Alma Mater Studiorum · Università di Bologna

CAMPUS DI CESENA
SCUOLA DI SCIENZE
Corso di Laurea in Ingegneria e Scienze Informatiche

Motori di Rendering a confronto in Blender: Teoria e Applicazioni

Relatore:
Dott.ssa
Damiana Lazzaro

Presentata da:
Marina Londei

Sessione II
Anno Accademico 2014/2015

PAROLE CHIAVE

Rendering

Blender

Ray tracing

Rendering engine

Light behaviour

Introduzione

0.1 Obiettivi

Lo scopo di questa tesi è la presentazione di diversi engine di rendering utilizzabili in Blender, il differente modo in cui operano per la resa dell'immagine e le possibilità che offrono attraverso lo studio dei singoli e il loro confronto.
Il mondo della computer grafica è entrato ormai a far parte di ogni ambito, dall'industria cinematografica e videoludica al campo medico, da quello architettonico al pubblicitario e a quello militare, e per questo motivo le tecnologie utilizzabili sono diventate numerose ed eterogenee; si è ritenuto quindi utile fornire delle informazioni su alcune di esse, spiegandone il funzionamento, pregi e difetti.
Ognuna di queste si fonda su un particolare modello matematico, ma tutte ricercano il realismo della resa, in particolare per quanto riguarda i fenomeni luminosi.
La luce è infatti la componente fondamentale per un buon rendering, in quanto è soltanto tramite essa che si possono descrivere i materiali e la composizione dei singoli oggetti in modo che risultino quanto più realistici possibile all'occhio dell'osservatore; il problema della sua resa è ciò che affligge gli studiosi di questo campo da decenni, alla ricerca di un modello matematico che possa descrivere in modo accurato ogni fenomeno luminoso.
L'elaborato vuole porre l'attenzione proprio su come ogni motore di rendering gestisca questo problema e su tutta la teoria matematica su cui essi si

fondano.
A conclusione dello studio si è voluto realizzare un progetto consistente nella modellazione 3D e nella successiva resa di una scena di interni per mostrare il livello di realismo che è stato possibile raggiungere.

0.2 Tecnologie utilizzate

Per lo studio dei singoli motori e la realizzazione conclusiva del progetto è stato necessario utilizzare un programma di modellazione 3D che potesse supportare una riproduzione quanto più fedele degli ambienti. Si è scelto di utilizzare Blender, programma open source per la realizzazione di modelli tridimensionali, il quale è in grado di supportare tutti e quattro i motori scelti. Per quanto riguarda l'analisi della scena, si è fatto uso di foto e video per lo studio delle proporzioni e della luce, e di una conoscenza personale del luogo rappresentato. La modellazione della scena e l'applicazione di materiali e texture sono avvenute tramite un computer personale, mentre per quanto riguarda la realizzazione delle immagini e del video è stato necessario utilizzare i computer universitari a causa delle risorse hardware richieste dal modello.

0.3 Struttura della tesi

Il lavoro è così strutturato: per prima cosa viene trattato il rendering, dandone una definizione e cercando di introdurre il lettore al mondo della resa 3D, ponendo l'accento sul rendering cosiddetto "physically based" e spendendo qualche parola sul software di modellazione Blender.

Il secondo capitolo tratta i modelli di illuminazione, concentrandosi in particolare su quello globale e spiegando il funzionamento delle due principali famiglie che questo modello comprende (ray tracing e radiosity). Il

capitolo successivo si interessa esclusivamente del ray tracing e si sofferma sul funzionamento dei diversi algoritmi che appartengono a questa famiglia.

Il quarto capitolo entra nel mondo dei quattro motori di rendering studiati per questa tesi descrivendoli in modo dettagliato per poi sottolinearne aspetti positivi e negativi, cercando di offrire una guida per l'utente su come utilizzarli e in quale contesto.

Infine, il quinto capitolo viene dedicato al progetto svolto di pari passo con l'elaborato, descrivendo e motivando le scelte effettuare e seguendo i passi compiuti per realizzarlo.

Indice

Capitolo 1

Blender e il rendering

1.1 Cos'è il rendering

Il rendering è il processo di generazione di un'immagine a partire da un modello 2D o 3D; è la fase fondamentale in cui si passa dalla descrizione del modello all'immagine vera e propria, durante la quale si aggiungono texture, luci e posizioni relative degli oggetti per ottenere l'immagine finale e completa a partire da un'elaborazione preliminare della scena. Il rendering veniva utilizzato inizialmente per risolvere problemi di determinazione di oggetti visibili e superfici nascoste da un certo punto di vista. Nel corso degli anni gli studi su questo processo hanno portato a soluzioni sempre migliori e precise, facendo sì che il realismo nella scena aumentasse e permettendo al rendering di essere utilizzato in un'ampia varietà di discipline quali fisica e astrofisica, astronomia, biologia, psicologia e studio della percezione. Per quanto riguarda l'ambito della computer grafica, dove viene valorizzato più che in altri campi, si utilizzano algoritmi di resa della scena che interpretano e definiscono il colore in ogni punto dell'immagine.

1.2 Physically based rendering ed equazione del rendering

Dal momento che il soggetto della tesi e l'obiettivo del progetto è il realismo della resa, è obbligatorio parlare di physically based rendering. Con questo termine ci si riferisce in modo molto generale al fatto che l'algoritmo utilizzato per il rendering si fondi su principi basati sulla fisica; in realtà il termine non è ancora ben definito nell'ambito grafico, e spesso viene utilizzato dandogli diversi significati. Si può dire che un rendering di questo tipo sia opposto a quello dei modelli "ad hoc", ovvero metodologie più semplici e veloci per dare un'iniziale rappresentazione dell'immagine, rimanendo però ad una descrizione superficiale degli oggetti senza addentrarsi nel loro esatto funzionamento. In generale è difficile poter dire con accuratezza che cosa significhi per un algoritmo essere "physically based", perché affermare che si fondi su principi della fisica fornisce una classificazione molto vaga. Pur non potendo effettuare una forte classificazione è possibile affermare che un physically based rendering è quello che si propone di risolvere l'equazione del rendering, che descrive il flusso di energia luminosa attraverso una scena. Essa è stata proposta nel 1986 da James Kajiya, e ogni tecnica di rendering che sia realistica ha come obiettivo la sua risoluzione. L'equazione è così definita:

In una determinata posizione e direzione, la luce uscente (Lo) corrisponde alla somma della luce emessa (Le) e di quella riflessa. La luce riflessa, a sua volta, è la somma della luce entrante (Li) da tutte le direzioni, moltiplicata per la superficie riflettente e per l'angolo d'incidenza.

Matematicamente:

$$L_o(x, \vec{w}) = L_e(x, \vec{w}) + \int_\omega (f_r(x, \vec{w'}, \vec{w}) L_i(x, \vec{w'})(\vec{w'} \cdot \vec{n}))\ d\vec{w'} \qquad (1.1)$$

Dove:
$L_o(x, \vec{w})$ è la luce uscente da una particolare posizione x e direzione $\vec{w}$;

$L_e(x, \vec{w})$ è la luce emessa nella stessa posizione e direzione;
$\int_\omega ... \, d\vec{w'}$ è la somma infinitesimale calcolata su un emisfero di direzioni entranti;
$f_r(x, \vec{w'}\vec{w})$, è la percentuale di luce riflessa in quella posizione e direzione da w';
$L_i(x, \vec{w'})$ è la quantità di luce entrante da posizione x e direzione w';
$(\vec{w'} \cdot \vec{n})$ è l'attenuazione della luce entrante dovuta all'angolo di incidenza.

Questa equazione possiede due importanti caratteristiche: è lineare (è composta soltanto da addizioni e moltiplicazioni) e possiede omogeneità spaziale (è la stessa in tutte le posizioni e direzioni).

1.3 Cos'è Blender

Blender è un software open-source, free e multipiattaforma che viene utilizzato per modellazione, animazione, compositing e rendering di immagini tridimensionali; la sua storia ha inizio nel 1995 presso lo studio di animazione olandese NeoGeo, e il suo rilascio sotto licenza GNU GPL è avvenuto nel 2002. Ad oggi Blender vanta una community molto estesa e in grado di fornire un supporto eccellente attraverso i numerosi forum ad esso dedicati e alla documentazione estesa (scritta interamente dagli utenti).
È un programma molto flessibile che può essere facilmente compreso e utilizzato da chi si avvicina per la prima volta al mondo della modellazione 3D, ma che offre strumenti complessi per i più esperti.
Esso è in grado di rappresentare diversi fenomeni fisici, tra i quali gli stati gassoso e liquido della materia e la gravità.
Blender offre delle primitive geometriche per modellare i diversi oggetti della scena, insieme a degli strumenti di manipolazione ed editing che consentono di realizzare qualsiasi forma si voglia. È inoltre possibile aggiungere e lavorare con le curve.
All'interno della scena è indispensabile avere una o più sorgenti luminose,

senza le quali non sarebbe possibile renderizzare alcunché, e una telecamera che fornisce l'inquadratura per il nostro rendering. È bene sottolineare che Blender rimane uno strumento, e nonostante l'ampia varietà di tool che fornisce un vero capolavoro si può ottenere soltanto tramite la pratica e lo studio, nonché la conoscenza di materie quali l'anatomia, la teoria dell'illuminazione e dei colori, i principi dell'animazione e in generale ogni conoscenza artistica, tecnologica e scientifica che c'è dietro ciò che vogliamo rappresentare.

Capitolo 2

Modelli di illuminazione

Un modello di illuminazione descrive i fattori che determinano il colore di una superficie in un determinato punto, descrivendo le interazioni tra le luci e gli oggetti tenendo conto delle proprietà delle singole superfici e della natura della radiazione luminosa incidente su esse. L'uso di un modello di illuminazione è fondamentale per ottenere una rappresentazione realistica delle superfici tridimensionali, altrimenti l'unica assunzione che si potrebbe fare è che esse siano illuminate uniformemente, facendole apparire piatte e perdendo la loro natura tridimensionale.
Seguendo il modello fisico, quando la luce colpisce una superficie, una parte viene assorbita e una parte riflessa; per alcuni tipi di materiali (quelli traslucidi) la luce viene anche rifratta.
Questi fenomeni possono essere osservati sia per i raggi di luce diretta, sia per i raggi di luce derivanti dalle inter-riflessioni tra gli oggetti. Ogni modello di illuminazione offre un certo livello di realismo a seconda di come riesce a rappresentare i diversi fenomeni, e di pari passo varia la complessità computazionale richiesta: più il modello è sofisticato, più aumenta il livello di realismo e più aumenta il costo computazionale. La scelta di un modello piuttosto che di un altro dipende dalla potenza di calcolo che si ha a disposizione, dalla complessità della scena che si vuole rappresentare e dalla necessità o meno che le immagini siano da produrre in tempo reale.

I modelli di illuminazione possono essere suddivisi in due grandi famiglie: modelli locali e modelli globali.
I modelli locali sono chiamati così perché non tengono conto dell'interazione degli oggetti con gli altri (e quindi delle inter riflessioni), ma si limitano a considerare ogni superficie in modo autonomo e calcolare per ogni punto di essa come viene illuminato dalla sorgente luminosa. L'interazione non viene calcolata, ma viene simulata tramite una luce ambientale.
I modelli di illuminazione globale, al contrario, calcolano il colore che indice sul punto in termini sia di luce emessa direttamente dalle sorgenti luminose sia di luce che raggiunge il punto dopo la riflessione e trasmissione della luce sia dalla propria superficie che dalle altre presenti nell'ambiente. Ovviamente l'approccio globale conferisce più realismo alla scena perché simula l'effettivo comportamento della luce in un ambiente in cui sono presenti diversi tipi di materiali che agiscono attivamente sulla visione.
Vista la natura dell'obiettivo che si è prefissati, ovvero il realismo e l'accuratezza fisica, verranno approfonditi soltanto i modelli di illuminazione globale.
Gli algoritmi che implementano questo modello sono diversi, e possono essere divisi in due grandi categorie:

- Ray tracing
- Radiosity

La differenza tra radiosity e ray tracing sta nel tipo di calcolo che viene effettuato per la resa della scena; nei paragrafi dedicati si tratterà la differenza in dettaglio.
Per ora si vuole sottolineare che il metodo radiosity è detto "view-independent" mentre il ray tracing è "view-dependent": il primo infatti calcola la diffusione della luce nell'ambiente indipendentemente dall'osservatore, considerando il contributo di ogni superficie e l'influenza che essere esercitano tra di loro, e soltanto in un secondo momento verranno calcolate una o più immagini per un certo punto di vista.
Al contrario, il ray tracing è un algoritmo "view-dependent" perché si fonda

sul concetto che gli unici punti della scena da rappresentare sono quelli visti dall'osservatore, ovvero i punti dai quali la luce riflessa o diffusa raggiunge il punto di vista.
Gli algoritmi view-dependent sono ottimi per rappresentare fenomeni speculari, ovvero altamente dipendenti dalla posizione dell'osservatore, ma comportano lavoro extra quando devono rappresentare fenomeni diffusivi che per grandi aree dell'immagine non presentano grandi variazioni; al contrario, gli algoritmi view-independent modellano efficientemente il fenomeno diffusivo ma richiedono eccessivi tempo e memoria per rappresentare la specularità.

2.1 Ray tracing

Uno dei primi metodi di sintesi dell'immagine nacque da un'idea trovata negli studi della fisica: quando dovevano progettare le lenti, i fisici disegnavano sulla carta il percorso seguito dai raggi della luce che nascevano da una sorgente, passavano attraverso la lente e poi proseguivano oltre; questo processo, consistente nel seguire i raggi della luce, era chiamato "ray tracing".
I primi ricercatori nell'ambito della computer grafica trovarono che simulare il comportamento della luce studiandone l'effettiva fisica fosse la soluzione migliore per rappresentare ottime immagini, ma nel 1960 i computer erano troppo lenti per poter abbracciare questa metodologia, perciò l'idea passò in secondo piano per molti anni. Col passare del tempo, vennero ideati molti altri algoritmi che cercavano di simulare diversi aspetti del fotorealismo: riflessioni, ombre e sfocature dovute al movimento degli oggetti. Questi algoritmi però non riuscivano a rappresentare tutti i casi in cui si verificavano questi fenomeni e inoltre spesso non era possibile farli coesistere nella resa della stessa immagine: accadeva infatti che venissero renderizzate immagini con ombre ma senza trasparenza, con riflessione ma senza dettagli di sfocatura.
Con l'aumentare della potenza dei computer aumentò anche il fascino per l'algoritmo di ray tracing che era stato abbandonato, in modo da riuscire a

simulare la vera fisica della luce; l'algoritmo venne quindi ripreso, studiato e migliorato aggiungendo una grande varietà di effetti ottici.
Ad oggi il ray tracing è una delle tecniche più potenti tra quelle della sintesi delle immagini, semplice e facile da implementare, ed è anche la più utilizzata.
Esistono alcuni fenomeni ottici che l'algoritmo non riesce ancora a rappresentare del tutto, come le inter-riflessioni diffuse della luce (il classico effetto del "color bleeding" della luce colorata che fa sì che un oggetto rosso su un tavolo bianco dia una colorazione rosastra a quest'ultimo) e gli effetti caustici (la concentrazione della luce in un punto, ad esempio le onde luminose che si possono notare sul fondo di una piscina).
La metodologia del ray tracing si basa sul seguire il percorso dei raggi infinitesimali della luce attraverso la scena finché non incontrano una superficie.
È una tecnica che consiste nel generare un'immagine basandosi sul percorso della luce attraverso i pixel e simulandone il comportamento nell'ambiente che si vuole creare, ovvero l'effetto che genera nell'incontrare i diversi oggetti della scena; l'obiettivo di questa categoria di algoritmi è quello di determinare e calcolare il contributo dei singoli raggi all'immagine.
Il realismo che è in grado di fornire questo algoritmo comporta un alto costo computazionale e proprio per questo motivo viene utilizzato in quei casi in cui l'immagine può essere renderizzata lentamente durante il corso del tempo: ad esempio, è ottimo nel caso di effetti speciali cinematografici, mentre non è adatto per applicazioni real time dove è richiesta velocità.
Il ray tracing è in grado di simulare una grande varietà di effetti visivi, quali riflessione, rifrazione e diffusione della luce. Esistono diversi modi per implementare un algoritmo di ray tracing, ma tutti devono tenere in considerazione particolari oggetti, fenomeni e aspetti:

- *Camera*: come e da dove la scena è vista. Le diverse camere generano dei raggi che vanno dal punto di vista alla scena.

- *Intersezioni raggio-oggetto*:occorre sapere precisamente dove un raggio interseca un oggetto. Inoltre occorre determinare alcune proprietà geo-

metriche dell'oggetto nel punto di intersezione, quali la nomale alla superficie o il materiale di cui è fatto.

- *Distribuzione della luce*: la luce è ovviamente la componente fondamentale per un buon rendering; è necessario quindi modellare il comportamento e la distribuzione di essa attraverso tutta la scena.

- *Visbilità*: per sapere se la luce che stiamo considerando colpisce un determinato punto di una certa superficie, occorre controllare se il percorso luce-punto non sia interrotto. Fortunatamente, per qualsiasi ray tracer ciò è facile da determinare dal momento che è possibile tracciare il raggio dalla superficie alla sorgente luminosa, trovare l'intersezione raggio-oggetto più vicina e confrontare la distanza di questa con la distanza della sorgente: se un raggio superficie-luce (chiamato "raggio ombra") interseca un oggetto prima di arrivare alla luce, il punto si trova in ombra (perlomeno rispetto alla sorgente luminosa). Il raggio ombra è in sostanza come un qualsiasi altro raggio, con la differenza che viene utilizzato per osservare l'ambiente intorno in cerca di ombre; nel momento in cui uno di questi raggi arriva effettivamente alla sorgente luminosa senza incontrare un oggetto opaco esso verrà considerato come un effettivo raggio luminoso, che porta quindi con sé energia.

- *Diffusione delle superfici*: di ogni oggetto è necessario conoscere il modo in cui appare, incluse le informazioni riguardo a come la luce interagisce con la superficie dell'oggetto e anche la natura della luce diffusa da essa (ovvero quella "re-irraggiata"). In particolare siamo interessati alle proprietà della luce diffusa verso la camera. Esistono modelli di diffusione delle superfici che possono simulare una grande varietà di comportamenti.

- *Ray tracing ricorsivo*:poiché la luce può raggiungere un oggetto dopo aver attraversato o essere rimbalzata su diverse altre superfici, è solitamente necessario tracciare raggi aggiuntivi che trovano la loro origine

sulle superfici, e ciò è particolarmente importante per superfici lucide quali metallo o vetro.

- *Propagazione dei raggi*: un'altra componente importante riguarda lo studio di ciò che succede alla luce quando attraversa un certo ambiente. Se renderizziamo una scena nel vuoto, l'energia della luce rimane costante per tutto il percorso di ogni raggio. Nonostante nessun osservatore sia mai stato nel vuoto, questa è l'assunzione che fa la maggior parte dei ray tracers, anche se esistono comunque modelli molosofisticati che simulano il comportamento della luce attraverso l'atmosfera terrestre, la nebbia, e così via.

2.2 Radiosity

L'algoritmo di radiosity è stato ideato per ben approssimare le interazioni tra le superfici dotate di riflessioni diffusive, considerate tutte riflettori Lambertiani, ovvero perfettamente diffusive.
In particolare questa metodologia vuole rappresentare in maniera più accurata possibile il modo in cui la diffusività di ciascuna superficie influenzi l'altra, caratterizzandone la luminosità e le sfumature di colore, modellando come la luce rimbalzi da un oggetto all'altro nella scena.
Questo approccio permette di ottenere effetti di luce "soft", come ombre soft-edged e colour bleeding, ovvero la colorazione di una superficie o parte di essa che risulta essere dipendente dalla riflessione di colore di superfici vicine: ad esempio si può pensare ad un oggetto rosso posto su un tavolo bianco che fa sì che l'area del tavolo circostante risulti essere rosastra. Il radiosity possiede due importanti peculiarità:

- Le fonti di luce sono trattate esattamente come qualsiasi altra superficie (è quindi possibile modellarne la forma), con la differenza che hanno un parametro di emissione positivo;

- Le equazioni di radiosity è risolta una volta soltanto, prima che la scena venga renderizzata, dopodiché questa può essere vista da qualsiasi punto senza necessità di re-computazione. Ciò causa un calcolo iniziale molto lungo, ma grazie a ciò è poi possibile riprodurre velocemente diverse viste utilizzando il metodo di Gouraud per lo shading.

Iniziamo a comprendere il radiosity con un esempio: immaginiamo di avere un mondo 3D "chiuso", una stanza con diversi oggetti illuminata da pannelli di luce. La stanza si trova in un equilibrio di energia luminosa poiché per ogni superficie (eccetto quelle luminose) la quantità di energia che la raggiunge è uguale a quella assorbita più quella re-irraggiata; quest'ultima, inoltre, diventa una frazione della luce che raggiunge gli altri oggetti nella stanza.
Data una superficie i, essa sarà potenzialmente illuminata da ogni altro oggetto nella scena; la quantità di luce rilasciata da i è chiamata radiosità di quella superficie, ed è formata da due componenti: la luce che la raggiunge dalle altre superfici e che viene diffusa da essa e la luce che viene generata direttamente dalla superficie (nel caso in cui sia una sorgente di luce).
La riflettività di una superficie ci informa riguardo la quantità della prima componente che riesce a diffondere; per fare ciò dobbiamo ovviamente sapere quanta luce raggiunge una superficie, ed è proprio sulla risoluzione di questo problema che si fonda il metodo radiosity. Sostanzialmente la luce che raggiunge la superficie i-esima è data dalla somma della frazione della radiosità di ogni altra superficie che la raggiunge.
Se con j indichiamo tutte le altre superfici, allora la frazione F_{ij} sarà la quantità di energia luminosa che raggiunge la superficie i dalla superficie j e verrà chiamata *Form Factor*.
Il form factor dipende dalla geometria della scena, inclusi l'angolazione e la distanza tra le superfici i e j. Per riassumere ciò che abbiamo detto, si ha:

$$\text{Radiosità}_i = \text{Emissione}_i + \text{Riflettività}_i \int \text{Radiosità}_j \cdot \text{FormFactor}_{ij}$$

dove l'integrale è su tutte le superfici j. Strettamente parlando, si intende su ogni punto di ogni superficie perché la radiosità non è costante su tut-

ta la superficie. Non è però pratico calcolare la radiosità sul singolo punto matematico; occorre decidere fino a che livello di approssimazione si vuole calcolare la distribuzione dell'energia luminosa. Visto che le superfici possono essere molto grandi generalmente vengono suddivise in *patches*(pezze), ovvero le nostre unità per il calcolo della radiosità, considerate energicamente uniformi.

Dopodiché possiamo porre una mesh triangolare al centro della patch (dove abbiamo calcolato la radiosità) e applicare il Gouraud shading a tempo di render. Il Gouraud shading è una particolare tecnica di shading (ovvero il processo che determina il colore dei pixel di una superficie) che si fonda sullo shading interpolativo. Questo si propone di ottenere l'informazione sul colore per ogni vertice del triangolo e determinare la restante superficie interpolando i valori trovati per i vertici. In questo modo però lo shading risulta indipendente per ogni triangolo, causando una netta visibilità dei bordi tra due triangoli adiacenti; il modello di Gouraud tiene conto dell'effettiva geometria del modello che si vuole rappresentare, e in particolare delle informazioni date da triangoli adiacenti. Dopo aver realizzato la griglia di triangoli per la superficie, per ogni vertice di essa si calcola la normale alla superficie (e non al triangolo): in questo modo il calcolo dello shading produce lo stesso valore su entrambi i lati di poligoni adiacenti, rendendo la colorazione complessiva uniforme.

Questo metodo richiede che sia nota la normale alla superficie in ogni vertice; se non è disponibile, la si approssima come media delle normali ai poligoni che condividono quello specifico vertice.

Questa metodologia ci permette di modellare tutti quegli effetti di luce soffusa che troveremmo in una stanza.

Le stesse sorgenti di luce non sono trattate in modo differente: ogni patch può diffondere ed emettere luce.

Poiché la radiosità di una patch dipende dalla radiosità di ogni altra patch avremo un grande sistema di equazioni (che aumenterà all'aumentare delle patches della scena) da risolvere per poter determinare la radiosità della sin-

gola. Per limitare il numero di equazioni da risolvere si cerca di ridurre il numero di patches per quelle aree su cui ci aspettiamo di avere un'illuminazione uniforme, mentre per zone in cui la luminosità varia rapidamente si utilizzerà un numero maggiore di suddivisioni.
Il sistema di equazioni da risolvere deriva dalla determinazione ogni possibile interazione di luce tra tutte le patch; questo è il motivo per cui l'algoritmo radiosity richiede molto tempo nella fase di modellazione, bilanciato però dal minor tempo di rendering, fase durante la quale viene semplicemente applicato Gourad per realizzare la colorazione finale.

