

# Rapport SI - TP 2

Startresse Guillard  
startresse.guillard@gmail.com

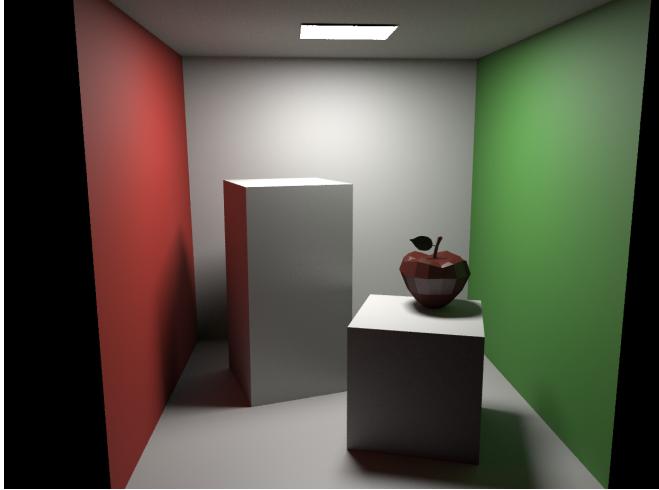


Fig. 1. Rendu final

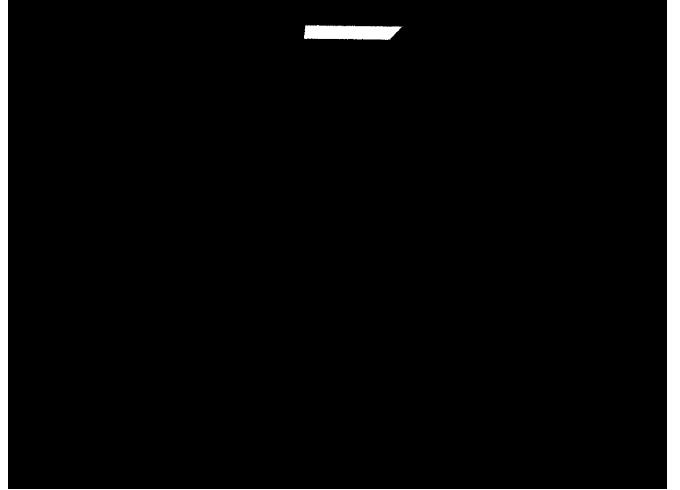


Fig. 2. Le

## I. TRAVAIL EFFECTUÉ

### A. Lancé de rayon

Le programme est une synthèse d'image sur un mesh grâce à un lancer de rayon. Pour chaque pixel on lance un rayon de ce pixel droit dans l'image et on calcule l'intersection  $p$  avec le mesh. Pour accélérer le calcul d'intersection, un BVH a été mis en place, cela a permis d'ajouter un peu de triangles dans la scène. L'image résultante est la somme des valeurs décrites dans les sections suivantes. On peut pondérer ces valeurs pour avoir un rendu quelque peu plus artistique (ie  $L1 * 0.6$  et  $L2 * 0.4$  pour éviter d'avoir le centre de l'image trop éclairé).

### B. Le

Tout d'abord pour composer l'image, on va calculer la valeur d'émission des sources et les poser directement sur l'image. L'image 2 montre le résultat de l'image entière avec seulement le Le pris en compte.

### C. LI

1) *Lumière*: Pour calculer l'éclairement direct d'une surface, on relance un rayon du point  $p$ . Pour avoir moins de rayons à lancer, au lieu de lancer un rayon aléatoirement et espérer toucher une source de lumière, on lance un rayon directement vers un point d'une source de lumière. Il suffit ensuite de regarder si ce rayon intersecte le BVH, si ce n'est pas le cas le point est éclairé par la source de lumière.

2) *Blinn-Phong*: Le modèle d'éclairement est le modèle de Blinn-Phong. N'étant pas spécifié par le mesh, le coefficient  $K$  a été arbitrairement mis à 0.7.

