Quentin Wendling - Nathan Urbain , CMI IRVIIJ

Rapport de devoir libre de Visualisation

Le projet a été réalisé en C++ avec Qt.

Interface

La première étape de notre projet a été de réaliser un environnement fonctionnelle sous OpenGL, pour cela nous avons donc utilisé QGLViewer pour afficher des objets 3D et pour pouvoir manipuler une caméra dans la scène. Le chargement d'objet 3D dans la scène a été réalisée avec l'aide de la librairie assimp, ce qui nous permet d'avoir un contrôle total sur les objets que nous voulons rendre aussi bien d'un point de vue géométrique que d'un point de vue matériaux.

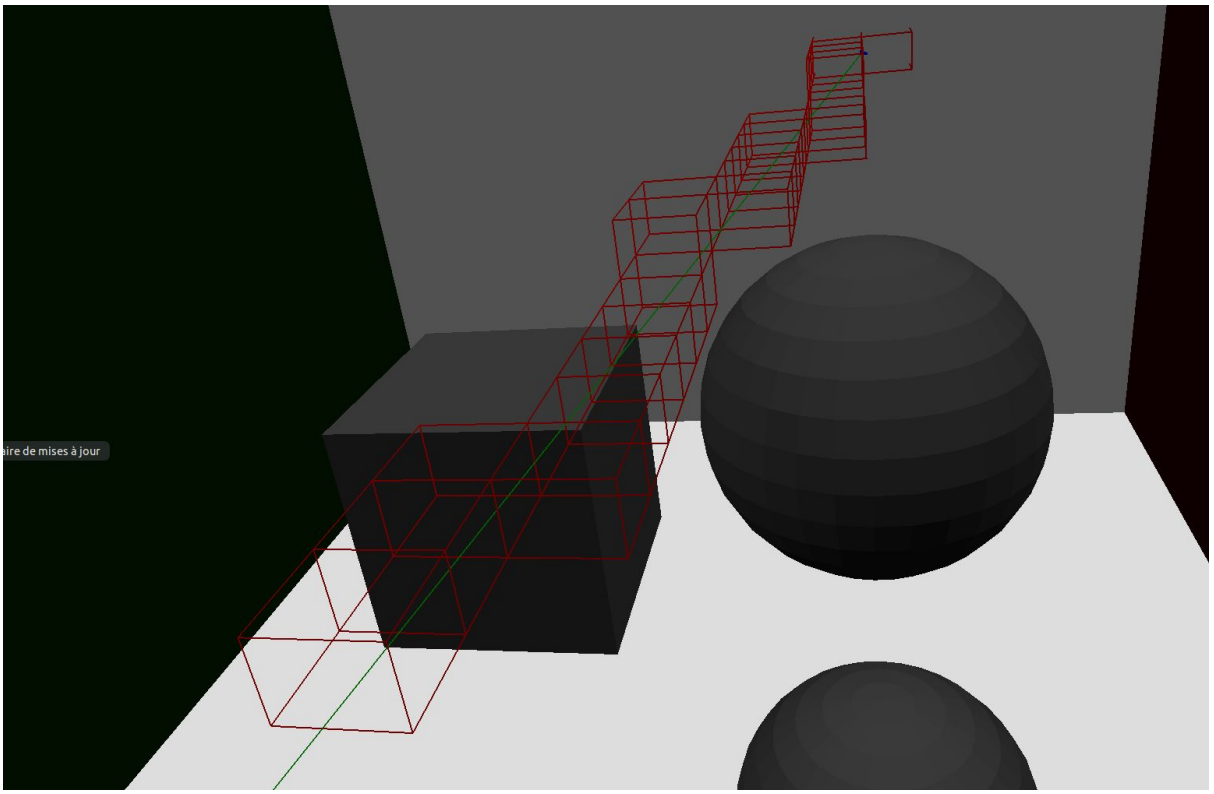
Classes de base

Avant de commencer le développement du premier lancer de rayon, les différentes classes de base ont été développées.

