REPORT PER LAB_04

Matteo giri

Introduzione

Questo è il report per l'esercizio di laboratorio **LAB_04**. Tutti e tre i task dell'esercitazione sono stati implementati e spiegati qui sotto.

Dettagli realizzativi

STEP 1: sviluppare la gestione degli shadow rays per la generazione di hard shadows
Per sviluppare la gestione degli hard shadows bisogna andare a costruire uno shadow ray dal punto colpito dal raggio alle varie sorgenti di luce (tutte considerate come puntiformi) ed andare a vedere se il raggio arriva al punto luce senza colpire altri oggetti in scena. Se il raggio arriva al punto luce allora la luce contribuisce al colore del punto considerato, altrimenti no.

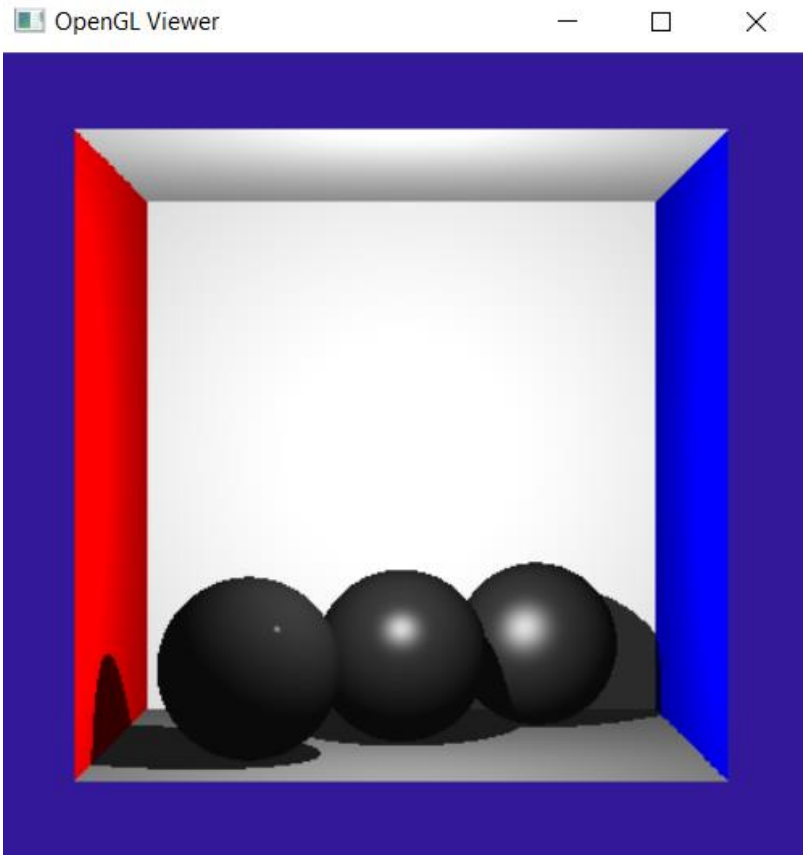
In particolare, per ogni luce nel sistema si vanno ad effettuare le seguenti operazioni:

```
else{ //HARD SHADOWS
    Vec3f pointOnLight = f->computeCentroid();
    Vec3f dirToLight = pointOnLight - point;
    dirToLight.Normalize();

    // altrimenti
    //    la luce i non contribuisce alla luminosita' di point.

    // creare shadow ray verso il punto luce
    n_ray = new Ray(point, dirToLight);

    // controllare il primo oggetto colpito da tale raggio
    new_hit = new Hit();
    colpito = CastRay(*n_ray, *new_hit, false);

    if (colpito) {
        n_point = n_ray->pointAtParameter(new_hit->getT());
        //calcola il vettore distanza fra il punto colpito dal raggio e il punto sulla luce
        dista.Sub(dista, n_point, pointOnLight);

        // se e' la sorgente luminosa i-esima allora
        // calcolare e aggiungere ad answer il contributo luminoso
        if (dista.Length() < 0.01) { //il punto colpito coincide con la luce
            if (normal.Dot3(dirToLight) > 0) {
                Vec3f lightColor = 0.2 * f->getMaterial()->getEmittedColor()*f->getArea();
                answer += m->Shade(ray, hit, dirToLight, lightColor, args); //Shade è il modello di illuminazione di phong
            }
        }
    }
}
```

