Synthèse d'Image - Lancer de rayon - TD04

Premières ombres & modèle de Phong

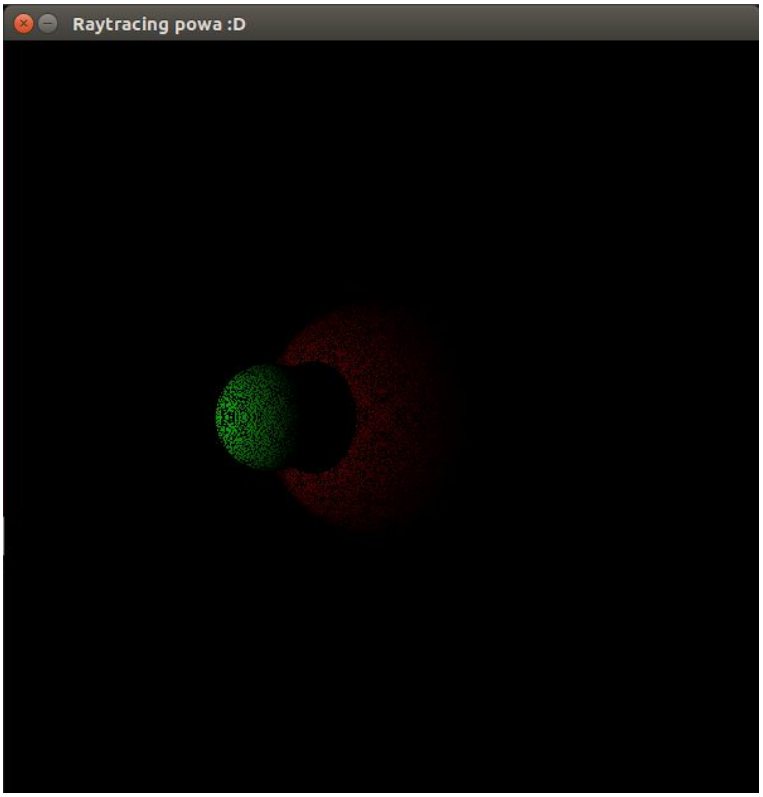
L'objectif de ce TD est de compléter notre premier raytracer en y ajoutant des ombres et de la réflection spéculaire.

Exercice 01 : Des ombres basiques

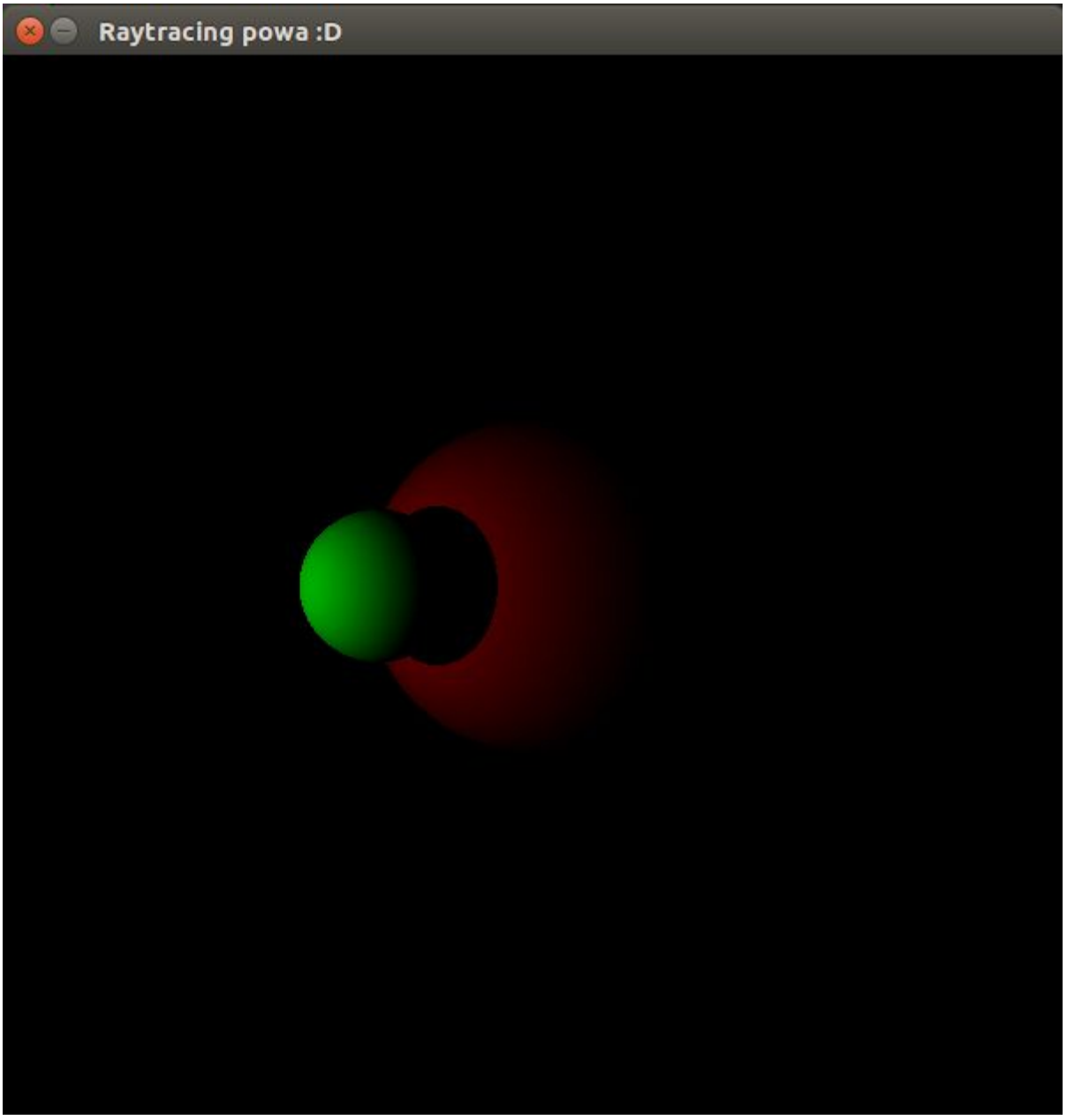
1. **Construisez une scène composées de deux sphère : une rouge en (0, 0, −5) de rayon 1.5 et une verte en (−1, 0, −3.5) de rayon 0.5. Placez une lumière de couleur (5, 5, 5) en (−3, 0, −1). Lancez votre programme. Quel est le problème ?**

