Lab 4 - Ray Tracing

4.1 Parte 1 - openGL

In questa esercitazione si richiedeva di implementare la gestione dei seguenti elementi:

- **Shadow Rays**
- **Reflection Rays**
- **Soft Shadows**

L'algoritmo di Ray Tracing viene richiamato alla pressione del tasto R della tastiera e viene richiamato per ogni pixel della viewport. La funzione TraceRay crea un oggetto Hit che serve per determinare il punto di incidenza con la scena. Questo oggetto deve essere fornito alla funzione CastRay che crea effettivamente il raggio e provoca l'intersezione di questo raggio con un oggetto della scena. La classe RayTree viene richiamata in questo punto e permette di disegnare i diversi raggi primari, riflessi e d'ombra alla pressione del tasto T.

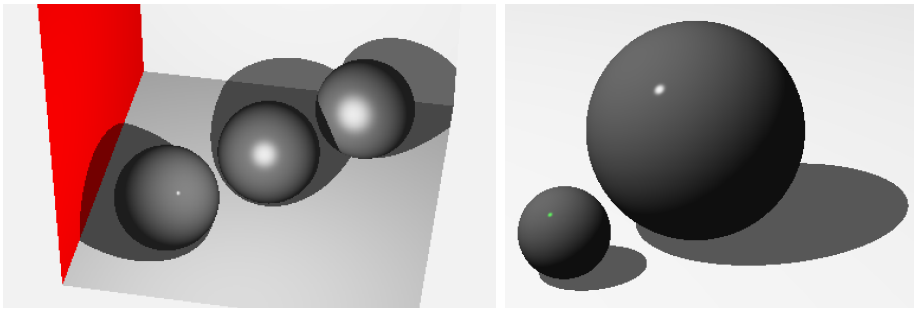
L'algoritmo vero e proprio comincia con la creazione del colore di risposta riempito inizialmente con il colore del background. In caso in cui non vi sia alcuna intersezione, l'algoritmo ritorna questo colore, altrimenti procede con l'aggiunta di tutti i contributi luminosi. Si ricavano tutti i valori utili al materiale, punto di intersezione e normale alla superficie nel punto di intersezione. Se il materiale intersecato è emittente, allora si ritorna direttamente il colore della luce senza effettuare altri calcoli. Successivamente si procede con la logica delle ombre.

4.1.1 Shadow Logic

Quando si esegue l'applicativo è possibile definire il numero di campioni d'ombra che si devono prendere, con il parametro `num_shadow_samples`. Quindi l'algoritmo per ogni luce della scena ripete la logica un numero di volte pari a questo parametro e poi fa la media di tutti i contributi che ottiene. La logica distingue i casi in cui il numero di campioni da ottenere sia pari a 0 o maggiore di 0. Se è 0 allora non vi devono essere ombre, quindi si lascia ciò che era già nel template: si fa direttamente il calcolo di Phong con la funzione `Shade` per ottenere il colore dell'oggetto intersecato dipendentemente dal colore e dall'intensità della luce.

Se invece vi è almeno un campione d'ombra, è necessario creare uno shadow ray che parta dal punto di incidenza sulla scena e che abbia come direzione la luce attualmente considerata. A questo punto, si calcola il punto di incidenza nuovo e si confronta la lunghezza del nuovo raggio con la distanza dalla luce. Se sono uguali allora il punto è in luce, altrimenti vi è qualcosa in mezzo e questo punto non ottiene il contributo luminoso della luce considerata. Se la distanza è praticamente la stessa, allora si applica il contributo richiamando `Shade`.

