TP Initiation à la recherche 1

Master Informatique - 1ère année

Année 2020-2021

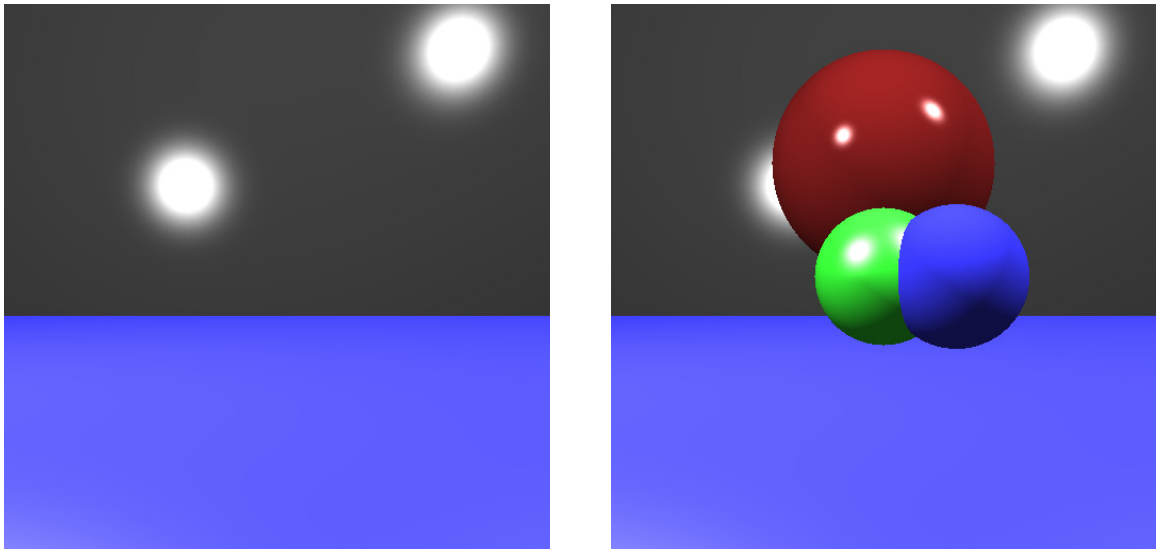
L'objectif de ce TP est de prendre en main les sources d'un code de lancer de rayons, qui sera utilisé dans le cadre de ce module.

Exercice 1

Récupérez l'archive des sources du lancer de rayons qui vous sont fournies avec cet énoncé. Après installation, vous disposez dans votre dossier `LR` :

- des sources de l'application, dans le dossier `Src` ;
- d'une scène de test dans le dossier `Scenes`
- d'un fichier de configuration pour l'utilitaire `cmake` (`CMakeLists.txt`) ;
- et d'un fichier de configuration pour le générateur de documentation `doxygen` (`Doxyfile`).

Dans un premier temps, générez la documentation html de l'application avec `doxygen` (un dossier `html` est créé - vous pouvez ouvrir la documentation à partir du fichier `index.html` qui s'y trouve). Générez le `Makefile` avec l'utilitaire `cmake`, puis compilez et tester l'application qui portera le nom `lr`. Vous utiliserez le fichier `Scenes/scene01.txt` pour les différentes questions de cet énoncé (le détail de ce fichier vous est donné en annexe A). Après exécution, vous devez obtenir une image nommée `image.ppm` qui correspond à l'image de la figure 1a.



(a) Première image obtenue avec l'application fournie, à partir de la scène présente dans `scene01.txt`.

(b) Aperçu de la même image après implantation de la méthode `intersecte` de la classe `Sphere`.

FIGURE 1 – Images obtenues à partir du fichier `scene01.txt` avant et après implantation de la méthode d'intersection de la classe `Sphere`.

Cette image correspond à une vue de la scène virtuelle, vue depuis la position de la caméra : un plan horizontal bleu (sol), un plan vertical sombre (mur du fond) et le reflet dans ce dernier de deux sources ponctuelles blanches placées en hauteur (non visibles sur l'image).