Per prima cosa si va a calcolare il centroide della luce considerata (che indipendentemente dal tipo di sorgente luminosa deve infatti essere considerata come puntiforme) e si calcola la direzione della luce dal punto che stiamo considerando. Poi si va a creare lo shadow ray vero e proprio andando a costruire il raggio che parte dal punto *point* e va verso la direzione della luce, e poi si casta il raggio tramite la funzione *CastRay.* Questa funzione costruisce un raggio che attraversa la scena e va a cercare il primo oggetto che viene colpito dal raggio. Se un oggetto (che può essere anche il punto luminoso) è stato colpito dal raggio allora la variabile *colpito* sarà true. Fatto questo si va a controllare l'oggetto che è stato colpito dal raggio: se la distanza fra il punto colpito dal raggio e il punto luce considerato è minore di una certa soglia (0.01) allora significa che il raggio ha colpito la luce senza imbattersi in altri oggetti; quindi, significa che la luce contribuirà al risultato.

Il contributo della luce viene calcolato secondo il modello di illuminazione di Phong, andando a passare alla funzione che lo calcola (*Shade*) il raggio *ray* che ha colpito il punto *hit* considerato, la direzione verso la luce e il colore della luce.

Risultato:



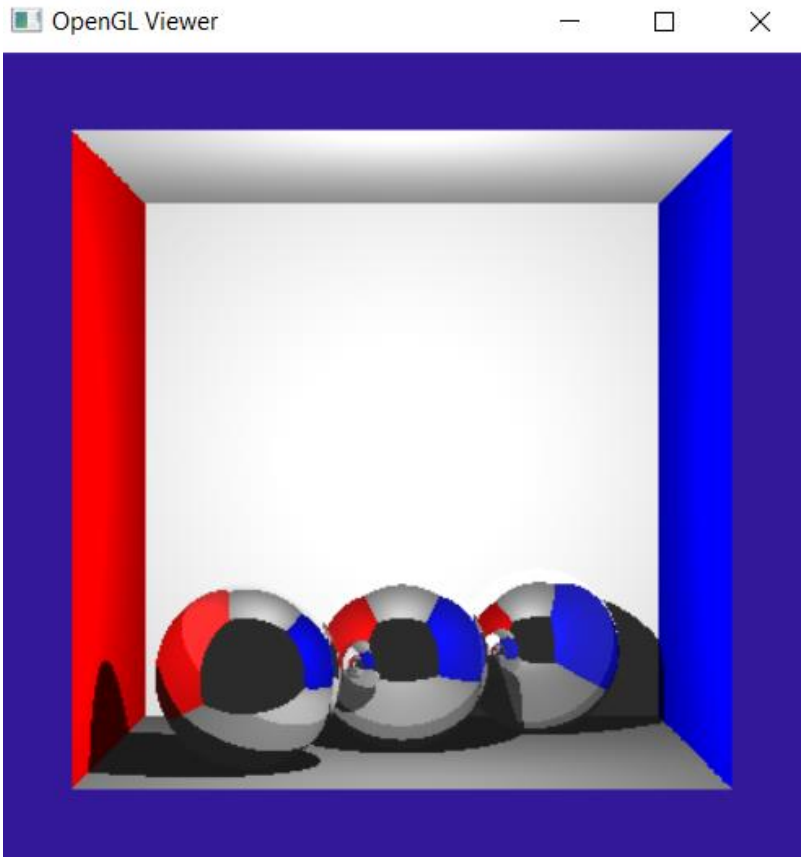
STEP 2: sviluppare la gestione ricorsiva dei reflection rays