2.2.1 Costruzione dell'equazione di radiosity

Volendo dare una definizione del termine, diciamo che la radiosity è la quantità di energia per unità di area che lascia una superficie nell'unità di tempo. Possiamo ignorare la componente temporale e procedere col descrivere le componenti dell'equazione:

- A_i è l'area della patch i
- B_i è la radiosità della patch i
- E_i è l'energia generata per unità di area nell'unità di tempo (componente considerata soltanto per le sorgenti di luce)
- R_i è il coefficiente di riflessione, che determina la frazione di energia luminosa incidente che viene riflessa
- F_{ij} è la frazione di energia luminosa che lascia la patch j e raggiunge la patch i (Form Factor)

Considerando una patch j, essa emetterà $\mathrm{B}_j d\mathrm{A}_j$ energia totale. Parte di questa energia, una frazione F_{ij}, colpirà la patch i:

$$\mathrm{F}_{ij}\mathrm{B}_j d\mathrm{A}_j$$

Il totale di energia che raggiungerà la patch i considerando ogni altra patch j sarà:

$$\int \mathrm{F}_{ij}\mathrm{B}_j d\mathrm{A}_j \tag{2.1}$$

Una parte di questa viene assorbita dalla patch i mentre il resto sarà diffuso; la quantità diffusa sarà quindi:

$$\mathrm{R}_i \int \mathrm{F}_{ij}\mathrm{B}_j d\mathrm{A}_j \tag{2.2}$$

Poiché la patch che stiamo considerando potrebbe essere una sorgente di luce e quindi emetterne, in generale si avrà che la luce emessa da una patch i sarà:

$$\mathrm{B}_i d\mathrm{A}_j = \mathrm{E}_i d\mathrm{A}_j + \mathrm{R}_i \int \mathrm{F}_{ij}\mathrm{B}_j d\mathrm{A}_j \tag{2.3}$$

In questa equazione gli unici elementi di cui non siamo a conoscenza sono i Form Factor, che dovranno essere calcolati per ottenere la radiosità delle singole patches. È utile sottolineare che nel calcolo della radiosità, e quindi dei Form Factors, si effettuano due approssimazioni per quanto riguarda la geometria: la prima è che trattiamo ogni patch come energicamente uniforme (in quanto non ci è possibile calcolare la radiosity per ogni punto della superficie); la seconda è che si lavora con patch di dimensione finita.
Possiamo quindi approssimare l'integrale dell'equazione di radiosity con una sommatoria:

$$\mathrm{B}_i = \mathrm{E}_i + \mathrm{R}_i \sum_{j=1}^{n} \mathrm{F}_{ij}\mathrm{B}_j \tag{2.4}$$

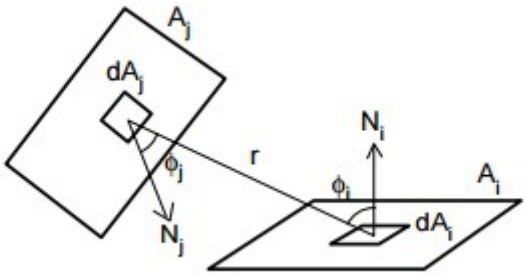
Per immagini monocrome basterà utilizzare una sola di questa equazione per ogni patch nel modello, e tutte le equazioni generate dovranno essere calcolate simultaneamente. Per patch colorate invece avremo bisogno di tre di queste equazioni, una per il rosso, una per il verde e una per il blu (RGB). Come possiamo notare, ogni B è dipendente dal valore di B di tutte le altre patches.

E_i sappiamo essere diverso da zero soltanto per le sorgenti luminose, mentre gli R_i dipendono dalle caratteristiche della superficie. Gli F_{ij} devono essere calcolati dipendentemente dalla geometria della scena; poiché le patches sono piatte nessuna quantità di energia che lascia una patch può colpire sé stessa, perciò $F_{ii} = 0$.

2.2.2 Calcolo dei Form Factor: metodo dell'emisfera

Abbiamo fatto l'assunzione secondo la quale la luce ricevuta da una patch è uniforme su tutta la sua superficie; ciò significa che possiamo considerare semplicemente un punto su i . Inoltre possiamo notare che la patch i può ricevere luce da ogni direzione entro un'emisfera centrata sul punto scelto.

Figura 2.1:

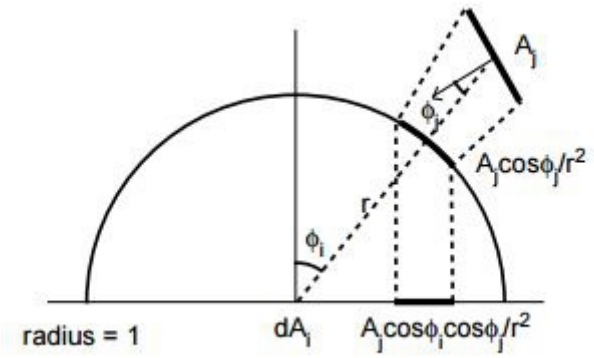


Dall'immagine possiamo dedurre che la quantità di energia che lascia un punto sulla superficie j in direzione di un punto della patch i è ridotta all'angolo sotteso verso quella direzione dalla patch j , in particolare sarà ridotta a $\cos(\phi_j)$; simmetricamente, la quantità di energia che colpisce la patch i sarà ridotta del suo angolo, $\cos(\phi_i)$. Poiché abbiamo assunto che le patch siano abbastanza piccole da essere considerate energicamente uniformi allora non dobbiamo fare altro che sommare ogni contributo. Ciò significa che il Form Factor sarà:

$$F_{ij} = \frac{1}{A_i} \int_{A_i} \int_{A_i} \frac{\cos(\phi_i)\cos(\phi_j)}{\pi r^2} dA_j dA_i \tag{2.5}$$

In questa formula ci sono due coseni, ognuno associato a una delle due patches: uno è per l'angolo della superficie che invia la radiazione luminosa (perché l'angolo determina quanta luce può diffondere in quella particolare direzione) e l'altro è per l'angolo della superficie che riceve radiazione (in quanto la possibilità di ricevere luce e diffonderla dipende dall'angolo che si viene a formare in direzione di quella luce).
Cominciamo a considerare la patch i come un punto, in modo da concentrarvi l'energia che arriva.
Una metologia che può essere utilizzata per calcolare l'energia che arriva dalle altre patches è quella di posizionare un emisfera di raggio unitario con centro in quel punto, orientata nel verso opposto alla superficie; l'energia che passa attraverso l'emisfera sarà quella che arriverà alla patch i. Se proiettiamo le altre patches sull'emisfera, queste proiezioni ci consentiranno di calcolare i Form Factor per queste.
Per prima cosa proiettiamo la patch j sull'emisfera di i, usando il centro di quest'ultima come centro di proiezione; l'area risultante proiettata sull'emisfera tiene conto dell'orientamento della patch j ma non di quello della patch i, in quanto abbiamo effettuato la proiezione usando il centro, che non ha direzione. Ovviamente se il contributo della patch j colpisce la patch i verticalmente, esso sarà maggiore rispetto al contributo ottenuto da patch che la colpiscono per obliquo. Possiamo tenere conto di ciò pesando il contributo di ogni patch a seconda di dove si trova sull'emisfera: avremo un peso molto alto vicino al centro dell'emisfera e, al contrario, un peso più basso man mano che ci si avvicina al piano.
Possiamo quindi constatare che l'area proiettata sull'emisfera è l'energia luminosa dovuta alla patch j e al suo orientamento rispetto al centro dell'emisfera; la sua posizione rispetto all'emisfera invece determina quanta di questa energia effettivamente riceverà i. Ovviamente, l'energia ricevuta è proporzionale all'area calcolata sull'emisfera e inversamente proporzionale al quadrato della distanza tra le due patches.

Figura 2.2: Metodo dell'emisfera



Dall'immagine notiamo che r è la distanza tra $d\mathrm{A}_i$ e il centro di A_j ; poiché l'emisfera ha raggio unitario, l'area del cerchio alla base è uguale a π. Come detto sopra, ciò che dobbiamo fare è proiettare la patch j sull'emisfera tenendo conto del suo angolo rispetto alla direzione r. Questo calcolo non tiene conto dell'orientamento della patch i, quindi il passo successivo è proiettare l'area dall'emisfera al piano sottostante.
L'area risultante, frazione dell'area totale π, è proporzionale al Form Factor. Proiettando l'area A_j sull'emisfera otteniamo $\mathrm{A}_j \cdot \frac{\cos(\phi_j)}{r^2}$. Il coseno è l'angolo di A_j mentre r^2 è dovuto alla legge dell'inverso del quadrato della distanza che nel caso della luce afferma che l'intensità di essa a partire da una sorgente è inversamente proporzionale al quadrato della distanza tra la superficie colpita e la sorgente (che nel nostro caso è la patch j che diffonde luce in direzione di i).
Dopo aver proiettato la patch j sull'emisfera, possiamo concentrarci solo sulla patch i considerando l'area che abbiamo ottenuto.
Proiettando l'elemento sull'emisfera su A_i otteniamo $\mathrm{A}_j \cdot \frac{\cos(\phi_j)\cos(\phi_i)}{r^2}$, che è ciò che ci serve per calcolare l'energia proveniente dalla patch j.
Se la patch che stiamo proiettando sull'emisfera è parallela alla patch i contribuisce con la quantità massima di energia che diffonde, mentre se si trova in basso, ad esempio all' orizzonte dell'emisfera allora il suo contributo sarà nullo.

In conclusione, considerando una piccola area $d\mathrm{A}_i$ sulla superficie i e ogni altra patch j possiamo approssimare il calcolo del form factor a:

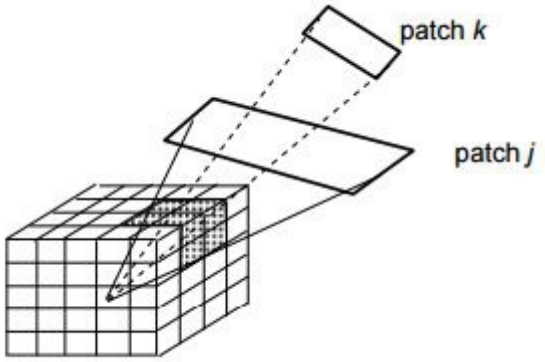
$$\mathrm{F}_{ij} \approx \int_{\mathrm{A}_j} \frac{\cos(\phi_j)\cos(\phi_i)}{r^2} \tag{2.6}$$

integrando su tutta l'emisfera.

2.2.3 Calcolo dei Form Factor: approccio dell'emicubo

La soluzione proposta dall'utilizzo dell'emisfera rimane una soluzione continua; nel 1985 Cohen e Greenberg proposero un metodo per il calcolo dei Form Factor utilizzando un numero finito di aree, approssimando l'emisfera ad un emicubo.

Figura 2.3: Metodo dell'emicubo



Si divide l'emicubo in piccole celle: più se ne usano, più accurato risulterà il calcolo. Come per l'emisfera, si proiettano le patches sull'emicubo: accadrà che alcune patches saranno proiettate su una sola faccia dell'emicubo, altre invece occuperanno più facce. Utilizzando lo z-buffering è inoltre possibile confrontare la profondità di due patches proiettate sulla stessa cella ed eliminare quella più lontana.
Ciò che fa l'approccio dell'emicubo è facilitare la proiezione della patch j , ma si è persa la possibilità di calcolare il peso del contributo della patch non potendo più lavorare con gli angoli in quanto l'emicubo ne ha soltanto

due (orizzontale e verticale). Si sa però quale cella sulla superficie approssima quale parte dell'emisfera, quindi riusciamo comunque a dare un peso ad ogni cella. Una cella che si trova nel centro di una faccia può essere vista direttamente dal centro della patch i, mentre una cella vicino allo spigolo dell'emicubo è vista obliquamente.
È possibile generare il peso delle singole celle al momento della costruzione dell'emicubo e incorporarlo ad ogni piccola area.
Ogni cella dell'emicubo viene considerata come una singola patch e le viene associato un valore delta di Form Factor precalcolato:

$$\Delta \mathrm{F}_p = \frac{\cos(\phi_j)\cos(\phi_i)}{r^2}\Delta \mathrm{A} \tag{2.7}$$

dove ϕ_p è l'angolo tra la normale alla cella p e il vettore che congiunge $d\mathrm{A}_i$ e p e A è l'area della cella. Infine, F_{ij} viene calcolato sommando i Form Factor delle celle su cui la patch A_j viene proiettata.

Il processo che utilizza radiosity necessita di una grande quantità di tempo, anche se una volta che la scena è stata computata scegliamo un punto di vista e utilizzando il Gouraud shading possiamo unire i colori delle patches; se volessimo cambiare il punto di visione non ci sarà bisogno di renderizzare nuovamente tutta la scena e ricalcolare la radiosity delle singole patches, ma basterà semplicemente riapplicare il Gourad shading. In ogni caso, il grande costo del radiosity ci porta a sceglierlo soltanto nel caso in cui l'accuratezza dell'immagine è di gran lunga più importante del tempo di resa e quando la scena è fissa ma vista più volte e da diversi punti. Molte delle funzionalità del radiosity possono essere emulate da ottimi ray tracers, sebbene forniscano una minore accuratezza.

Capitolo 3

Implementazioni dell'algoritmo di ray tracing

L'algoritmo di ray tracing si propone, come già detto, di costruire la scena seguendo il percorso dei raggi della luce, determinando così il colore corretto di ogni singolo pixel.
Il modo per colorare la scena consiste nello studiare tutti i raggi che colpiscono ognuno dei pixel ed effettuare una sorta di media dei colori che lo raggiungono; come si trovano però questi raggi?
Definire la colorazione di un raggio non è difficile: possiamo pensare ad esso come a un percorso seguito da una particella di luce (il fotone) mentre attraversa lo spazio. Nel mondo reale ogni fotone porta con sè dell'energia: questa energia viene trasferita alle celle ricettrici nella retina. Il colore che percepiamo per quel fotone è collegato all'energia trasportata da esso: colori diversi corrispondono a fotoni con diversa energia associata.
L'energia trasportata da un fotone è una radiazione elettromagnetica che può essere modellata sotto forma di onda elettromagnetica, la quale possiede una lunghezza d'onda e una frequenza; esse sono tra loro inversamente proporzionali, perciò minore sarà la lunghezza d'onda maggiore sarà la frequenza e quindi l'energia del fotone.
Lo spettro elettromagnetico indica l'insieme di tutte le possibili frequenze:

una parte di questo spettro non è visibile (lunghezze d'onda maggiori/minori di una certa soglia), mentre quella visibile dà origine alla luce e ai colori. Ad ognuna delle frequenze dello spettro è associata una diversa energia e quindi un particolare colore che i nostri occhi percepiranno.
Ovviamente può accadere che tipi diversi di fotone (e quindi colori diversi) arrivino simultaneamente ai nostri occhi, facendo in modo di percepire la somma dei colori che ci ha raggiunto; considerando quindi un pixel in una scena, occore determinare quali fotoni presenti contribuiranno effettivamente alla sua colorazione.
Esistono diverse tecniche che ci consentono di raggiungere questo obiettivo, che sono per l'appunto le differenti implementazioni utilizzabili per realizzare l'algoritmo del ray tracing.
Prima di analizzarle, occorre effettuare la differenziazione tra metodi biased e unbiased.
Un metodo unbiased utilizza modelli fisici e ottici accurati, e contempla molti fenomeni luminosi quali ad esempio la dispersione spettrale, la polarizzazione della radiazione elettromagnetica, la stenoscopia e l'aberrazione ottica. Questi metodi sono detti anche "fotorealistici" perché riproducono in maniera realistica l'ambiente e il comportamento della luce, a scapito ovviamente dei tempi di calcolo. I rendering fotorealistici si propongono di generare immagini così come le vedrebbe l'occhio umano.
La resa fotorealistica viene utilizzata in ambito architettonico, dove si ricerca la qualità senza compromessi, ed è fondamentale per l'industria degli effetti speciali dove spesso accade che footage del mondo reale e immagini generate al computer debbano essere unite.
Al contrario, un algoritmo biased risulta essere più veloce a scapito però della qualità finale dell'immagine che si allontana dalla realtà e la rappresenta in modo approssimativo. Esso utilizza artefatti di compressione: questi sono il risultato di una compressione dati applicata all'immagine che rimuove alcuni dati dal contenuto complessivo, lasciando tuttavia che il risultato sia comunque gradevole.

Come la parola ci suggerisce, "unbiased" significa "privo di errori" : in realtà, ovviamente, gli errori sono presenti anche in questo tipo di rendering nonostante cerchi, per approssimazioni successive, di raggiungere un risultato perfetto.
Emulare il comportamento della luce è estremamente complesso in quanto bisognerebbe studiare anche il tipo di atmosfera in cui è calata la scena; per questo motivo è sbagliato affermare che un algoritmo unbiased ci fornisca una resa perfetta, perché nessun metodo è in grado di fornircela. Ciò che lo differenzia però da un algoritmo biased è che quest'ultimo non fornirà mai una stima corretta quando si tratterà di risolvere l'equazione del rendering.
In termini di calcolo ciò significa che, utilizzando il metodo di Monte Carlo per l'integrazione per la risoluzione dell'equazione 1.1, dando un numero sufficientemente alto di punti N per il calcolo dell'integrale un algoritmo unbiased non produrrà alcun errore nel calcolo; al contrario, uno biased non potrà mai garantirci un risultato corretto indipendentemente dal numero di punti che gli forniremo.
Di seguito si analizzeranno nel dettaglio i diversi algoritmi del ray tracing, spiegandone il funzionamento e descrivendone pregi e difetti.

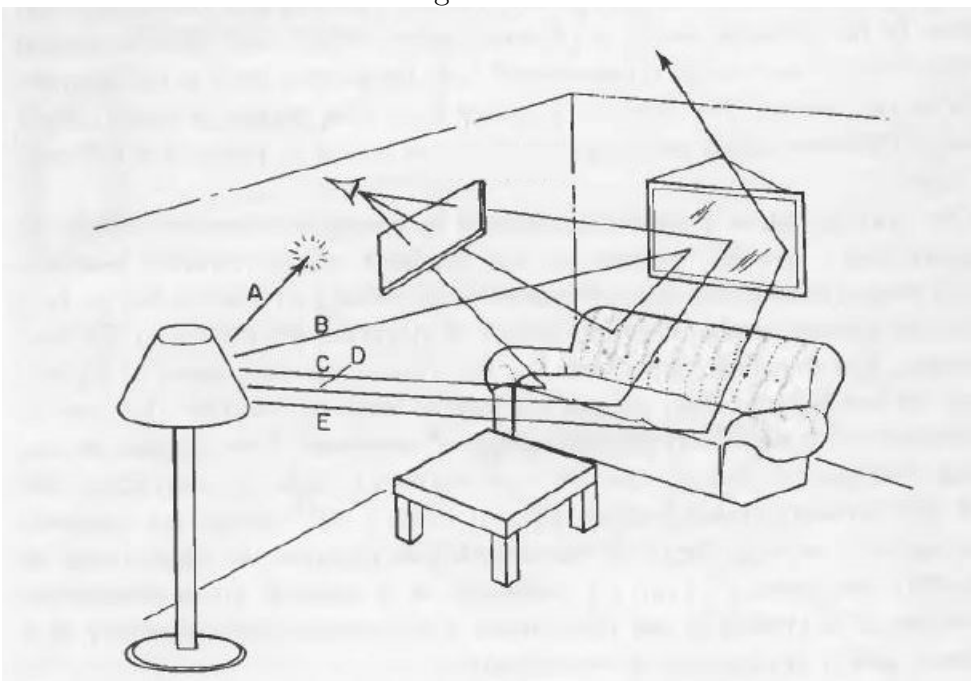
Metodo di Monte Carlo per l'integrazione L'integrazione col metodo di Monte Carlo si fonda sulla scelta di numeri random utilizzati per approssimare il risultato dell'integrale: in particolare, questi n numeri scelti casualmente saranno quelli in cui calcolare la funzione integranda. Il vantaggio di utilizzare questo metodo è la facilità di realizzazione e la possibilità di estenderlo ad integrali in più di una variabile di integrazione; dall'altra parte troviamo però che affinché il risultato sia accettabile (se non ottimo) il numero n di punti scelti deve essere molto grande: esso è infatti un metodo statistico e l'errore commesso nella stima diminuisce all'aumentare del numero di punti utilizzati.
Nel caso del calcolo dell'integrale dell'equazione d'illuminazione un errore maggiore porta ad un aumento del disturbo nell'immagine: questo può essere

eliminato aumentando il numero di punti di valutazione.

3.1 Forward path tracing

Immaginiamo nuovamente di trovarci in una stanza (figura 3.1), ad esempio un salotto, in cui sono presenti vari oggetti tra i quali un divano, uno specchio, una lampada (la nostra sorgente di luce) e un tavolo.

Figura 3.1:



Il piano rappresenta la posizione dell'occhio e la direzione di vista. Supponiamo che la lampada sia accesa: essa emetterà fotoni in ogni direzione e per ognuno di essi occorre studiare se ed eventualmente in che modo contribuirà alla scena; per ora concentriamoci sullo studio di alcuni fotoni in particolare. Il fotone A lascia la sorgente di luce dalla quale è stato generato e colpisce la parete: esso viene per la maggior parte assorbito, perciò non contribuisce all'immagine.
Il fotone C invece, dopo aver lasciato la lampada, colpisce il divano e qui

viene parzialmente assorbito prima di essere riflesso; nonostante ciò, da lì un altro fotone con una colorazione più debole lascia il divano e arriva fino al nostro occhio. È proprio per questo motivo che riusciamo a vedere il divano: la luce generata viene (in parte) riflessa dal divano e raggiunge il nostro punto di vista.
Il fotone B segue un percorso più complesso prima di arrivare all'occhio: dapprima viene riflesso dallo specchio, poi colpisce il divano e infine raggiunge il piano di vista.
Il fotone D, al contrario, colpisce prima il divano e poi lo specchio, per essere poi riflesso verso l'osservatore; il fotone E segue un percorso simile a questo, con la differenza che non raggiunge l'occhio e quindi non contribuisce alla resa dell'immagine.
In generale, i fotoni possono seguire percorsi molto più complicati, lasciando la sorgente di luce e "rimbalzando" per tutta la scena; ad ogni interazione con un oggetto però essi diventano sempre più tenui, meno luminosi, perciò dopo qualche "rimbalzo" il raggio è così soffuso che non si riesce a vedere.
Seguendo il percorso dei fotoni(e quindi dei raggi) abbiamo effettivamente svolto del ray tracing: più specificatamente abbiamo fatto del *forward path tracing*, in quanto abbiamo seguito i raggi a partire dalla sorgente fino all'interno della scena, tracciando il loro percorso in avanti proprio come i fotoni stessi si sarebbero spostati.

3.2 Backward path tracing

La tecnica del forward ray tracing descritta sopra è un'approssimazione del funzionamento del mondo reale ed in linea teorica è un buon metodo per realizzare immagini; c'è però un problema con una simulazione così diretta, che è il tempo necessario per generarla.
Una sorgente di luce genera milioni e milioni di fotoni ogni secondo, ognuno di questi con una frequenza e una direzione diverse; molti di questi colpirebbero oggetti che non vedremmo, neanche indirettamente, oppure sempli-

cemente uscirebbero dalla scena, ad esempio attraverso una finestra, oppure sarebbero così deboli da non contribuire in ogni caso. Se dovessimo davvero rappresentare una scena seguendo effettivamente ogni fotone dalla sorgente scopriremmo che il numero di essi che raggiunge l'occhio con un'intensità accettabile sarebbe esiguo, e occorrerebbe una grande quantità di tempo anche solo per rappresentare una scena poco illuminata.
Il forward ray tracing è indubbiamente un buon metodo e assolutamente preciso, ma il problema è il grande numero di fotoni che partono dalla sorgente e non giocano alcun ruolo nella resa dell'immagine: computazionalmente è un grande costo seguire quei raggi considerati inutili.
La chiave per risolvere questa eccessiva computazione sta nell'invertire il problema seguendo i raggi all'indietro anziché in avanti, chiedendoci quali fotoni sicuramente contribuiranno all'immagine: essi saranno quelli che colpiscono il piano di visione e poi arrivano al nostro occhio.
Prima di arrivare all'osservatore, i fotoni avranno seguito un determinato percorso generando un raggio; da quale oggetto però proviene un certo fotone? Se estendiamo il raggio all'interno della scena possiamo cercare l'oggetto più vicino che si trova sul percorso del fotone, potendo così affermare che esso proviene da quell'oggetto.
In questo tipo di approccio stiamo seguendo i raggi all'indietro, dall'occhio dell'osservatore alla sorgente: ciò ci permette di restringere l'attenzione soltanto verso quei raggi che sappiamo essere utili alla resa dell'immagine. Si può dire che stiamo cercando il primo oggetto colpito dal raggio nel suo percorso all'indietro, anche se in realtà ciò che si intende è la ricerca dell'oggetto che potrebbe aver emesso il fotone.
Poiché il nostro obiettivo è calcolare il colore dei singoli pixel (e quindi la loro luminosità), possiamo tracciare uno o più raggi che attraversano ogni pixel dell'immagine partendo dall'osservatore; questi raggi saranno chiamati *raggi primari*.

