Université de Limoges
Faculté des Sciences & Techniques
Faculté des Sciences et Techniques de Limoges

ISIR - TP2 - Éclairage et calcul d'ombres portées

Maxime MARIA

2022-2023

Dans ce TP, vous allez prendre en compte des sources lumineuses ponctuelles et calculer les ombres portées dans notre scène. Dans un premier temps, vous allez ajouter un nouvel objet à notre moteur, le plan.

1 L'objet plan

1. Ajoutez une classe `PlaneGeometry`, héritant de `BaseGeometry`, qui décrit la géométrie d'un plan. Un plan doit pouvoir se construire à partir d'un point (appartenant au plan) et d'une normale.
2. Ajoutez maintenant une classe `Plane` héritant de `BaseObject` qui décrit un objet plan. Elle doit donc avoir un attribut `PlaneGeometry` et implémenter la méthode `intersect`.
3. Dans `Scene::init`, ajoutez un plan horizontal positionné à $y = -2$ et un matériau `ColorMaterial` rouge, puis associez les deux. Lancez le calcul de l'image, vous devriez obtenir la Figure 1.

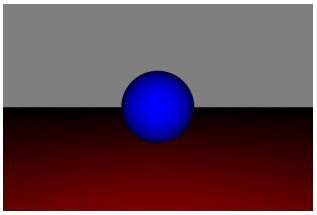


Figure 1 – Résultats de l'exercice 1.

2 Sources lumineuses ponctuelles

La classe abstraite `BaseLight` sert de base à la hiérarchie de sources lumineuses. Elle contient un attribut `_color` correspondant à la couleur de la source lumineuse et une méthode `sample` qui, à partir d'une position passée en paramètre, retourne un objet de type `LightSample`.

La classe `LightSample` possède quatre attributs. Pour le moment, ne vous souciez que des trois premiers. L'attribut `_pdf` (*Probability Density Function*) sera utilisé plus tard.

1. Ajoutez un attribut `float _power` à la classe `BaseLight` et prenez-le en compte dans le constructeur. Cet attribut, comme son nom l'indique, nous permettra de régler la puissance des sources lumineuses.
2. Ajoutez une classe `PointLight`, héritant de `BaseLight`, qui représente une source lumineuse ponctuelle. Cette classe ne possède qu'un seul attribut, sa position.
3. Implémentez la méthode `PointLight::sample`. La radiance correspond à la couleur de la lumière multipliée par sa puissance et pondérée par un facteur d'atténuation. Ici, ce facteur correspond à l'inverse de la distance entre le point d'éclairage et la source lumineuse au carré. Ainsi, plus la source lumineuse sera loin du point à éclairer, moins elle contribuera. Pour la PDF, laissez simplement 1.

3 Prise en compte des lumières

Jusque là, l'intégrateur utilisé pour calculer les images (`RayCastIntegrator`) ne prenait pas en compte les sources lumineuses (en fait, il simulait une source lumineuse sur la caméra...).

1. Ajoutez une classe `DirectLightingIntegrator` héritant de `BaseIntegrator`. Dans cet intégrateur, la méthode `Li` doit calculer l'éclairage des objets en fonction des sources lumineuses de la scène. Pour cela, si une intersection est trouvée entre le rayon primaire et la scène, il faut ajouter la contribution de chaque source lumineuse à la luminance finale : `Li += mtl->getColor() * lightSample._radiance * cosTheta`. Codez l'éclairage direct dans une méthode privée `DirectLightingIntegrator::_directLighting`.
2. Modifiez la méthode `Scene::init` et ajoutez une source lumineuse ponctuelle située en $(1, 10, 1)$ de couleur blanche et de puissance 100. Changez l'intégrateur dans la fonction `main` en utilisant la méthode `Renderer::setIntegrator`. Lancez le calcul de l'image, vous devriez obtenir la Figure 2(a).
3. Remarquez que le haut de la sphère est noir (cadre vert sur la Figure 2(a)) alors qu'il devrait être éclairé. Le problème est que la luminance calculée dépasse 1, donc quand la couleur du pixel est attribuée, la valeur des composantes dépasse 255 et ne peut donc pas être codée sur 8 bits. Pour pallier ce problème, il suffit de restreindre cette valeur entre 0 et 1. Pour cela, utilisez la fonction `glm::clamp` dans la méthode `Renderer::renderImage`. Lancez le calcul de l'image, vous devriez obtenir la Figure 2(b).

