Synthèse d'Image - Lancer de rayon - TD03

Un peu de lumière

L'objectif de ce TD est d'ajouter des lumières à votre scène et de calculer leur influence sur les objets intersectés par les rayons.

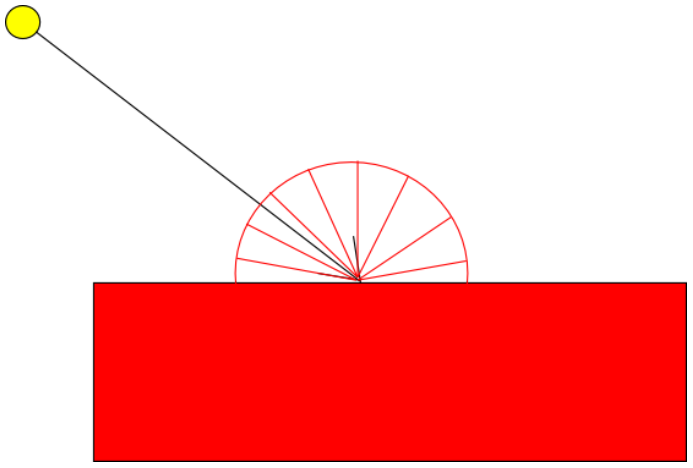
Exercice 01 : Une structure pour nos lumières

Nous allons avoir besoin de représenter des lumières dans notre programme. Une lumière sera définie par une position dans l'espace et une couleur.

1. **Dans le fichier raytracer.h, créez une structure *Light* représentant une lumière.**
2. **Ajoutez une fonction `createLight` prenant en paramètre une position et une couleur, et retournant un lumière.**
3. **Dans votre structure *Scene*, ajoutez un tableau (taille fixe pour le moment, par exemple 10) permettant de stocker des lumières ainsi qu'un compteur pour connaître le nombre de lumières dans la scène. Pensez à initialiser ce compteur lors de la création de la scène.**
4. **Ajoutez et implémentez la fonction `void addLightToScene(Scene* scene, Light light);` permettant d'ajouter une lumière dans votre scène.**

Exercice 02 : Modèle de lambert : réflection diffuse

L'étude des échanges lumineux entre surfaces relève d'une branche de la physique appelée radiométrie. Plusieurs modèles existent afin de décrire ces échanges. Nous allons commencer par implanter un modèle très simple ne prenant en compte que les réflections **diffuses** de la lumière. Une surface est dite diffuse si elle renvoie la lumière de la même manière dans toutes les directions :



Cela signifie que quelque soit votre position vous verrez un point de l'objet de la même manière. La couleur ne dépend pas de votre position mais uniquement de la position de l'objet par rapport à la source de lumière.

C'est l'effet inverse qui se produit pour des surface dites **spéculaire** comme le miroir : ce que vous voyez dépend de votre position car la surface renvoie la lumière dans une direction privilégiée.

Dans la vie réelle, très peu de surfaces sont purement diffuse ou purement spéculaire. Mais pour l'instant nous allons traiter le cas purement diffus.

Actuellement, dans votre programme, vous affichez la couleur de l'objet s'il y a une intersection. A présent vous allez devoir faire un calcul qui dépend :

- de la lumière
- de la position de l'intersection
- de la couleur de l'intersection
- et de la **normale** au point d'intersection, qui est un vecteur unitaire orthogonal à la surface au point considéré

1. **Dans votre programme, ajouter un champ *normal* à votre structure Intersection. Ce champ servira à stocker la normale au point d'Intersection entre un rayon et une sphère.**
2. **Modifiez votre fonction `intersectsSphere` de manière à calculer la normale au point d'intersection et à remplir le nouveau champ de la structure Intersection. Attention, la normale sera différente si l'origine du rayon est à l'intérieur de la sphère.**
3. **Modifiez également la fonction `throwRayThroughScene` pour remplir correctement les champs de l'intersection passée en paramètre.**

Votre programme est désormais prêt pour accueillir le calcul de la composante diffuse de la lumière. Pour chacune des lumières de votre scène, le calcul de la couleur du pixel est le suivant :

$$Ic * Lc * NdotL / norm(IL)^2$$

où

- `Ic = Couleur de l'intersection`
- `Lc = Couleur de la lumière`
- `NdotL = Produit scalaire entre la normale et le vecteur IL`
- `I = position de l'intersection`
- `L = position de la lumière`

S'il y a plusieurs lumières, il faut additionner chaque résultat pour obtenir la couleur finale du pixel.

2. **Créez une fonction `void lambertRaytracing(Scene scene, SDL_Surface* framebuffer)`, similaire à la fonction simpleRaytracing, mais qui à chaque intersection détectée, itère sur les lumières de la scène pour calculer la couleur finale du pixel.**

3. **Dans les fichier colors.h et color.c, ajoutez une fonction `Color3f clampColor(Color3f c)`, qui retourne une couleur avec ses 3 composantes clampées entre 0 et 1 (cherchez la définition d'un *clamp* sur Internet si vous ne la connaissez pas).**

4. **Dans votre fonction `lambertRaytracing`, une fois la couleur finale du pixel calculée, clampez cette couleur et servez vous en comme paramètre de la fonction `PutPixel` à la place de la simple couleur de l'intersection.**

5. **Dans votre fichier main.c, ajoutez à votre scène :**
   - **une sphère située en 0, 0, -5 et de rayon 1.5**
   - **une lumière située en (0, 0, 0) et de couleur (10, 10, 10)**

   Exécutez votre programme. Vous devrez voire votre sphère illuminée.

6. **Ajoutez d'autres sphère et d'autres lumières à votre scène et observez les résultats.**