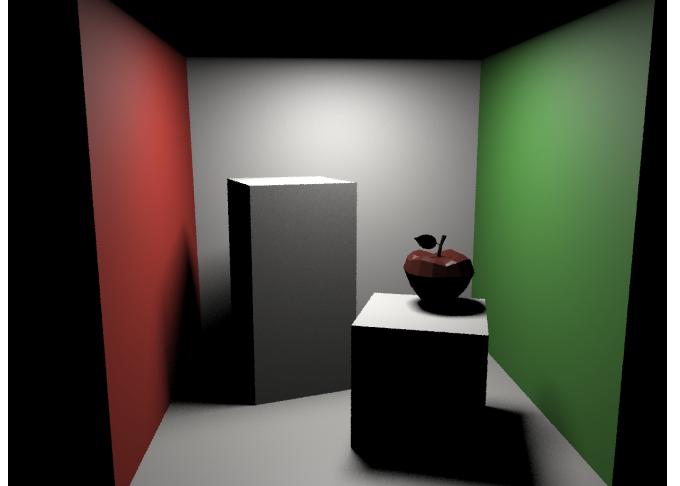


Fig. 3. L1

3) *Aléatoire sur les sources de lumière*: Pour choisir un point sur une source de lumière, il y a plusieurs façons de faire:

- Pour chaque source, choisir un point aléatoire sur cette source.
- Choisir une source aléatoirement (uniformément) parmi toutes les sources puis choisir un point aléatoire sur cette source.

Ces 2 méthodes ne conviennent pas dans certains cas. En effet s'il y a beaucoup de triangles d'émission la première méthode

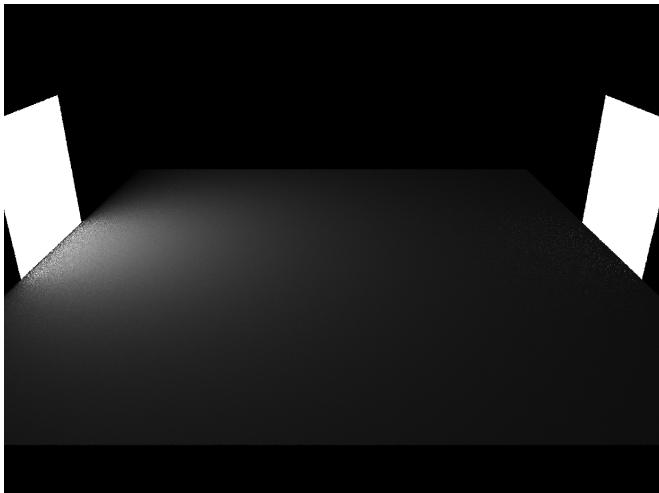


Fig. 4. Source uniformément aléatoire

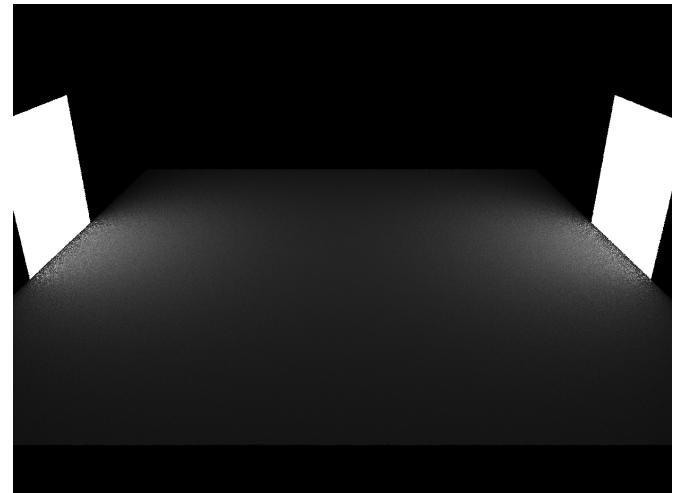


Fig. 5. Source aléatoire améliorée

va devenir très lente très vite. Dans la seconde, l'éclairage va alors favoriser les sources qui ont beaucoup de triangles. Dans les 2 cas, ces méthodes peuvent générer des résultats comme dans Fig 4.

Pour éviter ce problème, une méthode est de sélectionner une sources aléatoirement avec un aléatoire proportionnel à l'aire du triangle d'émission. L'algorithme de sélection est décrit à Algorithm 1. Trier les triangles permet une exécution plus rapide de l'algorithme.