Un esempio del risultato dopo l'applicazione di un campione d'ombra:



(a) (b)

4.1.2 Reflection Logic

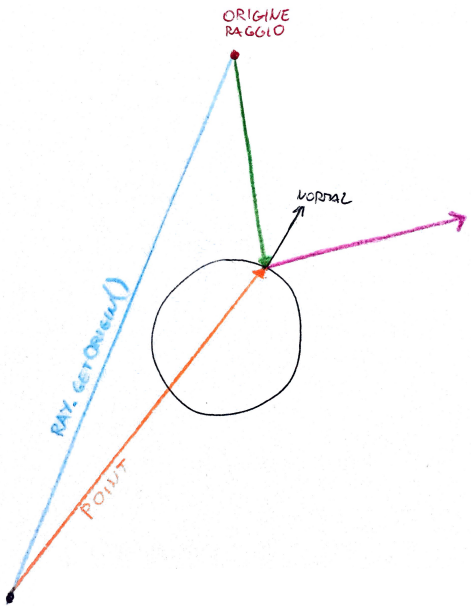


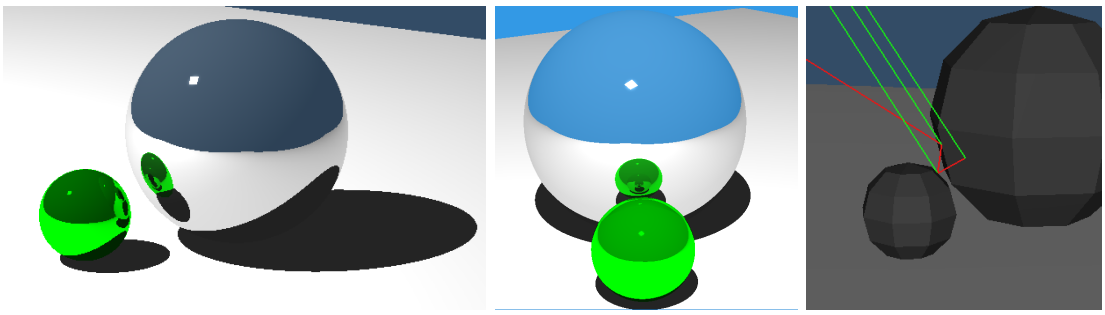
Figura 4.1

La logica dei riflessi è molto più semplice: prima di tutto si controlla se vi è una componente riflettente nel materiale dell'oggetto intersecato.
Se l'oggetto non è riflettente l'algoritmo termina, altrimenti si calcola il raggio riflesso e ricorsivamente si richiama `TraceRay` con il nuovo raggio finché vi sono rimbalzi (`num_bounces`).
Nell'immagine 4.1 è illustrata la situazione di un raggio incidente (verde) e il suo riflesso (viola). Il raggio incidente è ottenibile con la differenza tra il vettore *point* e il vettore che rappresenta l'origine del raggio.

Il raggio riflesso è ottenibile con la formula

$$R = 2 < n, l > n - l$$

ovvero due volte il prodotto scalare tra normale e raggio incidente, moltiplicato per la normale; il vettore risultante deve essere sottratto al raggio incidente.



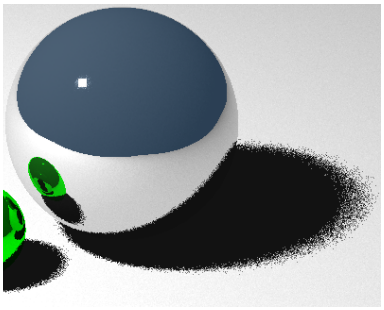
(a) Due sfere riflettive (b) Background più chiaro (c) RayTree di debug

[1]

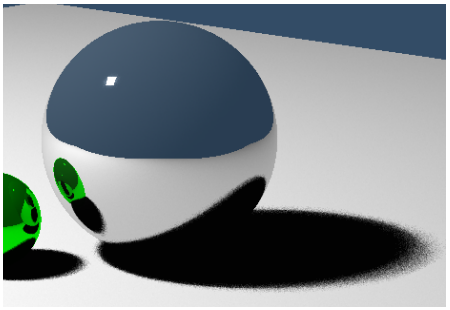
[1](Nota: fin da subito è stato modificato il file `reflective_spheres.obj` sostituendo il colore rosso riflettente con il verde)