La classe Triangle" représente des triangles qui sont les éléments de base de tous objets 3D. Ils sont représentés par trois points, une normale ainsi que les différentes informations de couleurs, ils disposent également d'un indice unique permettant de les différencier, utile pour la suite du projet. Enfin la classe dispose d'une méthode permettant de savoir si un point est à l'intérieur du triangle ainsi qu'une seconde calculant la normale du triangle.

La classe "Rayon" permet de représenter nos rayons, ils sont représentés par une origine ainsi qu'une direction. Ils disposent également d'une méthode permettant de savoir s'ils intersectent un triangle et une seconde renvoyant l'intersection la plus proche de l'origine dans une liste de triangle ignorant les intersections situées derrière l'origine.

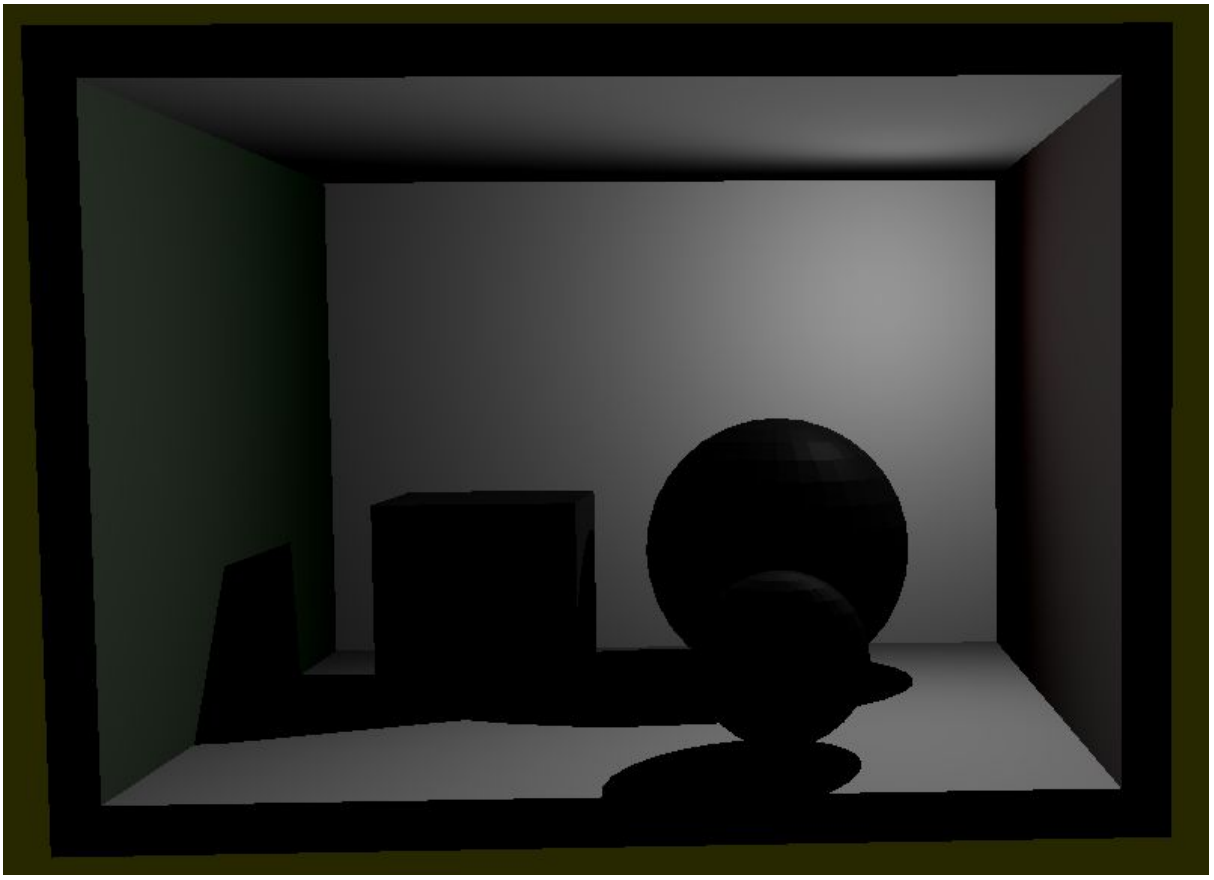
Nous disposons donc d'une méthode pour détecter le point d'intersection d'un rayon avec une scène 3D mais la complexité de la méthode la rend inutilisable en pratique. Nous avons donc implémenté un algorithme de 3DDDA (3D Digital Differential Analyser) qui consiste à voxelisé la scène dans une grille régulière et de parcourir cette grille le long d'un rayon avec le principe des droites de bresenham.



