            <technique_common>
                <perspective>
                    <xfov sid="xfov">50</xfov>
                    <aspect_ratio>2</aspect_ratio>
                    <znear sid="znear">0.1</znear>
                    <zfar sid="zfar">100</zfar>
                </perspective>
            </technique_common>
        </optics>
    </camera>
</library_cameras>
```

Library_materials e library_effects

La library_materials è la libreria dedicata alla definizione dei materiali, i quali rappresentano l'aspetto visivo degli oggetti geometrici. All'interno di ogni materiale viene istanziato un effetto contenuto nella library_effects, il quale conterrà i parametri necessari per il rendering.

Un effetto può seguire diversi profili in modo da incapsulare tutti i valori delle specifiche impiegate da una determinata piattaforma shader. COLLADA supporta tre profili specifici per OpenGL ES, OpenGL Shading Language e il linguaggio Cg; esiste inoltre un profilo comune nel quale sono riportati i valori per le funzioni assicurate da un qualsiasi shader generico; essi sono: constant, lambert, phong e blinn. Queste funzioni si differenziano per l'algoritmo utilizzato per il calcolo dell'effetto, per il quale vengono considerati parametri come trasparenza, luce emessa, indice di rifrazione, etc.

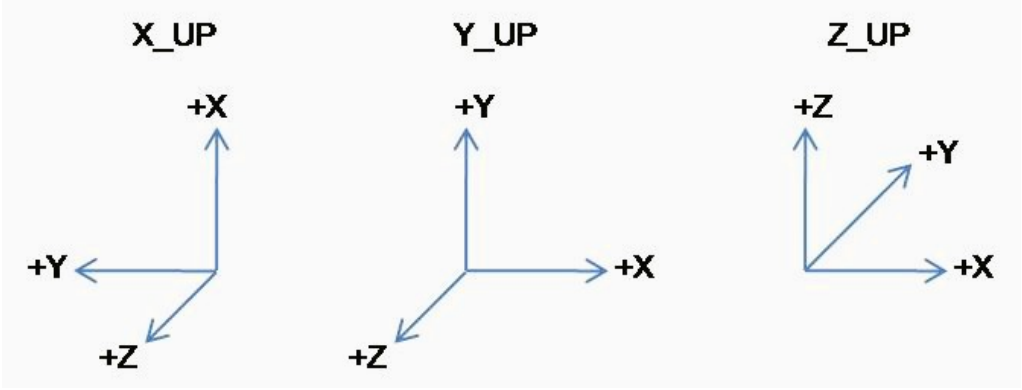
L'esempio seguente mostrerà la definizione di un materiale e il relativo effetto:

```
<library_materials>
    <material id="ID4" name="material">
        <instance_effect url="#ID3"/>
    </material>
</library_materials>
<library_effects>
    <effect id="ID3">
        <profile_COMMON>
            <technique sid="COMMON">
                <lambert>
                    <emission>
                        <color>1 1 1 1</color>
                    </emission>
                </lambert>
            </technique>
        </profile_COMMON>
    </effect>
</library_effects>
```

La semplice ed intuitiva struttura dell'esempio consente di notare immediatamente come all'interno del materiale sia istanziato l'effetto, il quale definisce un profilo comune che utilizza una tecnica Lambert avente come parametro il colore RGBA (1; 1; 1; 1) corrispondente alla luce emessa dalla superficie.

Asset

Questo elemento è presente come possibile figlio per tantissimi elementi e ha la funzione di contenere delle informazioni di contorno come la data di creazione o l'autore dell'elemento, ma anche dati estremamente importanti come l'unità di misura e il sistema di coordinate da utilizzare.



Possibili coordinate di sistema

```
<asset>
    <contributor>
        <author>Alessio Michele</author>
    </contributor>
    <created>2013-01-23T12:00:00</created>
    <unit name="meter" meter="1"/>
    <up_axis>Z_UP</up_axis>
</asset>
```

In questo esempio si esplicita che l'elemento, in cui sarà contenuto l'asset, è stato creato il 23-01-2013 alle ore 12:00 da Alessio Michele, utilizza il metro come unità di misura e imposta come coordinate di sistema la configurazione Z_UP.

Extra

Nel caso in cui alcune applicazioni non riescano a memorizzare determinate informazioni all'interno degli elementi previsti dallo schema, COLLADA ne permette l'estensione attraverso l'elemento extra, nel quale viene specificata una determinata tecnica rappresentativa, in base ad un proprio profilo che può contenere qualsiasi dato o elemento compatibile con XML.

Se per esempio un'applicazione volesse memorizzare il nome del pc con il quale è stato creato il file COLLADA, lo può fare nel seguente modo:

```
<COLLADA>
    …
    <extra>
        <technique profile="MioCAD">
            <pc>Alessio</pc>
        </technique>
    </extra>
    …
</COLLADA>
```

4 Realizzazione del progetto

Questo capitolo è dedicato alla descrizione della realizzazione del progetto.

Per comprendere ciò che è stato fatto, verrà inizialmente presentato l'approccio iniziale costituito per lo più dallo studio del contesto lavorativo e delle specifiche dello standard da implementare. Successivamente saranno esposti gli aspetti più significativi di tutte le librerie utilizzate nella computazione dei dati.

Una volta fatto il quadro generale, sarà fatta una panoramica del lavoro svolto, distinguendo le due fasi fondamentali costituite dalla realizzazione dell'import e dell'export.

Alla fine del capitolo, verranno presentati, oltre ai risultati ottenuti nei test fatti e comparati con quelli condotti sugli altri CAD, anche alcuni spunti per gli sviluppi futuri.

4.1 Approccio iniziale

Prima di poter iniziare il lavoro di stesura del codice vero e proprio, è stato necessario svolgere un'ampio studio di tutte le componenti fondamentali per la realizzazione del progetto.

Il primo aspetto affrontato è stato quello della computer grafica, per l'acquisizione della terminologia specifica e la comprensione delle tecniche utilizzate per le rappresentazioni grafiche in spazi tridimensionali. In questa fase è stato fondamentale l'apporto dato dai programmatori dell'azienda, che con la loro esperienza sono riusciti a presentare tali concetti, contestualizzandoli all'interno delle strutture utilizzate da Spazio3D.

