TP : Pathtracing (ou presque)

Jonathan Fabrizio
http://jo.fabrizio.free.fr/

Objectif

L'objectif de ce T.P. est de programmer un mini "presque" pathtracer.

The goal of this practical work is to implement a sort of a pathtracer.

1 Travail préliminaire — Preliminary work

Téléchargez le squelette du programme. Regardez les sources et essayez de comprendre ce qu'il fait et comment il fonctionne.

Download the skeleton of the program. Have a look to the source code and try to understand the content of the program.

2 Les intersections — Intersections

Deux ou trois fonctions sont vides ou presque, notamment `double intersect(const Ray &ray)` dans `Sphere` et `radiance(const Scene &scene, const Ray &ray, int depth)`.

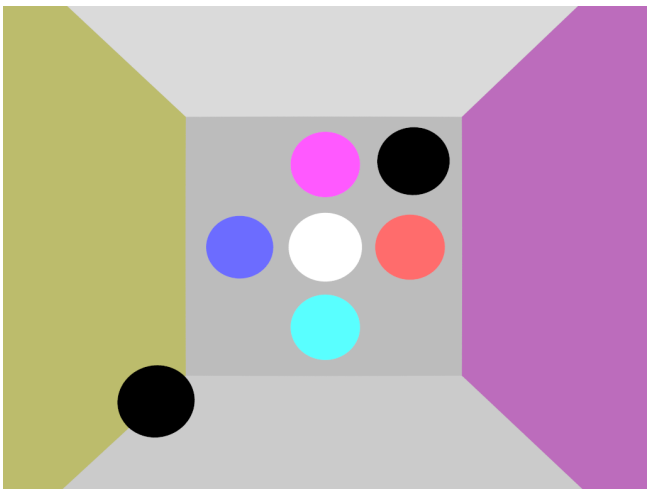
Question 1 : Complétez `double intersect(const Ray &ray)` afin de calculer les intersections des `Ray` avec `Sphere`.

Question 2 : Afin de voir un résultat et vérifier que l'intersection fonctionne, implémentez `radiance(const Scene &scene, const Ray &ray, int depth)`. Renvoyer simplement la couleur de l'objet intersecté sans tenir compte de l'illumination ou des ombres.

Two or three functions are empty or near empty, especially `double intersect(const Ray &ray)` in `Sphere` and `radiance(const Scene &scene, const Ray &ray, int depth)`.

Question 1 : Fill-in `double intersect(const Ray &ray)` to compute the intersections of the light rays `Ray` with the `Sphere`.

Question 2 : In order to see a result and check the correctness of the intersection, implement `radiance(const Scene &scene, const Ray &ray, int depth)`. Simply return the color of the intersected object without taking into account lighting and shadows.



3 La lumiere indirecte Indirect lighting

Maintenant que l'intersection fonctionne, implementez `radiance(const Scene &scene, const Ray &ray, int depth)` :

Question 1 : Implementez le cas `EMISSIVE` : renvoyez `get_emission()`. L'image ne change pas beaucoup.

Question 2 : Implementez le cas `DIFFUSE` : relancez `nb_samples` à l'aide de `generate_random_point_on_sphere()` et moyennez leur resultats pour trouver la contribution de la lumière.

Question 3 : Implementez le cas `REFLECTIVE` : relancez le rayon réfléchit sans décompter `depth`. Du coup implémentez `Vector reflected(const Vector normal)`.

Au besoin réglez `max_bounces` et `nb_samples`, limitez le nombre de samples pour limiter le temps d'execution.

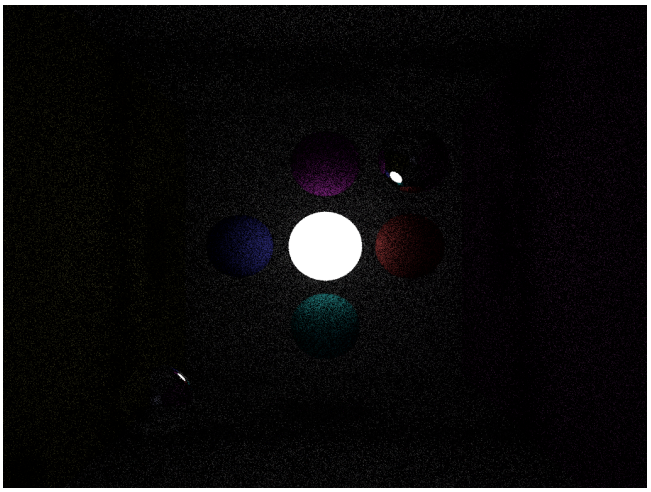
If the intersections are correct, you can implement `radiance(const Scene &scene, const Ray &ray, int depth)`:

Question 1 : Implement `EMISSIVE`: return `get_emission()`. That does not really change the image.

Question 2 : Implement `DIFFUSE`: cast `nb_samples` using `generate_random_point_on_sphere()` and compute the mean of all results to deduce the contribution of the light.