Tous ceci nous permis enfin de commencer les différentes méthodes de lancer de rayon et rendus. Rendus obtenus par dessin de la scène pixel par pixel grâce à un QPainter sous Qt et lancer de rayon en multiThread.

Phong

Premier type de rendu “Phong”, avec normalisation des coefficients (Kd_normalise = Kd/Pi et Ks_normalise = Ks/2*n). Ce rendu est effectué en deux temps, nous commençons par lancer un rayon depuis un pixel, et calculons son intersection. Puis nous lançons un autre rayon depuis ce point vers la lumière afin de savoir si celle-ci est visible ou non si elle ne l’est pas le point est dans l’ombre sinon nous calculons un éclairage de phong sur celui-ci.



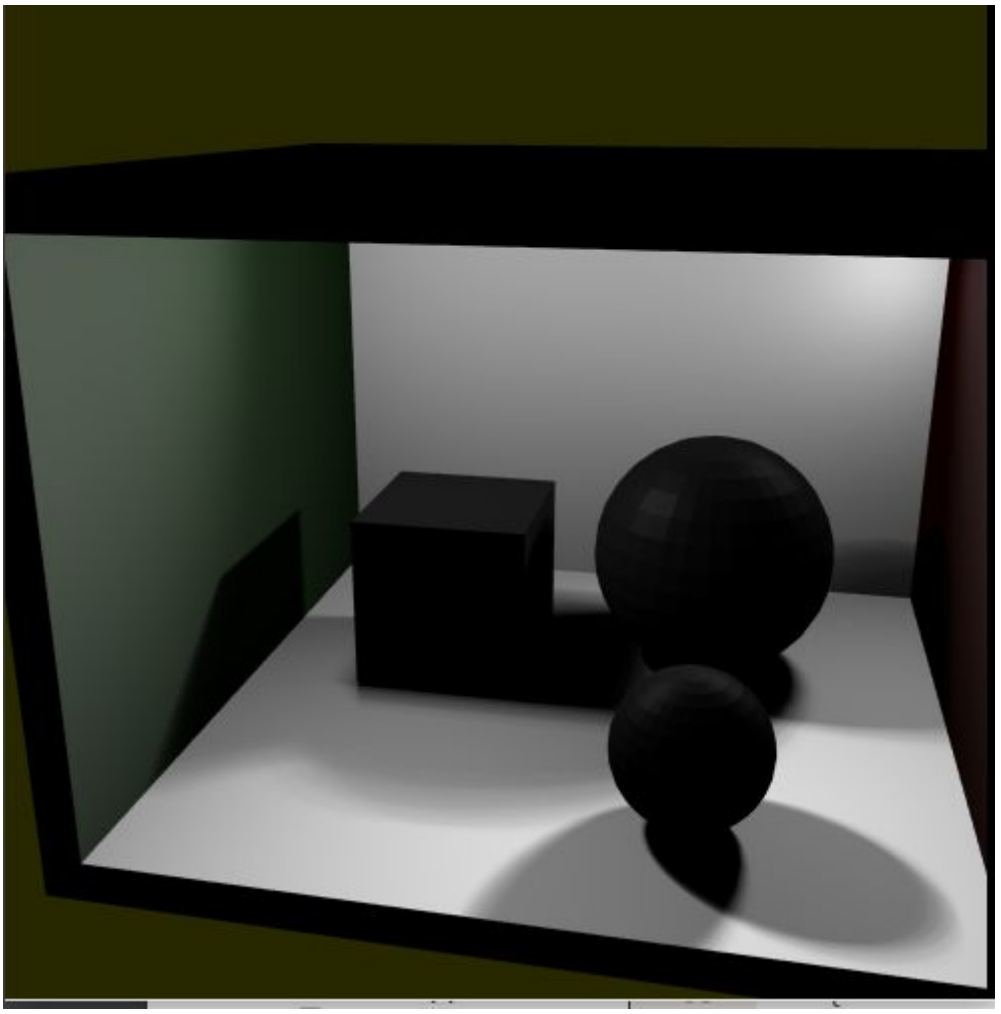
Lancer de rayon

Lancer de rayon “Stochastique” & “pénombre” & Réflexion “Floue”