Il passo successivo è stato quello di definire le esigenze primarie dell'azienda in relazione al progetto. Assieme al coordinatore del team d'ingegneri è stata stilata la seguente lista di linee guida:

- implementare prima l'importazione e poi l'esportazione;
- realizzare le funzioni per la rappresentazione delle geometrie previste da COLLADA con le strutture di Spazio3D;

- importare all'interno della scena le luci e le telecamere, in modo quanto più prossimo rispetto a quello previsto dallo standard;
- inserire nell'esportazione un metodo che renda possibile l'identificazione dei file creati con Spazio3D;
- le texture possono essere realizzate alla fine del progetto, perché non rientrano nelle priorità primarie;
- implementare il codice in modo da perdere il minor numero di informazioni possibili;

A questo punto, il lavoro di studio si è focalizzato sullo standard COLLADA. L'approccio iniziale, ha riguardato l'osservazione diretta degli elementi nei file esportati da parte di altri CAD, sfruttando la natura testuale propria delle applicazioni XML e la possibilità di visualizzarne la struttura già formattata all'interno di un qualsiasi browser. In particolar modo, sono state esaminate le differenze strutturali nella descrizione delle forme geometriche derivate da minimi perturbamenti delle entità all'interno della stessa scena. Infine, dopo aver compreso i concetti base di questo formato, sono state approfondite le specifiche descritte nel manuale ufficiale e studiate le possibili soluzioni per la loro rappresentazione in Spazio3D, con l'apporto delle funzionalità già presenti.

4.2 Librerie di supporto

Durante la realizzazione del progetto sono stati utilizzati dei costrutti già inseriti nel programma. Alcuni di questi appartengono alla libreria STL propria del C++ mentre altri sono stati implementati dai programmatori della BrainSoftware.

STL e contenitori

La Standard Template Library è una libreria standard del linguaggio C++, nella quale sono definiti alcuni algoritmi generici, iteratori e strutture dati, di uso comune.

Di questa libreria sono stati utilizzati due tipi di contenitori: i vector e i map. I primi sono un'evoluzione dei normali array, resi dinamici in modo tale da potersi ridimensionare automaticamente ogni qualvolta se ne presenti il bisogno. Il tempo richiesto per l'inserimento e la cancellazione di un elemento posizionato in coda è O(1), mentre se sono situati in testa oppure si vuole effettuare una ricerca, il tempo richiesto cresce linearmente diventando O(n). Il loro utilizzo si è reso necessario per gestire efficacemente le sequenze di vertici e indici. I contenitori map invece, sono degli array dinamici ordinati di tipo associativo rispetto a delle chiavi. Ogni elemento è costituito da una coppia (chiave, valore), il cui tipo è definito dall'utente e non sono consentiti elementi con la stessa chiave. Il tempo richiesto per l'accesso ai dati è O(log n).

Parser

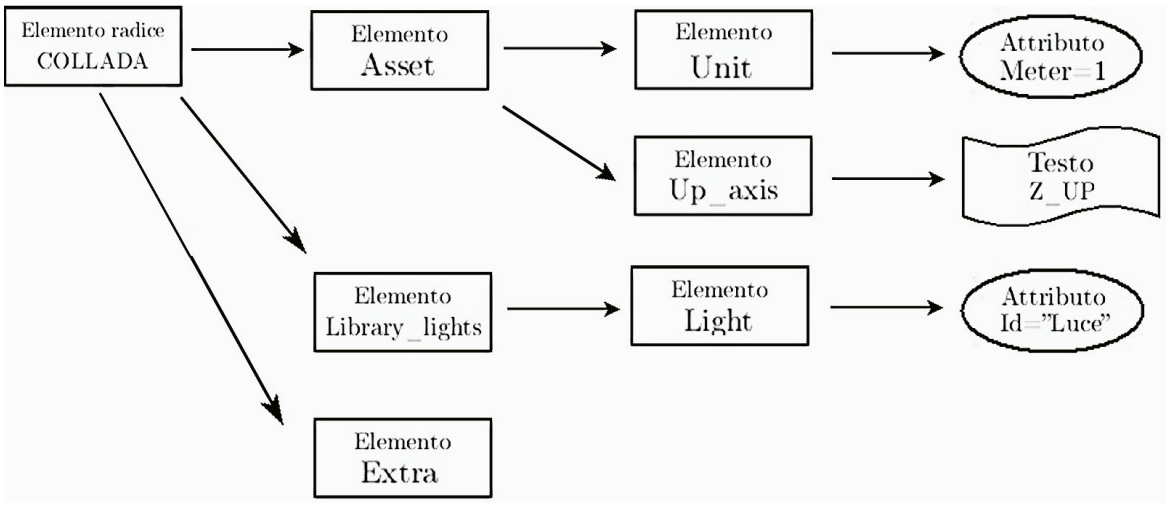
Come detto, COLLADA è uno standard per la rappresentazione di qualità grafiche digitali attraverso la definizione della struttura di un file testuale secondo uno schema XML. Generalmente, per leggere un documento istanza di XML viene utilizzato un apposito programma detto parser, in grado di determinare una struttura grammaticale a partire da un'analisi sintattica. Sfruttando le proprietà di COLLADA è stato

possibile utilizzare lo stesso parser XML già implementato all'interno di una classe di Spazio3D.

Questa classe realizza questo tipo di parser ma ne estende le funzionalità attraverso metodi in grado di leggere e scrivere file XML, appoggiandosi ad una struttura di tipo gerarchico. Così facendo, la gestione dei documenti COLLADA è stata notevolmente agevolata sia per l'importazione che per l'esportazione.

Nell'esempio verrà mostrata la struttura creata dal parsing del seguente codice:

```
<COLLADA>
    <asset>
        ...
        <unit meter="1"/>
        <up_axis>Z_UP</up_axis>
    </asset>
    <library_lights>
        <light id="Luce">
        ...
        </light>
    </library_light>
    ...
    <extra>
        ...