Ogni raggio può essere espresso nella forma

$$r(x, \vec{w}) = x + d \cdot \vec{w} \tag{3.1}$$

dove x è l'origine del raggio, $\vec{w}$ è la sua direzione e d è la distanza percorsa dal raggio. A questo punto dobbiamo trovare l'oggetto più vicino (con d minore) intersecato dal raggio, ovvero l'oggetto che viene potenzialmente visto attraverso il pixel.
Dato il punto di intersezione x dobbiamo trovare la sua luminosità in direzione del raggio r; per fare ciò dobbiamo conoscere la normale n della superficie nel punto x e la funzione di distribuzione bidirezionale della riflettanza (BRDF). Quest'ultima è una funzione che descrive come la luce viene riflessa da una superficie opaca: data la direzione della luce in ingresso, la normale alla superficie e la direzione di uscita, essa ci ritorna la quantità di luce lungo la direzione uscente. La quantità di luce riflessa L_r dovuta a una sorgente di luce puntiforme che ha intensità ϕ_l e si trova in posizione p può essere calcolata in questo modo:

$$L_r(x, \vec{w}) = f_r(x, \vec{w}, \vec{w'}) \frac{\vec{w'} \cdot \vec{n}}{\|p - x\|^2} V(x, p) \frac{\Phi_l}{4\pi} \tag{3.2}$$

dove $\vec{w'} = \frac{(p-x)}{\|p-x\|}$ è il vettore unitario in direzione della sorgente di luce.
La funzione di visibilità V viene valutata tracciando un *raggio ombra* da x verso la luce; se il raggio ombra interseca un oggetto tra x e la sorgente, allora V =0 e l'oggetto è in ombra, altrimenti V=1.
Il backward path tracing effettua una valutazione probabilistica di tutti i possibili percorsi della luce: in particolare, per ogni pixel vengono tracciati dei raggi (chiamati *sample rays*) in modo stocastico lungo tutti i possibili percorsi che potrebbero seguire. Ovviamente, un numero maggiore di samples porta a una valutazione migliore dell'integrale per il calcolo dell'equazione del rendering.
Il problema di questa metodologia è il disturbo generato nell'immagine: se viene scelto un numero troppo basso di samples, il risultato sarà un'imma-

gine disturbata (avremo infatti pixel molto luminosi che causeranno un gran numero di punti bianchi); per questo motivo in situazioni che presentano un'illuminazione complessa (ad esempio un modello architettonico) saranno necessari un gran numero di samples.
Il path tracing è in grado di fornire ottimi risultati con un numero basso di samples in quelle situazioni in cui l'illuminazione varia lentamente, ad esempio in una scena all'aperto in cui il cielo non presenta eccessive variazioni di colore; in questo caso non sono necessari troppi raggi di valutazione, in quanto la funzione che vogliamo integrare presenta una bassa variazione, perciò anche pochi samples possono darci una buona stima dell'integrale.
A causa dell'eccessivo costo del forward path tracing e quindi del suo ormai scarso utilizzo, il termine path tracing ha assunto il significato esclusivo di backward path tracing.

3.3 Bidirectional path tracing

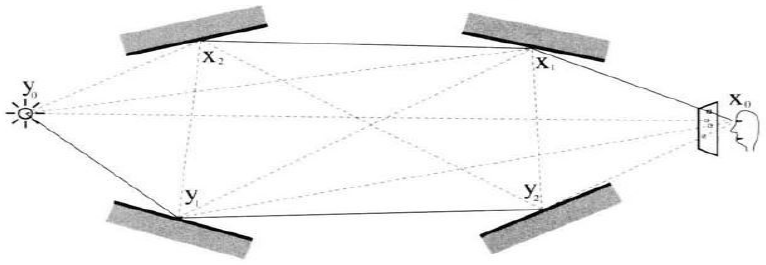
Come spiegato sopra, esistono due metodologie per la valutazione dell'integrale in un punto:

- *Sparare raggi* dalle sorgenti di luce e generare cammini all'interno della scena, smettendo di seguire il singolo percorso dopo un certo numero di rimbalzi ; la luce risultante viene utilizzata per definire il singolo pixel dell'output.

- *Raccogliere* i raggi da un punto sulla superficie. Il singolo raggio viene proiettato dalla superficie nella scena, seguendo un percorso che rimbalza sui vari oggetti e che termina quando viene intersecata una sorgente di luce; la luce è quindi inviata indietro lungo il percorso creato, fino al pixel. Generare un singolo path significa creare un "sample", ovvero un campione; per un singolo pixel sulla superficie vengono generalmente generati 800 campioni (fino ad arrivare a 3000).

Il bidirectional path tracing combina questi due approcci per ottenere una convergenza dell'integrale più veloce: l'idea è quella di considerare che i raggi vengano "sparati" simultaneamente sia dalla sorgente di luce che stiamo considerando che dal punto di vista.
Questa metodologia sfrutta il fatto che alcuni percorsi possano essere valutati in modo migliore partendo dall'occhio dell'osservatore mentre altri vengano valutati più facilmente se si parte dalla sorgente di luce. Un particolare esempio sono le caustiche: se si volessero rappresentare utilizzando il path tracing tradizionale occorrerebbe tracciare dei raggi di riflessione random tali che andrebbero incontro a un gran numero di rimbalzi speculari prima di colpire la sorgente di luce; partendo però dalla luce il problema diventa più semplice in quanto il raggio deve semplicemente essere riflesso dall'oggetto, colpire la superficie con componente diffusiva e raggiungere il piano di vista.
Il bidirectional path tracing inizia tracciando due percorsi attraverso la scena: uno che parte dalla sorgente luminosa e uno che parte invece dall'occhio dell'osservatore. Il passo successivo consiste nel combinare le informazioni dei due percorsi e ciò si realizza connettendo tutti i vertici del percorso, dove per vertice del percorso si intende il punto di intersezione lungo quel path, inclusi i punti finali (la sorgente di luce e l'occhio). Se chiamiamo y il percorso seguito dalla luce e x il percorso seguito dai raggi che partono dal punto di vista avremo che x_i saranno i vertici del percorso che parte dall'occhio, mentre y_j sono i vertici sul percorso della luce. Per ogni coppia di vertici x_i y_j viene calcolata la funzione di visibilità V tramite un raggio ombra.

Figura 3.2:



La stima finale per ogni pixel è calcolata come somma delle medie pesate delle intensità delle diverse combinazioni dei percorsi.
Utilizzando questo metodo è possibile calcolare anche i contributi delle sorgenti luminose secondarie.
L'algoritmo lavora producendo risultati migliori in situazioni dove la luce indiretta è importante (scene di interni), questo perché gli eventuali disturbi di luce che avremmo col path tracing classico dovuti al numero di superfici ad alta riflessività e/o diffusività vengono ridotti, in quanto i path della luce incontrano quelli dell'occhio e producono quindi risultati più attendibili e con un disturbo minore.

3.4 Photon Mapping

Il photon mapping è una metodologia biased che si basa sullimmagazzinare informazioni sull'immagine e riusarle ogni volta sia possibile.
Esso è una buona alternativa alle tecniche basate puramente sul metodo Monte Carlo, in quanto queste necessitano un grande numero di campioni per convergere al risultato ottimo.
È un algoritmo "a due passi" che separa la soluzione trovata per l'illuminazione dalla geometria; le informazioni luminose vengono rappresentate in una struttura dati chiamata "photon map", e questa separazione risulta essere decisamente utile in quanto i diversi termini dell'equazione 1.1 possono essere calcolati separatamente e salvati in differenti mappe.
Durante il primo passo si costruisce la mappa di fotoni tramite il photon tracing nel quale i fotoni vengono emessi dalla sorgente di luce e tracciati lungo tutta la scena: quando un fotone interseca un oggetto viene salvato nella photon map; il secondo passo è il *rendering*, dove si utilizzano le informazioni della photon map creata al primo passo.
Esso è un algoritmo molto flessibile in quanto le diverse parti dell'equazione di rendering possono essere calcolate utilizzando altre tecniche; inoltre è il

principale algoritmo utilizzato per la resa degli effetti caustici della luce.

Primo passo: photon tracing La prima fase consiste nell'emettere fotoni in numero discreto dalla sorgente di luce e nel tracciarli poi nella scena; i fotoni vengono emessi secondo la distribuzione di potenza della luce: per una sorgente puntiforme essa sarà uniforme in ogni direzione, mentre per una sorgente ad area si deve scegliere un punto random sulla superficie, considerare l'emisfera centrata in questo punto e infine scegliere una direzione di emissione sull'emisfera.
In realtà la strategia di emissione può essere scelta in base alla necessità, in particolare considerando la sparsità di una scena: se un ambiente è molto "sparso", ovvero ci sono pochi oggetti e lontani tra loro, è conveniente concentrare l'emissione dei fotoni verso le geometrie che ci interessano davvero, altrimenti la maggior parte dei fotoni verrà sparata inutilmente, causando un inutile aumento di tempo di calcolo.
La possibilità di pilotare il tipo di emissione si riflette sull'ottima riuscita da parte del metodo della rappresentazione di effetti caustici, in quanto è appunto possibile concentrare la luce (e quindi più fotoni) in un punto specifico.
Si segue il percorso dei fotoni sulla scena e, qualora il singolo fotone intersechi un oggetto di materiale con componente diffusiva, esso viene memorizzato nella photon map: in particolare, in questa vengono memorizzati il punto di intersezione, la direzione di provenienza del fotone e la sua intensità.
Esso può essere o assorbito o diffuso (a seconda del materiale dell'oggetto): per determinarne il comportamento si utilizza un metodo probabilistico chiamato *Russian Roulette*; la probabilità è ovviamente influenzata dal materiale sul quale si sta lavorando.
Questo metodo è molto utile per rimuovere fotoni "non importanti" e concentrarsi su quelli che davvero forniscono un contributo all'immagine: in sostanza, si utilizza una sorta di campionatura probabilistica per evitare di calcolare parti poco incisive per la resa finale.
Immaginiamo di avere un materiale con una certà riflessività d e un fotone in

entrata che ha intensità ϕ_p . Utilizziamo la Russian Roulette per decidere se questo fotone verrà assorbito o sarà riflesso; lo pseudocodice seguente illustra la procedura:

p;

ϵ = *random();*

$\epsilon \in [0\text{-}1]$ è un numero random uniformemente distribuito

if ($\epsilon < p$)

reflect photon with power ϕ_p

else

photon is absorbed

In generale, il nostro algoritmo probabilistico si basa sulla riflettività del materiale della superficie: se possiede un valore pari a 0.5 significherà che il 50% dei fotoni verranno riflessi a piena intensità, mentre l'altro 50% verrà assorbito. È bene sottolineare che se il fotone incontra una superficie perfettamente speculare questo non viene memorizzato nella mappa, ciò perché questo tipo di illuminazione può essere computata più efficientemente tramite puro ray tracing durante il passo di rendering. La mappa che utilizziamo per lo store dei fotoni deve essere fatta in modo tale da poter permettere una ricerca veloce delle informazioni durante il rendering, fase durante la quale verrà interrogata milioni di volte. La struttura dati migliore per la memorizzazione della mappa è un kd-tree, ovvero un albero binario in cui ogni nodo è un punto k-dimensionale.

Secondo passo: Rendering In questa fase si procede alla realizzazione dell'immagine finale calcolando la luminosità di ogni pixel.
Per migliorare l'efficienza del metodo, l'equazione del rendering viene studiata analizzando separatamente quattro fattori: illuminazione diretta, riflessione speculare, caustiche e illuminazione indiretta.
In particolare, come già detto, la riflessione speculare viene calcolata nella

maggior parte dei casi utilizzando procedure di puro ray tracing, dal momento che riescono a rappresentare bene le riflessioni.
I passi che l'algoritmo utilizza per calcolare la luminosità dei pixel sono i seguenti:

- Dato il punto di intersezione in esame, trova gli N fotoni più vicini servendosi delle informazioni contenute nella photon map
- Considera S la sfera che contiene questi N fotoni
- Per ogni fotone divide la quantità di flusso (intensità) del singolo per l'area S
- La somma dei risultati di ogni fotone considerato rappresenta la luminosità totale dell'intersezione che stiamo considerando, nella direzione in cui il raggio di luce l'ha colpita.

Conoscendo ora il funzionamento dell'algoritmo è facile comprendere come possa essere realizzato al meglio l'effetto delle caustiche dal momento che si può direttamente visualizzare la photon map, poiché siamo in grado di analizzare le caratteristiche degli N fotoni più vicini al punto che stiamo considerando e utilizzarle per la costruzione dell'immagine. Per la resa delle caustiche viene utilizzata una photon map dedicata per evitare di sovraccaricare la mappa primaria, in quanto sarebbero necessarie molte più informazioni in una singola mappa per riprodurre effetti caustici accurati.

Gli algoritmi presentati sono, a parte il photon mapping, tutti classificabili come unbiased.
Ora che si è studiato il loro funzionamento, assieme alle possibilità che offrono, si è in grado di passare all'analisi dei singoli motori di rendering presi in esame, per scoprirne le funzionalità e il modo migliore per sfruttarli al massimo.

Capitolo 4

Motori di rendering a confronto

Un motore di rendering è un programma che interagisce con un'applicazione host 3D per fornire più potenzialità e possibilità a quest'ultimo.
YafaRay, LuxRender, RenderMan e Cycles utilizzano Blender come applicazione host, con la differenza che l'ultimo dei citati è un addon abilitato di default nel software mentre gli altri tre devono essere scaricati e successivamente abilitati nelle user preferences.
Sia YafaRay che LuxRender che RenderMan possono essere utilizzati come addons anche in altre applicazioni di modellazione tridimensionale (RenderMan, in particolare, è stato ideato principalmente per il software "Maya").

Cycles, YafaRay, Lux render e RenderMan: una prima classificazione

Tutti e quattro questi motori di rendering si propongono di risolvere l'equazione di illuminazione, che descrive come ogni punto dell'oggetto che si sta analizzando sia illuminato in funzione della sua posizione nello spazio; inoltre, ognuno di questi utilizza il metodo di Monte Carlo per la risoluzione dell'integrale dell'equazione.
Esistono però molte differenze tra di essi, in particolare per l'algoritmo utilizzato per risolvere l'equazione.

Abbiamo infatti che:

- Cycles utilizza il backward path tracing
- Luxrender è in grado di utilizzare il backward path tracing, il bidirectional path tracing e il photon mapping (si predilige l'utilizzo del path tracing bidirezionale)
- YafaRay utilizza il photon mapping
- RenderMan utilizza il backward path tracing

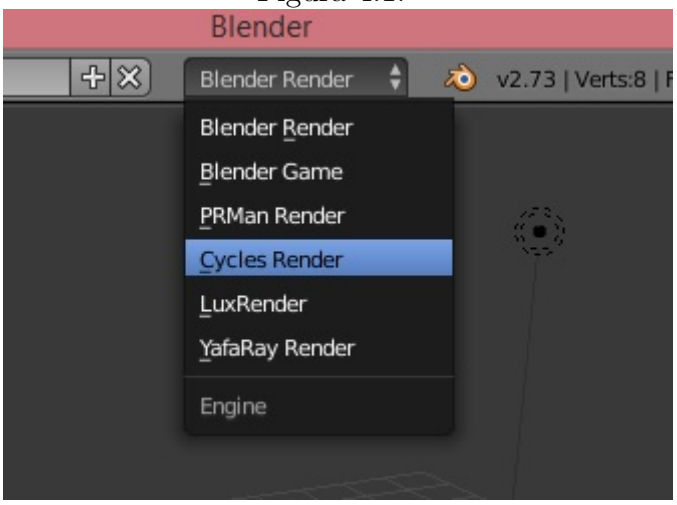
Per quanto riguarda i motori compresi in questo studio abbiamo che Cycles, LuxRender e RenderMan sono unbiased, mentre YafaRay è un motore biased (eventualmente lo è anche LuxRender se lo si considera nella sua versione con photon mapping).
Tutti e quattro i motori sono ray tracers, poiché gli algoritmi utilizzati fanno parte della famiglia del ray tracing; le diverse metodologie implementate fanno sì che ognuno di questi realizzi in modo differente la scena finale, favorendo determinati effetti di resa e richiedendo più o meno tempo per concludere il lavoro.

4.1 Cycles

4.1.1 Introduzione

Cycles è un motore di rendering unbiased incorporato di default in Blender a partire dalla versione 2.6 di quest'ultimo; è un programma che si focalizza sull'interattività e sulla facilità d'uso, offrendo all'utente un'elevata quantità di strumenti di lavoro.
Per poterlo usare basta semplicemente selezionarlo come rendering engine attivo dal menù che si trova nell'header dell'interfaccia (figura 4.1).

Figura 4.1:



Cycles, nel caso l'hardware lo consenta, è in grado di utilizzare la GPU per renderizzare: esso infatti può effettuare il rendering in modalità CUDA, sfruttando le schede grafiche NVIDIA.

4.1.2 Pannello di rendering

Questo motore offre un pannello di impostazioni molto vasto per quanto riguarda il rendering: si possono infatti personalizzare numerose opzioni in base a ciò che vogliamo ottenere.
È possibile ad esempio scegliere il numero di samples per il rendering finale (numero di samples per ogni pixel); ovviamente, più samples vengono utilizzati, minore sarà il disturbo nell'immagine. Cycles offre la possibilità di scegliere il numero di samples da utilizzare anche per la preview, ovvero per l'anteprima di rendering visualizzata nella viewport. Soltanto in cycles è possibile usufruire della preview.
Un'altra opzione personalizzabile è il numero di "rimbalzi" da far effettuare ai raggi luminosi nella scena; in particolare, è possibile determinare:

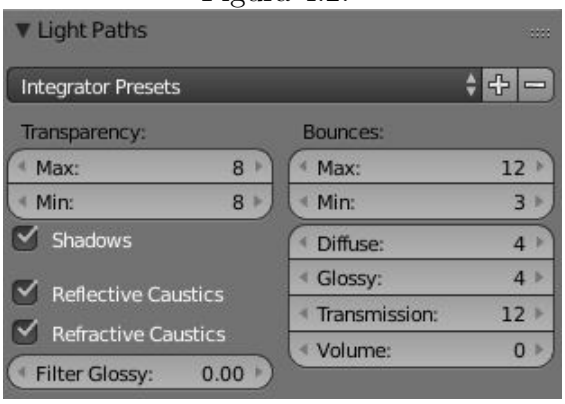
- Rimbalzi di luce diretta, impostando i valori di minimo e massimo per il numero di bounces; per ottenere una qualità migliore, il valore di

max bounces dovrebbe essere settato al massimo inseribile (1024), ma per avere rendering più veloci solitamente si diminuisce.

- Rimbalzi di luce diffusa
- Rimbalzi di luce riflessa
- Rimbalzi di luce trasmessa/rifratta

Cycles offre inoltre all'utente la possibilità di scegliere se renderizzare o meno le caustiche: il path tracing in linea di principio è in grado di rappresentarle (con un numero sufficientemente alto di samples), ma nella pratica ciò porterebbe sia a un aumento potenzialmente inutile dei tempi di rendering sia a un disturbo troppo elevato; viene lasciata perciò all'utente la libertà di abilitare o disabilitare questa opzione.

Figura 4.2:



Per quanto riguarda gli oggetti, considerandoli dal punto di vista del calcolo dei percorsi luminosi, è possibile impostare a quali tipi di raggi possono essere visibili o meno:

- Camera
- Componente diffusiva
- Componente riflettiva

- Raggi di trasmissione
- Ombre

Utilizzando questa impostazione è possibile, ad esempio, far sì che un oggetto che emette luce si invisibile alla camera (una sorgente di luce è di default invisibile).

4.1.3 Editor dei nodi

La principale caratteristica di Cycles è l'utilizzo dei cosiddetti "nodi" per settare e definire le proprietà dei materiali, delle luci e del background della scena. Si può pensare ad essi come a nodi di un albero, il quale determina l'output dell'oggetto sul quale stiamo lavorando; ogni nodo trasforma un certo input che gli viene dato in output, e i nodi sono tra loro connessi tramite socket. Essi sono un'espressione visiva di operazioni matematiche e possono essere considerati in tutto e per tutto come un linguaggio visuale di programmazione.
Come già detto, la definizione di ogni proprietà della scena viene decisa tramite la composizione di nodi; in realtà il loro utilizzo non è obbligatorio, ma in tal caso ci si priverebbe quasi totalmente dell'abilità espressiva del motore. Per accedere all'editor dei nodi occore cambiare il tipo di editor a "Node editor" dal menù a tendina in basso a sinistra.

Si possono indivuare due principali classi di nodi: shaders e texture.

Shaders

Questa classe di nodi lavora sull'aspetto della superficie di un oggetto, studiando e definendo il comportamento fisico del materiale di cui è fatto (quindi si avrà, ad esempio, uno shader per un particolare tipo di metallo,

uno per il vetro, e così via). All'interno di questa classe possiamo individuare quattro tipologie di shader disponibili tra i nodi:

- BSDF che descrivono la riflessione della luce, la rifrazione e l'assorbimento sulla superficie di un oggetto
- Emission che descrivono l'emissione della luce da una superficie
- Volume che descrive la dispersione della luce dentro un volume
- Background che descrive l'emissione della luce dall'ambiente

Ogni nodo shader ha come input un colore e fornisce uno shader come output; tutti gli output possono essere combinati tra di loro tramite particolari nodi quali Mix e Add shaders.
Il risultato finale viene quindi utilizzato dal motore di rendering per computare tutte le interazioni della luce.

Textures

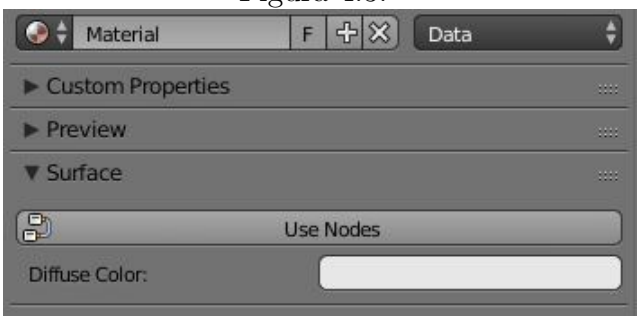
Cycles mette a disposizione diversi tipi di texture, fornendone sia di preimpostate che permettendo all'utente di utilizzare un'immagine scelta.
Ogni tipologia di texture (es: Image texture, Environment texture, Noise texture) corrisponde a un singolo nodo che accetta parametri in ingresso (diversi a seconda del tipo di texture) e restituisce un colore o un valore.
Per lo UV mapping e il texture painting occorre utilizzare un nodo di tipo "Image texture".

Oltre a quelli che rientrano in queste due classi, esistono nodi per definire informazioni geometriche, modificare le coordinate delle texture e gestire e modificare la colorazione della mesh su cui si sta lavorando.

4.1.4 Utilizzo dei nodi

Per iniziare a lavorare con i nodi e definire un materiale tramite essi è necessario, dopo aver scelto l'oggetto a cui si vuole assegnare il nuovo materiale, scegliere di utilizzare i nodi cliccando sul bottone "use nodes".

Figura 4.3:



A questo punto si può iniziare con il lavoro.
Aprendo l'editor noteremo che sono già presenti due nodi: material output e diffuse bsdf.
Il nodo "material output" è necessario in quanto informa che il materiale dell'oggetto su cui si sta lavorando è stato definito tramite i nodi; in sostanza, quindi, che il nostro output deriva da tutto l'albero di nodi che andremo a costruire.
Il nodo diffuse bsdf determina invece soltanto il tipo di materiale di cui è fatta la nostra mesh e quindi il tipo di interazione con la luce; è il nodo che per default viene aggiunto quando si sceglie di lavorare coi nodi, e descrive semplicemente una superficie perfettamente diffusiva. Questo nodo ci permette di selezionare il colore da applicare alla mesh e settare un valore per la roughness.
Esistono diversi materiali di base che Cycles offre, e sono i seguenti:

- Diffuse che identifica una superficie diffusiva
- Translucent che identifica una superficie traslucida, ovvero che trasmette la luce incidente su essa in modo diffuso
- Glossy che descrive una superficie lucida; è usato per metalli e specchi
- Anisotropic definisce una superficie lucida ma fornendo un controllo della riflessione; in particolare, un materiale anisotropico modifica il

suo aspetto (il modo in cui riflette) a seconda di come è ruotato, questo perché è dipendente dalla direzione

- Toon permette di creare una superficie con un effetto luminoso tipico di un cartone animato
- Transparent rappresenta una superficie con trasmissione perfetta (senza rifrazione) della luce, che la attarversa come se non ci fosse alcuna geometria
- Glass per definire una superficie di vetro. Questo shader tende a produrre disturbi a causa delle caustiche che genera. Modificando i valori dell'IOR (Indice di rifrazione) e della roughness (la ruvidità) si possono definire tutti i possibili tipi di vetro (dai più "limpidi" ai più disturbati)
- Velvet rappresenta il velluto
- Subsurface scattering definisce un meccanismo di trasporto della luce nel quale essa penetra la superficie di un oggetto, interagisce con esso e i singoli raggi vengono "sparpagliati" per poi uscire in una direzione differente rispetto a quella di entrata. In generale la luce, una volta entrata sotto la superficie dell'oggetto, verrà riflessa per un certo numero di volte seguendo angolazioni irregolari all'interno del materiale e solo infine tornerà fuori, con un angolo diverso rispetto a quello che avrebbe avuto se fosse stata riflessa direttamente fuori dalla superficie. È uno shader molto particolare che serve per il corretto di rendering di materiali quali la cera, la pelle e il marmo
- Emission definisce una superficie sorgente di luce.