Le lancer de rayon stochastique revient à appliquer phong mais cette fois en divisant chaque pixel en sous pixel. Cette division est effectuée avec le principe des disques de Poisson approximés par jittering, chaque pixel est donc divisé en une grille régulière et un rayon est aléatoirement lancé dans chaque case de cette grille. Nous avons ensuite créé une classe permettant de définir des sources de lumière surfacique, une source surfacique est définie comme un triangle dans lequel nous tirons aléatoirement un certain nombre de points qui nous serviront à rendre des pénombres.

En effet lorsque nous allons vérifier si un point est visible nous allons vérifier depuis quelle portion de la source surfacique il est visible ce qui va nous permettre d’avoir de la pénombre.

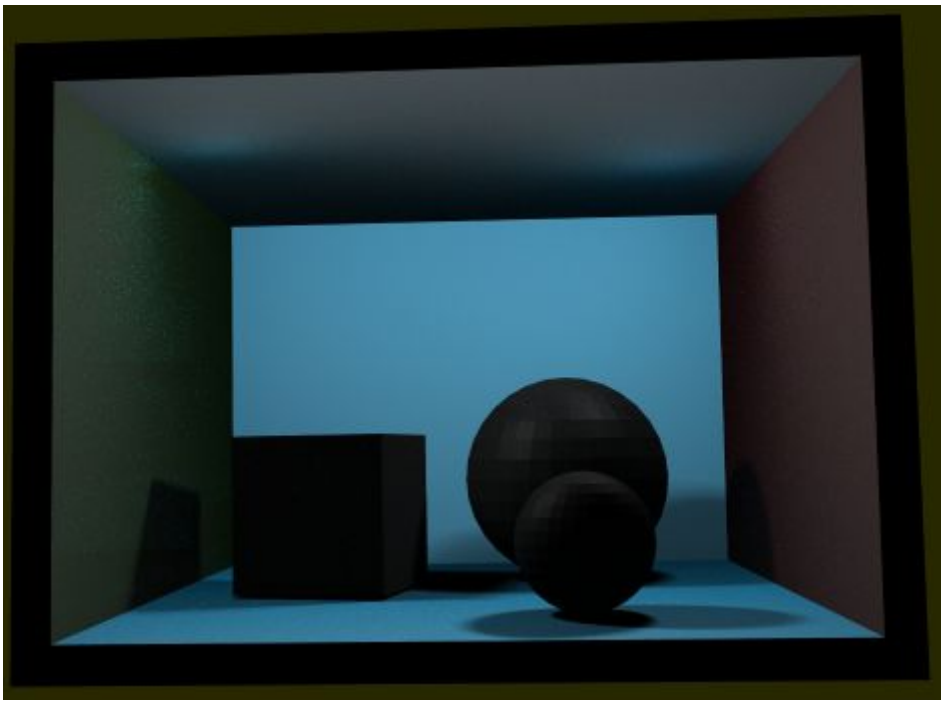
La réflexion floue, lors du calcul de la couleur spéculaire, utilise une direction aléatoire dans un cône centré sur la direction de réflexion parfaite, méthode choisie à l’instar de la direction de réflexion parfaite seule.