4.1.3 Soft Shadows

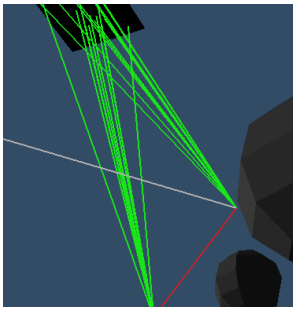
Le soft-shadows sono state ottenute modificando la logica delle ombre e tracciando tanti raggi che intersecano randomicamente l'area della luce considerata quanto indicato con il parametro `num_shadow_samples`. Ogni contributo viene sommato e al termine del ciclo che interseca tutti gli shadow ray viene fatta la media di questi contributi.



(a) 4 campionature



(b) 16 campionature

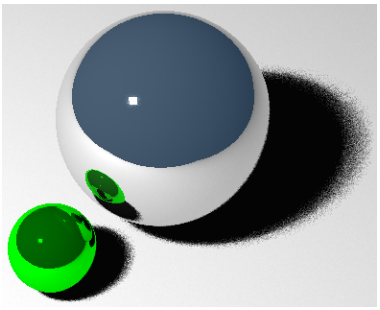


(c) RayTree di debug

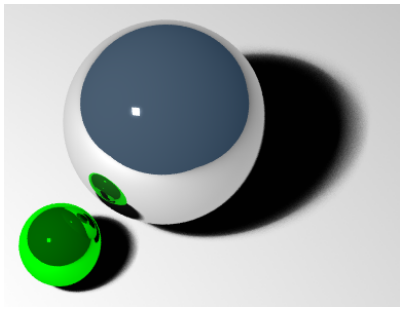
Antialiasing

Per avere un migliore riscontro visivo nella rappresentazione delle soft-shadow, si è scelto di studiare un metodo per rappresentare un effetto più sfumato e quindi di applicare un filtro di antialiasing. Il metodo più semplice è il seguente: invece di sparare un raggio al centro del pixel attuale, si sparano più raggi in posizioni randomiche e si fa la media dei contributi ottenuti.

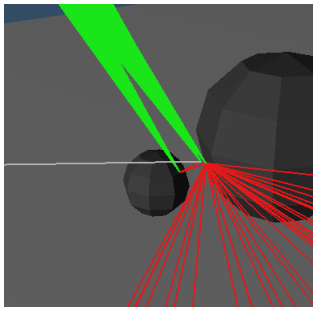
Nella classe `glCanvas` vi è la funzione `TraceRay` che richiama l'omonima funzione della classe `RayTracer`. In questa funzione è stata aggiunta una logica che cicla tante volte quanto specificato nel parametro `num_antialias_samples` (che è stato aggiunto ad `argParser.h`). Ad ogni ciclo si randomizza la posizione del raggio e si da in pasto alla funzione `TraceRay`. Questo algoritmo di antialiasing ha un grande costo computazionale poiché aumenta esponenzialmente il numero di raggi e quindi il tempo di rendering.



