



ÉCOLE  
**CENTRALE**LYON

# ÉCOLE CENTRALE LYON

MOS 2.2 INFORMATIQUE GRAPHIQUE  
INTRODUCTION AU RAYTRACING

## Rapport de BE

*Élève :*  
Hugo MAILFAIT

*Enseignant :*  
Nicolas BONNEEL

## Table des matières

<b>Introduction</b>	<b>2</b>
<b>1 Crédit de la première scène</b>	<b>2</b>
1.1 Les opérations fondamentales . . . . .	2
1.2 Apport de l'éclairage . . . . .	3
1.3 Création de la scène . . . . .	3
<b>2 La gestion des rayons secondaires</b>	<b>4</b>
2.1 Les ombres portées . . . . .	4
2.2 La correction gamma . . . . .	5
2.3 Les surfaces "miroir" . . . . .	5
2.4 Les surfaces transparentes . . . . .	6
<b>3 L'éclairage indirect</b>	<b>7</b>
3.1 Résumé théorique . . . . .	7
3.2 Résultat et améliorations . . . . .	7
<b>4 Les ombres douces</b>	<b>9</b>
4.1 Une première méthode naïve . . . . .	9
4.2 Modification de la méthode d'échantillonnage . . . . .	10
4.3 Amélioration du modèle de caméra . . . . .	11
<b>5 Les maillages</b>	<b>11</b>
5.1 Création d'un premier maillage . . . . .	12
5.2 Ajout d'une <i>Bounding Box</i> . . . . .	12
5.3 <i>Bounding Volume Hierarchies</i> . . . . .	13
<b>6 Les textures</b>	<b>13</b>
<b>7 Déplacement de la caméra</b>	<b>14</b>

# Introduction

Le rendu est un processus informatique calculant l'image d'une scène créée dans un logiciel de modélisation 3D. Cette scène comporte à la fois des objets et des sources de lumière et elle est observée d'un point de vue précis. Ce processus peut faire appel à différentes techniques de calcul , dont les plus connues sont le *raytracing* et la rastérisation. Les méthodes de *raytracing* sont plus lentes à réaliser mais permettent de simuler correctement et facilement les interactions lumière-matière. Elles sont notamment utilisées au cinéma.

Le principe du *raytracing* est de simuler le comportement de la lumière. On utilisera notamment le principe de Helmholtz selon lequel on peut suivre les rayons de lumière en sens inverse (depuis le capteur vers les sources de lumière) et obtenir le même résultat. Des rayons de lumière seront donc lancés depuis la caméra en direction de chaque pixel de l'image.

## 1 Crédit de la première scène

### 1.1 Les opérations fondamentales

Les opérations fondamentales dans un *raytracer* sont la génération de rayons et le calcul d'intersections entre un rayon et une primitive géométrique.

Les points d'intersections entre un rayon et une sphère résolvent à la fois l'équation de la sphère et l'équation du rayon généré. Il s'agit donc de l'ensemble des t tels que :

$$\|C + t.V - O\|^2 = R^2$$

avec O le centre de la sphère de rayon R, V le vecteur directeur du rayon de lumière, C son origine, et t un paramètre le long du rayon.

La résolution de cette équation aboutit à un polynôme de degré 2. Selon les valeurs du discriminant, il y a 0, 1 ou 2 solutions. Dans le cas où il y en a 2, on récupère celles positives. Si là encore il y a deux choux possibles, on sélectionne la plus proche de la caméra, donc la valeur de t la plus faible.

En plus de cette mise en place de l'algorithme d'intersection, il faut également définir les classes de base nécessaires au projet. Il faut notamment une classe Vector pour manipuler les coordonnées, une classe Ray, qui représente un rayon de lumière et une classe Sphere. L'ensemble de ces éléments permet d'aboutir à l'image suivante :

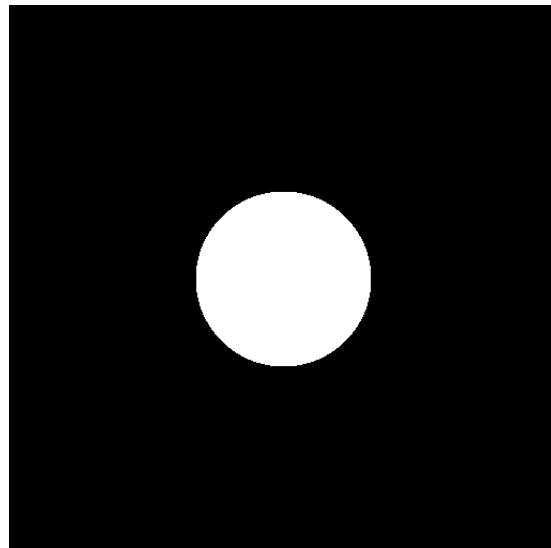


FIGURE 1 – Intersection de rayons avec une sphère

## 1.2 Apport de l'éclairage

Dans l'image précédente, il est impossible d'observer le relief de la sphère. L'image semble être en 2D. Pour résoudre ce point, on se propose d'apporter de l'éclairage. Les pixels éclairés ne sont alors plus tout blancs ou tout noirs : leur intensité dépend de la position du pixel par rapport à la source de lumière et de la distance les séparant.