Per sviluppare la gestione ricorsiva dei reflection rays bisogna andare a verificare se il materiale dell'oggetto del punto colpito è riflettivo, e in caso positivo andare a castare il raggio riflesso e aggiungerne il suo contributo al risultato:

```
// se (il punto sulla superficie e' riflettente & bounce_count>0)
//se questo punto della superficie non è per nulla riflettente non va usato il raytracing
if (reflectiveColor.Length() != 0 && bounce_count > 0) {
    // calcolare ReflectionRay  R=2<n,l>n -l
    Vec3f vRay = ray.getDirection();
    Vec3f reflectionRay = vRay - (2 * vRay.Dot3(normal)*normal);
    reflectionRay.Normalize();

    Ray* new_ray = new Ray(point, reflectionRay);

    //invocare TraceRay(ReflectionRay, hit,bounce_count-1)
    // aggiungere ad answer il contributo riflesso
    answer += TraceRay(*new_ray, hit, bounce_count - 1)*reflectiveColor;
}
```

Per prima cosa si va a vedere se il materiale ha una componente riflessiva perché, se non ha questa componente allora è inutile andare a creare il raggio riflesso. Si controlla anche il *bounce_count* che serve per andare a limitare il numero di "rimbalzi" che il raggio può effettuare su oggetti riflessivi (per evitare che il raggio rimbalzi all'infinito). Se l'oggetto ha una componente riflessiva allora si va a calcolare il raggio riflesso tramite la formula vista a lezione e si richiama ricorsivamente *TraceRay* con questo nuovo raggio (diminuendo il *bounce_count* ogni volta). Il risultato di traceRay viene puoi aggiunto ad *answer* che è poi il colore finale che verrà inserito nel pixel considerato.

Risultato (con numero di rimbalzi = 5):



STEP 3: Implementare una strategia per la gestione di soft shadows mediante risorse luminose ad area anziché puntiformi

Per implementare le soft shadows bisogna andare a castare più shadow rays verso le luci che non sono puntiformi (a differenza di come facevamo prima che consideravamo tutte le luci come puntiformi):

```
if (args->softShadow) { //SOFT SHADOWS
    int hMax = 30;
    for (int h = 0; h < hMax; h++) {
        new_hit = new Hit();
        pointOnLight = f->RandomPoint();  //scelta casuale di un punto nell'area della luce
        dirToLight = pointOnLight - point;
        dirToLight.Normalize();
        n_ray = new Ray(point, dirToLight);
        new_hit = new Hit();
        colpito = CastRay(*n_ray, *new_hit, false);

        if (colpito) {
            n_point = n_ray->pointAtParameter(new_hit->getT());
            //calcola il vettore distanza fra il punto colpito dal raggio e il punto sulla luce
            dista.Sub(dista, n_point, pointOnLight);

            if (dista.Length() < 0.01) { //il punto colpito coincide con la luce
                if (normal.Dot3(dirToLight) > 0) {
                    Vec3f lightColor = 0.2 * f->getMaterial()->getEmittedColor()*f->getArea();
                    answer += m->Shade(ray, hit, dirToLight, lightColor, args) * (1.0 / hMax); //Shade è il modello di illuminazione di phong
                }
            }
        }
    }
}
```

In particolare, se volessimo avere delle soft shadows complete dovremmo castare un raggio per ogni punto della luce, ma dato che ciò potrebbe essere dispendioso in termini di risorse e tempo si va a castare un numero *hMax* di raggi in punti random della luce. Il procedimento è poi uguale al caso delle hard shadows visto nel punto 1. L'unica differenza è che alla fine il contributo di ogni shadow ray viene normalizzato per il numero di shadow rays. Questo viene fatto per evitare che il valore finale sia sempre troppo elevato rispetto a quello che deve essere (perché andremmo a sommare tutti gli shadow rays ad answer quando invece il contributo dovrebbe essere "unitario").

Risultato (con 30 raggi, ovviamente aumentando il numero di raggi il risultato migliora):