    </extra>
</COLLADA>
```



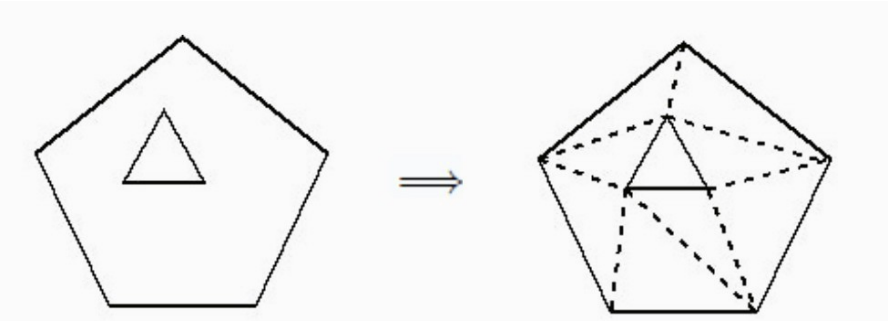
Rappresentazione della struttura creata dal parsing del codice sopra descritto

Covering

Spazio3D gestisce facce da tre o quattro vertici, perciò, nel caso in cui ci sia un poligono composto da cinque o più vertici, viene utilizzata un'altra classe già implementata capace d'eseguirne il covering, cioè la copertura della superficie attraverso un insieme di facce triangolari.

Questa classe utilizza algoritmi complessi che richiedono la risoluzione di molti calcoli, ma danno la possibilità di gestire la copertura di facce anche se al loro interno sono presenti dei fori.

Una particolarità di questa classe consiste nel lavorare con vertici che giacciono in un sistema di coordinate bidimensionali x e y. Per impiegare correttamente questi metodi con dei punti che si estendono in uno spazio tridimensionale, è quindi necessario eseguire su di essi un lavoro d'adattamento.



Copertura della mesh con un foro al suo interno

Blocchi

In Spazio3D esiste la possibilità di raggruppare delle entità all'interno di blocchi che fungono da contenitori. A questi blocchi possono essere assegnate delle trasformazioni, che verranno applicate in cascata agli oggetti in esso contenuti.

La possibilità di inserire un blocco all'interno di un altro blocco verrà utilizzata per realizzare una struttura gerarchica delle entità, esattamente come avviene in COLLADA per gli elementi node all'interno della descrizione di una scena.

4.3 Implementazione dello standard

La seguente analisi approfondirà le due fasi principali del progetto realizzato: l'import e l'export.

4.3.1 Import

Per leggere le informazioni contenute in un file COLLADA è stata utilizzata la classe precedentemente presentata, che realizza il parser XML. L'interpretazione e l'implementazione delle informazioni contenute nella struttura creata, ha seguito l'ordine prestabilito durante l'approccio iniziale del progetto.

La difficoltà realizzativa dell'import è stata maggiore rispetto all'export, perché in questa fase è stata richiesta la gestione, per ogni specifica, di tutte le sue possibili rappresentazioni.

Geometrie

Il primo aspetto considerato è stato quello delle geometrie mesh, che come già spiegato, in COLLADA sono descritte mediante sette tipologie di primitive: Line, Linestrips, Polygons, Polylist, Triangles, Trifans e Tristrips.

All'interno di ogni geometria sono presenti degli array che contengono tutti i punti che la descrivono, perciò la prima operazione svolta è la loro memorizzazione all'interno di un contenitore map avente come coppia chiave-valore l'id univoco dell'array e il puntatore al vector dei vertici (ognuno composto da tre valori float corrispondenti alle coordinate X, Y e Z). Questa fase è necessaria per poter gestire l'eventuale istanziazione di un array contenuto all'interno di un'altra geometria.

Spazio3D supporta degli oggetti per la rappresentazione di linee tridimensionali, gestibili semplicemente specificando la sequenza dei vertici che le compongono, esattamente allo stesso modo previsto per le Linestrips.

Questa caratteristica ha permesso l'immediata traduzione delle Linestrips, semplicemente riportando la sequenza degli indici che le compongono. Per le Line invece, si è deciso di generare una nuova linea solamente quando l'indice finale di una non corrisponde a quello iniziale della successiva. Ad esempio, se la sequenza degli indici è 0 1 2 3 3 4 4 5, non vengono create le quattro linee (0; 1) (2; 3) (3; 4) e (4; 5) ma solo due (0; 1) e (2; 3; 4; 5).

Le rimanenti cinque primitive costituiscono rappresentazioni diverse per la definizione d'insiemi di facce. Tra queste, le più semplici sono certamente le Triangles, Trifans e Tristrips perchè descrivono facce triangolari. All'interno di Spazio3D possono essere gestite facce da tre o quattro vertici, perciò è possibile riprodurre esattamente le facce gestendo correttamente la sequenza degli indici secondo la primitiva specificata.

Polygons e Polylist sono le primitive che hanno presentano le maggiori difficoltà perchè descrivono facce con un numero arbitrario di vertici e nel primo caso possono contenere anche dei fori. Nel caso in cui una faccia sia costituita da meno di cinque vertici e non abbia fori al suo interno, viene facilmente ricreata senza dover compiere alcuna operazione d'attamento. Se invece la faccia non rientra in questa casistica, vengono utilizzati i metodi della classe Covering per la sua copertura. Purtroppo, gli algoritmi implementati all'interno di questa classe lavorano solo sulle coordinate x e y dei punti; perciò prima d'utilizzarli dovranno essere portati tutti in pianta, aumentando ulteriormente il tempo già oneroso, necessario alla loro computazione. Questa operazione avviene mediante una matrice di trasformazione calcolata rispetto al piano generato da tre punti presi a caso (ovviamente tra quelli che compongono la faccia stessa). Una volta calcolata la copertura, tutte le facce calcolate vengono riportate nella corretta posizione applicando, ad ognuna di esse, la matrice di trasformazione inversa rispetto a quella precedentemente utilizzata. Dopo aver ricavato tutte le facce della geometria, viene creato un oggetto mesh per racchiuderle assieme ai rispettivi vertici.

Ogni linea e ogni oggetto mesh viene inserito all'interno di un contenitore di tipo blocco. In questo modo si concretizza la possibilità di avere una geometria composta da un numero arbitrario di primitive e di poterla memorizzare all'interno di un map

attraverso l'inserimento di una sola coppia formata dall'id della geometria e da un puntatore al blocco.