#### Algorithm 1: Sélection aléatoire relative à l'aire

```

Data: Une liste des sources src triée par aire
      décroissante.;

Un nombre aléatoire u entre 0 et 1

Résultat: Une source de lumière sélectionnée
      aléatoirement relativement à son aire
area_accumulator = 0;
while u > area_accumulator do
    | s = src.next();
    | area_accumulator += s.area()/src.area();
end
return s;
```

#### D. L2

1) *Direction aléatoire*: L'éclairage indirect (L2) consiste à relancer des rayons depuis *p*. Si ces rayons intersectent le BVH (en *q*), on va ajouter la valeur de L1 en *q* à *p*. L'accumulation de toutes ces valeurs L1 indirectes est la valeur L2 en *p*.

Les rayons aléatoires sont générés dans la demi-sphère unité selon la méthode 35 du Global Illumination Compendium.

2) *Réutilisation des L1 pré calculés*: Pour éviter de recalculer des L1 aux points où ils ont déjà été calculé lors de la passe L1 (Fig 3.), on peut réutiliser ce résultat. Cependant, ceci ne fonctionne que si *q* est visible dans l'image (sinon son L1 n'aura pas été calculé). Pour vérifier cela, on inverse *q* pour trouver ces coordonnées dans l'images (*px*, *py*). On regarde

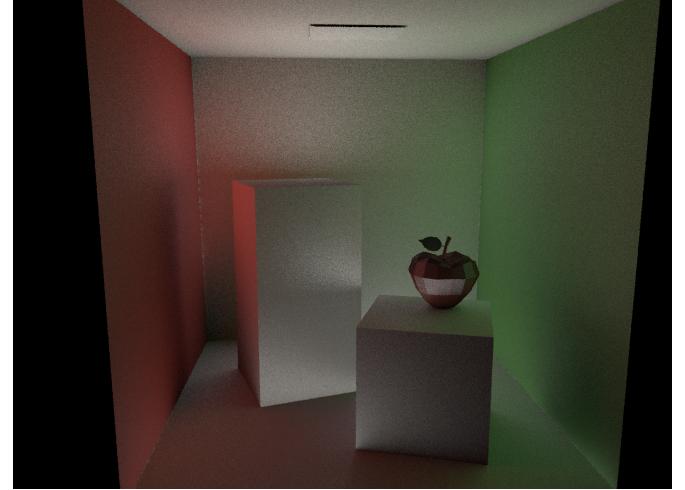


Fig. 6. L2

alors si le rayon lancé de (*px*, *py*) (déjà lancé donc il suffit de l'avoir stocké et de vérifier) intersecte le BVH en *q* ou sur un autre point. Si il intersecte en *q* il suffit de récupérer le L1, sinon il faut le recalculer. Pour recalculer le L1, comme sa valeur précise est moins importante dans le L1 car on va moyenne plusieurs rayons, on peut lancer moins de rayons (facteur 1/8 par exemple).

On peut voir en Fig 7 l'utilisation des L1 sans recalculs pour calculer L2. On voit que cela fonctionne bien au premier plan, là où tous les rayons vont intersecter de la géométrie affichée dans l'image. On voit aussi que dans le coin au fond, la lumière n'est plus bonne, car le L2 des points sur les murs ont besoin du L1 de derrière les boîtes, or celui-ci n'a pas été pré-calculé. On voit en Fig 8 le L2 des points où le L1 a été recalculé. On voit bien qu'au premier plan qu'aucun L1 n'est recalculé, car ils sont tous dans le screen space. Dans le fond de la boîte en revanche, comme beaucoup de géométrie est cachée, on va beaucoup plus recalculer les L1. L'image 6 est donc l'agrégation de ces 2 images (pas besoin de les calculer