Exercice 2

Le fichier scène utilisé dans la question précédente contient également la description de 3 sphères. Ces dernières n'apparaissent pas sur l'image, car le code de la méthode chargée de déterminer si un rayon

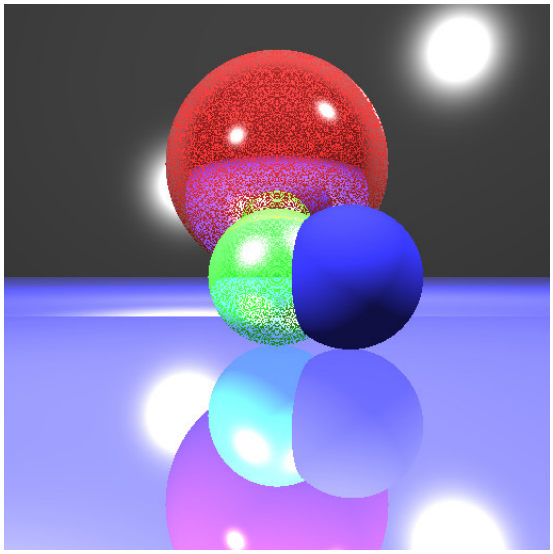
intersecte une sphère n'a pas été écrit. En vous appuyant sur ce qui a été vu en cours et en vous aidant de la documentation html pour les différentes classes à utiliser, complétez le code de la méthode

`bool Sphere::intersecte(const Rayon& r, Intersection& inter)`

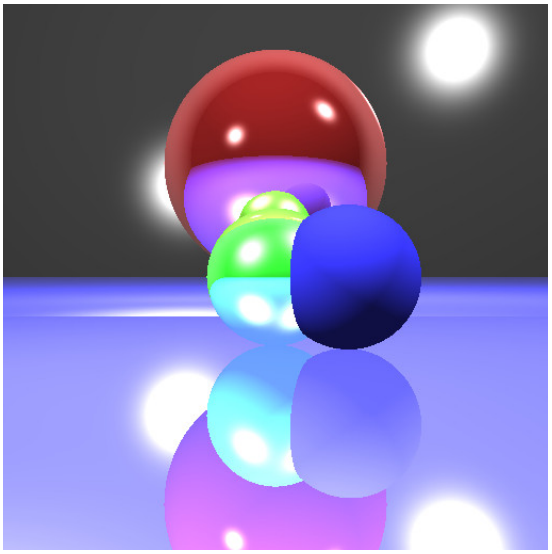
Aucune autre classe ne doit être créée et aucune autre classe ou méthode ne doit être modifiée. A l'issue de ce travail, vous devez obtenir l'image qui figure sur la figure 1b.

Exercice 3

La profondeur de réflexion du lancer de rayons est fixée par la constante `PROF`, définie dans le fichier `lr.cpp`, et passée à la méthode de génération d'image de la caméra. A ce stade, elle a été initialisé à 0, ce qui signifie qu'aucun rayon réfléchi n'est lancé. Modifiez cette constante en lui affectant la valeur 1, puis relancer votre application. Vous devez obtenir l'image qui figure que la figure 2a.



(a) Prise en compte d'une réflexion après la première implémentation de la méthode `intersecte` de la classe `Sphere`.



(b) Aperçu de la même image après correction de la méthode `intersecte`. Les artefacts visibles précédemment ont disparu.

FIGURE 2 – Aperçu des images obtenues après implémentation de la méthode `intersecte` de la classe `Sphère`.

Vous noterez que si la réflexion des sphères sur le plan horizontal, ainsi que sur les sphères elles-mêmes, apparaît bien, la qualité du rendu des sphères rouges et verte laisse à désirer [1] ... Le problème illustré ici est celui de la précision numérique de vos processeurs. Lorsque vous effectuez un test sur la valeur du paramètre `t`, vous considérez que la valeur calculée est exacte. Ceci n'est jamais le cas, du fait que les valeurs réelles obtenues sont représentées sur un nombre fini de bits. Ainsi, la comparaison par rapport à la valeur nulle qui est faite dans l'algorithme fournira parfois des valeurs erronées, simplement parce que la valeur calculée n'est pas exactement égale à 0, mais en est une valeur très proche.