(a) Prima

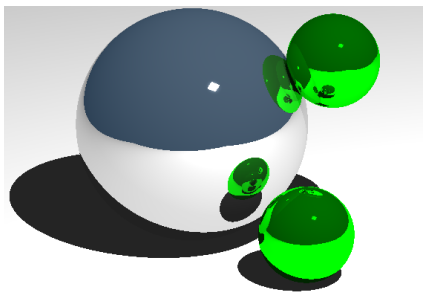


(b) Dopo 30 campioni di antialiasing

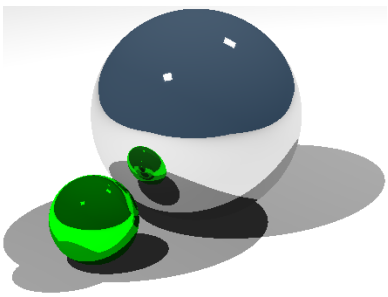


(c) RayTree di debug

4.1.4 Sperimentazioni



(a) Due palline verdi



(b) Due luci

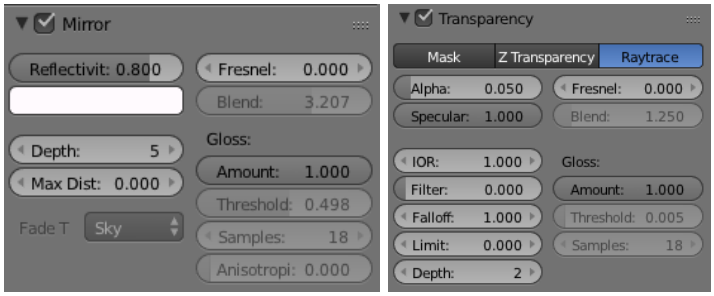
(c) Tre luci

4.2 Parte 2 - Blender

Per questa parte è stata modificata la scena contenente la batteria cambiando materiali e aggiungendo oggetti riflettenti e trasparenti.

Inizialmente è stato modificato il materiale dei fusti della batteria, selezionando solo la parte delle pelli e rendendola in gran percentuale trasparente attivando la modalità `Transparency` con un alpha pari a 0.2. In questo modo tutti i fusti hanno almeno due materiali: il bianco dei fusti, il nero dei ring (anelle di ferro che tirano le pelli) e il trasparente delle pelli.

Successivamente sono state aggiunte alla scena 3 NURBS patches a cui è stato applicato il materiale *Specchio* che riflette con un fattore di riflessione dell'80%.



(a) Materiale riflettente (b) Materiale trasparente

Tutta la scena è stata rinchiusa in un grande cilindro con materiale rosso che crea una sorta di stanza ed è stata illuminata con una luce centrale bianca.

Figura 4.2

Successivamente, la scena è stata arricchita con due oggetti posizionati dietro lo specchio centrale. A sinistra è stata posizionata una sfera con materiale riflettente e una luce blu che la illumina; mentre a destra è stato posizionato un toro con material trasparente con una luce verde che lo illumina dall'alto.

La scena prima del render non presenta le trasparenze, né i riflessi:

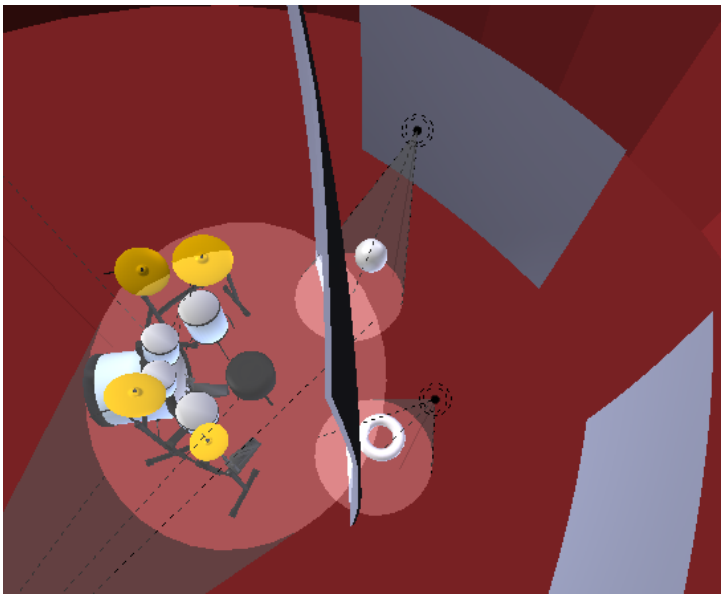


Figura 4.3

Il risultato del render invece porta con se tutti gli effetti speciali dovuti ai materiali e alle luci:

Figura 4.4