Verrà ora proposto uno pseudo-codice che a grandi linee racchiude l'insieme di operazioni appena descritte:

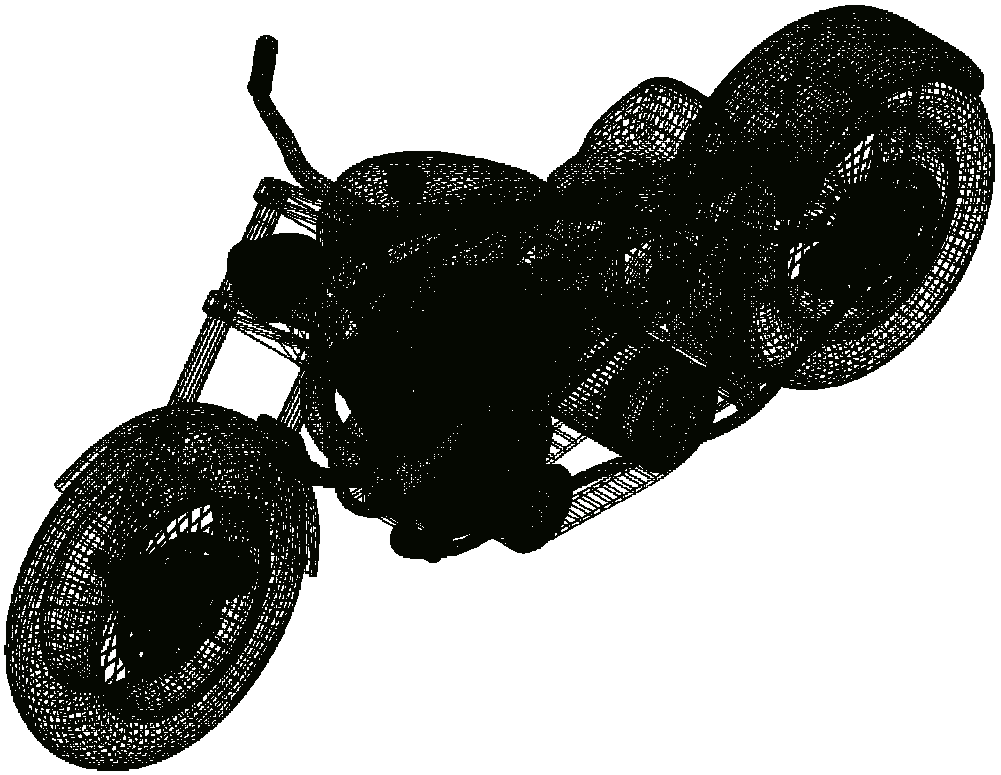
```
Individua la Library_geometry
Per ogni geometria
{
    Per ogni array
    {
        Carica l'array di valori float;
        Inserisci nel vector i punti tridimensionali creati raccogliendo
        a tre a tre i valori dell'array;
        Crea la coppia id dell'array e puntatore al vector;
        Inserisci la coppia nel contenitore map per i vertici;
    }
}
Per ogni geometria
{
    Carica la sequenza degli indici;
    Richiama il vettore di vertici specificato;
    Se la primitiva è Line
    {
        Crea la prima linea e inserisci il primo indice;
        Scorri la sequenza degli indici
        {
            Se l'indice finale di una linea corrisponde all'inizio della
            successiva
                Accoda l'indice finale alla sequenza descrittiva della linea;
            Altrimenti
                Termina la linea con l'indice finale e crea una nuova
                linea con il corrispondente indice d'inizio;
        }
        Inserisci le linee nel blocco;
    }
    Se la primitiva è Linestrips
    {
        Crea una linea e assegnale l'intera sequenza degli indici;
        Inserisci la linea nel blocco;
    }
    Se la primitiva è Triangles
    {
        Per ogni gruppo di tre indici
            Crea una faccia e assegna gli indici facendo corrispondere il
            terzo vertice con il quarto;
        Crea un oggetto mesh e inserisci l'insieme di facce e vertici;
        Inserisci l'oggetto mesh nel blocco;
    }
    Se la primitiva è Trifans
    {
        Scorri la sequenza degli indici a partire dal terzo
            Crea una faccia con: il primo indice, l'indice corrente e quello
            precedente, facendo corrispondere il terzo vertice con il quarto;
        Crea un oggetto mesh e inserisci l'insieme di facce e vertici;
        Inserisci l'oggetto mesh nel blocco;

    }
    Se la primitiva è Tristrips
    {
        Scorri la sequenza degli indici a partire dal terzo
            Crea una faccia con l'indice corrente e i due precedenti,
            facendo corrispondere il terzo vertice con il quarto;
        Crea un oggetto mesh e inserisci l'insieme di facce e vertici;
```

```
        Inserisci l'oggetto mesh nel blocco;

    }
    Se la primitiva è Polylist
    {
        Scorri la sequenza degli indici
            Se la faccia è composta da tre o quattro vertici
                Crea la faccia con i relativi indici, facendo corrispondere
                se necessario il terzo vertice con il quarto;
            Altrimenti
            {
                Ricava la matrice di trasformazione per portare in pianta i
                punti e la relativa matrice inversa;
                Applica la matrice ai punti;
                Utilizza i metodi della classe Covering per ottenere le facce
                corrette;
                Applica la matrice inversa ai punti;
            }
        Crea un oggetto mesh e inserisci l'insieme di facce e vertici;
        Inserisci l'oggetto mesh nel blocco;
    }
    Se la primitiva è Polygons
    {
        Scorri la sequenza degli indici
            Se la faccia è composta da tre o quattro vertici e non ha fori
                Crea la faccia con i relativi indici, facendo corrispondere
                se necessario il terzo vertice con il quarto;
            Altrimenti
            {
                Ricava la matrice di trasformazione per portare in pianta i
                punti e la relativa matrice inversa;
                Applica la matrice sia ai punti del perimetro esterno sia ai
                punti dei fori;
                Utilizza i metodi della classe Covering per ottenere le facce
                corrette;
                Applica la matrice inversa sia ai punti del perimetro esterno
                sia ai punti dei fori;
            }
        Crea un oggetto mesh e inserisci l'insieme di facce e vertici;
        Inserisci l'oggetto mesh nel blocco;
    }
}
Crea la coppia id della geometria e puntatore al blocco;
Inserisci la coppia nel contenitore map per le geometrie;
```

Ovviamente, il codice reale è molto più complesso ed articolato, avendo un gran numero di controlli per la correttezza dei dati e la suddivisione delle operazioni in funzioni. Se presenti, vengono inoltre inseriti i valori delle normali ai relativi vertici, tranne nel caso in cui le facce siano generate dalla classe Covering, in quanto in alcuni casi si potrebbe perdere la corrispondenza con i punti della faccia originale.