Question 3 : Implement `REFLECTIVE`: cast reflected ray without decrementing `depth`. You need to implement `Vector reflected(const Vector normal)`.

If needed change `max_bounces` and `nb_samples`, limit `nb_samples` to reduce executing time.



4 La lumiere directe Direct lighting

Nous avons un pathtracer. Il n'est pas correct car l'envoie des rayons ne tient pas compte de la BRDF mais cela permet de comprendre le principe.

Un problème c'est qu'à moins de faire beaucoup de rebonds (et donc d'avoir un temps d'execution trop long), l'image est trop sombre. Nous allons donc ajouter l'apport des lumieres directes même si ça fausse encore un peu plus notre pathtracer. Dans `radiance(const Scene &scene, const Ray &ray, int depth)` :

Question 1 : Ajouter dans le cas `DIFFUSE` la contribution des lumières directes. Pour faire simple, lancer un rayon par lumiere. Quel est le problème avec cette approche ?

Question 2 : Améliorer le resultat en lançant plusieurs rayons par lumiere. Quelle est l'amélioration ?

Au besoin réglez `max_bounces` et `nb_samples`, limitez le nombre de rebonds pour limiter le temps d'execution.

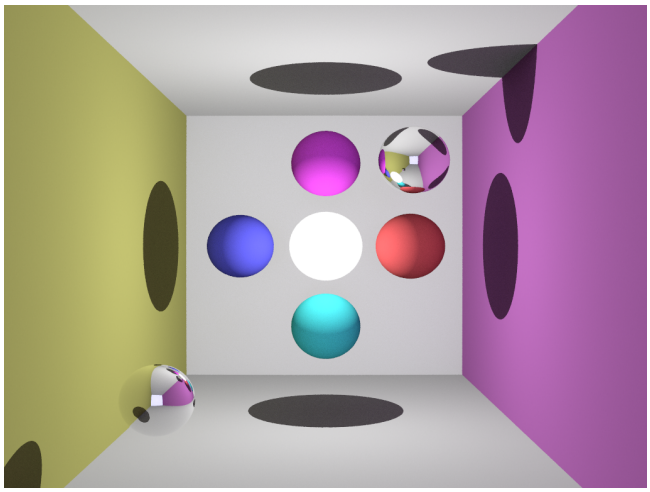
We have our pathtracer. It is not correct because we do not cast rays according to the BRDF but I do hope that you understand the principle.

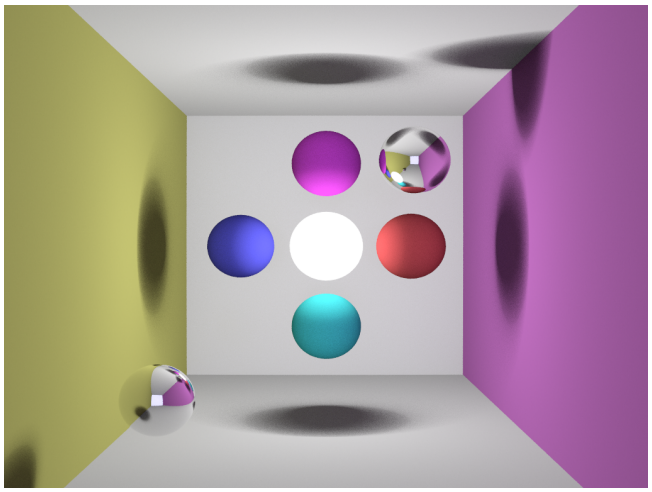
A major problem is that the image is too dark (unless computing a lots of bounds but this increase too much the executing time). So we will add the contribution of direct lighting even if the result will be far from a real pathtracer. In `radiance(const Scene &scene, const Ray &ray, int depth)` :

Question 1 : Append in `DIFFUSE` the contribution of direct lighting. To do so, cast a ray in the direction of the light source. What is the problem with this strategy?

Question 2 : Improve the result by casting multiple rays. What is the improvement?

If needed change `max_bounces` and `nb_samples`, limit `max_bounces` to reduce executing time.





5 Tests Tests

Tester les différents cas, l'impact de chaque parametres. En particulier :

Question 1 : Supprimer l'encodage en sRGB, surtout dans le cas "indirect lighting only". Que constatez-vous ?

Question 2 : Entre `max_bounces` et `nb_samples` qui est le plus impactant dans le cas "direct" et "indirect" ?

Question 3 : Il vous reste du temps ? Ajouter le cas transparent ou changer l'envoie des rayons pour tenir compte de la BRDF.

Test different cases. Measure the impact of each parameter. In particular:

Question 1 : Remove sRGB encoding, especially for "indirect lighting only". What do you see?

Question 2 : Between `max_bounces` and `nb_samples` which one is the most painful for "direct" and "indirect" lighting?

Question 3 : You still have the time? Add the `TRANSPARENT` case or cast rays according to the BRDF.