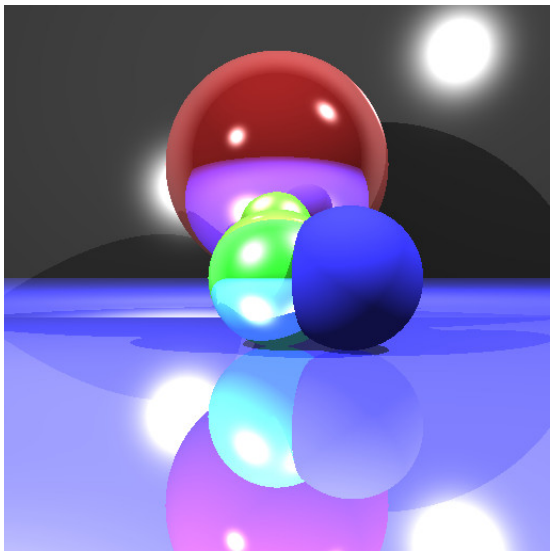
Dans le cas qui nous intéresse ici, lorsqu'un rayon réfléchi est lancé depuis l'une des sphères rouge ou verte, l'origine du rayon se trouve sur la sphère correspondante. Dans certains cas se produit alors une auto-intersection avec la sphère de départ du rayon, qui conduit à une erreur d'évaluation de l'intensité lumineuse portée par le rayon. Pour éviter ce genre de problème, on définit généralement une constante réelle très petite, par rapport à laquelle les tests sont effectués. En notant cette constante `SP_EPSILON` (*epsilon* pour la classe `Sphere`) et en lui associant la valeur 0.0001, modifiez le code de votre fonction `intersecte`, de manière à ne plus faire de tests par rapport à une valeur exacte. Vous devez alors obtenir l'image apparaissant sur la figure 2b.

Exercice 4

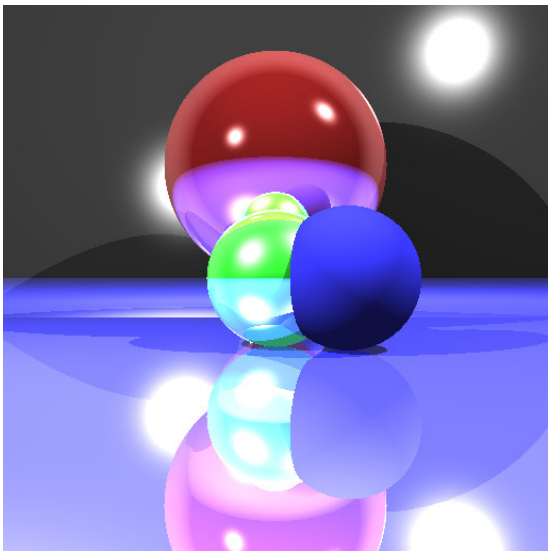
Si les reflets sont désormais présents et de bonne qualité, aucune ombre n'apparaît sur les images. Ceci est normal, dans la mesure où le code de la méthode `coupe` de la classe `Sphere` renvoit toujours `false`. De ce fait, l'algorithme considère qu'une sphère ne peut pas occulter la lumière provenant d'une source

1. Le matériau de la sphère bleue est purement diffus (mat). En lancer de rayons, aucune réflexion n'est calculée sur ce type de matériau.

et aucune ombre générée par une sphère ne peut être produite. Complétez le code de cette méthode, en vous inspirant de ce que vous avez développé pour la méthode `intersecte`, en vous rappelant que ici, aucun point d'intersection ne doit être calculé. A l'issue de cette question, vous devez obtenir l'image qui est présentée sur la figure 3a. La modification de la profondeur de réflexion du lancer de rayons à 2 permet d'obtenir l'image de la figure 3b.



