Esercitazione 4

RAYTRACING
La prima parte dell'esercitazione prevede lo sviluppo del ray tracing, un algoritmo che implementa il modello di illuminazione globale (illuminazione diretta - illuminazione indiretta: riflessione, rifrazione).

Colore di un oggetto
Ogni oggetto è costituito da uno o più materiali; ogni materiale è caratterizzato da diverse componenti che combinate definiscono il colore dell'oggetto:

- componente diffusiva - luce riflessa in tutte le direzioni, dipende dall'orientamento della superficie e dalla posizione della luce
- componente riflessiva - componente luminosa che si ottiene riflettendo la direzione della luce rispetto alla normale della superficie
- componente emissiva - emissione luminosa uniforme in tutte le direzioni, presente solamente se l'oggetto è una sorgente luminosa
- componente ambientale (non dipende dal materiale) - luce globale sempre presente e costante, garantisce un livello minimo di illuminazione alla scena

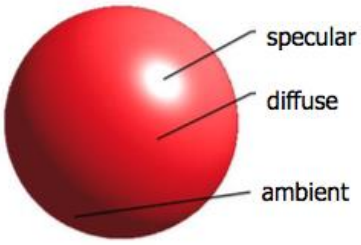

Figura 1 - Componenti luminose del materiale

Il progetto fornito mostra inizialmente una scena in cui sono presenti due sfere di colore scuro e implementa il Ray Casting.

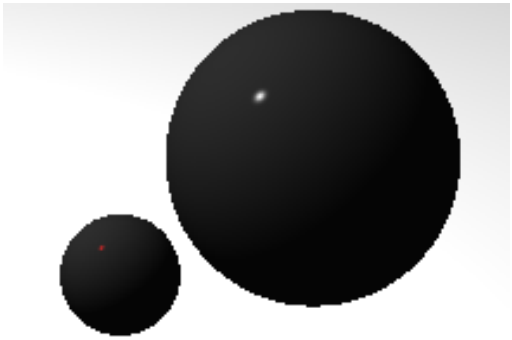

Figura 2 - Scena iniziale

Il Ray Casting applica un modello d'illuminazione che tiene conto solamente dei contributi forniti dalla luce diretta. In dettaglio:

- per ogni pixel della viewport si costruisce un raggio che parte dall'eye point e passa per il pixel considerato
- si determina, se esiste, il primo oggetto intersecato dal raggio appena costruito; l'intersezione definisce il colore del pixel:
    - se l'oggetto è una sorgente luminosa, il colore del pixel è uguale al colore della luce
    - altrimenti si applica il modello d'illuminazione di Phong (che tiene conto del materiale dell'oggetto)

- se il raggio non interseca nessun oggetto il pixel viene colorato con il colore di sfondo predefinito (background)

Il Ray Casting, tuttavia, non tiene conto delle sorgenti luminose indirette (riflessione, rifrazione).

Pseudocodice:

```
For every pixel(x, y) {
   construct a ray from eye point
   color[x, y] = rayCasting(ray);
}

rayCasting(ray) {
   hitObject(ray, hitPoint, object);

   if (ray do not intersect any object)
      return (background color);

   color = object color;
   if (object is light)
      return (color);
   else
      return (lighting(hitpoint, color));
}
```

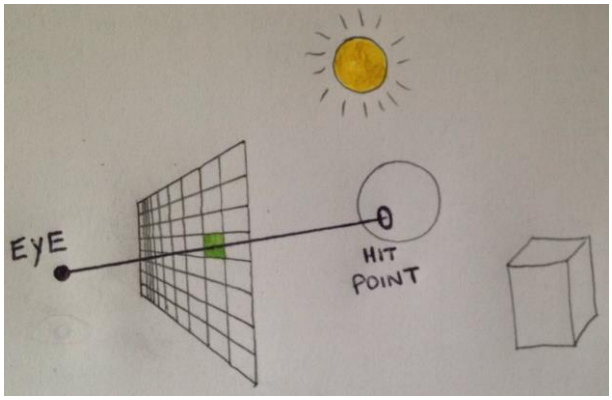

Figura 3 - Ray Casting

Per completezza, di seguito vengono mostrate le componenti diffusiva, riflessiva ed emissiva dei materiali (formato RGB) delle due sfere:

Big Sphere
diffuse 0.1 0.1 0.1
reflective 0.8 0.8 0.8
emitted 0 0 0

Small Sphere
diffuse 0.1 0.1 0.1
reflective 0.8 0 0
emitted 0 0 0

Punto 1

Il primo punto dell'esercitazione richiede di implementare la gestione degli *shadow rays* per la generazione di ombre. Per ogni luce in scena si costruisce un raggio (*shadow ray*) che parte dal punto di incidenza sulla scena in direzione della luce considerata (dirToLight = pointOnLight - hitPoint; Ray(hitPoint, dirToLight)). A questo punto si determina, se esiste, il primo oggetto intersecato dal raggio appena costruito (CastRay). Infine, se il punto di intersezione (intersection point) e il punto sulla luce (pointOnLight) coincidono (a meno di un piccolo *delta*) la luce contribuisce a determinare il colore del punto, altrimenti significa che c'è un oggetto in mezzo (pertanto il contributo luminoso non viene considerato). L'immagine illustra quanto appena spiegato; la luce L_0 contribuisce alla colorazione del punto, mentre la luce L_1 non contribuisce.

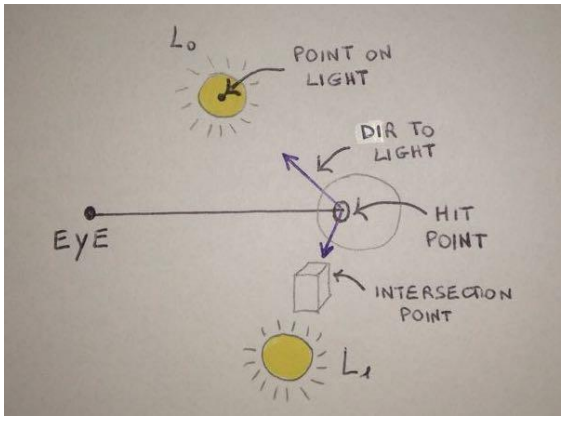

Figura 4 - Shadow rays

Pseudocodice:

```
For every pixel(x, y) {
   construct a ray from eye point
```

```
   color[x, y] = rayCasting(ray);
}

rayCasting(ray) {
   hitObject(ray, hitPoint, object);

   if (ray do not intersect any object)
      return (background color);

   color = object color;

   for each light {
      construct a ray from hitPoint to light
      hitObject(lightRay, hitPoint, object);
      if (ray do not intersect any object)
         color = color + lighting(hitpoint, color);
   }

   return (color);
}
```

A livello implementativo:

```
long num_lights = mesh->getLights().size();
for (int i = 0; i < num_lights; i++) {
  Face *f = mesh->getLights()[i];
  Vec3f pointOnLight = f->computeCentroid(); // Centro della luce
  Vec3f dirToLight = pointOnLight - hitPoint;
  dirToLight.Normalize();

  newRay = new Ray(hitPoint, dirToLight);
  newHit = new Hit();
  bool hitSomething = CastRay(*newRay, *newHit, 0);

  if (hitSomething) {
    intersectionPoint = newRay->pointAtParameter(newHit->getT());
    distance.Sub(distance, intersectionPoint, pointOnLight);

    if (distance.Length() < 0.01) {
    if (normal.Dot3 (dirToLight) > 0) {
      Vec3f lightColor = 0.2 * f->getMaterial()->getEmittedColor() * f->getArea();
      answer += m->Shade(ray, hit, dirToLight, lightColor, args); // Phong
    }
    }
  }
}
```

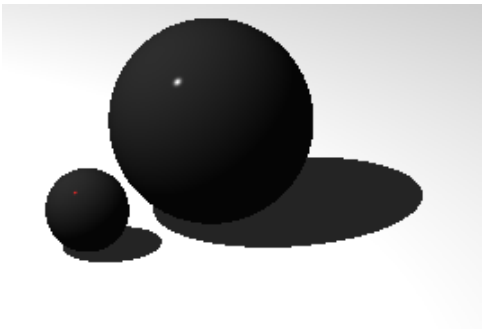

Figura 5 - Scena con shadow rays

Punto 2

Il secondo punto dell'esercitazione richiede di arricchire quanto implementato con la logica di gestione dei raggi riflessi (reflection rays).