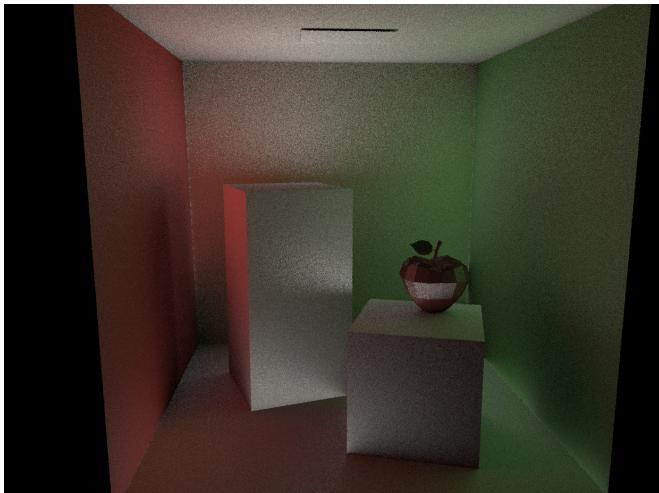


Fig. 7. L2 réutilisant les L1 déjà calculés

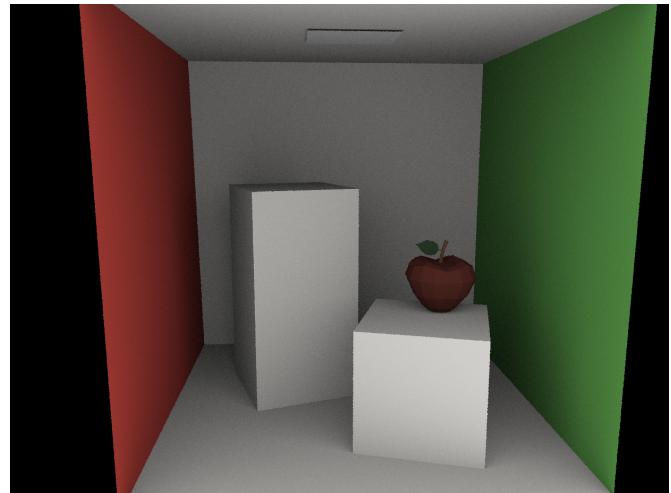


Fig. 9. Occultation ambiante



Fig. 8. L2 avec recalc de L1

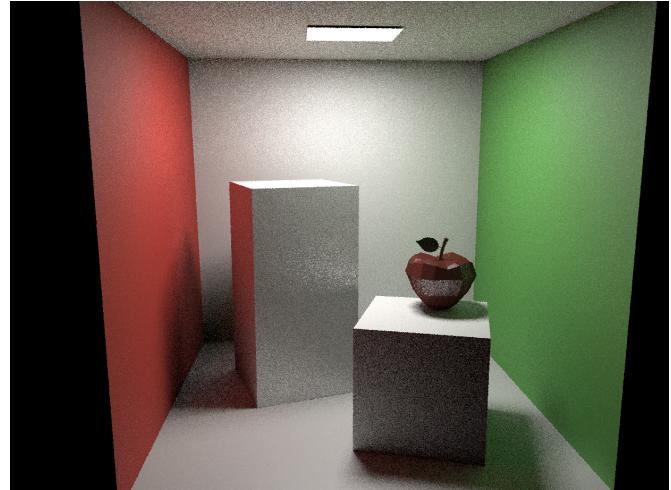


Fig. 10. 1 image

indépendamment, pour chaque rayon on regarde dans quel cas on se trouve).

#### E. Occultation ambiante

L'occultation ambiante n'est pas présente dans l'image 1 car elle permet de remplacer L2 si elle n'est pas calculable. Elle est calculée en lançant des rayons depuis  $p$  (Compendium 35) et en regardant si les rayons intersectent le BVH. Si le rayon n'intersecte pas le BVH, on accumule l'éclairage en ce pixel. La géométrie est donc éclairée par son accessibilité à l'extérieur.

#### F. Moyennage pour anti aliasing

Afin de réduire le bruit sur un image, on peut lancer plus de rayons, ou calculer plusieurs images et moyenner. Les images suivantes ont toutes été calculées avec le même nombre de rayons, mais elles ont été calculées un certain nombre de fois afin de faire à la fois de l'anti aliasing et du lissage. En effet, pour éviter l'aliasing de base, au lieu de lancer le rayon depuis

$(px, py)$  on le lance depuis  $(px + u01(), py + u01())$  avec  $u01()$  générant des nombres aléatoires entre 0 et 1. Quand on calcule plusieurs images superposées, cette variation aléatoire sert d'anti aliasing. Les images 10 à 14 ont été générées avec 16 rayons de L1 et 16 rayons de L2.