Importazione delle geometrie di una moto

L'importazione delle spline e delle convex mesh non è stata realizzata. Nel primo caso basterà adattarle alle spline proprie di Spazio3D mentre nel secondo caso dovrà essere implementato un algoritmo per il calcolo del guscio convesso dell'insieme dei vertici di una o più geometrie richiamate.

Materiali

Ad ogni mesh può essere associato un materiale, il quale sarà costituito da un particolare effetto creato dalla combinazione di colori e texture associate alle proprietà fisiche della luce. In COLLADA i colori sono rappresentati da quattro valori float che rappresentano i valori normalizzati dei rispettivi colori RGBA. Spazio3D gestisce i colori attraverso una palette di 256 elementi di cui all'incirca metà sono già preassegnati. Avendone perciò a disposizione solo un numero ristretto, non viene occupato un posto libero nel caso il colore richiesto non figuri tra quelli presenti ma invece viene prima eseguita una ricerca per vicinanza alla tonalità. Solo nel caso in cui la distanza sia troppo grande viene inserito un nuovo colore. Per il calcolo della distanza è stata sfruttata la possibilità di associare i valori RGB a dei punti con coordinate XYZ; ne consegue che, la distanza tra due colori diventa la distanza euclidea dei corrispondenti punti, mentre la ricerca può essere paragonata al trovare il punto più vicino a quello dato in un intorno sferico di raggio pari alla massima tolleranza desiderata.

Verrà ora proposto uno pseudo-codice che descrive le operazioni necessarie per l'assegnazione di un colore:

```
Crea un punto con i valori XYZ pari a quelli RGB;
Inizializza la distanza minima trovata con un valore molto grande;
Per ogni elemento della palette
{
    Se l'elemento è vuoto
        Memorizza l'indice;
    Altrimenti
        {
        Crea un punto con i valori XYZ pari a quelli RGB dell'elemento;
        Calcola la distanza tra i punti;
        Se la distanza è inferiore a quella minima precedentemente trovata
        memorizza l'indice dell'elemento;
}
Se la distanza è inferiore ad una certa tolleranza data
    Utilizza l'indice dell'elemento con distanza minima;
Altrimenti
{
    Se c'è un elemento vuoto
        Assegna all'elemento i valori RGB del colore richiesto;
    Altrimenti
        Lancia eccezione;
}
```

Luci

Per importare le luci non è stato necessario compiere alcun adattamento in quanto trovavano una corrispondenza esatta rispetto a quelle presenti in Spazio3D, con l'unica eccezione che riguarda la luce direzionale, perchè non prevista.

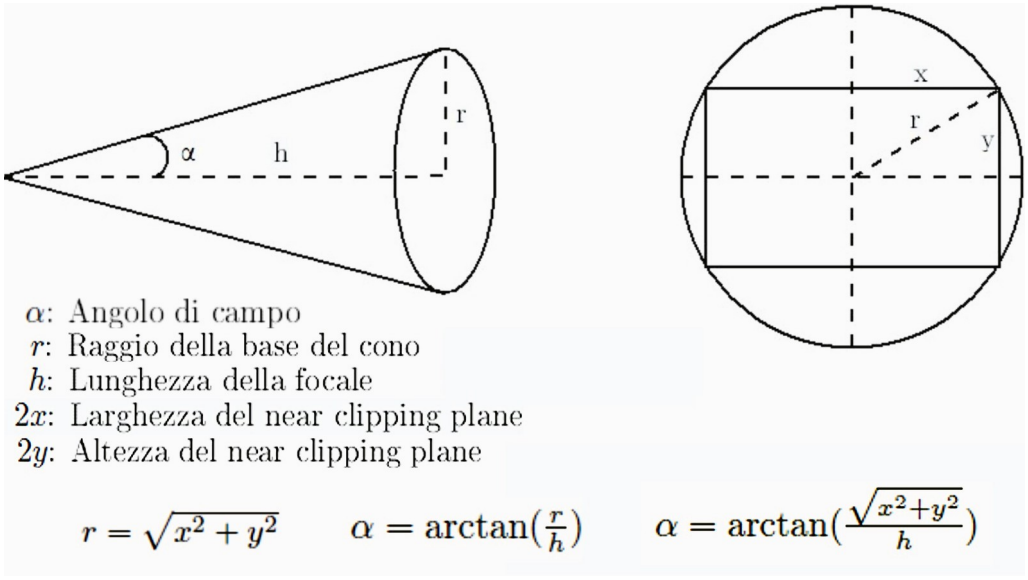
Per la sua rappresentazione si è scelto di approssimarne l'effetto utilizzando un punto luce con le stesse proprietà e di posizionarla lungo la direzione prevista ad una distanza molto grande rispetto agli altri componenti della scena. In questo modo i raggi che colpiscono le superfici delle entità risultano essere quasi paralleli, come se fossero appunto originati da una luce direzionale. Come una luce d'ambiente anche quella direzionale non ha attenuazione, perciò ai coefficienti costanti, lineari e quadratici, vengono dati valori molto prossimi allo zero.

Telecamere

Per quanto riguarda le telecamere, COLLADA ne gestisce due tipologie: ortogonali e prospettiche, che come visto catturano proiezioni di spazio differenti e vengono posizionate all'interno della scena modificando l'orientamento comune prefissato attraverso l'applicazione delle matrici di trasformazione proprie del node in cui vengono istanziate.

Spazio3D invece, ne mette a disposizione una con un volume di vista rappresentabile mediante un cono, avente l'apice posizionato nel punto d'osservazione e la lunghezza della focale infinita.

La soluzione scelta per adattare queste diverse tecniche, prevede la circoscrizione del piano rettangolare del volume di vista più vicino al punto d'osservazione (presente sia nella proiezione ortogonale sia in quella prospettica) nella circonferenza corrispondente alla sezione del cono passante per il piano stesso. Dal raggio della circonferenza così ottenuta, si risale al valore dell'angolo di campo. La conoscenza di questo valore è resa necessaria dal fatto che Spazio3D lo utilizza come parametro fondamentale per la definizione della telecamera stessa.



Formule per il calcolo dell'angolo di campo

Scena

Come visto, COLLADA permette la definizione di più scene all'interno dello stesso file e di poterne istanziare una, quella effettivamente visualizzata. Spazio3D può gestire una sola scena e perciò viene importata solamente quella istanziata.