Il raggio riflesso viene generato ovviamente solamente se l'oggetto colpito ha una componente riflettente.

Per calcolare il raggio riflesso si ricorre a semplici considerazioni geometriche.

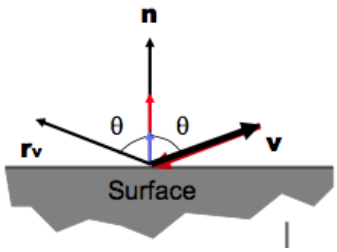

Figura 6 - Raggio riflesso

$$r_v + v = 2\cos\theta\, n$$
$$r_v = 2(n \bullet v)n - v$$
$$v\ opposite \rightarrow\ r_v = v - 2(n \bullet v)n$$

Il processo di generazione dei raggi riflessivi è ricorsivo e termina o quando il raggio interseca un oggetto opaco (senza componente riflessiva) o quando si è raggiunto un determinato numero di salti (bounceCount).

A livello implementativo:

```
Vec3f reflectiveColor = m->getReflectiveColor();
if (reflectiveColor.Length() != 0 && bounce_count > 0) {
   Vec3f rayVector = ray.getOrigin();
   Vec3f reflection = (2 * normal.Dot3(rayVector) * normal) - rayVector;
   reflection.Normalize();
   Ray ray = Ray(hitPoint, reflection);
   answer += TraceRay(ray, hit, bounce_count - 1);
   answer = answer * reflectiveColor;
}
```

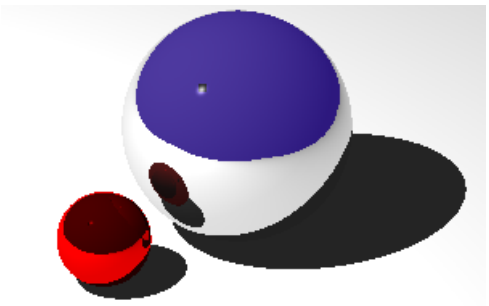

Figura 7 - Scena con shadow rays e reflective rays

Punto 3

Per implementare le soft-shadows si è modificata leggermente la logica di calcolo delle ombre considerando num_shadow_samples raggi luminosi in posizione casuale sulla luce stessa, calcolandone il contributo e facendo la media. Il risultato è una forma più smooth delle ombre.

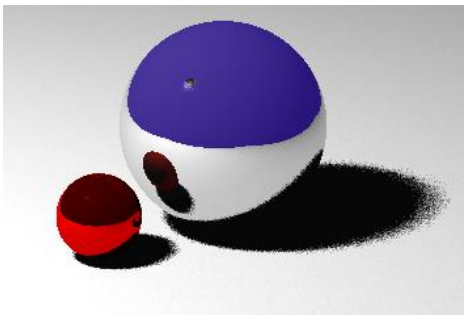

Figura 8 - Soft shadows

RAYTRACING - Blender

Per la seconda parte dell'esercitazione (Digital Art) si è deciso di modificare le due scene già create aggiungendo effetti di riflessività, trasparenza e rifrazione.