Al posto del nostro diffuse iniziale potremmo aver avuto qualsiasi altro dei nodi semplici elencati sopra: la libertà espressiva sarebbe comunque rimasta limitata in quanto, di per sè, il singolo nodo di tipo shader non fornisce molte opzioni; la vera potenza risiede nella possibilità di combinare i diversi tipi di nodi per creare qualsiasi tipo di materiale. Vediamo un piccolo esempio:

Figura 4.4:

In questo caso abbiamo utilizzato un nodo di tipo "Diffuse BSDF" e uno "Transparent BSDF", combinati tra loro tramite un nodo "Mix Shader" che, come suggerisce il nome, permette di combinare due shader e ottenere un particolare output.
Questo nodo consente anche di definire quanto ciascuno dei due shader influisca sull'output finale: ciò è possibile tramite il controllo del "Fac" che assume valori compresi tra 0 e 1; in particolare, 1 rappresenta il 100% di influenza del primo materiale (quello collegato alla prima delle due socket per gli shader), mentre lo 0 ne rappresenta l'assenza e quindi il 100% di influenza del secondo materiale (collegato alla seconda socket).
Di default il Fac ha un valore di 0.5, perciò il 50% del materiale sarà influenzato dal primo shader e altrettanto dal secondo.
Oltre al nodo "Material Output" esistono anche i nodi "Lamp Output" e "World Output": essi svolgono lo stesso ruolo del nodo per l'output del materiale con la differenza che si occupano rispettivamente delle informazioni sulla sorgente luminosa e di quelle relative al background.
Dopo aver assegnato un materiale al nostro oggetto, si potrebbe applicare una texture (in particolare, prendendo in esame l'esempio fornito prima, potremmo decidere di applicarla o alla componente diffusa o a quella glossy, ottenendo effetti diversi) per rivestirne la superficie. Cycles offre diversi tipi di texture:

- Noise texture genera un disturbo simile a delle nuvole sul materiale

dell'oggetto

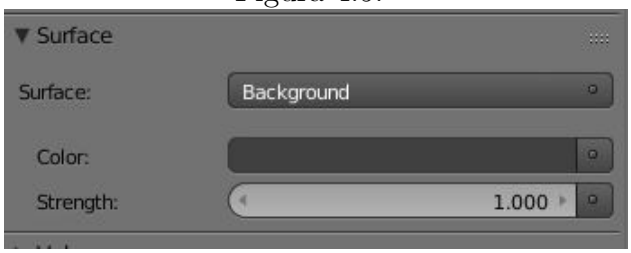
- Wave texture genera delle righe (o degli anelli) che ricoprono il materiale.
- Voronoi texture genera delle celle secondo la tassellatura di Voronoi
- Musgrave texture genera un disturbo che può essere utilizzato, ad esempio, per rendere al meglio materiali organici
- Gradient texture genera una sfumatura, un gradiente, sul materiale
- Magic texture genera un colore psichedelico
- Checker texture fornisce una quadrettatura del materiale
- Brick texture suddivide la superficie del materiale in mattoncini
- Image texture ricopre la superficie del materiale con un'immagine scelta dall'utente. È possibile selezionare qualsiasi tipo di immagine, scalarla e posizionarla sull'oggetto; ogni mesh a cui si voglia applicare l'immagine deve prima essere "unwrappata" affinché la texture possa essere mappata sull'oggetto e quindi correttamente applicata.

4.1.5 Background

Per quanto riguarda la gestione del background, e quindi del "mondo" in cui la scena è immersa, esiste il tab apposito dove è possibile lavorare con la superficie ambientale. Il primo passo da fare è scegliere come superficie il tipo "Background" dal menù a tendina:

Questa opzione viene scelta di default, ed è l'unica che consente di ottenere effetti per il background; qualsiasi altra scelta non produrrebbe alcun risultato.
Di default viene anche impostato un colore di sfondo, ma è possibile scegliere altri tipi di effetti semplicemente cliccando sul bottone a fianco al colore.

Figura 4.5:



Otterremo a questo punto un menù che ci consentirà di scegliere tra varie texture; oltre ai tipi già presentati, ne esistono altri due, particolarmente adatti per realizzare il background: Environment Texture e Sky Texture. La prima richiede un'immagine che sarà utilizzata come mappa d'ambiente, e auspicabilmente dovrà essere in formato latlong (latitudine-longitudine); la seconda invece crea un cielo di cui si può specificare la direzione del sole, la torbidità del cielo e l'albedo del terreno (ovvero la percentuale di luce incidente sul terreno che sarà riflessa da esso).
Per ogni tipo di texture scelto è possibile modificarne l'intensità (tramite lo slider "Strength") e ottenere più o meno luminosità.

Tutti i nodi di tipo texture possono ricevere l'input da un nodo di tipo Mapping che serve per trasformare coordinate, e risulta essere quindi molto utile per modificare l'aspetto della texture sull'oggetto.
Oltre ai nodi facenti parte delle classi Shader e Texture, ne esistono molti altri che consentono di usufruire di un'ampia gamma di funzionalità e che sono in grado, collegandoli ad altri nodi in maniera più o meno complessa, di generare svariati effetti, anche ricercati.
Essendo le combinazioni dei nodi e i conseguenti effetti ottenibili in numero potenzialmente infinito (la grandezza dell'albero non ha limiti in termini di nodi utilizzabili) è impossibile riuscire a trattare ogni possibile effetto; alcuni motori di rendering, come YafaRay, sebbene offrano molta libertà, rimangono comunque limitati dai percorsi pre-impostati, ed è quindi ragionevole pensare di presentarli nella loro interezza.

Poiché Cycles è uno dei motori più utilizzati e apprezzati, sul web è possibile trovare qualsiasi tipo di tutorial, sia riguardo funzionalità e utilizzo generale del motore, sia sulla realizzazione di specifici oggetti; per quanto riguarda lo studio di Cycles e la scoperta delle sue funzionalità, essi si sono rivelati il miglior strumento di apprendimento.

4.1.6 Considerazioni su Cycles

Cycles risulta essere un motore con un gran numero di funzionalità che riesce a soddisfare sia richieste semplici, garantendo facilità di utilizzo, sia lavori più articolati che richiedono uno studio e una complessità maggiore di realizzazione. È un engine che viene utilizzato sia da professionisti che da studenti alle prime armi, grazie alla sua grande flessibilità e al realismo che offre. Di per sé non richiede particolari risorse: è in gran parte la complessità della scena che si costruisce che influisce sulle capacità hardware richieste.
Ovviamente, un utilizzo massiccio di ad esempio image texture di grandi dimensioni e qualità (per esempio le immagini HDRI che contengono molte più informazioni rispetto alle semplici immagini .png o .jpg) peserà sulla fluidità del lavoro e sul tempo di rendering soprattutto per quanto riguarda la parte iniziale di raccolta delle informazioni della scena; la durata del rendering vero e proprio è determinata dal tipo di ambiente che si vuole rappresentare: ad esempio, la quantità di riflessioni tra gli oggetti incide molto sui tempi di rendering, e ovviamente un maggior numero di campioni richiesti per aumentare la qualità dell'output.

4.2 YafaRay

4.2.1 Introduzione

Il progetto YafRay (Yet Another Free Raytracer) nacque nel 2001 e la prima release pubblica fu nel 2002.

Su richiesta della community di Blender, YafRay venne aggiunto come plugin per Blender dalla release 2.34, nell'agosto del 2004; l'ultima versione di YafRay rilasciata fu la 0.0.9 nel 2006, dopodiché l'intero programma venne completamente riscritto.
YafaRay è il risultato di questa riscrittura "from scratch": il nome venne cambiato proprio per indicare che si trattava di un engine completamente nuovo. È free e open-source.

4.2.2 Installazione e configurazione

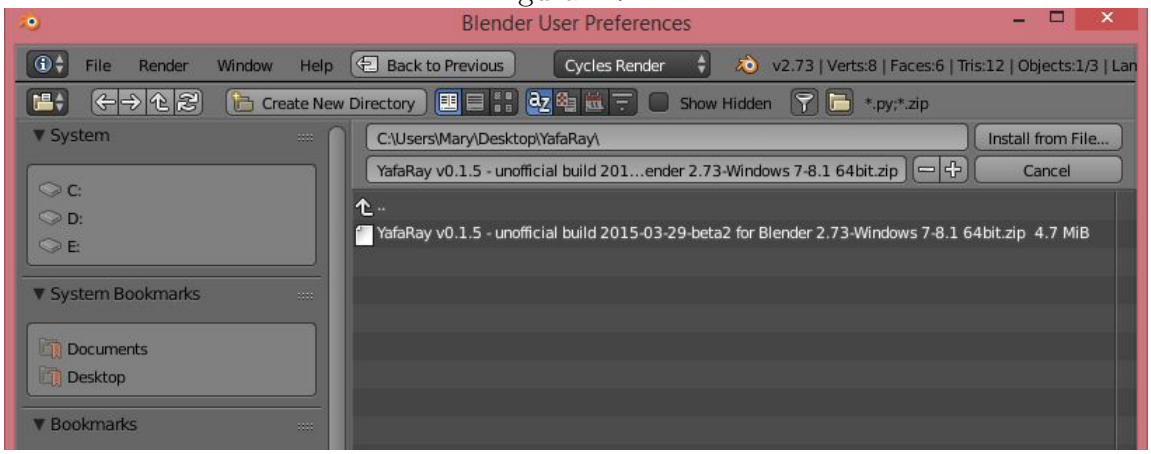
Installare YafaRay per poterlo utilizzare immediatamente come motore di rendering è molto semplice.
Innanzitutto ci si collega al link http://www.yafaray.org/download (sito ufficiale del motore YafaRay, sezione download) e da questa pagina si può selezionare la versione che si preferisce dell'engine e il sistema operativo che si possiede; al momento l'ultima versione stabile è la 0.1.5, disponibile per windows (a 64 e 32 bit), per Linux 64bit e per OSX.

Figura 4.6:

Una volta selezionata la versione basterà scaricarla come archivio zip; la locazione in cui si decide di salvare l'archivio è indifferente, ma va ricordata per l'aggiunta a Blender dell'add-on.
Per effettuare il setup di YafaRay occorre ora aprire Blender e navigare su File->User Preferences e selezionare il tab Addons. Da qui si clicca sul pulsante "Install from file" in basso a sinistra della finestra e si aprirà un File

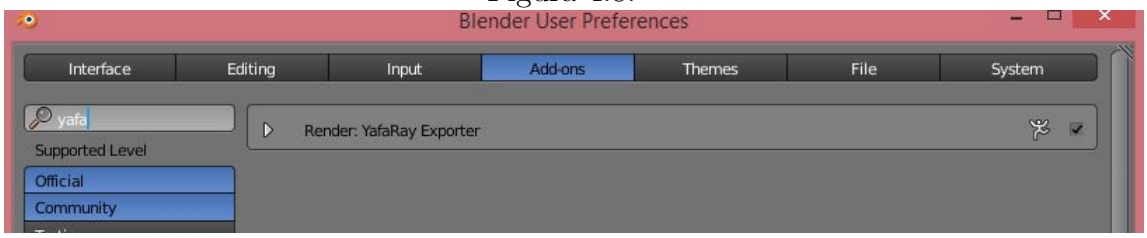
Explorer: a questo punto basta navigare nella cartella in cui si è salvato l'archivio di YafaRay, selezionarlo e cliccare su "Install Addon".

Figura 4.7:



Ora nella tab degli addons è comparsa "Render: YafaRay Exporter" come opzione selezionabile: si spunta il quadratino a destra della scritta e prima di chiudere la finestra delle User Settings si clicca su "Save User Settings".

Figura 4.8:



A questo punto, se non ci sono stati problemi, si può utilizzare YafaRay come engine di rendering. Per selezionarlo è sufficiente cliccare sul menù a tendina nell'header dell'interfaccia di Blender e sceglierlo come motore di rendering attivo; è ora possibile cominciare a lavorare con YafaRay.

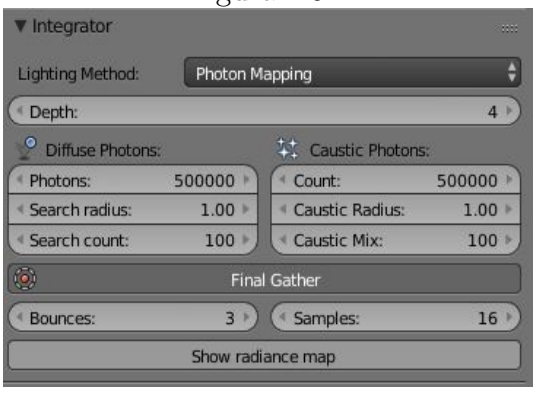
4.2.3 Pannello di rendering

Il pannello per le impostazioni di Rendering di YafaRay presenta notevoli differenze con quello di Cycles (e anche con quello del Renderer interno). Nella sezione "Integrator" è possibile selezionare la modalità di illuminazione (e quindi di rendering) che si vorrà utilizzare. È stato affermato nei capitoli precedenti che YafaRay utilizza il photon mapping; in realtà fornisce la possibilità di scegliere tra i seguenti tipi di metodi per l'illuminazione:

- Direct Lightning
- Path tracing
- Photon Mapping
- Bidirectional

Il direct lightning è il metodo più veloce per quanto riguarda il tempo di rendering, ma non riesce a rappresentare la componente indiretta della luce. YafaRay risulta essere più performante con l'utilizzo del Photon Mapping, algoritmo che è particolarmente indicato per scene di interni.

Figura 4.9:



Fotoni

I fotoni, come sappiamo, propagano energia luminosa: per questo motivo, più fotoni utilizziamo e più sarà accurata la stima della luce e la resa della scena; aumentare il numero di fotoni però causa un aumento del tempo necessario per la costruzione della photon map. Sceglierne in numero troppo basso invece porterà a una sorta di sfocamento ai bordi degli oggetti, in particolare in quei punti dove c'è un passaggio repentino tra luce e ombra (ad esempio negli angoli) e dove quindi c'è maggior bisogno di informazioni.
Nelle scene che non presentano caustiche, o comunque non in gran numero, è possibile utilizzare mappe con una densità bassa di fotoni (1000-2000).

Depth

Questa impostazione permette di controllare il numero di "rimbalzi" consecutivi da far effettuare ai fotoni, sia quelli che controllano le caustiche sia quelli che si occupano della componente diffusa; a seconda del tipo di fotoni, però, assume un diverso significato:

- Depth per fotoni delle caustiche: indica la quantità massima di eventi consecutivi di rifrazione e/o di riflessione per i fotoni delle caustiche prima di incontrare una superficie diffusiva
- Depth per fotoni diffusivi: quantità di rimbalzi sulle superfici diffusive; una profondità maggiore produrrà una photon map più densa e un color bleeding più preciso e studiato. Oltre una certa soglia l'aumento di questo valore produrrà effetti limitati in quanto molti fotoni verranno assorbiti dalle superfici diffusive

Poiché i rimbalzi più importanti sono i primi tre, in generale scegliere un valore compreso tra 6 e 8 sarà ottimo per qualsiasi tipo di scena.

Search radius e search count

L'obiettivo del photon mapping è quello di avere il maggior numero di fotoni colpiti entro il raggio minore.

"Search radius" e "search count" sono due parametri che vengono utilizzati per gestire la ricerca dei fotoni vicini al punto che si sta valutando, con l'obiettivo di avere il maggior numero di fotoni nella circonferenza. Il parametro "search count" determina il numero minimo di fotoni inscritti nella circonferenza di raggio "search radius": solitamente i valori settati sono rispettivamente un range di 50-350 e 1.00, ma ovviamente dipende dalla scena che si sta realizzando: se essa ha una densità di fotoni bassa non è consigliato utilizzare un valore basso per il raggio, in quanto molte aree non avranno alcun fotone. In generale il raggio dovrebbe essere inversamente proporzionale al numero di fotoni emessi.

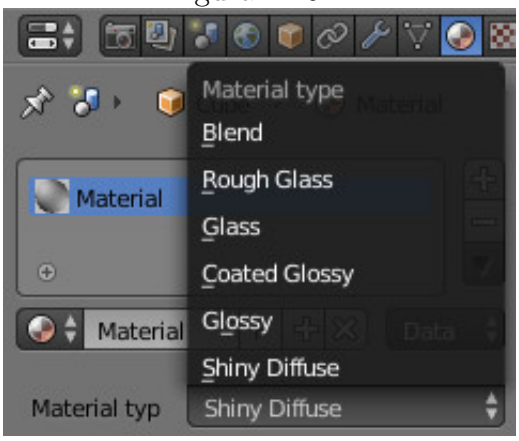
Final gather

Il final gather è una tecnica che serve per migliorare e completare la photon map e che, dopo il photon tracing, fornisce una migliore approssimazione della luminosità locale utilizzando diversi rimbalzi di luce. In sostanza, una volta che la photon map è stata costruita, viene sparato un raggio dal punto in cui è situata la camera; dal punto di intersezione del raggio con la superficie diffusiva vengono sparati dei raggi samples in modo random sulla superficie dell'emisfera centrata nel punto di intersezione per campionarla, ed essi a loro volta incontreranno altre superfici. Alla fine di questa operazione si ottiene la radianza calcolata per ognuno dei samples nell'intersezione con le superfici e si produce una media della luce indiretta che raggiunge il punto di intersezione considerato inizialmente.
Il numero di samples che si seleziona avrà un impatto direttamente proporzionale al tempo di rendering, mentre il numero di rimbalzi non influirà in modo significativo; inoltre il numero di bounces indicato sarà raggiunto soltanto per una frazione di samples sparati (alcuni possono essere assorbiti dalle superfici prima di effettuare il numero di rimbalzi indicato).

4.2.4 Materiali

YafaRay propone cinque diversi tipi di materiali generici, ognuno dei quali offre diverse possibilità per raggiungere effetti più o meno avanzati:

Figura 4.10:



Glass

Viene generato considerando rifrazione, riflessione e assorbimento della luce incidente; l'ambiente circostante ha un grande impatto sull'aspetto di questo materiale, e può produrre delle caustiche in presenza di determinate condizioni:

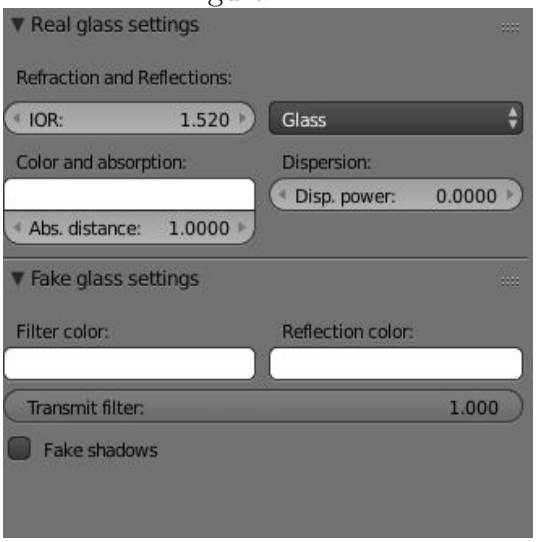
Viene generato considerando rifrazione, riflessione e assorbimento della luce incidente; l'ambiente circostante ha un grande impatto sull'aspetto di questo materiale, e può produrre delle caustiche in presenza di determinate condizioni:

- Esiste una sorgente di luce e/o un background abilitato a sparare raggi relativi alle caustiche
- Il materiale ha IOR > 1
- Si sta utilizzando un metodo di illuminazione globale per il rendering oppure è stato abilitato "Use caustics" in caso si stia usando il Direct Lightning

- I raggi delle caustiche hanno un valore di profondità abbastanza alto da passare attraverso tutte le superfici trasparenti

Questo tipo di materiale offre diverse impostazioni che permettono di ottenere risultati differenti:

Figura 4.11:



Absorption

Definisce la quantità di luce assorbita dal materiale ma anche il colore del vetro (e quindi delle caustiche); maggiore sarà il valore di questo parametro, più luce sarà assorbita. Se si imposta il colore bianco, l'assorbimento sarà nullo.

Reflection color

Determina il colore della luce riflessa; la quantità di questa dipende dall'IOR: più alto è l'indice, più il vetro sarà riflettente. La riflessione produce caustiche.

Refraction: IOR

La rifrazione è il cambiamento di direzione quando la luce passa da un mezzo con un certo indice di rifrazione ad un altro con indice differente. La rifrazione produce caustiche.
Di seguito gli IOR per alcuni materiali:

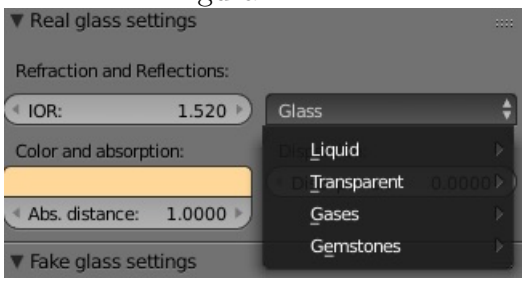
- Ghiaccio: 1.31
- Acqua: 1.33
- Vetro comune: 1.52
- Diamante: 2.52

Dispersion

Essa causa la separazione della luce bianca nelle sue componenti con diversa lunghezza d'onda. Aumentando il valore di questa impostazione, si produce un effetto più visibile.

Per il vetro YafaRay offre la possibilità di scegliere l'IOR da una lista molto fornita di materiali reali, evitando all'utente di dover effettuare continui test per raggiungere l'effetto desiderato. È possibile scegliere tra quattro principali categorie:

Figura 4.12:



Ognuna di esse propone una grandissima varietà di oggetti del mondo reale; alcuni esempi sono benzina, latte, shampoo, vodka per quanto riguarda la categoria dei liquidi; zucchero, sale, plexiglass, lenti a contatto per la categoria dei materiali trasparenti; elio, idrogeno, diossido di carbonio per i gas; infine, ogni varietà di gemme e pietre preziose.

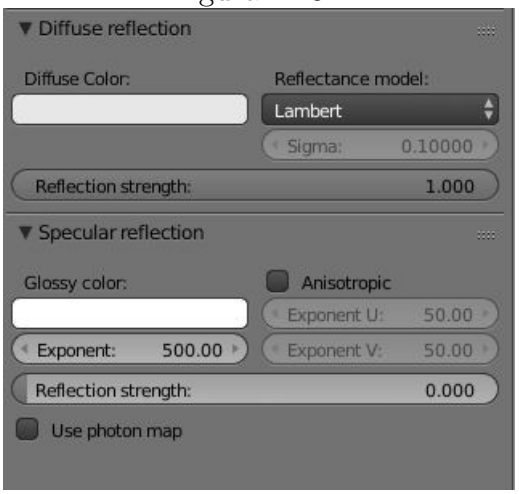
Rough Glass

È sostanzialmente il materiale glass che utilizza un modello glossy(lucido) per la componente di rifrazione. L'unica opzione aggiuntiva che si ha è quella riguardante l'esponente di asperità del materiale: maggiore sarà questo valore, maggiormente ruvida sarà la superficie.

Glossy

Il materiale glossy di YafaRay può essere utilizzato per superfici quali plastica, vernice di automobile, metallo raffinato, legno lavorato.

Figura 4.13:



Questo tipo di materiale, come si vede dall'immagine, possiede due colori da poter impostare: diffuso e glossy. Il parametro indicato come "Reflection strength" in entrambi i pannelli serve ad indicare quanta è l'intensità delle due componenti: più sarà la componente diffusa, meno sarà quella lucida

(ovvero la componente riflettiva).
Se impostiamo la glossy reflection a 0 (specular reflection) avremo che la maggior parte del colore che sarà visibile sull'oggetto sarà quello diffuso, con un piccolo bordo di colore lucido; al contrario, se questo valore sarà pari a 1, sarà visibile solo il colore glossy e l'effetto lucido del materiale. Il parametro Exponent controlla la sfocatura della riflessione del materiale: più alto sarà il valore dell'esponente, minore sarà la sfocatura e più "limpido" sarà l'oggetto. Solitamente per i metalli vengono utilizzati valori sopra il 200. Questo materiale è utilizzato anche per realizzare riflessioni anisotropiche, ovvero quelle che non sono uguali in tutte le direzioni. Se si abilita questo parametro, il valore dell'esponente di cui si è parlato sopra viene diviso nelle componenti verticale e orizzontale (U e V).

Coated glossy

Questo materiale rappresenta sostanzialmente una superficie laccata. Il pannello di impostazioni è del tutto uguale a quello del materiale glossy, con l'aggiunta di un sotto pannello in cui è possibile settare l'IOR della laccatura in superficie e il colore della sua riflessione. Un ottimo utilizzo di questo materiale può essere la resa della vernice delle auto.

Shiny diffuse

Questo tipo di materiale ha diverse applicazioni; può essere utile per:

- Materiali diffusivi senza alcuna componente speculare
- Riflessioni perfettamente speculari con o senza effetto Fresnel
- Superfici che emettono luce

Per quanto riguarda la componente diffusiva, le impostazioni rimangono le stesse che per i materiali precedenti, con la differenza che ora abbiamo anche una componente di emissione; il colore di questa è determinato dal "Diffuse color" e uno slider permette di controllare l'intensità di emissione

della luce da parte dell'oggetto.
Così come per il glossy, anche per questo materiale abbiamo due insensità che si influenzano: maggiore sarà quella della componente speculare, meno visibile sarà la componente diffusiva. Un'intensità pari a 1 per la prima comporterà un materiale perfettamente speculare.
È possibile abilitare l'opzione Fresnel: in questo modo si avrà che la quantità di riflessione speculare dipenderà dal punto di vista; in particolare significa che una superficie sarà più riflettiva negli angoli radenti alle superfici piuttosto che in quelli perpendicolari.
Le altre componenti riguardano la trasparenza e la traslucidità. I due termini descrivono due fenomeni simili ma che non vanno confusi: il primo si riferisce a un materiale trasparente, che lascia filtrare la luce; il secondo invece è un tipo di materiale che permette alla luce di passargli attraverso ma soltanto in modo diffuso: ciò significa che non è possibile guardarvi attraverso.
In entrambi i casi la luce viene filtrata dalla superficie, e viene colorata e affievolita a seconda delle proprietà del materiale; lo slider "Transmit filter" controlla l'intensità del color filtering: con valore pari a 0, non c'è alcun filtraggio di luce.