Prima di esaminare il contenuto della scena, viene eseguita una scansione di tutti i node contenuti nella libreria delle scene e nella libreria dei node, memorizzando in un map la coppia id del node ed un puntatore ad esso. Questa scansione servirà nel caso in cui all'interno della scena sia presente l'istanziazione di un node, rendendo perciò il suo ritrovamento quasi immediato. Ogni node, rappresentando un raggruppamento, viene associato ad un blocco e tutte le trasformazioni ed istanziazioni vengono applicate al blocco stesso.

La lettura della scena avviene attraverso l'ispezione del relativo albero dei node. Si è optato per la creazione di due alternative: mantenere inalterata la struttura o semplificare gli elementi node in caso in cui abbiano un solo figlio o siano figli unici. La prima opzione prevede la costruzione di un blocco in corrispondenza di ogni node e l'inserimento nello stesso delle istanziazioni presenti. Nel secondo caso invece, il contenuto del node dovrà essere inserito nel padre e le trasformazioni che conteneva, dovranno essere applicate ai figli.

Si possono trovare file con gerarchie di node che non contengono alcun tipo d'istanziazione, perciò vuoti e quindi non devono essere rappresentati in alcun caso. Per questo motivo prima di esaminare il contenuto di un node viene sempre eseguito un controllo.

Il contenuto di una scena viene istanziato solo all'interno di un node, perciò non è possibile determinare se i node al livello più alto siano o no dei veri gruppi. Essendo possibile specificare l'autore del file, si può capire se esso sia il risultato di un'esportazione diretta da Spazio3D. In tal caso, vengono eseguite delle semplificazioni iniziali in modo tale da mantenere inalterata l'intera struttura.

COLLADA prevede quattro elementi per descrivere le varie trasformazioni: rotate, scale, translate e matrix. Si è scelto di riportare tutte queste trasformazioni sotto forma di matrice e di moltiplicarle tra loro, in modo tale da avere una sola matrice da applicare ai figli del node. Per la non commutatività della moltiplicazione delle matrici, l'ordine delle moltiplicazioni stesse, dovrà essere inversa rispetto all'ordine in cui appaiono.

Rotazione

$$(x;y;z;\alpha) \Longrightarrow \begin{bmatrix} x^2+(1-x^2)\cos\alpha & (1-\cos\alpha)xy-(\sin\alpha)z & (1-\cos\alpha)xz+(\sin\alpha)y & 0 \\ (1-\cos\alpha)xy+(\sin\alpha)z & y^2+(1+y^2)\cos\alpha & (1-\cos\alpha)yz-(\sin\alpha)x & 0 \\ (1-\cos\alpha)xz-(\sin\alpha)y & (1-\cos\alpha)yz+(\sin\alpha)x & z^2+(1-z^2)\cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Traslazione

$$(v_x;v_y,v_z) \Longrightarrow \begin{bmatrix} 1 & 0 & 0 & v_x \\ 0 & 1 & 0 & v_y \\ 0 & 0 & 1 & v_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scala

$$(s_x;s_y;s_z) \Longrightarrow \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Conversione delle trasformazioni COLLADA in matrici

Asset

Ogni elemento asset può contenere come informazioni l'unità di misura e il sistema di coordinate da utilizzare. Per poter utilizzare queste informazioni, esse vengono convertite in matrici. L'unità di misura viene associata ad una matrice di cambio scala, il cui valore è rapportato con quello di riferimento di Spazio3D. Le coordinate di sistema vengono tradotte in matrici di rotazione in modo tale da riportare la rappresentazione degli elementi in un sistema d'assi comune.

4.3.2 Export

La realizzazione dell'export si è rivelata molto semplice e veloce, sia per la facilità nella composizione della struttura (grazie ai metodi contenuti nella classe adibita al parsing), sia per la facilità dell'adattamento verso lo schema COLLADA.

Geometrie

Le esportazioni delle geometrie sono state quasi immediate perché Spazio3D permette di riportare ogni entità ad una rappresentazione comune fatta di sole facce e vertici.

Ogni faccia, composta da tre o quattro vertici e senza fori, viene rappresentata mediante una Polylist, semplicemente specificando il numero di vertici che la compongono.

Per quanto riguarda le linee, vengono tutte descritte dalla primitiva Linestrips perché, come precedentemente visto, hanno la stessa struttura.

L'esempio seguente presenta una semplice esportazione di un cubo formato da sei facce ognuna di quattro vertici.

```
<library_geometries>
    <geometry id="Cubo-mesh">
        <mesh>
            <source id="Cubo-mesh-posizioni">
                <float_array id="Cubo-mesh-posizioni-array" count="24">
                    1 1 -1
                    1 -1 -1
                    -1 -1 -1
                    -1 1 -1
                    1 1 1
                    1 -1 1
                    -1 -1 1
                    -1 1 1
                </float_array>
                <technique_common>
                    <accessor source="#Cubo-mesh-posizioni-array" count="8"
                    stride="3">
                        <param name="X" type="float"/>
                        <param name="Y" type="float"/>
                        <param name="Z" type="float"/>
                    </accessor>
                </technique_common>
            </source>
            <vertices id="Cubo-mesh-vertici">
                <input semantic="POSITION" source="#Cubo-mesh-posizioni"/>
            </vertices>
            <polylist count="6">
                <input semantic="VERTEX" source="#Cubo-mesh-vertici"
                offset="0"/>
                <vcount>4 4 4 4 4 4 </vcount>
                <p>
                    0 1 2 3
                    4 7 6 5
                    0 4 5 1
                    1 5 6 2
                    2 6 7 3
                    4 0 3 7
                </p>
            </polylist>
        </mesh>
    </geometry>
</library_geometries>
```

Luci

Ogni luce di Spazio3D trova un'esatta corrispondenza con quelle di COLLADA. La loro esportazione si riduce perciò nel riportare semplicemente i valori dei relativi parametri descrittivi.