In particolare:

- per il piano del tavolo si è modificata la componente di riflessività (*Reflectivity*) impostando come valore 0.9 e si è messo come colore di riflessione un marrone chiaro

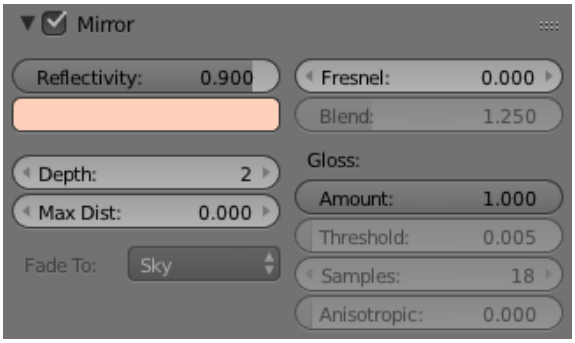

Figura 9 - Impostazioni piano del tavolo

- per i bicchieri si è modificata solamente la componente di riflessività impostando come valore 0.2

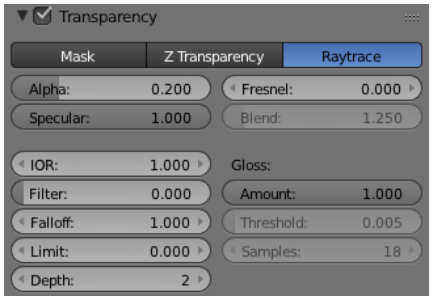

Figura 10 - Impostazioni bicchieri

- per la bottiglia si sono modificate le componenti di riflessività di rifrazione impostando rispettivamente i valori 0.2 e 0.9. Per aggiungere un ulteriore dettaglio si è inserita una mesh sferica all'interno della bottiglia, visibile una volta deselezionata l'opzione *Traceable*

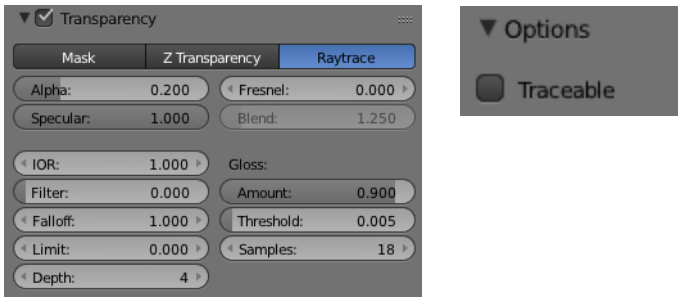

Figura 11 - Impostazioni bottiglia

Il risultato ottenuto è il seguente:

Figura 12 - Rendering scena 1

Per la seconda scena invece si è semplicemente messo un piano riflessivo di fronte alla mesh e un piano di appoggio con una texture che simula una materiale in legno.

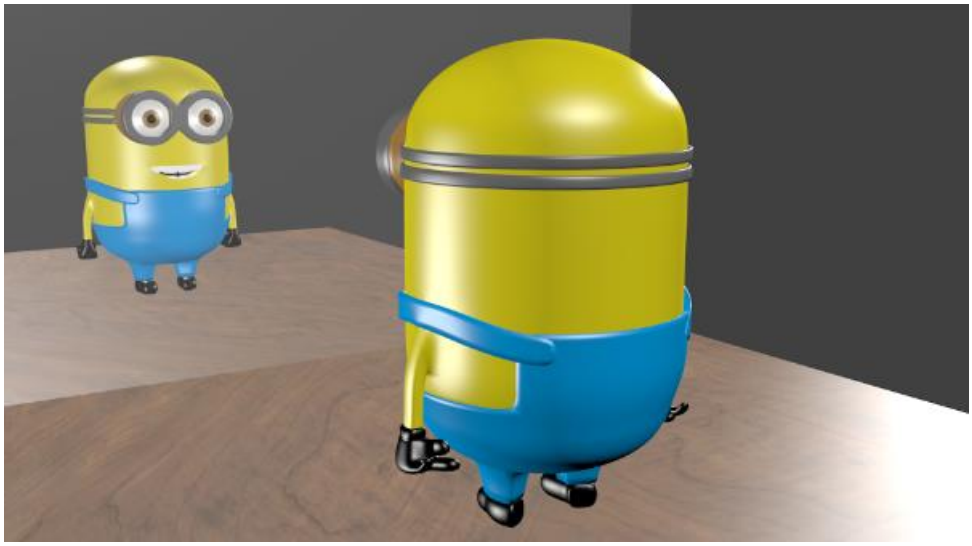

Figura 13 - Rendering scena 2