Blend material

Con questa feature è possibile unire due materiali già definiti dall'utente per crearne un terzo come mix di quei due: basta semplicemente creare un nuovo materiale, impostarlo di tipo "Blend" e nel pannello che viene mostrato scegliere i due materiali da mischiare.
Tramite questo materiale è possibile utilizzare e unire le proprietà di due diversi materiali, ad esempio unire una riflessione glossy con la trasparenza di un altro materiale.
Il "Blend Value" controlla la percentuale di ognuno dei due materiali nel materiale di output: un valore pari a 0.5 indica un uguale contributo da entrambi.

4.2.5 Background

Anche in YafaRay le impostazioni per il mondo possono essere accedute tramite la tab "World".
YafaRay offre cinque tipi di background.

Single color

La luce viene generata di un unico colore; solitamente questa opzione viene usata per dei test o delle scene semplici in cui l'obiettivo non è l'estetica. Spuntando "IBL" è possibile aumentare o diminuire l'intensità del colore.

Gradient

Questa opzione genera un background formato da quattro colori: due sopra l'orizzonte e due sotto. Ogni coppia di colori viene mischiata insieme in modo da generare una sfumatura, mentre i due colori vicini all'orizzonte non vengono uniti, così da definire una chiara linea d'orizzonte.
Utilizzare questo tipo di background è molto utile per determinare una sorta di finto terreno che estende il piano reale introdotto dall'utente fino alla linea d'orizzonte, generando l'effetto di un piano continuo e infinito.

Texture

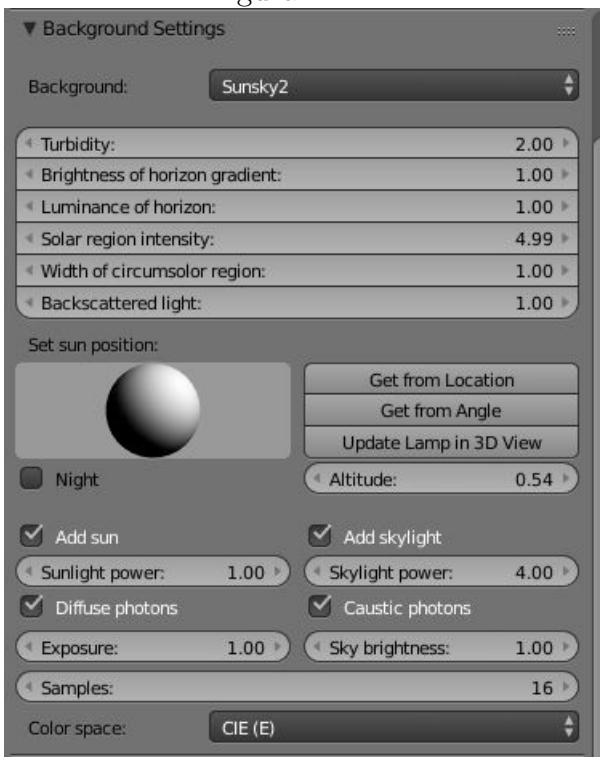
È possibile utilizzare una texture come background; questo significa che, una volta descritta una texture nell'apposita tab, potremo selezionarla come sfondo.
Ciò comporta che possiamo utilizzare un'immagine per il mondo, in particolare potremmo voler utilizzarne una di tipo HDR per fornire luce alla scena. Una HDRI (High Dynamic Range Image) è, in generale, un'ottima soluzione di illuminazione in quanto fornisce contemporaneamente sia un background per la scena sia la fonte di luce; questo tipo di immagine infatti non contiene soltanto informazioni sul colore ma anche dati sulla luminosità della scena che rappresenta.

Anche in questo caso bisognerà spuntare "Use IBL" per poter utilizzare l'immagine come sorgente di luce; altri parametri utili sono "Rotation" che determina la rotazione dell'immagine e "IBL samples" che consente di determinare la quantità di campioni da utilizzare per rappresentare le ombre morbide: più campioni saranno utilizzati, meno saranno disturbate le ombre (ovviamente, il tempo di rendering aumenterà).

Sunsky 1-2

Queste due opzioni, simili tra loro per le funzionalità che offrono, ma leggermente diverse per il modo in cui le gestiscono, rappresentano lo spettro completo del cielo in diverse condizioni meteo.

Figura 4.14:



Turbidity

Serve ad impostare la torbidità del cielo. Valori compresi tra 3 e 5 indicano un cielo normale; valori inferiori si riferiscono a cieli molto limpidi, mentre oltre il 5 avremo un cielo nebbioso.

Brightness of horizon gradient e Luminance of horizon

Queste due impostazioni servono per controllare il comportamento dell'orizzonte; in particolare, la prima controlla la luminosità dell'orizzonte rispetto al cielo, mentre la seconda definisce quanto è grande la sfumatura che li separa (in sostanza, quanto è alto l'orizzonte).

Solar region intensity e Width of circumsolar region

Con questi parametri è possibile controllare l'intensità del disco solare e la sua dimensione. Maggiori saranno i valori, più luminoso ma anche più piccolo sarà il sole.

Backscattered light

Questa impostazione definisce la quantità di luce riflessa dalla torbidità atmosferica; valori alti per questo parametro determinano un cielo più luminoso e quindi un disco solare più piccolo.

Controlli per la posizione del sole

È anche possibile determinare la posizione del sole, intesa come angolazione rispetto alla scena; esistono due modi per farlo: muovendo la sfera presente nel pannello, trascinando il cursore sopra la sfera, oppure sfruttando una sun lamp presente nella scena e ottenendone angolazione, posizione o entrambe.

Altitude

Questo controllo permette di gestire l'altitudine della visuale, relativamente al centro dello sfondo: se l'altitudine aumenta, la linea di orizzonte si abbassa. È un ottimo modo per controllare la posizione dell'orizzonte quando la telecamera è molto sopra il livello del terreno.

Night

Spuntando questa opzione è possibile ottenere un cielo notturno in cui il sole agisce da Luna per la scena. Questa scelta è disponibile soltanto per il sunsky 2.

Esistono poi ulteriori impostazioni da poter settare; esse sono:

- Add sun che aggiunge un modello realistico del Sole alla scena, considerando la posizione e le impostazioni settate precedentemente; è possibile controllarne l'intensità con lo slider apposito
- Add skylight abilita l'emissione di luce da parte del cielo
- Skylight power controlla l'intensità di luce emessa
- Samples determina il numero di campioni utilizzati per il sole e il cielo
- Sky brightness controlla la luminosità dei colori del cielo; non influisce sulla quantità di luce emessa. Opzione non disponibile per il sunsky1

4.2.6 Considerazioni su YafaRay

YafaRay risulta essere un ottimo motore di rendering esterno, semplice da utilizzare e con diverse funzionalità interessanti (si sottolinea la grande offerta di effetti per il background).

Con questo engine è possibile lavorare in modo professionale perché, nonostante sotto diversi aspetti sia molto semplice, offre comunque strumenti in

grado di ottenere risultati complessi.
Durante la fase di studio delle sue potenzialità si è notato che YafaRay pone l'accento più che sui materiali sulla luce dell'ambiente che si sta rappresentando, al contrario di Cycles che fonda tutto sulla loro accurata descrizione. Molte delle personalizzazioni che in Cycles si effettuano tramite i nodi, in YafaRay sono comprese nel momento della scelta del materiale e nelle impostazioni conseguenti.
Sostanzialmente lo studio del modo in cui la luce interagisce con gli oggetti più che durante la scelta dei materiali viene effettuato al momento della scelta della luce ambientale: si sceglie il tipo di materiale e poi sarà in gran parte il tipo di luce a determinare la resa effettiva.
Gli aspetti negativi di questo motore possono essere ritrovati nella mancanza di supporto per alcuni tipi di oggetti (ad esempio, YafaRay non supporta il particle system di blender nella resa dell'immagine) e nell'impossibilità di utilizzare la GPU per il rendering.

4.3 LuxRender

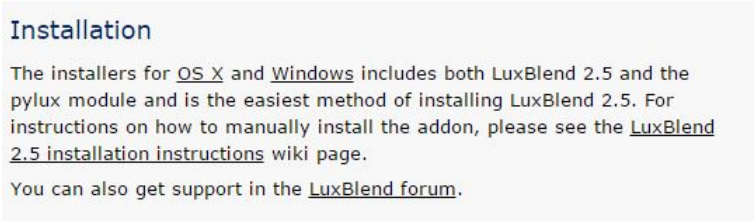
4.3.1 Introduzione

LuxRender è un software free e open-source che può essere utilizzato come add-on su Blender, Cinema4D, 3D Studio Max,Maya e altri.
Questo engine è basato su PBRT, un programma di ray tracing physically based. Esso, strumento molto potente e ben strutturato, è nato per esclusivo uso accademico e non artistico; poiché venne rilasciato sotto licenza GPL (General Public License, licenza per sofwtare libero) è stato possibile attingere al suo codice sorgente e creare un nuovo programma.
Nel 2007 iniziò l'attività di progettazione e programmazione di LuxRender, che ottenne il 24 giugno 2008 la sua prima release ufficiale.

4.3.2 Installazione e configurazione

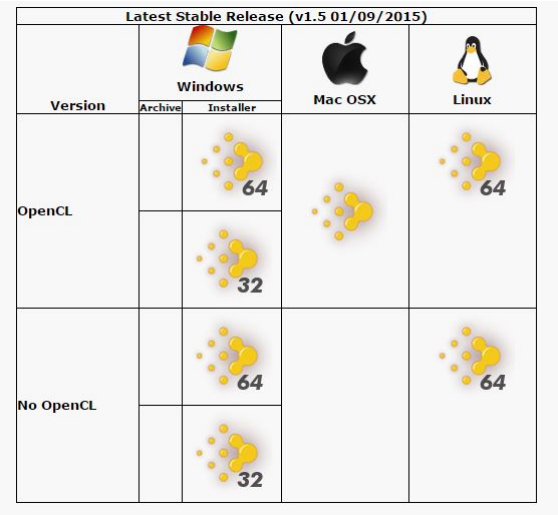
Per scaricare e successivamente installare LuxRender occorre collegarsi al sito ufficiale dell'engine al seguente link: http://www.luxrender.net/en_GB/index e selezionare il package relativo a Blender.
A questo punto, nella finestra che si aprirà, selezioneremo il nostro sistema operativo sotto il paragrafo "Installation":

Figura 4.15:



La scelta, al momento, è soltanto tra OSX e Windows; in realtà i due link rimandano entrambi alla stessa pagina, nella quale potremo scegliere la versione di LuxRender da scaricare (al momento l'ultima release stabile è la 1.5), il sistema operativo e la versione OpenCL o non OpenCL (la scelta dipende dalla possibilità o meno di usare la GPU per renderizzare).

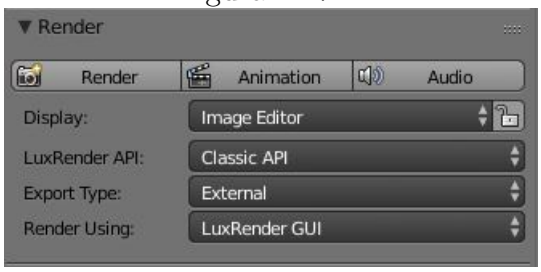
Figura 4.16:



A questo punto partirà il download del file eseguibile; una volta scaricato basterà eseguirlo ed esso salverà un archivio .zip in una locazione di memoria sul nostro computer.
La procedura da adesso è analoga a quella seguita per YafaRay: basterà recarsi nelle User preferences di Blender e aggiungere come add-on LuxRender selezionando l'archivio ottenuto in precedenza, abilitarlo come motore di rendering e salvare le modifiche.
Per accertarci che LuxRender sia stato installato correttamente lo selezioniamo dal solito menù a tendina dei render attivi, costruiamo la nostra scena e proviamo a renderizzare.

Nelle impostazioni del rendering dovremmo avere selezionate queste opzioni:

Figura 4.17:

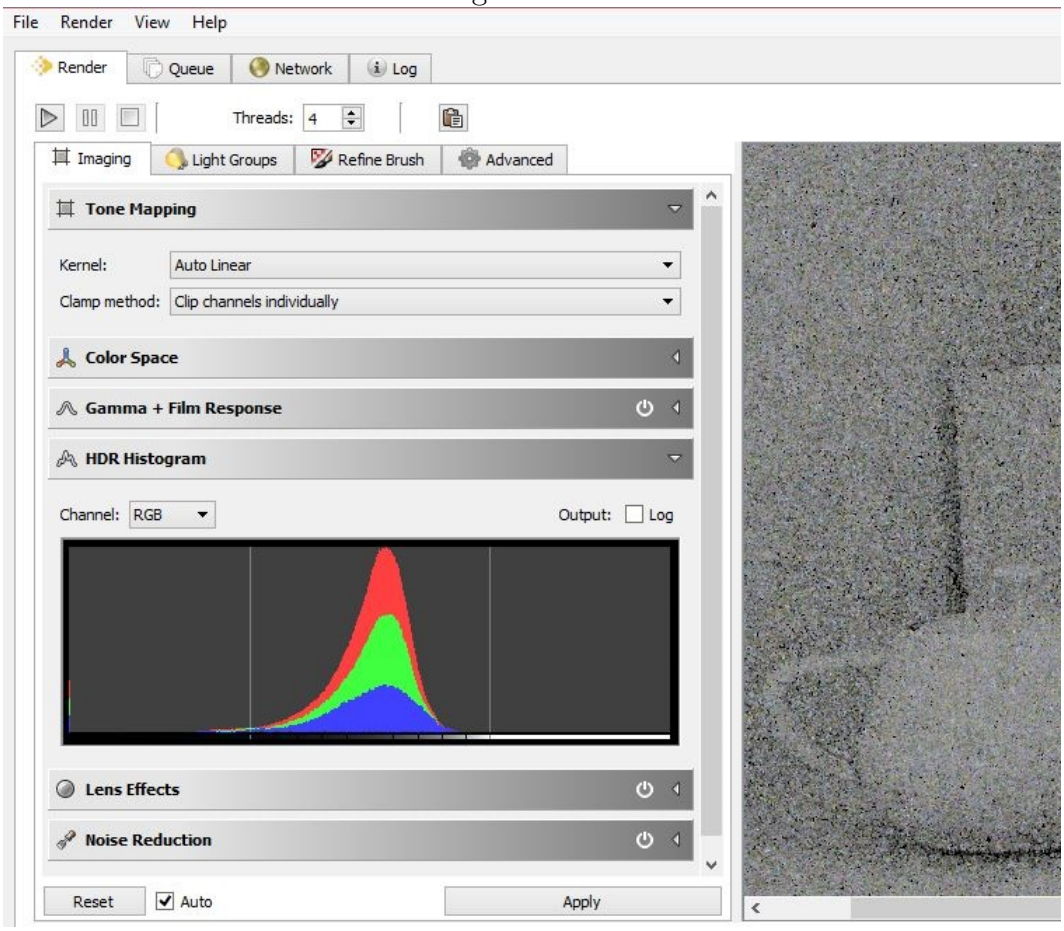


LuxRender utilizza un'interfaccia esterna a Blender per renderizzare la scena, da cui è possibile controllare i progressi del rendering ed scegliere tra diverse opzioni agendo sull'immagine.

4.3.3 Interfaccia esterna di LuxRender e post processing

Come già accennato, LuxRender fa utilizzo di un'interfaccia esterna per mostrare il progresso del rendering in corso:

Figura 4.18:



Da qui si può mettere in pausa il rendering e farlo ripartire quando ne abbiamo bisogno o addirittura terminarlo; a differenza degli altri motori di rendering, infatti, LuxRender continuerà a migliorare l'immagine finale fino a che l'utente non ne risulterà soddisfatto (è anche possibile, nel pannello di rendering, selezionare il numero massimo di samples da utilizzare per ogni pixel e/o il tempo massimo di rendering, per impostare un limite al rendering).
Dall'interfaccia è possibile zoomare l'immagine e spostarla per visualizzarla interamente e selezionare il numero di thread da utilizzare (è possibile cambiare questo valore anche mentre il rendering è attivo).
La tab "Render" contiene tutte le impostazioni per l'immagine che sta ren-

derizzando, e offre quattro diverse tab per lavorare sull'output, ognuna delle quali si occupa di un aspetto diverso.

Imaging

Questa tab contiene vari controlli per gestire il post-processing dell'immagine renderizzata.
Ognuno di questi box può essere espanso per accedere alle sue funzionalità; alcuni possiedono un interruttore che serve ad abilitare e disabilitare velocemente l'effetto. Alcune di queste opzioni possono essere applicate anche mentre il rendering è in corso, per esempio la funzione che consente di modificare il tipo di tone mapping o quella relativa all'istogramma HDR mostrato relativo all'immagine; altre, in particolare "Lens effects" e "Noise reduction", è consigliabile applicarle una volta terminato il rendering, sia a causa del tempo che richiedono (che andrebbe quindi a pesare sul tempo di realizzazione), sia per via di ciò che fanno: la prima permette di applicare svariati effetti all'immagine, la seconda si occupa di ridurre l'eventuale disturbo dell'immagine.

Light groups

Questa tab contiene i controlli relativi ai parametri di ogni light group indicato nella scena; è infatti possibile, durante la fase di modellazione, creare dei gruppi di luci che potranno essere gestiti a tempo di rendering grazie alle impostazioni dell'interfaccia.
È ovvio che le impostazioni riguardanti le luci in sè vanno aggiustate correttamente prima che inizi il rendering, in quanto le variazioni che si possono applicare utilizzano come punto di partenza le informazioni fornite dalla scena, perciò una stessa modifica nell'interfaccia a due luci con intensità diversa porterà a due risultati diversi, proprio perché le informazioni di partenza sono differenti.
Per ogni gruppo di luci LuxRender permette all'utente di spegnerle/accenderle (ad esempio si può pensare ad una scena di interni che vuole essere rappre-

sentata sia in ambiente diurno che notturno, avendo quindi la possibilità di decidere se accendere luci artificiali o solari), cambiarne il colore e l'intensità, potendola andare a modificare sia in termini di temperatura sia di valore vero e proprio.

Refine brush

In questa tab ci sono tutti gli strumenti che ci consentono di indicare quali aree sono da considerare più importanti nella nostra immagine e quali meno, ripassandole con una sorta di pennello. È possibile indicare sia le aree di maggiore importanza (ovvero quelle dove LuxRender concentrerà i suoi sforzi durante il rendering) sia quelle meno significative; si può modificare la grandezza del pennello così come la sua intensità.

4.3.4 Pannello di rendering

Il pannello di rendering di LuxRender consente di poter gestire diverse opzioni. Innanzitutto è possibile selezionare l'export type tra interno ed esterno: selezionando il primo l'interfaccia di LuxRender non verrà aperta, e il motore lavorerà "dietro le quinte" fornendoci solo il risultato finale (opzione consigliata se si sta renderizzando un'animazione); selezionando il secondo invece sarà possibile usufruire dell'interfaccia e delle sue funzionalità.

Figura 4.19:

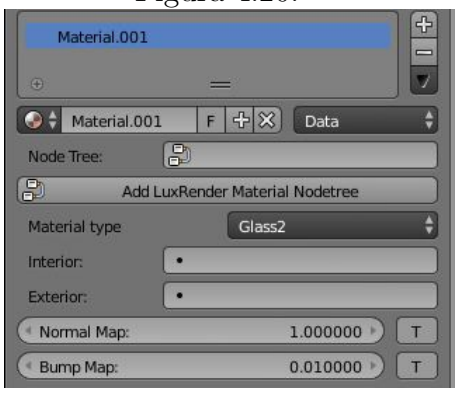
Questa tab permette di gestire molte opzioni di rendering, tra le quali selezionare la modalità di resa: bidirectional path tracing, photon map o eventualmente anche il direct lightning. Come accennato in precedenza possiamo impostare il numero massimo di samples per ogni pixel o eventualmente il tempo massimo di rendering (lasciando valori pari a 0, il rendering si fermerà soltanto se gli verrà ordinato di farlo).
È anche possibile indicare la "Halt treshold" e il "Convergence step".
La prima opzione permette a LuxRender di bloccare il rendering automatico non appena arriva a una certa qualità di immagine; questa qualità viene settata utilizzando un valore compreso tra 0 e 10 che indica la percentuale di pixel che possono fallire in un test di convergenza. Ogni volta che viene aumentato il numero, saranno necessari 10x pixel che possano passare il test: una soglia di 3.0 comporta che il 99.9% di pixel debbano passare, mentre un valore pari a 5.0 richiede che ne passi il 99.999%. Il passo di convergenza indica invece l'intervallo (inteso come campioni per pixel) per testare la convergenza, e di conseguenza per fermare il rendering.

4.3.5 Materiali

LuxRender presenta una buona varietà di materiali che possono essere gestiti sia dal loro normale pannello come in YafaRay sia, a partire dalla versione 1.3, utilizzando un editor basato sui nodi che intende sostituire totalmente il vecchio sistema di definizione dei materiali.
Se si volesse utilizzare l'editor dei nodi, e quindi generare un albero per il materiale che abbiamo scelto, basterà semplicemente cliccare sul bottone "Add LuxRender Material Nodetree" (figura 4.20).

L'editor dei nodi di LuxRender è analogo a quello di Cycles per quanto riguarda il funzionamento, mentre riguardo il numero di nodi e la varietà di essi che offre risulta essere carente rispetto a Cycles; per esempio il tipo output contiene soltanto il nodo di tipo "Material", ciò significa che tramite l'editor non è possibile andare ad agire sul mondo o sulle luci.

Figura 4.20:



Di seguito i materiali offerti da LuxRender, utilizzabili sia tramite i nodi sia tramite impostazioni base.

Matte e Matte Translucent

Matte è il materiale base di LuxRender e rappresenta una normale superficie diffusiva; vista la sua semplicità e conseguente velocità di renderizzazione è utilizzato per oggetti di sfondo, come i muri. Gli unici parametri che offre sono quelli relativi al colore del materiale e alla sua ruvidità (quest'ultimo valore è espresso in gradi).
Matte Translucent può essere considerata una sua estensione e permette alla luce di essere trasmessa attraverso il materiale; può essere utilizzato ad esempio per rappresentare lampade o materiali organici. Oltre al colore principale del materiale (indicato con "Reflection Color") è possibile indicare il "Transmission color", ovvero il colore che avrà la luce trasmessa da esso (per risultati migliori si consiglia di utilizzare il bianco puro); anche questo materiale possiede un controllo per la ruvidità.

Glossy, Glossy translucent e Glossy Coating

Il materiale Glossy rappresenta una superficie diffusiva con una copertura di vernice; è molto adatta per diversi tipi di plastica, porcellane e legno lavorato, ma anche per alcuni tipi di stoffa. Glossy translucent è invece una

superficie diffusiva che possiede sia la componente di trasmissione che quella di vernice: in sintesi, si può dire che sia una miscela di Matte Translucent e Glossy. È molto indicato per materiali organici quali pelle, foglie, occhi, latte.
Entrambi possiedono le opzioni relative al colore base e alla ruvidità, che per tutti e due i materiali determina quanto brilante è la superficie; è inoltre possibile specificare il colore della componente speculare, ovvero sostanzialmente della vernice che ricopre il materiale. Per questa esiste anche l'IOR da poter essere settato, per determinare il tipo di materiale della copertura. Il glossy translucent permette inoltre di specificare il colore che avrà la luce trasmessa.
Differentemente dagli altri due, il glossy coating rappresenta una sorta di copertura lucida per un qualsiasi materiale di base; per esso non viene infatti specificato alcun colore, bensì occorre indicare qual'è il materiale base (tra quelli definiti precedentemente dall'utente) a cui applicare questo livello. Anche per questo materiale è possibile gestire il colore speculare, l'IOR e la ruvidità.

Metal e Shiny Metal

Questi due materiali simulano ogni tipo di oggetto metallico; per il materiale "Metal" si è inoltre in grado di specificare il tipo di metallo che vogliamo rappresentare, scegliendo tra oro, alluminio, argento e rame. La differenza tra i due sta nel fatto che lo Shiny Metal è composto da una base metallica ricoperta da una superficie brillante e lavorata; per esso possiamo inoltre scegliere il colore della riflessione, ovvero il colore di base, e il colore speculare, cioè quello della superficie.
Per entrambi possiamo inoltre settare il valore di ruvidità.

Mirror

Questo materiale rappresenta una superficie perfettamente speculare; si può pensare ad esso come allo shiny metal senza copertura. Di esso possiamo

determinare il colore e aggiungere una sorta di "pellicola" sulla sua superficie, causando la comparsa di una colorazione arcobaleno per la riflessione.

Glass e Rough Glass

Entrambi questi materiali possiedono le impostazioni classiche dei materiali vetrosi, quali l'IOR, il colore della luce trasmessa e il colore della luce riflessa; inoltre, anche per essi è possibile impostare un valore di ruvidità. La differenza che intercorre tra i due è che, come è facile dedurre dal nome, il rough glass è di per sè un vetro ruvido, ideale per rappresentare vetri ghiacciati e perfino alcuni tipi di ghiaccio.

Velvet

Questo materiale rappresenta il velluto, ma è ideale anche per diversi tipi di stoffa. Di esso possiamo, tramite l'opzione "Thickness", determinare la quantità di "lanugine" sulla superficie.

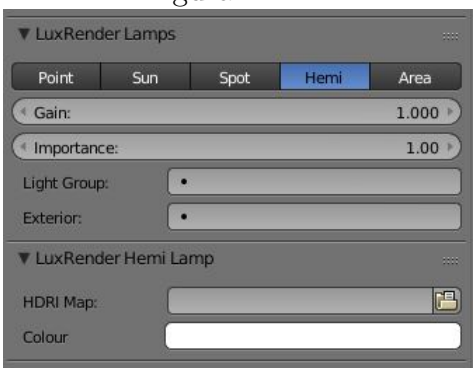
LuxRender offre la possibilità di unire due materiali, come YafaRay: si selezionano le due componenti tra i materiali già creati dall'utente e poi si determina la percentuale di influenza di ciascuno dei due.
Per quanto riguarda l'emissione, nei materiali in cui ciò sia supportato, occorre spuntare l'apposita opzione nel pannello di gestione del singolo materiale e determinarne poi l'intensità.