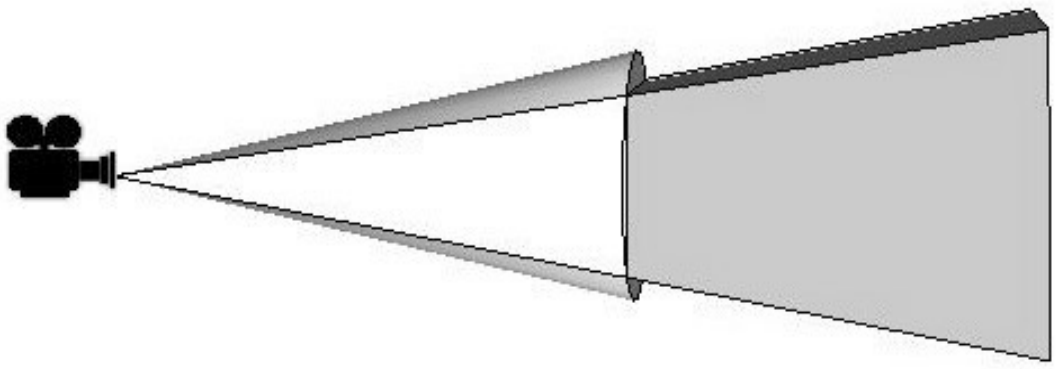
L'unica considerazione degna di nota riguarda l'esportazione di una luce precedente creata dall'importazione di una luce direzionale. Non potendo distinguere questo caso da quello in cui essa sia effettivamente una luce posizionata in un punto molto lontano della scena, viene riportata come un normale punto luce.

Telecamere

L'esportazione delle telecamere ha richiesto un procedimento inverso rispetto all'importazione. La proiezione di Spazio3D viene riportata al caso prospettico perché è quello che più l'approssima.

Il calcolo dei parametri necessari avviene inscrivendo nella sezione circolare del cono il near clipping plane, posizionato ad una distanza adeguata, calcolata in relazione all'unità di misura utilizzata.

Nell'immagine sottostante si nota come il tronco di piramide della proiezione prospettica approssimi meglio quella del cono:



Scena

L'esportazione della scena, avviene all'interno di un node padre, il quale poi sarà interpretato come inizio della descrizione. Per ogni blocco presente nella scena, viene creato un elemento node, in modo da preservare il raggruppamento logico delle entità.

Ogni entità, che sia una geometria, una luce oppure una telecamera, per essere interpretata correttamente, deve poter essere associata alla proprie trasformazioni. Per questo motivo, queste entità vengono istanziate all'interno di un proprio node, nel quale viene inserito l'elemento matrix che conterrà la relativa matrice di trasformazione.

Di seguito è presentata l'esportazione di una scena in cui appare la geometria mesh del cubo precedentemente vista, assieme ad una telecamera traslata di 10 lungo l'asse delle X.

```
<library_visual_scenes>
    <visual_scene id="Scena">
        <node>
            <node id="Cubo">
                <matrix>
                    1 0 0 0
                    0 1 0 0
                    0 0 1 0
                    0 0 0 1
                </matrix>
                <instance_geometry url="#Cubo-mesh">
            </node>
            <node id="Telecamera">
                <matrix>
                    1 0 0 10
                    0 1 0 0
                    0 0 1 0
                    0 0 0 1
                </matrix>
                <instance_camera url="#Camera-camera"/>
            </node>
        </node>
    </visual_scene>
</library_visual_scenes>
<scene>
    <instance_visual_scene url="#Scena"/>
</scene>
```

4.4 Risultati

Per verificare le prestazioni del lavoro realizzato, sono stati confrontati i tempi necessari per importare ed esportare gli stessi file rispetto a CAD differenti come SketchUp e Blender.

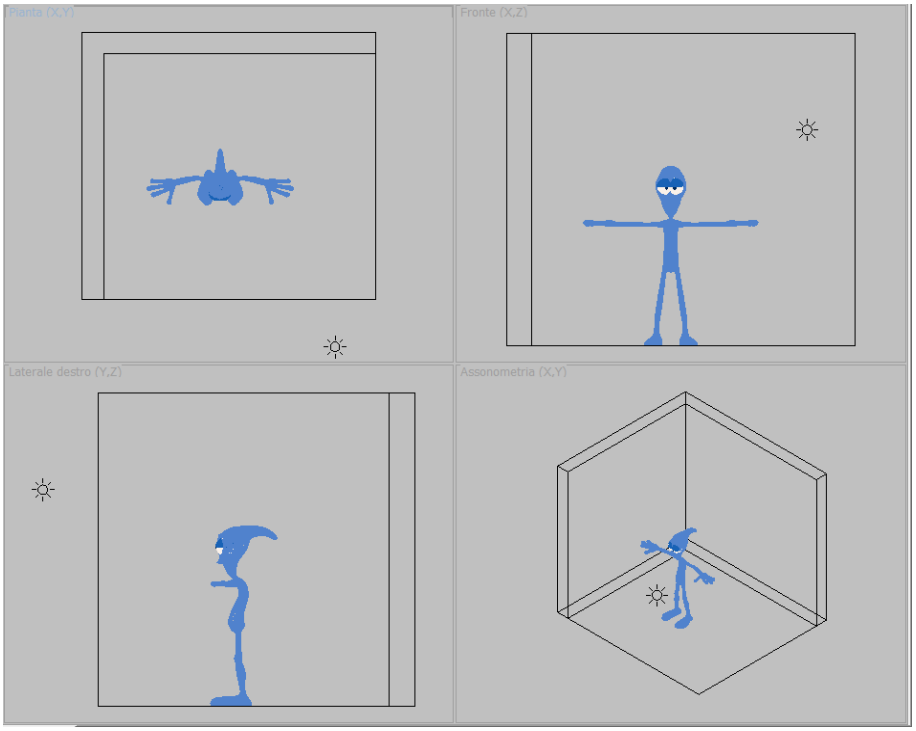
I file impiegati sono circa una ventina e si differenziano, oltre che per dimensioni e complessità, anche per le diverse tecniche rappresentative utilizzate. Ciò ha permesso di verificare la correttezza di tutte le funzionalità create.

Dall'analisi dei primi risultati sono emerse prestazioni molto carenti, se non proprio inappropriate; l'importazione di una semplice cucina composta da un tavolo e da cinque sedie, anche se composte da centinaia di migliaia di facce, richiedeva più di due ore. Attraverso il debug sono state individuate alcune funzioni spesso utilizzate, che in relazione al progetto, al loro interno eseguivano controlli e cicli inutili per la manipolazione delle stringhe. Il tempo per manipolare le stringhe, corrispondenti alle sequenze dei vertici e degli indici, esplodeva in modo esponenziale rispetto al loro aumento. Potenziando ed in parte ricreando alcune funzioni per adattarle a queste casistiche, le prestazioni sono notevolmente aumentate fino ad arrivare ad essere in linea con gli altri CAD. La stessa cucina precedentemente descritta, ora viene caricata in meno di due secondi.