La gestion des ombres dans un raytracer est simple à implanter, mais quelques précautions doivent être prises. Basiquement il suffit d'envoyer un rayon partant du point d'intersection et allant vers la lumière pour laquelle on veut calculer la contribution. Si ce rayon est intersecté avant d'atteindre la lumière, alors la lumière ne contribue pas et le point est dans l'ombre par rapport à cette lumière.

2. **Dans votre fonction `lambertRaytracing`, ajoutez les quelques lignes de code permettant d'ignorer la contribution lumineuse si le point d'intersection est dans l'ombre.** Vous devez obtenir le résultat suivant :



3. **Quelle peut être la provenance de ces artefacts ?**
4. **Proposez et implémentez une solution permettant de les éviter.**

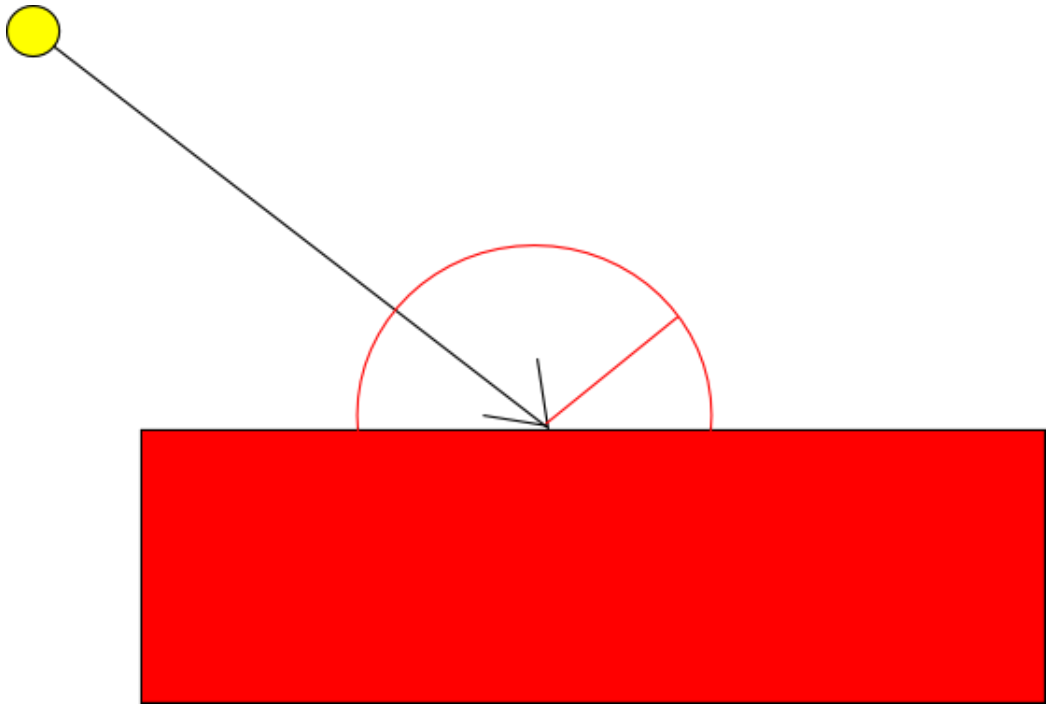


Introduction au spéculaire

Le modèle de Lambert nous permet de voir nos objets en 3D, ce qui est déjà pas mal. Néanmoins, il ne permet pas de faire le rendu d'objets brillants, comme le plastique ou la porcelaine. Le modèle de **Phong** va nous permettre de palier à cela.

Dans un premier temps, il faut comprendre ce qu'est la **réflection spéculaire**.

La réflection diffuse consiste à dire que la lumière arrivant en un point repart de la même manière dans toutes les direction (quelque soit le point de vue de l'observateur, il verra un point de la même couleur). Au contraire, la réflection spéculaire privilégie une direction particulière : Le symétrique de la direction d'incidence par rapport à la normale :



Dans le cas d'une réflection spéculaire pure (miroir par exemple), la lumière est totalement réfléchie selon cette direction (elle rebondit comme une boule de billard). Dans le cas de matériaux brillants mais non spéculaire pur, on utilise le modèle de Phong qui **associe une composante diffuse à une composante spéculaire** :

```
Idiff * Lc * NdotL / distance²  +  Ispec * Lc * specFactor / distance²
```

où

- **Ic :** Composante diffuse de l'intersection
- **Lc :** Couleur de la lumière
- **NdotL :** Produit scalaire entre la normale et le vecteur IL
- **distance :** norme du vecteur IL
- **Ispec :** Composante spéculaire de l'intersection
- **specFactor :** le facteur spéculaire calculé en fonction de l'angle de réflection de la lumière avec la normale et de la position de l'intersection par rapport à la caméra.