(a) Prise en compte des ombres après implantation de la méthode `coupe` de la classe `Sphere` - `PROF` = 1.



(b) Prise en compte des ombres après implantation de la méthode `coupe` de la classe `Sphere` - `PROF` = 2.

FIGURE 3 – Aperçu des images obtenues après implémentation de la méthode `coupe` de la classe `Sphère` pour deux profondeurs de recursion différentes.

Annexe A - format de fichier

Dans le cadre de cette application simplifiée du lancer de rayons, un format basic de scènes est proposé. Il s'agit d'un format texte, permettant de spécifier les différents éléments composant une scène de manière intuitive et lisible. Le format pourra également être commenté : toute ligne commençant par le caractère `#` sera considérée comme un commentaire.

Couleurs et matériaux

La couleur

Le modèle de couleurs utilisé par l'application de lancer de rayons est le modèle (R,V,B) (rouge, vert, bleu). Chaque couleur est alors obtenue par mélange d'un pourcentage de chacune de ces trois couleurs primaires. Les valeurs prises par chacune des composantes du triplet seront donc des nombres réels compris entre 0 (0% de la composante) et 1.0 (100% de la composante). Une valeur 1.0 pour chaque composante fournira du blanc, tandis qu'annuler chaque composante fournira du noir.

La couleur sera explicitement précisée pour la *couleur de fond*, c'est à dire la couleur que doit prendre un pixel si aucun objet n'y est visible. La syntaxe sera alors :

```
fond r v b
```

Les matériaux

Les matériaux sont définis de manière à pouvoir être utilisés dans le modèle d'illumination de Phong, décrit dans l'annexe B. Ses constituants sont :

- une couleur de base : (r, v, b) ;
- un coefficient de réflexion ambiant k_a : l'intensité de la source ambiante sera atténuée par ce coefficient ;
- un coefficient de réflexion diffus k_d : si une source est visible, sa contribution à la composante diffuse de l'éclairage sera pondérée par ce coefficient ;

— un coefficient de réflexion spéculaire k_s , qui est un marqueur de la brillance du matériau. Si une source est visible, sa contribution à la composante spéculaire de l'éclairage sera pondérée par ce coefficient ;
— un indice de brillance s, qui permet de régler la directionnalité de la brillance de l'objet : plus sa valeur sera élevée, plus le matériau sera brillant.

La syntaxe d'un matériau sera alors :

```
materiau r v b ka ks kd s
```

Les sources de lumière

Les sources de lumière seront des sources ponctuelles isotropes, qui disposeront d'une position (x, y, z) et d'une intensité lumineuse donnée selon le modèle (r,v,b) : (I_r, I_v, I_b). La syntaxe de définition d'une source sera alors :

```
source x y z Ir Iv Ib
```

Remarque : Une source particulière sera définie (éventuellement par défaut si elle n'est pas présente dans le fichier) ; appelée source **ambiante**, elle permet de représenter l'ensemble des intensités lumineuses dans lequel baigne la scène, issues d'une infinité de réflexions entre objets (voir le modèle d'illumination en annexe B). Elle n'a donc pas de localisation précise et est définie uniquement par son intensité :

```
ambiant Ir Iv Ib
```

Géométries

Un objet va être défini par sa géométrie, mais également par le matériau dont il est composé. Le format suppose que lorsqu'un matériau a été défini, il s'applique à toutes les géométries qui suivent, tant qu'un nouveau matériau n'est pas redéfini.

Les sphères

Une sphère est définie par tous les points (x, y, z) de l'espace vérifiant l'équation :

$$(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 = R^2$$

avec (x_c, y_c, z_c) les coordonnées du centre de la sphère et R son rayon. Le format proposé fournira alors ces 4 informations :

```
sphere xc yc zc R
```

Les plans

Un plan est défini par tous les points (x, y, z) de l'espace vérifiant l'équation :

$$a \times x + b \times y + c \times z + d = 0$$

avec (a, b, c) les coordonnées du vecteur normal (perpendiculaire) au plan et d la distance par rapport à l'origine. Le format proposé fournira alors ces 4 informations :

```
plan a b c d
```

Autres géométries

D'autres géométrie, suivant le même principe, seront ajoutées lors du TP suivant.