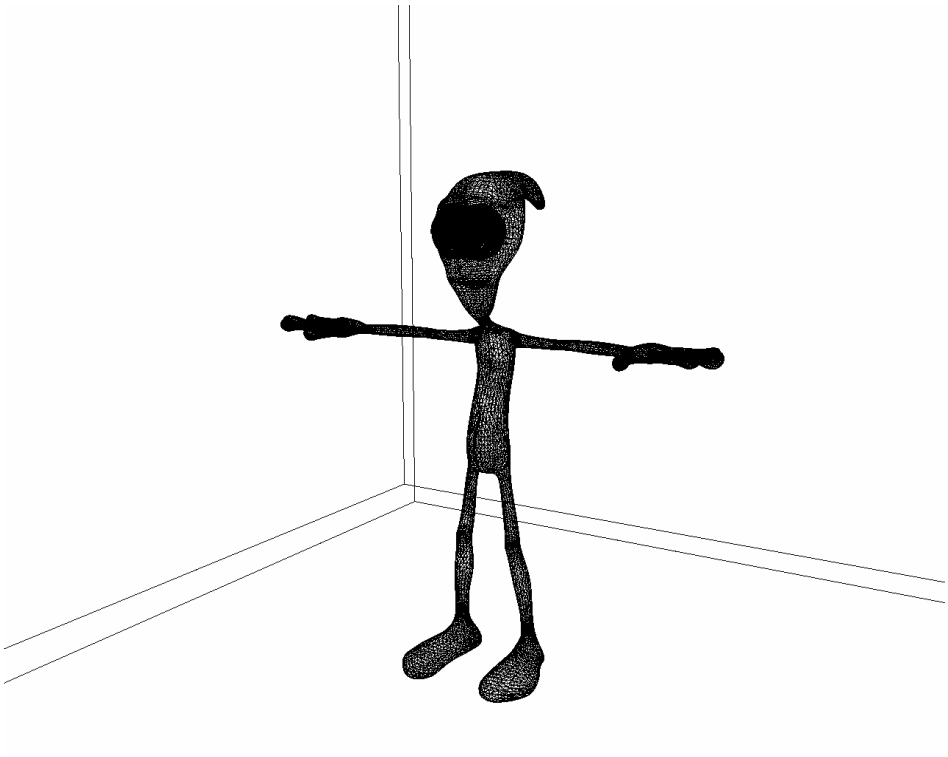
Anche l'esportazione soffriva delle stesse scarse prestazioni e la causa di ciò era ancora una volta l'errata gestione delle stringhe, le quali venivano riallocate e ricopiate per ogni singolo vertice aggiunto. Il tempo necessario per la gestione di un solo migliaio di vertici diventava così molto oneroso. Dopo le dovute modifiche,

come previsto, anche il computo delle esportazioni è diventato comparabile a quelli di SketchUp e Blender.

Le seguenti immagini mostrano il risultato di una importazione assieme alla relativa visione fotorealistica.



Visuale della scena da diverse prospettive



Visuale della scena con modellazione solida

Visuale della scena in fotorealismo

4.5 Sviluppi futuri

Il lavoro svolto ha reso possibile l'utilizzo di COLLADA per l'interscambio delle qualità digitali più comuni con prestazioni equiparabili a quelle degli altri sistemi CAD.

Allo stato attuale, molte informazioni non vengono ancora gestite e potrebbero essere realizzate all'interno di un futuro progetto di sviluppo. La gestione ad esempio delle texture sarebbe relativamente semplice perché già presenti all'interno delle funzionalità di Spazio3D mentre per altri elementi, come nel caso delle luci direzionali o di telecamere che utilizzino delle proiezioni ortogonali e prospettiche, è richiesto un lavoro d'implementazione direttamente nella sua architettura.

Un ultimo aspetto da considerare, riguarda l'estensione del formato. Come visto, COLLADA permette l'aggiunta di elementi non specificati nello schema all'interno degli elementi extra in modo da definire un profilo in grado di servirsi delle tecniche proprie dell'applicazione. Con il loro utilizzo, potrebbero essere inseriti tutti quei dati specifici che Spazio3D gestisce ma non sono previsti dallo standard.

Bibliografia

[1] Moller A., Schwartzbach M., Introduzione a XML, Pearson, 2007.

[2] Holzner S., XML Tutto & oltre, Apogeo, 2001.

[3] Grimaldi F., Manuale delle macchine utensili CNC, Hoelpli, 2007.

[4] Reisdorph K. Henderson K., Borland C++ Builder 5, Apogeo, 1997.

[5] Shreiner D., OpenGL Programming Guide, Pearson, 2009.

[6] Arnaud R., Barnes M., COLLADA, Taylor & Francis, 2006.

[7] Salvemini M., Computer grafica: Introduzione alle tecniche automatiche di rappresentazione, Gruppo Editoriale Jackson, 1982.

[8] Khronos Group, COLLADA - *www.khronos.org/collada*.

[9] BrainSoftware, Spazio3D - *www.spazio3d.com*.

[10] World Wide Web Consortium, XML - *www.w3.org*.

Ringraziamenti

Voglio ringraziare il prof. Giorgio Clemente per l'aiuto e la disponibilità concessami per la realizzazione di questa tesi.

Un sentito ringraziamento ai miei genitori Lino e Nadia per tutti gli sforzi fatti per permettermi di portare a termine il mio percorso di studi. Un grazie lo dedico anche a mio fratello Marco che da sempre condivide con me la passione per l'informatica.

Un ringraziamento speciale va alla mia fidanzata Greta per aver sempre creduto in me, supportandomi e sopportandomi, nei momenti difficili affrontati in questi anni.

Ringrazio i miei amici: Andrea, Bruno, Davide e Vincenzo, per aver condiviso con me in tutti questi anni momenti indimenticabili ed irripetibili.

Dedico questa tesi a mio cugino Michael, per sempre nel cuore…