4.3.6 Background

Una delle particolarità di LuxRender è il modo in cui il background può essere impostato: mentre per gli altri motori di rendering è possibile specificare un'immagine che rappresentasse il mondo dalle impostazioni dello stesso, in LuxRender questo viene fatto in modo diverso: dopo aver aggiunto una sorgente luminosa alla scena, occorre andare nel pannello ad essa corrispondente e specificare la mappa HDR che si vuole utilizzare. È importante

sottolineare che l'unico tipo di luce che supporta una mappa hdr è il tipo "Hemi"; se ad essa si associa, come in questo caso, una mappa HDR, è necessario aggiungere un'ulteriore sorgente luminosa (ad esempio un "Sun") per rappresentare effettivamente il sole.

Figura 4.21:

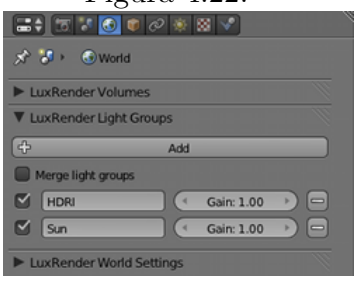


Se non si vuole utilizzare un'immagine è anche possibile specificare il colore che possiederà la luce. Tramite i valori "Gain" e "Importance" si può calibrarne la luminosità.

4.3.7 Light groups

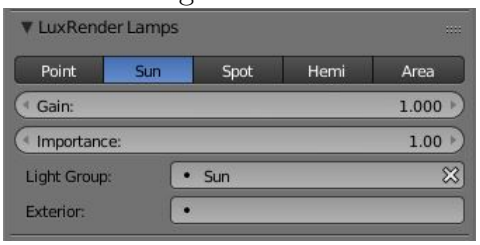
Come già accennato, LuxRender offre la possibilità di creare dei gruppi di luci, cosa che sarà utile nella fase di post processing.
Per creare un light group basta semplicemente recarsi nella tab "World" e aggiungere un nuovo gruppo (o più gruppi, a seconda delle necessità):

Figura 4.22:



A questo punto qualsiasi luce abbiamo nella scena o che aggiungeremo sarà possibile comprenderla in un gruppo, sul quale è possibile agire in fase di rendering e applicare le modifiche a tutte le luci che fanno parte di quel gruppo. Per aggiungere una sorgente luminosa a un light group esistente occorre andare nella tab relativa alla luce che stiamo considerando e nella voce "Light Group" selezionare il gruppo desiderato dalla lista.

Figura 4.23:



4.3.8 Considerazioni su LuxRender

LuxRender è un motore che risulta essere non particolarmente intuitivo per un utente non esperto; il suo sistema di nodi inoltre non offre nulla di più di quello di Cycles e, anzi, è carente di alcune funzionalità. Pur offrendo funzionalità relativamente semplici, la sua interfaccia non rispecchia questa caratteristica.
Utilizzando principalmente il bidirectional path tracing come tecnica di rendering, è molto indicato per scene di interni, ed è in grado di rappresentare in modo buono le caustiche.
LuxRender è però anche lento in proporzione al risultato che offre: anche per una scena semplice, con forme non elaborate e materiali base, il tempo che impiega per raggiungere un risultato accettabile è alto.
La forza di questo motore risiede nell'interfaccia dedicata al post processing, nella quale è possibile giocare con diversi effetti e modificare l'immagine.

4.4 Renderman

4.4.1 Introduzione

Renderman (abbreviazione di Renderman Interface Specification) è un motore di rendering sviluppato dalla Pixar Animation Studios che utilizza Maya come software 3D di appoggio.
Da marzo 2015 è stata resa pubblica e scaricabile da chiunque una versione ad uso esclusivamente personale e non-commerciale, sia per Maya che per il software KATANA; per quanto riguarda Blender, è possibile scaricarlo e utilizzarlo, ma il plug-in è ancora in fase di sviluppo e rifinitura. RenderMan, così come Cycles, è un path tracer, ma può utilizzare anche il direct lightning.
Lo sviluppo di RenderMan inizia nel 1970 presso l'università dello Utah dove il futuro fondatore della Pixar, Ed Catmull, stava svolgendo la sua tesi di dottorato sui problemi del rendering.
Da qui il lavoro si spostò presso la Lucasfilm, dove Catmull e altri grafici furono incaricati di lavorare su software grafici che potessero realizzare immagini complesse e altamente fotorealistiche; per raggiungere questo obiettivo svilupparono per l'appunto RenderMan, che ottenne questo nome solo nel 1989.
Nel 1983 nacque la Pixar come divisione della Lucasfilm, e divenne indipendente nel 1986 quando Steve Jobs la comprò.
Questo motore di rendering è stato e viene correntemente utilizzato dalla Pixar per la realizzazione dei suoi film, sia d'animazione che non; il primo per cui è stato utilizzato fu Star Trek: l'ira di Khan nel 1982. La lista è molto lunga e copre sostanzialmente tutti i film d'animazione realizzati negli ultimi anni; tra i titoli più significativi ricordiamo Jurassic Park(1993), Toy Story (1995), Titanic (1997), gli ultimi quattro titoli di star wars e Avatar (2009). Per quest'ultimo film sono state necessarie fino a 47 ore di rendering per il singolo fotogramma.

4.4.2 Installazione e configurazione

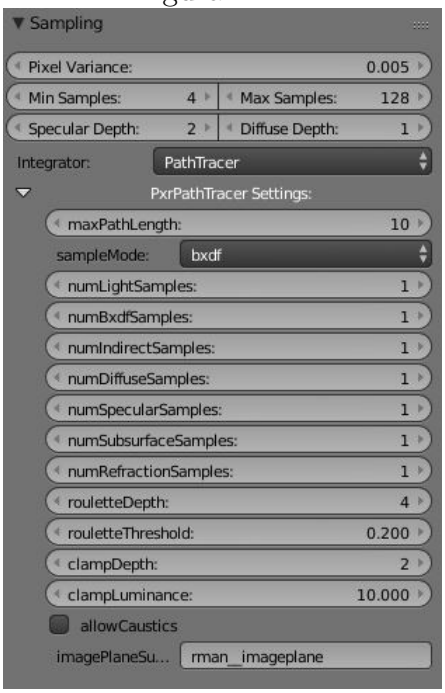
Per poter installare e utilizzare RenderMan è necessario registrarsi creando un proprio account; i dati di questo verranno poi richiesti in fase di installazione. Collegandosi al link http://renderman.pixar.com/view/get-renderman si può procedere con i passi per il download dell'engine.
Lo step successivo richiederà per l'appunto la registrazione di un account; una volta concluso questo passo si potrà scaricare l'installer per RenderMan. Scaricato l'eseguibile basterà farlo partire; in questa fase verrà richiesto di immettere i dati di registrazione precedentemente scelti.
Dopo aver installato RenderMan basterà seguire i passi già effettuati per YafaRay e LuxRender per abilitarlo tra gli add-ons di Blender; una volta spuntata "Prman for RenderMan" e salvate le impostazioni, RenderMan potrà essere scelto come motore di rendering attivo dal menù a tendina in alto.

4.4.3 Pannello di rendering

Il pannello di rendering di RenderMan ci consente di gestire molte impostazioni: è possibile impostare la "Pixel Variance", ovvero un valore massimo per il quale un pixel computato si distacchi dal vero valore del pixel e, a differenza di Cycles in cui si inserisce il numero di samples, i valori minimo e massimo di samples richiesti per il render.
RenderMan ottimizza il rendering selezionando il numero di campioni da utilizzare dal range impostato dall'utente: ciò significa che per quei punti dell'immagine che non necessitano troppe valutazioni si utilizzerà un numero di campioni pari ai valori minimi del range, mentre un altro numero di campioni sarà utilizzato soltanto per quelle parti che necessitano di più samples.
È inoltre possibile determinare la lunghezza massima del path dei raggi (in altre parole, quindi, il numero massimo di rimbalzi da fargli effettuare) e anche specificare il numero per ogni tipo di campione (per la luce diffusa, per la luce speculare, ecc...).

Si può scegliere inoltre di renderizzare o meno le caustiche.

Figura 4.24:



4.4.4 Materiali

Anche RenderMan offre un sistema di nodi, il quale però, per quanto riguarda Blender, non offre ancora molta varietà di opzioni. I nodi vengono divisi in tre categorie: PRMan bxdfs, che descrivono i materiali base, PRMan Lights, che comprendono tutti i tipi di sorgenti di luce, e i PRMan Patterns che permettono di aggiungere sia texture sia nodi di manipolazione (nodi mixer, map, ecc...).
In RenderMan occorre fare una distinzione preliminare per i materiali: esiste una classe, i "PxrLM", che comprende i materiali base rivisitati in modo più "semplice" rispetto agli altri disponibili; in pratica, utilizzano un'interfaccia progettata per essere facile nell'utilizzo e che contiene il set minimo di parametri per poter utilizzare al meglio il materiale. Di questa classe fanno parte cinque materiali base:

- PxrLMDiffuse che rappresenta un semplice materiale diffusivo
- PxrLMSubsurface che rappresenta un materiale con proprietà di subsurface scattering
- PxrLMMetal che rappresenta una superficie metallica
- PxrLMPlastic che rappresenta una superficie plastica; viste le numerose impostazioni che possiede può essere utilizzata per generare diversi altri tipi di materiali; settando il suo "Incandescence color" è possibile realizzare un materiale incandescente, simulando una sorta di emissione luminosa
- PxrLMGlass rappresenta un materiale vetroso e in generale è utile per creare materiali con proprietà di rifrazione, come ad esempio gelatina. Poiché questo materiale utilizza la rifrazione, è bene sapere che aumentando o diminuendo il valore di specular depth nel pannello di rendering si otterranno effetti diversi; in particolare, con una profondità più alta, il vetro risulterà più trasparente. È possibile impostare anche la ruvidità del materiale

Gli altri materiali forniti da RenderMan sono sostanzialmente gli stessi descritti sopra con impostazioni leggermente differenti e più complete, ma che comunque sono state già affrontate e spiegate per gli altri motori di rendering.
Va però fatta una menzione per tre materiali particolari presenti esclusivamente in questo motore di rendering:

- PxrDisney: è un materiale che sostanzialmente racchiude tutti gli altri, vista la grande quantità di parametri che è possibile settare. Questa numerosità però non causa un aumento della difficoltà nel suo utilizzo, tutt'altro: il principio e l'obiettivo di questo materiale è la semplicità d'uso e la flessibilità, grazie alle impostazioni molto intuitive che offre. È possibile settare il valore di ogni possibile caratteristica che un materiale può avere, mischiandole tra loro per creare effetti disparati: si

può infatti controllare la quantità di componente metallica, speculare, lucente, l'anisotropia e la ruvidità, oltre a poter impostare il colore di base del materiale e l'eventuale colore della luce emessa.

- PxrSkin: è il materiale utilizzato per riprodurre la pelle e materiali come onice e cera, in cui c'è una forte componente di subsurface scattering. Esso utilizza i principi più avanzati del rendering physically based per arrivare a un risultato quando più vicino alla realtà, ed è in grado di rappresentare in modo assolutamente migliore rispetto a un normale shader di tipo subsurface scattering tutti i minimi dettagli della superficie. Le impostazioni che offre sono molto particolari: è possibile settare il colore dello scattering a raggio corto, medio e lungo, informazioni che saranno combinate tra di loro per ottenere il colore finale; il calcolo del risultato finale è effettuato considerando anche i pesi che possiedono le tre distanze (settabili anch'essi dall'utente) e le lunghezze, per le quali si intende la distanza che la luce corrispondente a uno dei tre colori impostati (near, mid e far) rispettivamente percorrerà.

- PxrMarschnerHair: è il materiale utilizzato per i capelli. Offre parametri per gestire sia la componente diffusiva, che quella speculare: per la prima è possibile settare colore e intensità, mentre la seconda prevede un set completo di impostazioni per gestire la rifrazione, la specularità, la trasmissione.

In RenderMan le texture non possono essere gestite tramite l'apposito pannello come negli altri motori di rendering, ma soltanto attraverso il node editor, aggiungendo un nodo di tipo "texture" e collegandolo alla componente desiderata del materiale a cui la stiamo applicando. Il background viene gestito solo ed esclusivamente tramite le luci, che possono essere utilizzate sia come normale sorgente di luce sia come mappatura per un'immagine HDR; anche le impostazioni sulle ombre sono interamente gestite nel pannello della luce.

4.4.5 Considerazioni su RenderMan

Questo motore di rendering è indubbiamente il più potente tra quelli esaminati, nonché quello che offre più personalizzazione ed effetti. L'interfaccia di utilizzo risulta essere abbastanza semplice, nonostante la complessità di risultato che può far raggiungere, e questo è ovviamente qualcosa di auspicabile per ogni engine. Esso però non riesce a esprimere il massimo delle funzionalità su Blender, a causa del fatto che, come già detto, RenderMan nasce per lavorare su Maya e per quanto riguarda Blender il plug-in è ancora in fase di perfezionamento.
A causa della novità che rappresenta e della fase di sviluppo in cui si trova, documentazione e tutorial al momento scarseggiano per quanto riguarda il suo utilizzo in Blender, e questo ha causato sicuramente qualche difficoltà nell'apprendimento e nella scoperta di feature. Un motore con una tale espressività e potenza sicuramente non tarderà a prendere piede.

Capitolo 5

Progetto di tesi: realizzazione di una scena d'interni

Come sintesi degli studi effettuati, iniziati durante il periodo di tirocinio e proseguiti e approfonditi durante il periodo di tesi, è stata realizzata una scena che potesse mostrare al meglio differenti tipi di materiali e le loro interazioni con la luce, con varietà di situazioni, ambienti e oggetti; la scena di interni è la tipologia che racchiude tutto ciò. In particolare, la scelta di un ambiente moderno favorisce ancor di più la varietà di materiali presenti. Per il progetto è stata scelta un'ambientazione che presentasse sia ambienti relativamente ristretti che ampi, per mostrare come la luce si comporta e agisce nelle diverse situazioni, creando effetti visivi diversi.

5.1 Motivazioni delle scelte progettuali e studi svolti

Dopo aver preso in considerazione e studiato i motori presentati in questa tesi è stato scelto Cylces per la realizzazione della scena. I motivi che hanno portato a questa scelta sono diversi; in generale si può dire che questo motore si è rivelato essere il migliore in termini di rapporto tra qualità e risorse richieste. Si presenta ora il risultato raggiunto dopo il periodo di tirocinio.

Figura 5.1: Scena d'interni con Cycles: 300 samples

Figura 5.2: Scena d'interni con YafaRay

Le due immagini rappresentano la stessa scena realizzata con due diversi motori di rendering: Cycles e YafaRay. È facile notare che la resa di Cycles è decisamente più realistica rispetto a quella di YafaRay; quest'ultimo impiega un tempo minore per il rendering rispetto al primo, ma in modo non abbastanza significativo per poterlo considerare migliore. La luce non è curata quanto in Cycles né lo sono i materiali, per i quali oltretutto c'è una scelta più ristretta.
YafaRay, insieme a LuxRender, è in effetti più adatto per rappresentare particolari riflessioni e rifrazioni della luce (ricordando per l'appunto le caustiche), mentre incontra difficoltà in ambienti dove prevale la luce diffusa. Esso risulta essere comunque un buon motore di rendering in quanto il risultato è in ogni caso piacevole, ma la limitatezza dei materiali e degli effetti ottenibili (si veda ad esempio la differenza di resa del tappeto nelle due figure) non ha permesso la scelta di questo motore.
LuxRender si è rivelato inadatto per via della sua richiesta di risorse hardware: non è stato possibile renderizzare la scena presentata in precedenza in quanto le risorse fisiche non erano sufficienti a supportare il lavoro di LuxRender.
Nonostante una migliore offerta di materiali rispetto a YafaRay, questa sua eccessiva richiesta ha portato a scartare la possibilità di utilizzare questo motore; in ogni caso, la qualità che offre rapportata anche al tempo richiesto rimane comunque non paragonabile a quella di Cycles, anche se risulta essere un ottimo motore per rappresentare fenomeni non diffusivi.
La resa delle caustiche, infatti, risulta essere accurata in YafaRay e LuxRender, mentre è scarsa o del tutto assente in Cycles e RenderMan.

Figura 5.3: Effetti caustici della luce: resa di LuxRender

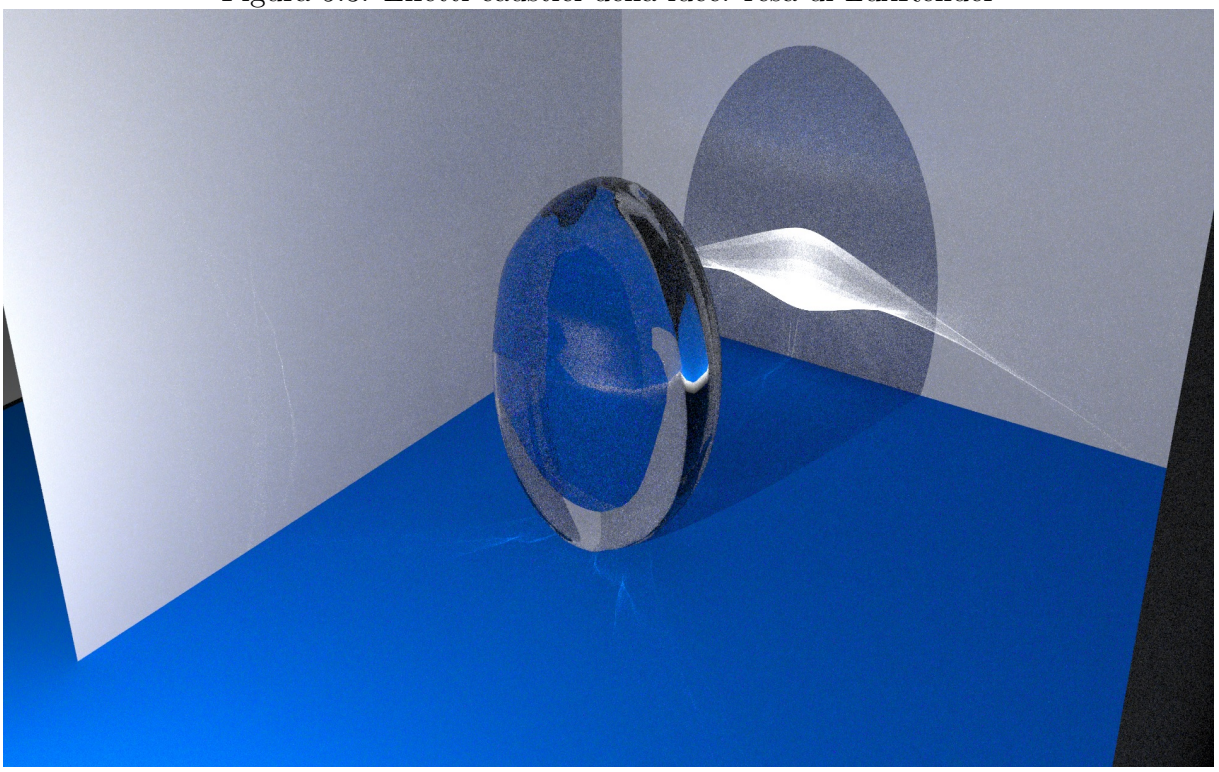


Figura 5.4: Effetti caustici della luce: resa di YafaRay

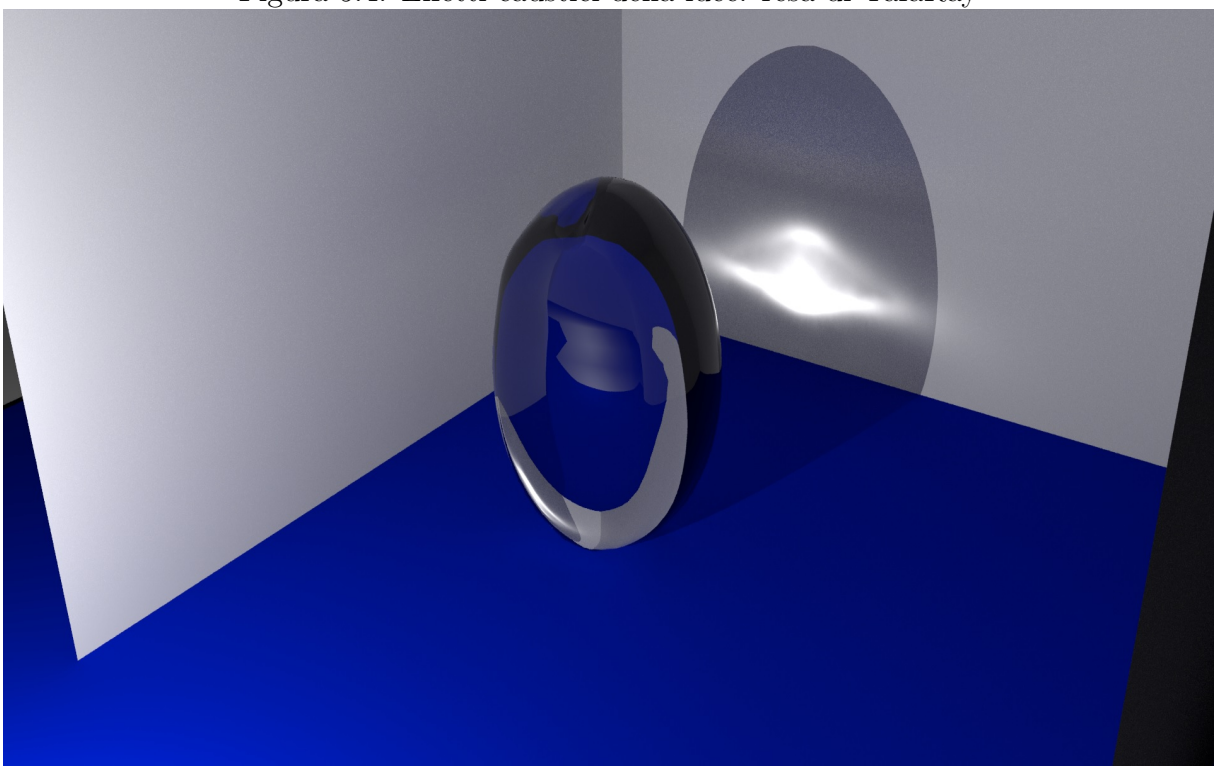


Figura 5.5: Effetti caustici della luce: resa di Cycles

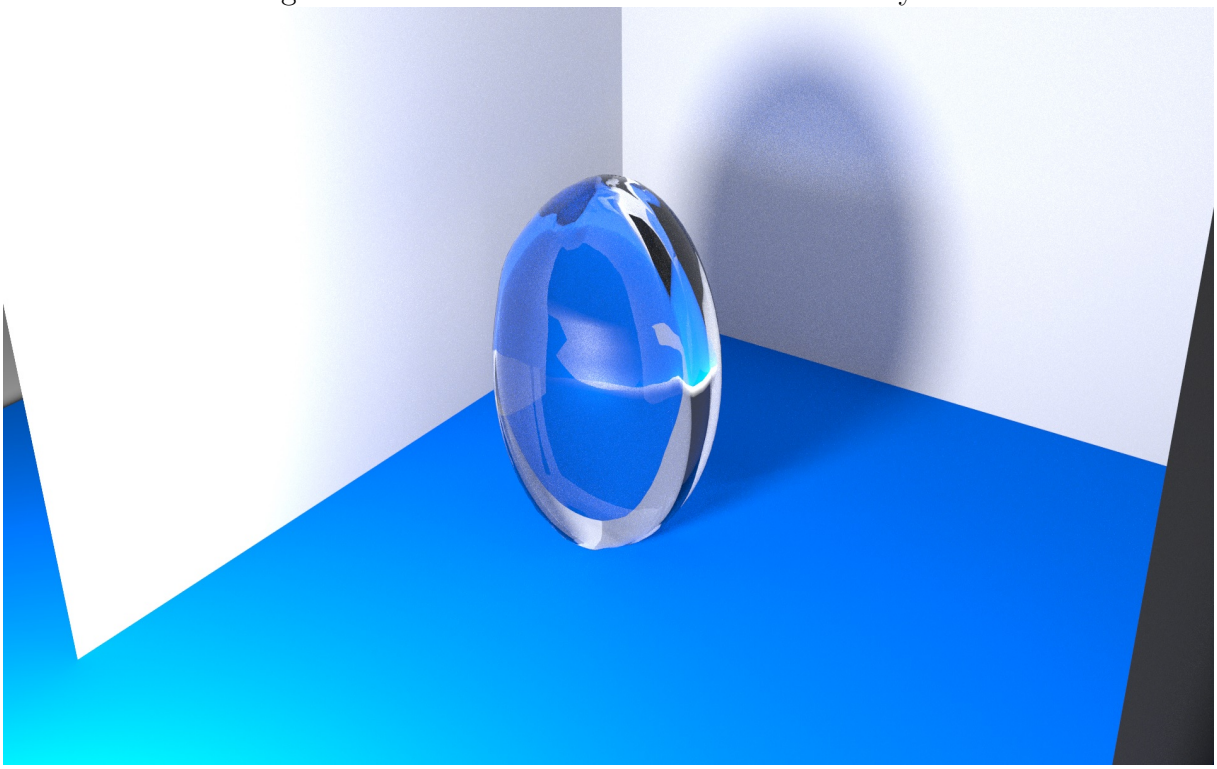
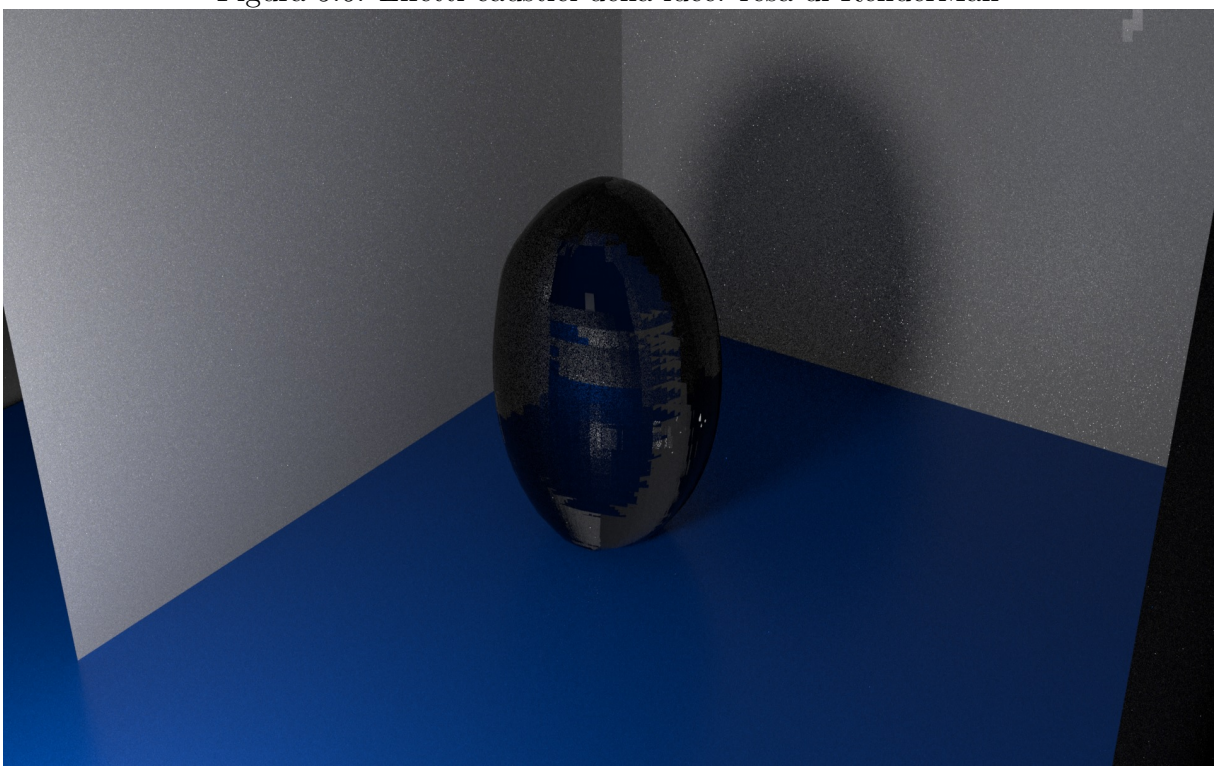