Annexe B - le modèle d'illumination

Le modèle d'illumination utilisé par cette application de lancer de rayon est un modèle empirique très simple, dû a B. T. Phong [2]. Son objectif est d'approcher au mieux ce que l'on percoit de l'éclairage, sans utiliser un quelconque modèle physique. Dans ce modèle, l'éclairage reçu de la scène par un point P se calcule comme la somme de 3 composantes :

$$I(P) = I_a(P) + I_d(P) + I_s(P)$$

avec :

2. Bui Tuong Phong, Illumination for computer generated pictures, Communications of the ACM, June 1975, https ://doi.org/10.1145/360825.360839

— $I(P)$ l'éclairage direct en un point P, la valeur que l'on souhaite calculer ;
— $I_a(P)$ l'éclairage ambiant donné par :

$$I_a(P) = K_a \times I_{S_a}$$

avec K_a le coefficient ambiant de l'objet et I_{S_a} l'intensité de la source ambiante (valeur moyenne de l'éclairage général de la scène). Ce terme permet d'attribuer un éclairage aux point des objets qui ne voient aucune des sources de la scène. Notez que cette source devra être spécifiée à part de toute autre source ;
— $I_d(P)$ l'éclairage diffus donné par :

$$I_d(P) = K_d \times (\overrightarrow{L}.\overrightarrow{N}) \times I_S$$

avec $\overrightarrow{L}$ le vecteur incident [3], $\overrightarrow{N}$ la normale [4] à la surface au point P, K_d le coefficient de réflexion diffuse du matériau et I_S l'intensité de la source considerée.
— $I_s(P)$ l'éclairage spéculaire donné par :

$$I_s(P) = K_s \times (\overrightarrow{V}.\overrightarrow{R})^s \times I_S$$

avec $\overrightarrow{V}$ le vecteur direction de visée [5], $\overrightarrow{R}$ le vecteur de réflexion spéculaire [6] en P, s l'indice de brillance du matériau et K_s son coefficient de réflexion et I_S l'intensité de la source considéréee.
Les différents vecteurs apparaissant dans ces formules sont illustrés dans la figure 4.

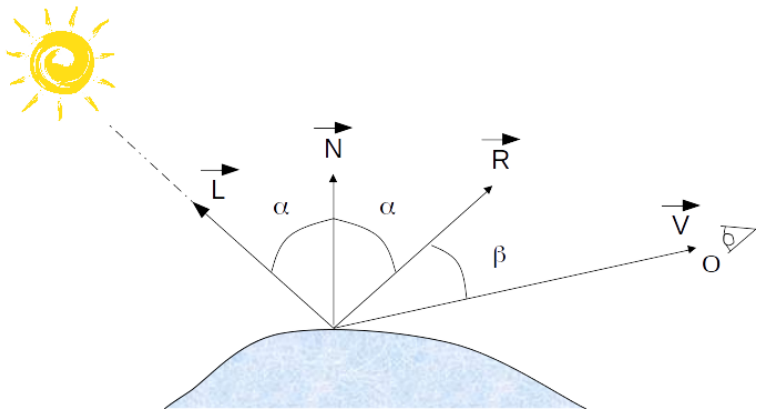


FIGURE 4 – Géométrie du calcul des termes diffus et spéculaire de l'éclairage

3. Vecteur normalisé d'origine le point où l'éclairage est calculé, et dirigé vers la source de lumière
4. Vecteur perpendiculaire à la surface
5. Vecteur normalisé d'origine le point où l'éclairage est calculé, dirigé vers la position de l'observateur
6. Symétique du vecteur $\overrightarrow{L}$ par rapport à la normale en P