A noter que la première partie de la formule est exactement la même que dans le modèle de Lambert.

Nous allons devoir faire pas mal de modifications dans notre code pour implanter ce modèle. En effet, avant chaque objet n'avait pour le moment qu'une seule couleur (et donc les points d'intersection aussi). Ici on a deux composantes de couleur : diffuse et specular. S'ajoute à ça un coefficient de brillance nécessaire au calcul du facteur spéculaire. Nous allons donc devoir revoir totalement la représentation des couleurs des objets.

Exercice 02 : Matériau

1. **Dans un nouveau module *shading* (shading.h, shading.c), créez une structure `Material` permettant de stocker une couleur *diffuse*, une couleur *spéculaire*, ainsi qu'un attribut *shininess* correspondant à la brillance de la surface.**

2. **Modifiez vos structure `Sphere` et Intersection pour qu'elles utilisent non plus une couleur mais un matériau.**

3. **Modifiez toutes les fonctions de votre raytracer qui utilisaient/créaient des couleurs pour leur faire manipuler des matériaux (pas la peine de modifier `lambertRaytracing` pour le moment).**

Exercice 03 : Réflection et modèle de Phong

La composante **`specFactor`** définie dans la formule de Phong se calcule comme suit:

```
specFactor = specAngle ^ shininess
où
specAngle = max(0, R.V)
```

où

- **`shininess`** : Coefficient de brillance de la surface
- **`R`** : Vecteur directeur de la lumière, réfléchit par rapport à la normale de l'intersection.
- **`V`** : Position de l'intersection (V pour View direction)

1. **Dans votre module geometry, implémentez la fonction `Vector3D reflect(Vector3D V, Vector3D N)` qui calcule le vecteur V réfléchit par rapport au vecteur N. Le calcul du vecteur est le suivant :**

   ```
   R = V - N * 2.0 * V.N
   ```

   Attention, les deux vecteurs passés en paramètre doivent être normalisés.

2. **Implémentez la fonction `phongRaytracing`. Cette fonction est sensiblement similaire à la fonction `lambertRaytracing` sauf qu'elle calcule la couleur du pixel en fonction de ses composantes spéculaires et diffuses.**

3. **Ajoutez diverses sphères et lights dans votre scène pour tester de votre tout premier raytracer fraichement terminé !**