Figura 5.6: Effetti caustici della luce: resa di RenderMan



L'ultimo motore analizzato è stato RenderMan. Come detto nel paragrafo dedicato, esso è stato da poco reso gratuito dalla Pixar Animation Studios, la sua casa produttrice.
A causa della data di rilascio recente la sua documentazione, come già detto, risulta essere scarsa e per questo motivo le funzionalità e i materiali che offre sono ancora in fase di scoperta e di testing da parte degli utenti.
Questo motore, per molti aspetti simile a Cycles, offre una qualità molto elevata anche con un numero basso di samples, dettagli più curati e pochi disturbi dovuti alle inter riflessioni della luce.
Essendo più potente richiede anche un maggiore tempo di rendering, e la durata è significativa anche per scene molto semplici (si vedano le figure 5.7 e 5.8).

Figura 5.7: Scena di prova RenderMan: 4-100samples

Figura 5.8: Scena di prova Cycles: 70 samples

Come si può notare, oltre che il minor disturbo a parità di campioni, RenderMan è anche in grado di rappresentare in modo migliore i fenomeni contemplati dalla radiosity (lo si può notare in particolare nel color bleeding della parete a destra e del pavimento: i colori si influenzano l'un l'altro a causa delle inter riflessioni tra i due piani).
Alla luce di queste sperimentazioni, per il tipo di studio che si è voluto affrontare, si è scelto Cycles in quanto offre un'ottima qualità anche in presenza di risorse nella media (il suo "peso" viene assorbito solamente dal tempo impiegato nel rendering). La grande varietà di impostazioni offerta copre sostanzialmente ogni tipo di materiale ed effetto che si voglia rappresentare. Per una scena abbastanza complessa come quella che si è realizzata per il progetto è indubbiamente la scelta migliore, sia per la resa che per i tempi richiesti: YafaRay avrebbe probabilmente impiegato meno tempo ma l'output non avrebbe raggiunto il livello desiderato; LuxRender non sarebbe stato in grado di elaborare la scena; infine, RenderMan avrebbe sicuramente portato a risultati migliori in termini di studio e rappresentazione degli ambienti, ma avrebbe richiesto tempi troppo elevati.
La scelta di Cycles è stata motivata anche dall'elevata quantità di documentazione e tutorial presente, la quale ha permesso di sfruttare al meglio il motore e le sue potenzialità.

5.2 Svolgimento del progetto

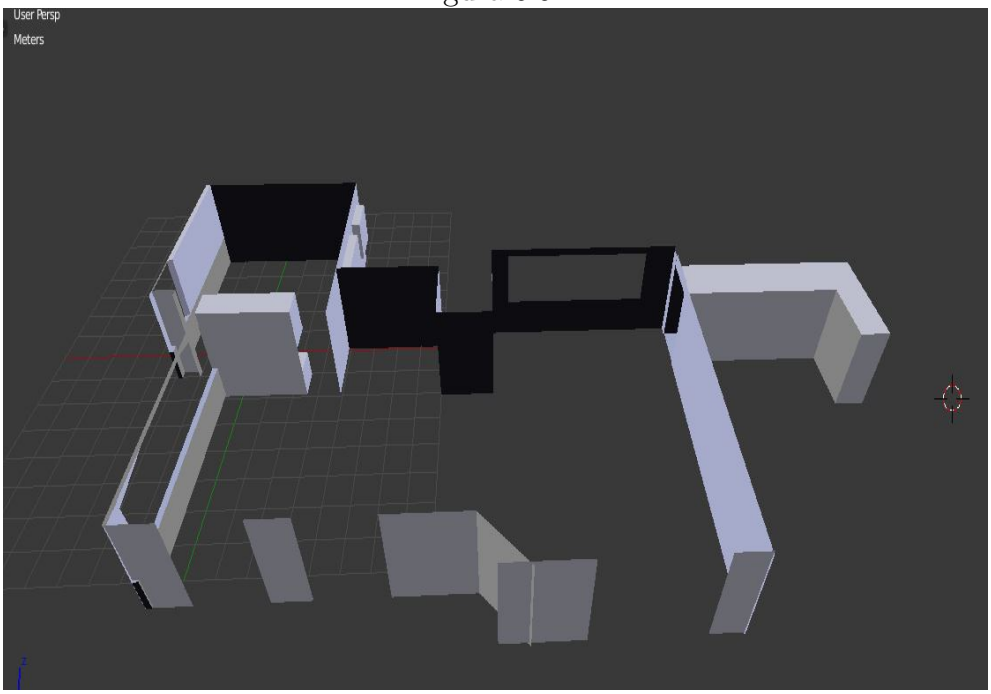
Gli ambienti scelti per la scena sono stati una cucina e un salone: il primo di questi è relativamente ristretto, mentre il secondo è ampio e possiede diversi punti di ingresso per la luce; si è optato per una situazione diurna, con assenza di luci artificiali.
Come prima cosa sono stati effettuati dei video e delle foto degli ambienti da varie angolazioni e in diversi momenti della giornata per poter scegliere quale tra le diverse situazioni fosse la migliore da poter essere rappresentata; in seguito si è passati ad una modellazione iniziale delle stanze e alla suddivi-

sione degli ambienti in modo da definirne le grandezze e le proporzioni; una volta in possesso della struttura è stato possibile procedere alla costruzione del mobilio, anche questa parte fondamentale per comprendere la divisione degli spazi e determinare la grandezza degli oggetti da inserire in proporzione al modello reale.
La parte finale della modellazione ha riguardato la realizzazione degli ultimi oggetti d'arredamento e di decorazione (soprammobili, piante da interni, lampade a muro).
Una volta costruita l'intera scena si è dato il via all'applicazione di materiali e texture, fase sicuramente tra le più interessanti, ma anche impegnativa e lunga.

Fase di modellazione

Le prime cose modellate sono state le mura della casa: è stato scelto questo approccio perché la necessità principale era dividere gli ambienti cercando di mantenere le giuste proporzioni. Per facilitare il raggiungimento dell'obiettivo si è utilizzato il sistema metrico di Blender per determinare l'altezza e la lunghezza delle mura.
Una volta modellate, sono state eliminate le parti relative alle finestre (aggiunte in seguito tramite add-on specifico).

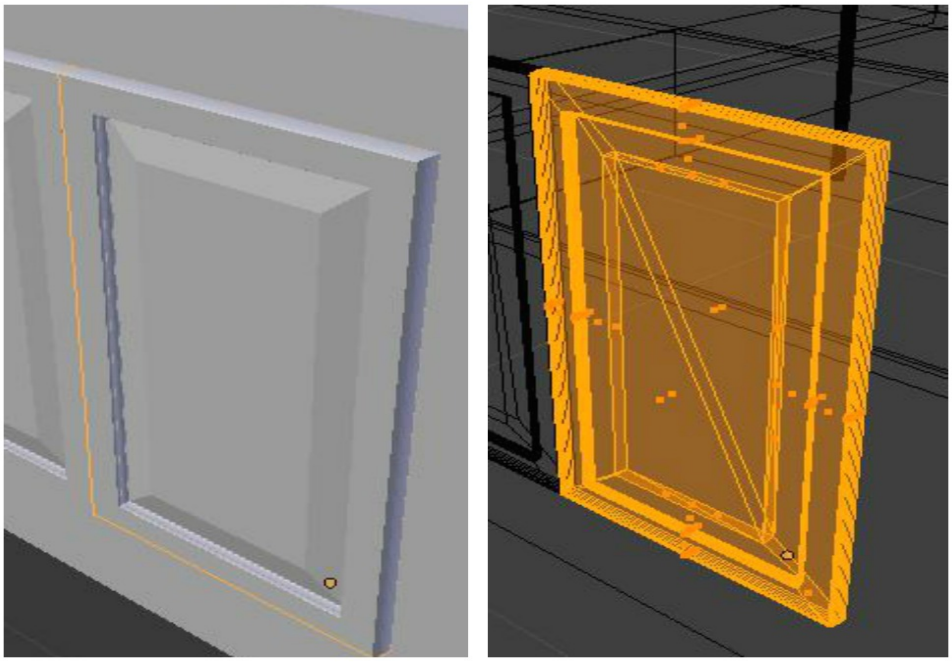
Figura 5.9:



Pavimento e soffitto sono stati aggiunti subito dopo questa fase. In seguito si è passati alla modellazione del mobilio, dedicandosi in particolare a quei mobili utili a suddividere ulteriormente gli ambienti e a descrivere gli spazi; un esempio significativo è la panca presente in cucina, che oltre a fungere da seduta divide in due ambienti la cucina.
Di per sè la sua geometria non è stata difficile da modellare, in quanto si tratta

semplicemente di un cubo scalato secondo esigenza lungo le dimensioni; ciò che ha richiesto più tempo sono state le ante degli sportelli, che possiedono una geometria più complessa:

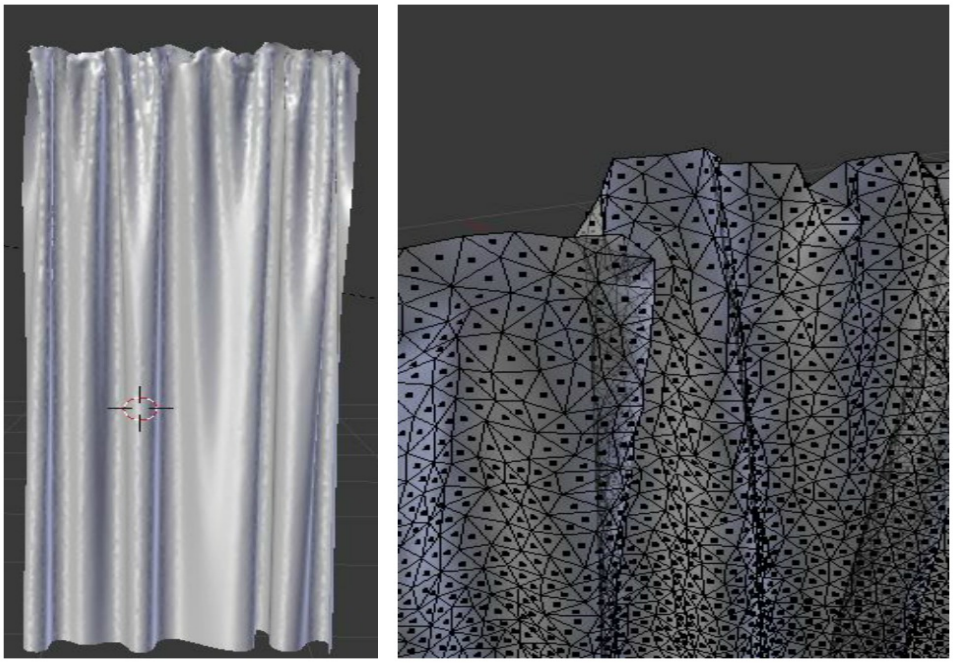
Figura 5.10:



Un altro tipo di oggetto che ha richiesto più lavoro rispetto agli altri e soprattutto l'utilizzo della fisica fornita in Blender è stata la tenda. Per realizzare le tende si è partiti da un semplice piano che è stato suddiviso più volte tramite l'operatore "Subdivide" fornito da Blender, il quale in modalità di editing permette di suddividere una superficie o un oggetto in parti uguali, per quante volte si ha necessità. Questo strumento è particolarmente utile per controllare al meglio la geometria dell'oggetto e lavorare con più precisione su una parte più o meno grande di esso.
In questo caso, la suddivisione è servita per rendere al meglio l'effetto del tessuto e delle sue pieghe; applicando poi il modificatore "Cloth" si sono assegnate le caratteristiche della stoffa al piano. In questo modo, animando la scena, il piano cadeva a terra piegandosi su sè stesso e comportandosi

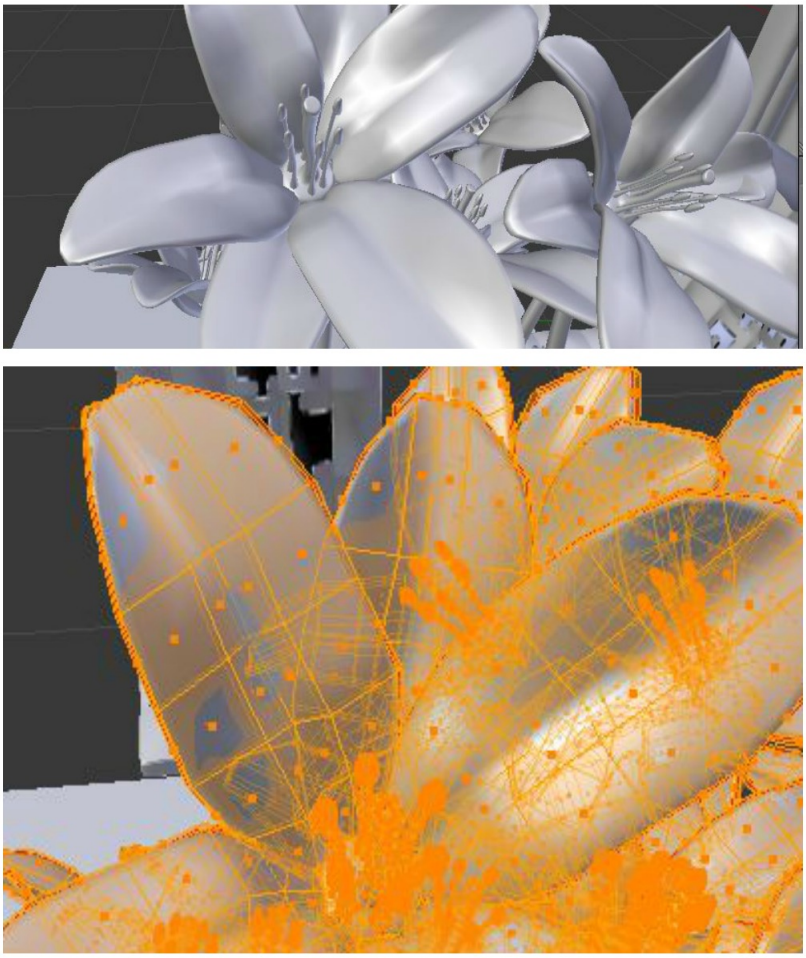
esattamente come il tessuto; è bastato definire quindi un gruppo di vertici in cima al piano a rappresentare il punto di attaccatura della tenda e impostarlo come "Pinning Group", per far sì che il piano rimanesse fisso in quei punti e il resto scendesse con naturalezza, col perno nella parte alta. Come si nota dall'immagine che segue, la suddivisione è stata fatta per un numero abbastanza alto di volte, e ciò ha permesso di raggiungere un risultato realistico.

Figura 5.11:



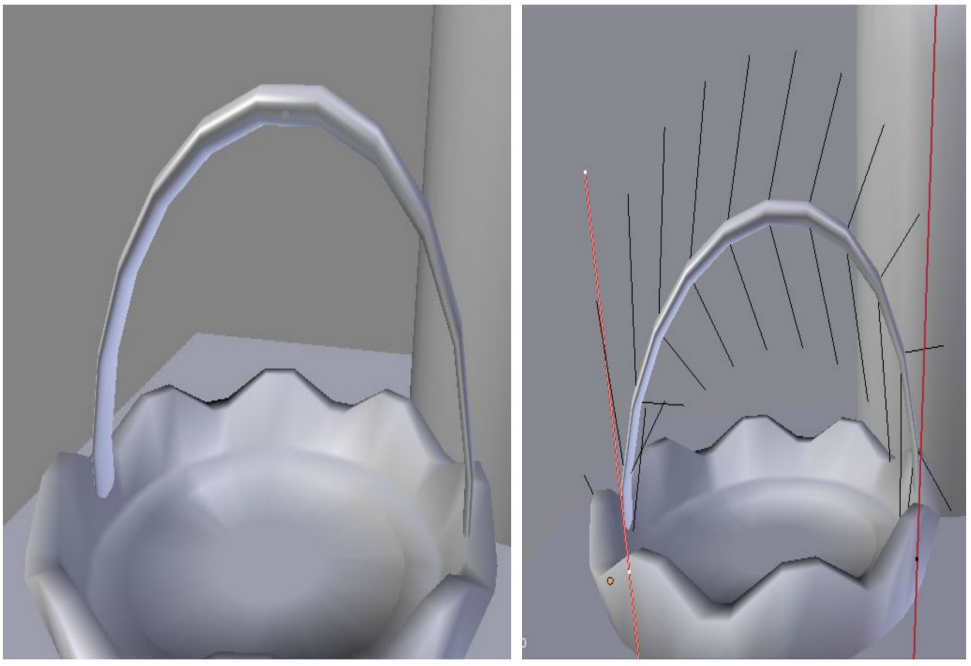
Molti dettagli hanno richiesto più tempo dell'oggetto intero; un esempio sono i fiori, nei quali i particolari risultano essere molto studiati.
In questo caso, sia per la realizzazione dei petali che degli stami, sono state utilizzate geometrie di base a cui è stato applicato il modificatore "Subdivision surface": al contrario dell'operatore di suddivisione questo modificatore, come suggerisce la parola, altera l'aspetto del nostro oggetto, agendo sulle sue facce per dargli un'apparenza più morbida, permettendo così di raggiungere particolari effetti di curvatura.

Figura 5.12:



Nel corso della modellazione sono state utilizzate anche le curve per la realizzazione di alcuni oggetti che presentavano particolari curvature. L'idea è stata quella di modellare la forma dell'oggetto andando ad agire sulla curva (si è utilizzata la curva di Bezier) per poi estruderla sulla base di una forma specificata. Per chiarire il concetto, se si volesse realizzare una tubatura che segue curvature particolari, dopo aver descritto il percorso tramite la curva è sufficiente aggiungere un oggetto di tipo "cerchio" per descrivere la forma che assumerà l'estrusione (eventualmente anche un quadrato o qualsiasi forma geometrica bidimensionale). Per realizzare ciò occorre, dal pannello "data" relativo alla curva, selezionare come "Bevel Object" la nostra forma 2D; a questo punto otterremo il nostro oggetto con la curvatura desiderata.

Figura 5.13:



Per poter applicare materiali e texture al nuovo oggetto bisogna convertire la curva a mesh; ciò è possibile premendo la combinazione di tasti Alt+C. Per realizzare invece la pianta posizionata vicino al televisore è stato utilizzato un particolare add-on di Blender che permette l'aggiunta e la gestione

degli alberi. È un tool molto comodo perché consente, in poco tempo, di creare piante e arbusti con notevoli opzioni di personalizzazione: è infatti possibile determinare il numero di livelli dell'albero (e quindi le ramificazioni), il numero di suddivisioni per ogni nodo e la quantità e sparsità dei rami periferici. Anche le foglie sono gestibili attraverso questo strumento, scegliendone forma e numero.

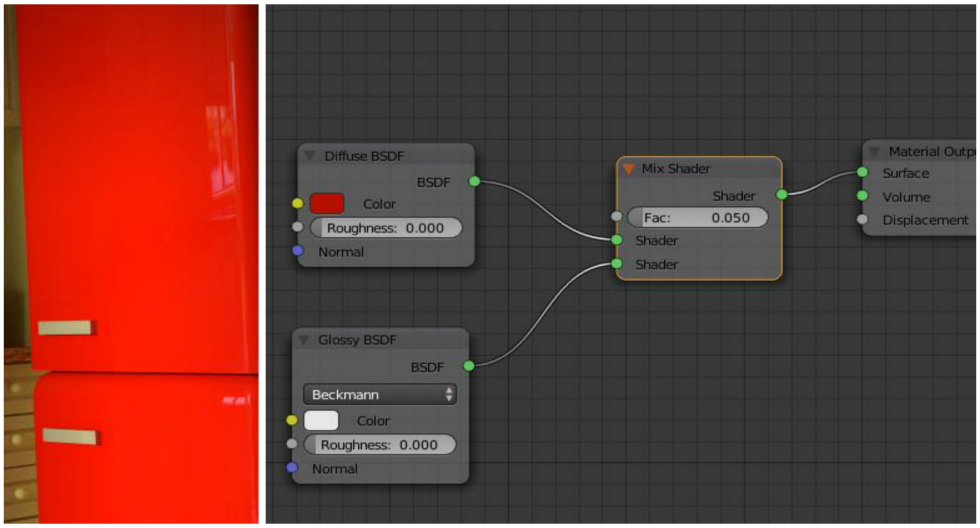
Figura 5.14:

I restanti oggetti sono stati modellati a partire da forme geometriche semplici e combinate tra loro, sulle quali sono state applicate operazioni di scalatura e rotazione sia sull'intero oggetto che su facce, spigoli e vertici di esso, estrusione, suddivisione di superfici e cancellazione di facce, nonché ancora una volta il modificatore "Subdivision Surface".