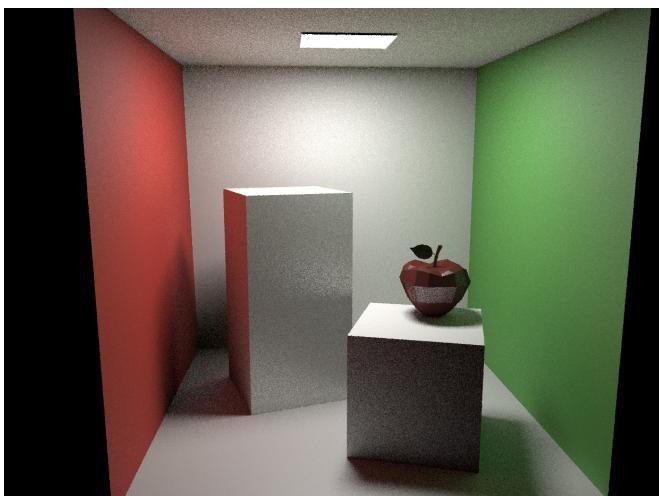


Fig. 11. 2 images



Fig. 14. 32 images

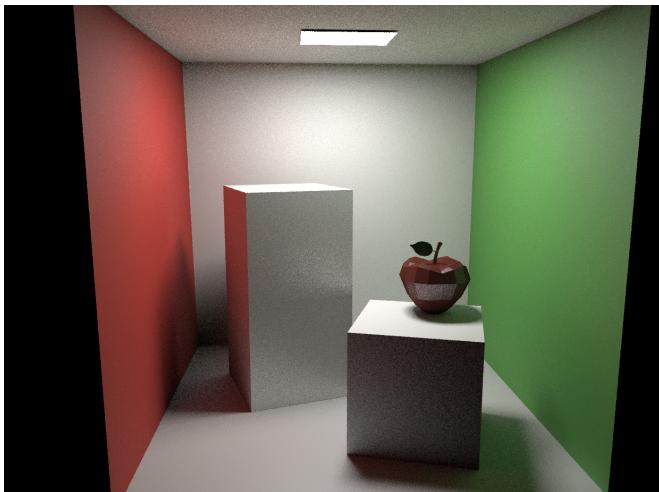


Fig. 12. 4 images

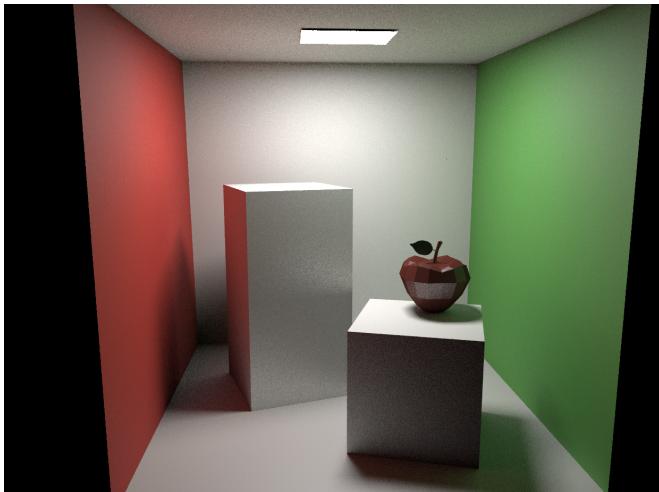


Fig. 13. 8 images

## II. AMÉLIORATIONS

### A. Lissage des normales

Comme on le voit sur l'image, le rendu de la pomme low poly est quelque peu étrange. C'est dû à la discréétisation forte des angles le long de la courbe de la pomme. Il serait intéressant d'interpoler les normales afin de lisser l'objet pour avoir un rendu plus lisse et agréable, on pourrait de plus mieux voir le reflet de Blinn-Phong sur le dessus de la pomme, qui a ici du mal à se distinguer avec le L2 de la pièce.

### B. Stockage 3D des L1

Comme décrit en I-D2, stocker les L1 déjà calculés permet de sauver beaucoup de temps. Il pourrait être intéressant de stocker tous les L1 calculés lors du parcourt des L2. Cependant cela pose de nombreux problèmes. Comment stocker ? (par discréétisation 3D ?). Comment accéder depuis un point 3D ? Pourrait-on interpoler des L1 stockés ? etc...