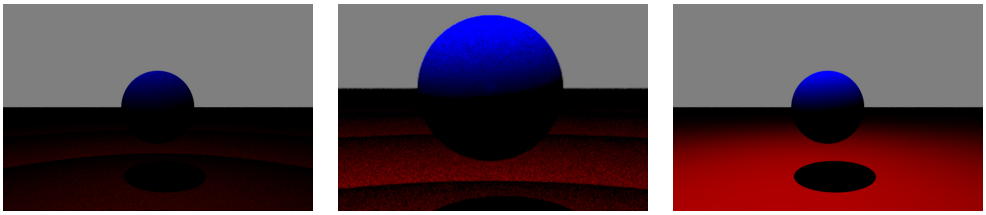
(a) (b)

FIGURE 2 – Résultats de l'exercice 3.

4 Calcul d'ombres portées

Les images calculées n'ont aucune ombre. Pourtant, la sphère devrait projeter une ombre sur le plan. Pour calculer les ombres portées, il va falloir lancer de nouveaux rayons pour déterminer si le point observé est visible depuis la source lumineuse (s'il ne l'est pas, alors le point observé est dans l'ombre).

1. Modifiez la méthode `DirectLightingIntegrator::Li` afin de lancer des rayons d'ombrage et déterminer si le point observé est éclairé ou non (vous ne calculerez l'éclairage que si le point est éclairé...).
2. Lancez le calcul de l'image, vous devriez obtenir la Figure 3(a). Il y a clairement un problème... La Figure 3(b) propose la même image zoomée et avec un contraste plus élevé pour visualiser le bruit. Ce bug est simplement dû à des problèmes de précision numérique lors du calcul d'intersection. Ainsi, pour certains points, l'intersection trouvée par le rayon d'ombrage est l'origine du rayon elle-même, on parle d'auto-intersection.
3. Pour éviter ce problème, il suffit de déplacer légèrement l'origine du rayon d'ombrage selon la normale de la surface en utilisant la fonction `Ray::offset` (que vous devez implémenter). Lancez le calcul de l'image, vous devriez obtenir la Figure 3(c).



(a) (b) (c)

FIGURE 3 – Résultats de l'exercice 4.

5 Recherche d'intersection pour les rayons d'ombrage

Pour l'instant, nous utilisons la méthode `intersect` pour tester nos rayons d'ombrage. Or, cette méthode cherche l'intersection la plus proche entre le rayon et la scène. Pour les rayons d'ombrage, il n'est pas nécessaire de chercher l'intersection la plus proche mais seulement de savoir s'il y a une intersection entre le point observé et la source lumineuse. Il est donc possible de réduire les temps de calcul.

1. Ajoutez une méthode `intersectAny` à la classe `Scene`. Cette méthode doit simplement retourner `true` si une intersection valide (entre `p_tMin` et `p_tMax`) est trouvée et `false` sinon (elle n'a donc pas de paramètre `HitRecord`). Pour cela, vous allez devoir adapter la hiérarchie `BaseObject` pour ajouter une nouvelle méthode `intersectAny`.
2. Dans la méthode `DirectLightingIntegrator::_directLighting`, utilisez `intersectAny` à la place de `intersect`. Lancez le calcul de l'image, vous devriez obtenir la même image que précédemment (*cf.* Figure 3(c)).