Fase di scelta e definizione dei materiali

I materiali di base offerti da Cycles sono buoni ma non riescono ovviamente a coprire l'infinita varietà di sfumature e differenze della realtà; utilizzando i tipi di nodi presenti e combinandoli tra loro è però possibile generare potenzialmente ogni aspetto visivo.
Lo studio dei singoli materiali ha richiesto la maggior parte del tempo che è stato dedicato al progetto, e si è svolto in parte seguendo e prendendo spunto da tutorial in rete, in parte tramite prove e tentativi basati su nozioni apprese in precedenza.
Ad alcuni oggetti sono state applicate delle texture per ottenere effetti particolari che con la semplice modellazione e creazione dei materiali sarebbe stato difficile o impossibile raggiungere; un esempio sono i pavimenti, pensati come semplici piani a cui è stata in seguito sovrapposta un'immagine.
Un materiale particolare, non di difficile realizzazione ma che ha garantito una resa peculiare, è stato quello del frigorifero: la superficie si presenta lucida e in grado di riflettere l'ambiente circostante, mantenendo un colore luminoso e forte.
Di seguito l'albero dei nodi relativo alla sua realizzazione:

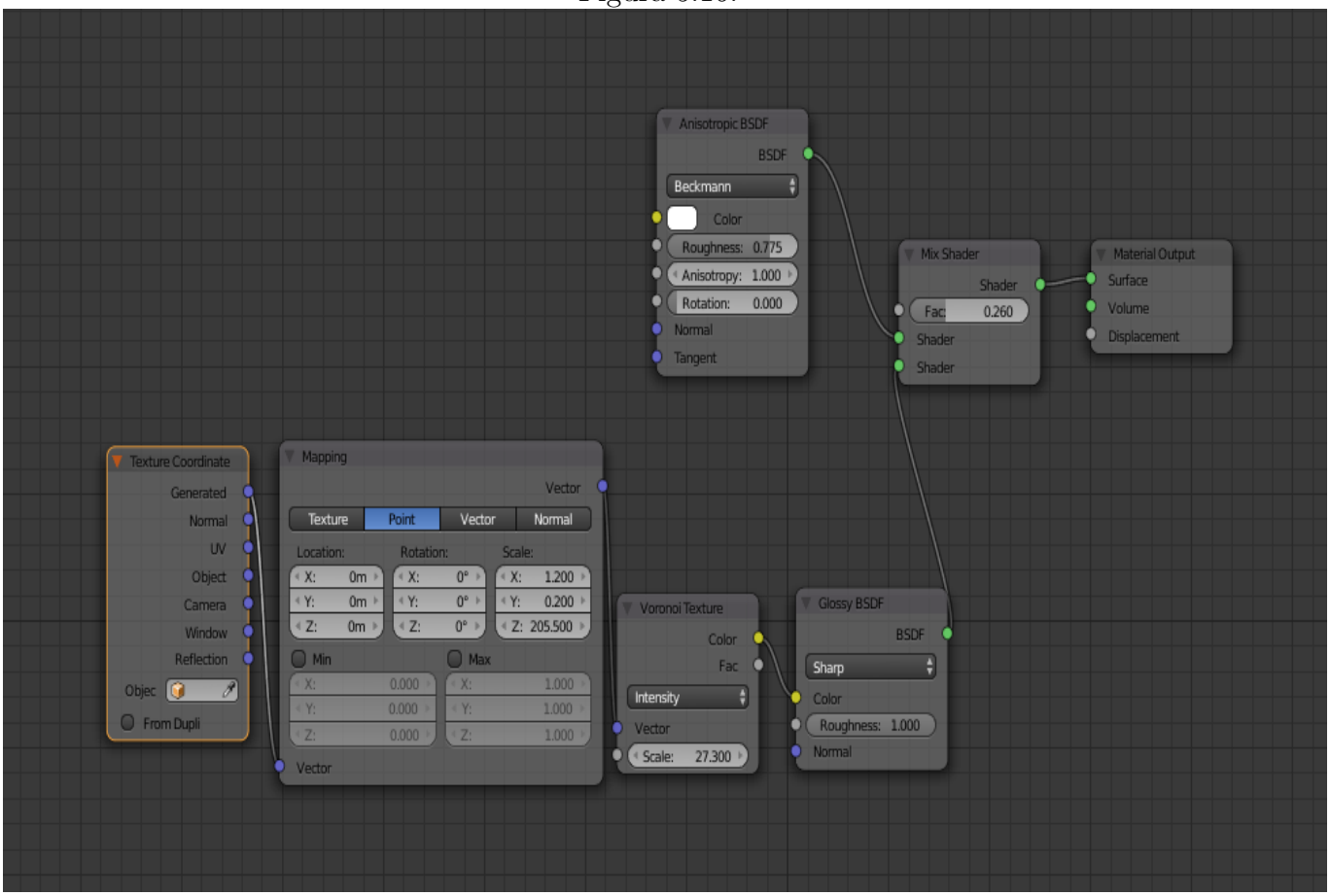
Figura 5.15:



Il colore bianco della componente glossy ci assicura la chiara riflessione dell'ambiente circostante; il valore del fac determina una forte influenza di questa componente.

Passando a qualcosa di più complesso, un materiale interessante è quello della pentola: la sua superficie è anisotropica, e questo le conferisce un aspetto molto particolare.

Figura 5.16:



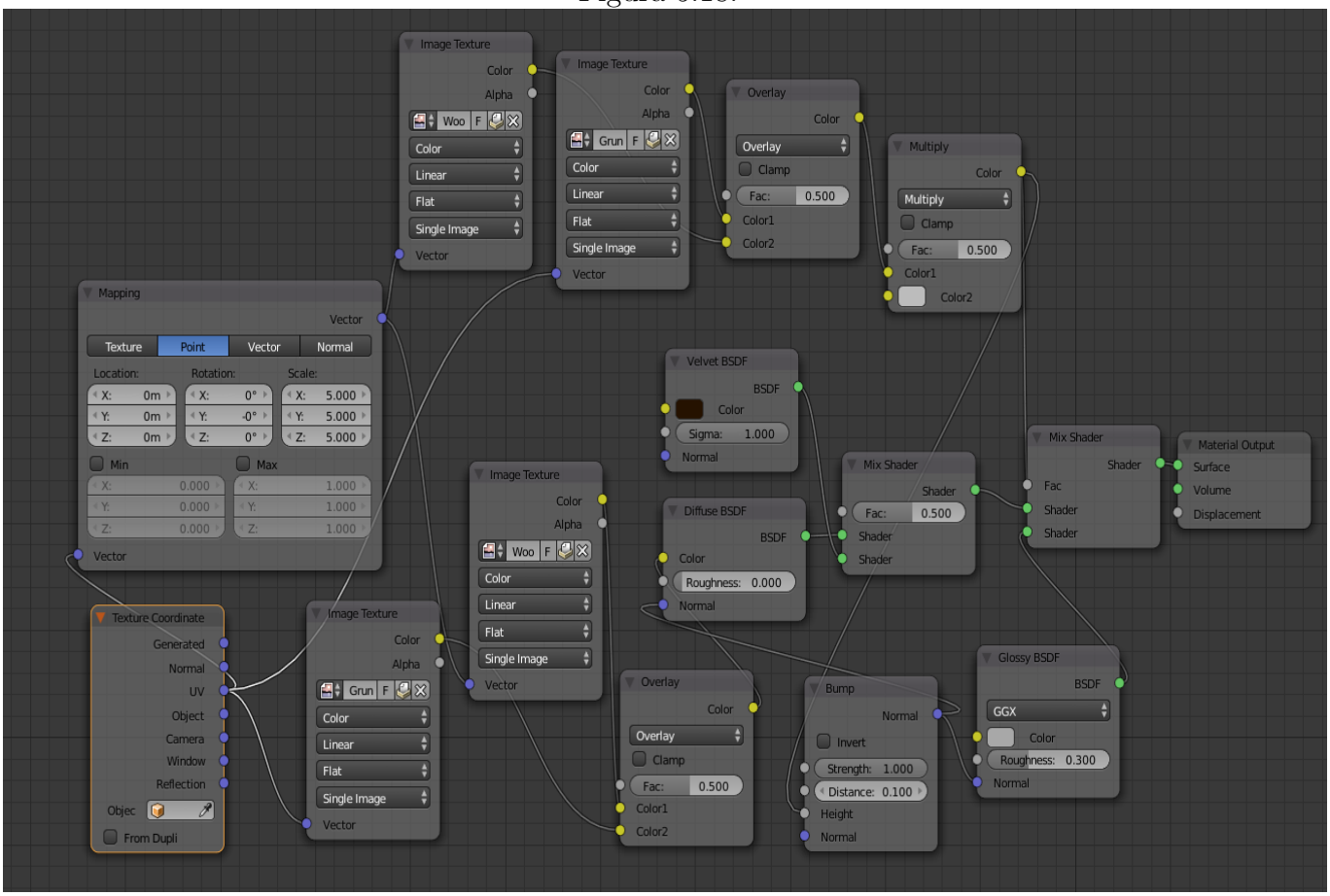
In questo caso l'output è dato da un mix tra materiale anisotropico e glossy; questa componente, inoltre, è legata a una texture di tipo "Voronoi" per dare l'idea che il materiale sia stato lavorato e il metallo battuto. Ovviamente la scalatura è molto grande, in quanto la tessellatura di Voronoi non deve essere visibile a occhio nudo ma soltanto partecipare al raggiungimento dell'effetto complessivo.

Figura 5.17:

Come si vede dall'immagine, l'effetto riflettente è quello tipico del metallo delle pentole.

Decisamente più articolato è il materiale utilizzato per rappresentare il legno di alcuni mobili.

Figura 5.18:



Osservando i nodi possiamo notare che l'output è generato da una componente glossy, una componente derivante dal mix tra materiale diffusivo e velluto, e una terza componente frutto di operazioni di unione tra due image texture; in particolare, queste due texture rappresentano la componente speculare della normale texture. Utilizzare anche la componente speculare di una texture serve per aggiungere un maggiore realismo al nostro materiale, in quanto la componente speculare avrà le stesse caratteristiche della componente diffusiva.

Dall'altra parte troviamo che il nodo *diffuse* ha come componenti due image

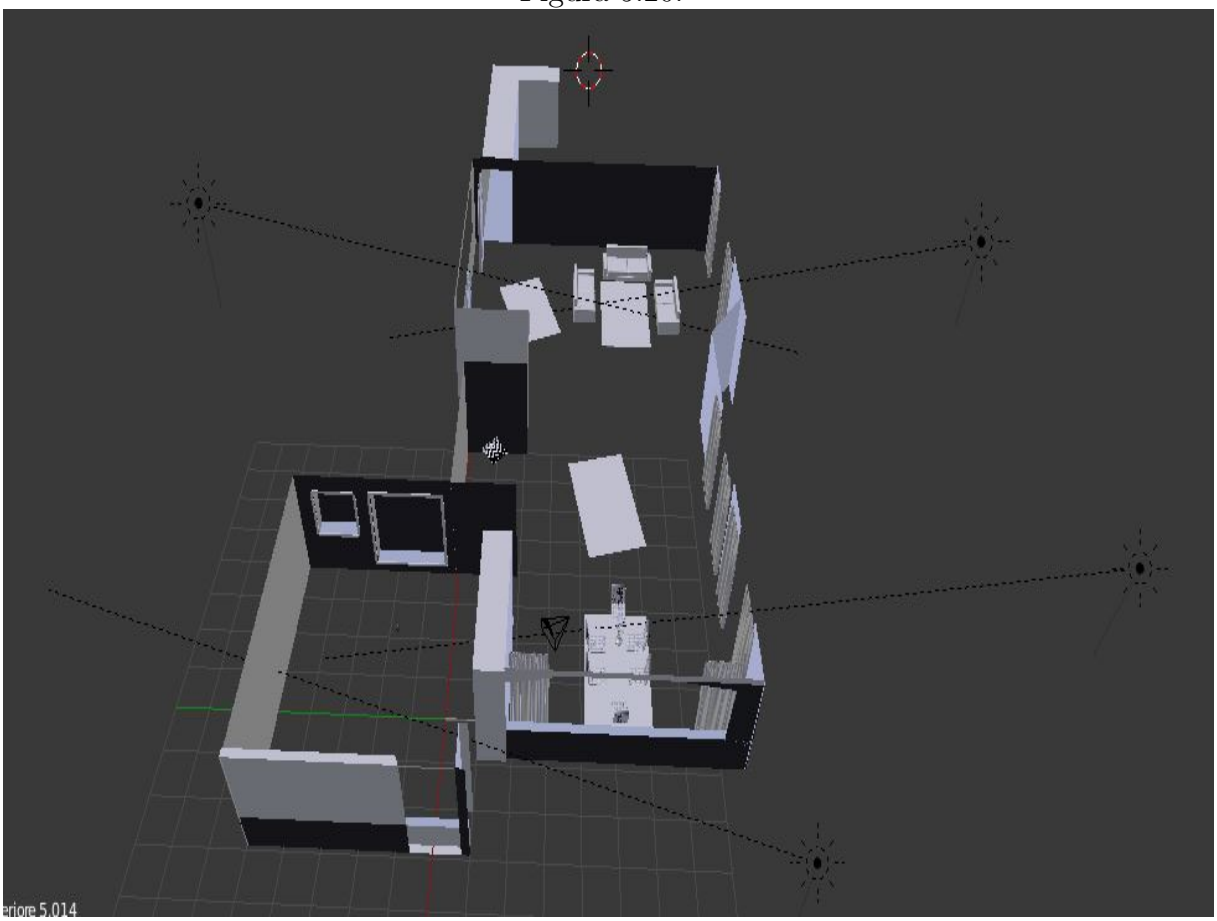
texture, che rappresentano questa volta il colore "normale" della texture; in particolare, una rappresenta un'immagine del legno, mentre l'altra è relativa a un materiale sporco, rovinato e graffiato. Queste due texture sono state sovrapposte e poi utilizzate come input per la componente diffusiva.
Infine, il nodo "Bump" è utilizzato per dare un effetto ancora più realistico al legno: di per sè questo tipo di nodo serve a generare delle perturbazioni sul materiali, "innalzando" le componenti; in sostanza, è in grado di simulare scavature o increspature degli oggetti alterando le interazioni della luce (la geometria 3D dell'oggetto non viene però modificata). In questo caso la superficie non subisce eccessive alterazioni, ma l'effetto finale risulta essere molto realistico.

Figura 5.19:

Fase di posizionamento delle luci e costruzione dell'ambiente

Una volta completate tutte le fasi precedenti si è potuto procedere al posizionamento della luce e in particolare all'analisi del modo migliore con cui generarla.
Inizialmente, soprattutto in seguito alla scelta effettuata per la scena del tirocinio, si è optato per utilizzare un'immagine HDR: grazie agli strumenti di Cycles è possibile estrarne le informazioni luminose al meglio e utilizzarle per controllare a proprio piacimento la luce dell'ambiente. In particolare, variando l'intensità e l'esposizione dell'immagine, è possibile individuare la posizione dei punti luminosi presenti in essa, spostarli e in generale gestirne la luminosità a seconda dei bisogni legati a ciò che vogliamo rappresentare. L'utilizzo di un'immagine di questo tipo influisce ovviamente sul tempo richiesto dal rendering in quanto fornisce informazioni sì utili, ma al contempo più complesse rispetto a un'immagine con un formato normalmente usato; inoltre, non sempre è facile controllare la posizione della luce che fornisce e raggiungere l'effetto desiderato. Nel caso di un ambiente mediamente complesso come quello preso in esame, in seguito a diversi test di resa, si è preferito scegliere una semplice immagine in formato png e posizionare delle luci di tipo "Sun" all'esterno della casa, gestendone la direzione, il colore e in particolare l'intensità, e ciò ha permesso un maggiore controllo del risultato rispetto alla scelta precedente.
In particolare, sono state inserite due luci di questo tipo dalla parte del salone che possedeva più finestre, in modo da poter far entrare più luce, e anche perché facendo così anche la cucina sarebbe stata illuminata.

Figura 5.20:



Le due luci a destra rappresentano il sole, e sono quindi quelle che hanno intensità maggiore delle altre; la luce a sinistra ha un'intensità quasi pari a zero, e serve soltanto per creare l'effetto della luce esterna, mentre quella posizionata obliquamente possiede un valore intermedio, utilizzata per illuminare la cucina ma in modo minore rispetto al salone.

Di seguito alcune immagini di diversi spazi della casa renderizzate.

Figura 5.21: In questa immagine è ben visibile l'effetto di riflessione del vetro sulla parete, reso in maniera ottimale dal motore. Non sono rappresentate le caustiche, ma essendo la luce non eccessivamente forte l'effetto è comunque realistico. Il materiale degli altri oggetti è l'argento, e le riflessioni su queste superfici sono state trattate accuratamente. Il velluto di cui è fatta la tenda presenta colorazioni diverse per via dell'incidenza della luce.

Figura 5.22:

Figura 5.23: Questa immagine è stata scelta per esaltare la resa realistica del legno, che possiede alcune increspature dovute alla lavorazione. Un altro effetto reso in maniera ottimale sono le ombre sul soffitto vicino alle finestre. Sulla superficie del tavolo è possibile notare un riflesso colorato, in particolare sul bordo, dovuto all'interazione della luce con le tende.

Figura 5.24:

Figura 5.25: In questa immagine possiamo notare la compresenza di una zona illuminata con una in penombra.

Figura 5.26:

Figura 5.27:

Figura 5.28:

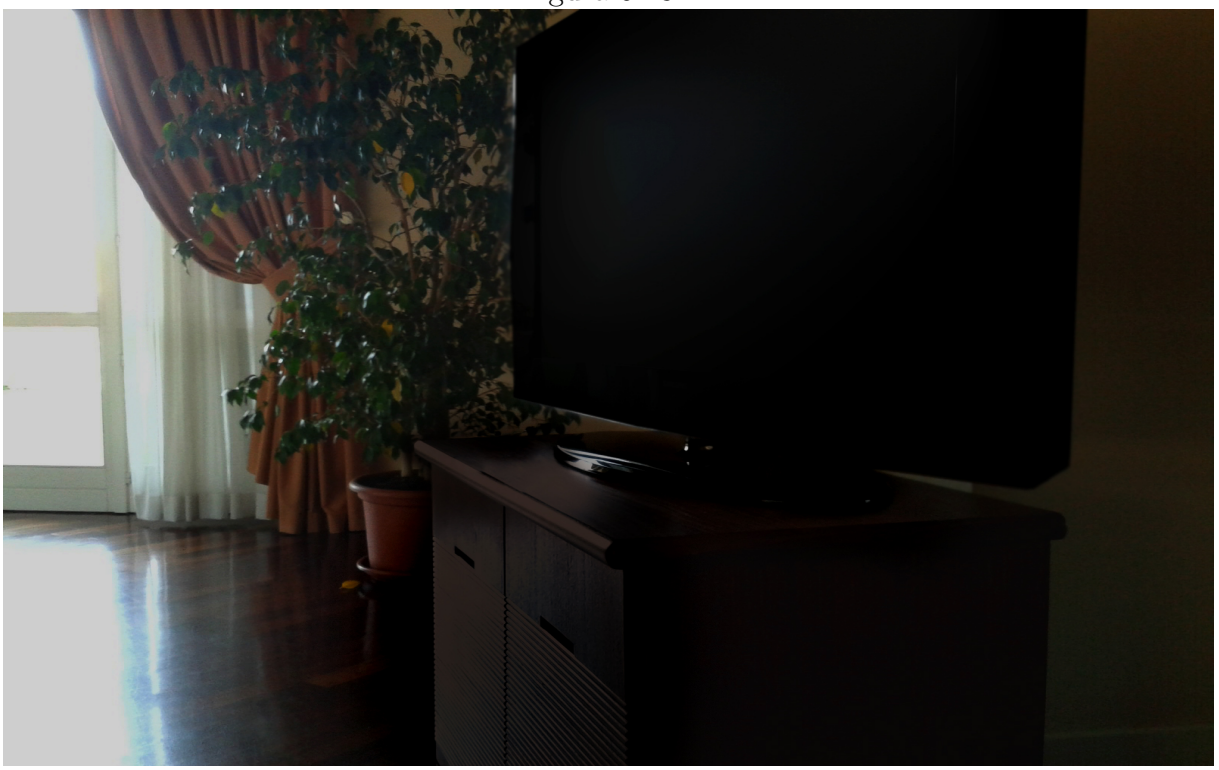


Figura 5.29: In questo caso si può ben notare come viene rappresentato il velluto, sia su una superficie piccola come quella dei cuscini sia su una più grande come quella dei divani. Questo materiale risulta essere molto scuro nelle zone in cui è in ombra ma, al contrario, dà un effetto molto luminoso. Il "sigma"(un valore che controlla l'influenza della parte scura del velluto)dei due velluti è differente, e ciò si riflette nella resa: i cuscini, possedendo un velluto con sigma di valore massimo, presentano una differenziazione più accentuata nel passaggio tra chiaro e scuro rispetto ai divani.

Figura 5.30:

Figura 5.31: Questa immagine è stata scelta per via della varietà dei materiali presenti. Come si può notare, ci sono tre metalli differenti per proprietà e quindi per aspetto: il metallo del lavandino è opaco e le riflessioni non sono precise; il metallo della pentola, già analizzato in precedenza, riesce bene a gestire l'ambiente circostante secondo la proprietà anisotropica che possiede; il materiale della padella invece è in grado di riflettere perfettamente il piano di cottura sotto di essa, essendo un metallo lucido.

Figura 5.32:

Figura 5.33: Da questa angolazione è possibile vedere più oggetti della cucina, in particolare il forno, i fornelli, la macchina del caffé e il frigorifero. L'immagine è stata scelta, riprendendo il discorso sul materiale fatto prima, per sottolineare come le riflessioni siano trattate in modo realistico: il frigorifero infatti riflette la finestra e parte del piano cottura, simulando il comportamento reale della luce.

Figura 5.34:

Figura 5.35: Dall'altra parte della cucina troviamo la zona in penombra in cui la luce arriva debolmente, in quanto la finestra si trova soltanto da un lato. È interessante notare l'ombra più debole della sedia sul muro e il materiale vetroso del portafrutta che lascia vedere il contenuto ma al contempo ne altera leggermente le forme a causa della rifrazione.

Figura 5.36:

Figura 5.37:

Figura 5.38:

Figura 5.39:

Figura 5.40:

Conclusioni

Analizzando il lavoro effettuato e i risultati ottenuti, si può concludere che l'obiettivo di realismo nella scena realizzata è stato raggiunto. Il tempo dedicato al progetto ha portato ad un apprendimento dettagliato del funzionamento di Blender, nonché delle conoscenze di base della teoria del rendering e dell'illuminazione, indispensabili per muoversi in questo campo.

Essendo questo il primo vero approccio al mondo della modellazione 3D i risultati raggiunti possono essere considerati buoni seppur non perfetti, in quanto il livello di dettaglio può essere ancora aumentato e perfezionato così come la precisione nel rispettare il modello reale.
La realizzazione di questo tipo di progetto si è rivelata molto interessante in ogni sua parte, in quanto ha portato alla conoscenza di diversi aspetti della realtà e ha consentito di perfezionare continuamente le tecniche di modellazione.

Com'è ovvio che sia sono state riscontrate delle difficoltà, legate in particolare al primo approccio coi nuovi motori di rendering e alla resa fedele del modello reale che si è preso in considerazione: non sempre è stato facile e immediato ricostruire gli oggetti del modello, soprattutto per quelli che possedevano forme particolari e ricercate; per questo motivo sono state introdotte diverse semplificazioni di modellazione, pur mantenendo la struttura e il mobilio fondamentali. Inoltre, sono state fatte delle modifiche a diversi oggetti d'arredamento e mobili sia per esplorare una maggior varietà di ma-

teriali, sia per facilitare la resa anche in termini di tempo di rendering.

Per la parte teorica e matematica, avendo già avuto un'infarinatura degli argomenti nel corso di Computer Graphics, lo studio è stato più semplice a livello di apprendimento, ma comunque corposo per il tempo dedicato.

Bibliografia

Alvaro Luna Bautista (2013)- *Understanding photon mapping* - <https://docs.google.com/document/d/1OZFmrUXsopMsuhHU_IVSo2FBh0XQx_hG7ryrrfBWgks/edit >

Eric P. Lafortune & Yves D. Willems - *BI-DIRECTIONAL PATH TRACING*, Department of Computing Science, Katholieke Universiteit Leuven, Belgium

Lazzaro D.(2015) - *Computer Graphics*, Ingegneria e Scienze Informatiche, Alma mater studiorum-Università di Bologna

Matt Pharr & Greg Humphreys(2010) -*Physically Based Rendering-From theory to implementation-Second Edition* , Morgan Kaufmann Publishers

Zack Waters - *Photon mapping* <http://web.cs.wpi.edu/∼emmanuel/courses/cs563/write_ups/zackw/photon_mapping/PhotonMapping.html>

Radiosity Rendering(chapter 5) <http://www.cs.bath.ac.uk/ pjw/NOTES/75-ACG/ch5-radiosity.pdf>

Cycles documentation
<http://www.blender.org/manual/render/cycles/index.html>

LuxRender user documentation
<http://www.luxrender.net/wiki/Main_Page>

YafaRay user guide
<http://www.yafaray.org/documentation/userguide>

Pixar's RenderMan user guide
<http://renderman.pixar.com/resources/current/RenderMan/home.html>

Scratchapixel - *Introduction to Ray Tracing: a Simple Method for Creating 3D Images* - <http://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-ray-tracing/raytracing-algorithm-in-a-nutshell >

Grammofono realizzato da: thesage -
<http://www.blendswap.com/user/thesage>

Textures: <http://www.textures.com/>

Ringraziamenti

Desidero innanzitutto ringraziare la Dott.ssa Damiana Lazzaro per avermi dato l'opportunità di realizzare questa tesi e per avermi seguita durante le diverse fasi di stesura e del progetto, mostrandosi sempre disponibile per qualsiasi consiglio e chiarimento.

Ringrazio anche i miei zii Orietta e Loris per aver messo a disposizione casa loro per essere utilizzata come soggetto del progetto realizzato.

Un ringraziamento particolare va a Elisabetta, Francesca, Lisa e Luca, le persone che più che mai sento vicine in questo momento e che hanno sempre creduto in me.

Infine,vorrei ringraziare tutti gli amici che ogni giorno mi supportano (e sopportano) e che, in un modo o nell'altro, vicini o lontani che siano, mi rendono fiera di essere loro amica. Non riuscirò mai a scrivere i nomi di tutti. Sono stati tre anni molto particolari, a volte sofferti, ma che rivivrei di nuovo perché mi hanno regalato molti dei momenti più belli che conservo: le risate e i momenti di confronto, ma anche l'ansia per gli esami; rivivrei anche quella, perché è stato proprio in quei periodi che ci siamo dimostrati uniti, lasciando nessuno indietro ma aiutandoci l'un l'altro per poter dire, alla fine di ogni sessione, di esserci riusciti, tutti e insieme, arrivando finalmente alla fine di un percorso.

Grazie di tutto questo, di esserci sempre stati, e qualsiasi strada sceglierete vi auguro di essere pronti e determinati per poter affrontare qualsiasi cosa, come abbiamo fatto fin'ora.