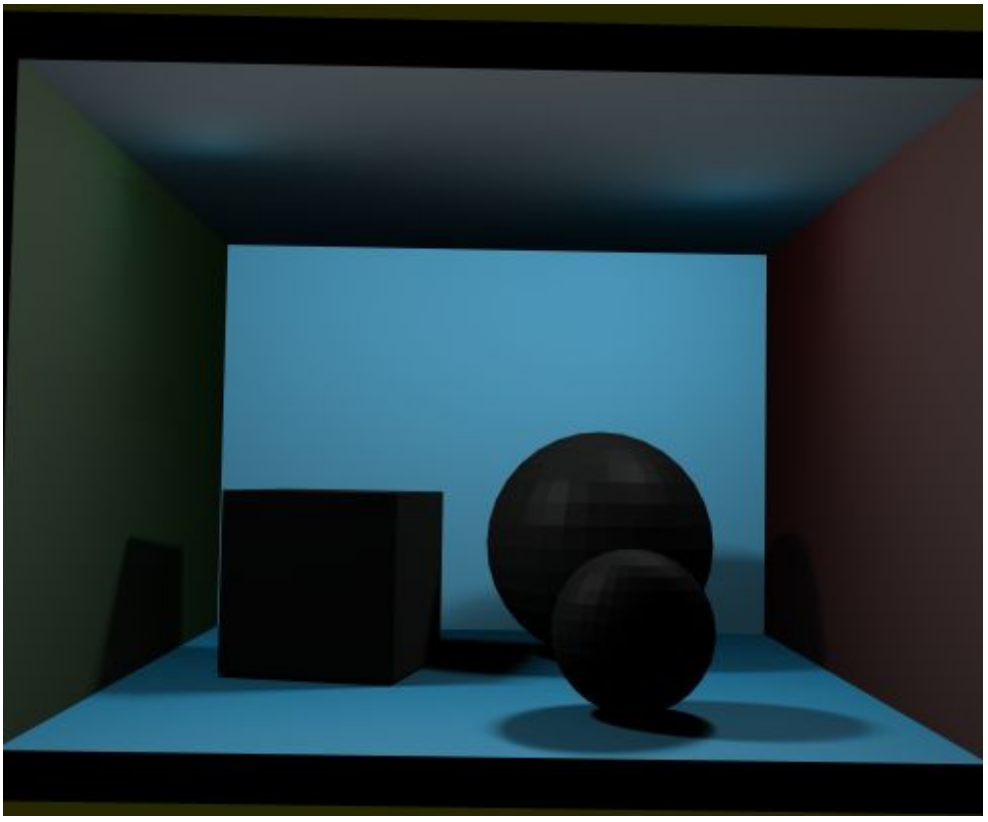
Travail supplémentaire

Lancer de rayon avec Photon mapping

Par la suite l'utilisation d'une photon map pour le calcul de l'éclairage indirect a été ajouté. La photon map est obtenu en tirant des photons aléatoires dans une sphère centrée sur les points de nos sources surfaciques. Au contact d'une surface chaque photon va selon le principe de la roulette russe soit émettre un photon de même énergie dans la direction de réflexion parfaite, soit être absorbé.
Ainsi une fois un point d'intersection détecté avec notre rayon lancé, un autre rayon dans une direction aléatoire choisie dans un hémisphère centré sur ce point est lancé. Ensuite, nous cherchons les photons dans un rayon de 1 autour du point d'intersection du dernier rayon lancé avec la scène. Nous utiliserons enfin une gaussienne pour pondérer la contribution de chacun de ces photons à la couleur du pixel.



Meilleur rendu



900 rayons par pixels

Information d’utilisation :

Pour lancer le projet compiler le projet et lancer l’application. La librairie PCL doit être disponible, et la librairie Assimp compilé dans le dossier du même nom.

Le contenu du dossier assimp/lib doit ensuite être mis dans le dossier bin à la racine du projet.

Touches :

Entré (pavé numérique) -> Lance le rendu (photon map)
Shift + clic gauche -> Lancer de rayon de debug
shift + clic droit -> Déplacement de la lumière active
Tab -> Changement de la lumière active
P -> Pour placer la source de lumière ponctuel utilisé dans phong
R -> Lancer de rayon phong
B -> Lancer de rayon stochastique