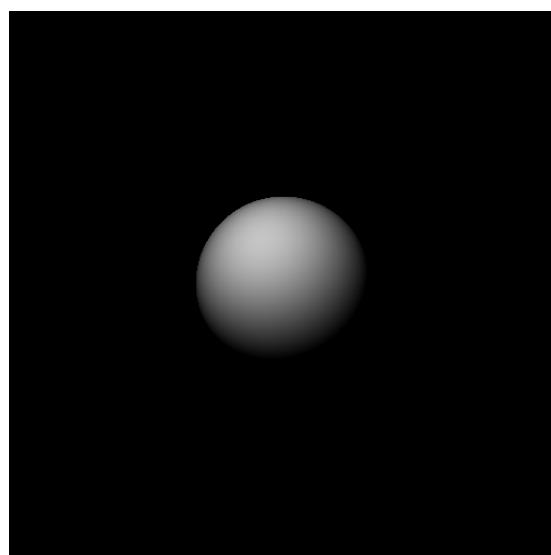


FIGURE 2 – Sphère éclairée (matériau diffus)

## 1.3 Création de la scène

Dans cette partie, on cherche à créer une scène, à savoir un tableau de plusieurs sphères. Cette nouvelle classe "Scene" doit alors elle aussi disposer d'une méthode permettant de gérer les intersections. Elle est définie comme l'intersection la plus proche de l'origine du rayon parmi les intersections avec l'ensemble des sphères de la scène. Pour notre scène, on décide de créer des murs, un sol et un plafond à l'aide de sphères géantes.

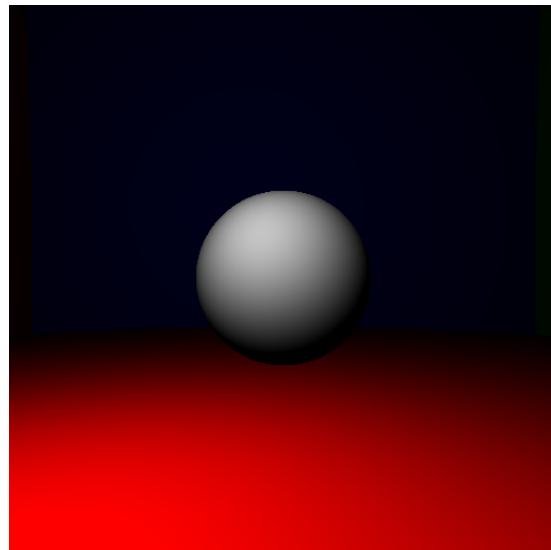


FIGURE 3 – Réalisation de la première scène

## 2 La gestion des rayons secondaires

### 2.1 Les ombres portées

Dans cette section, on aimerait observer la projection de l'ombre des objets sur les autres objets. Algorithmiquement, il faut donc générer un nouveau rayon depuis le point d'intersection trouvé vers la source de lumière. Si une intersection est détectée pour ce nouveau rayon et que la distance à ce point est inférieure à la distance entre la source et la première intersection, alors il y a une ombre !

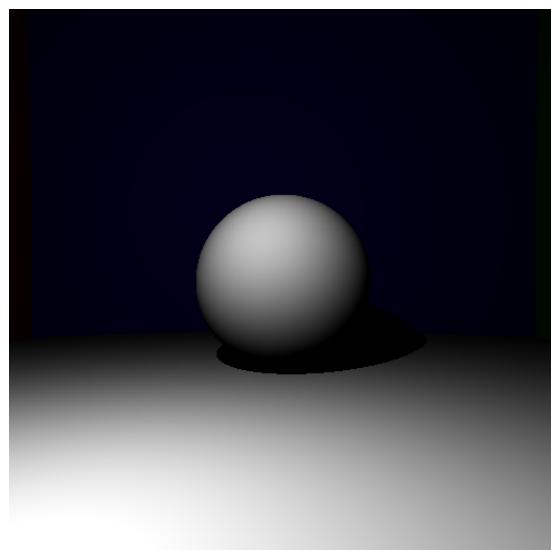


FIGURE 4 – Réalisation de la première scène

## 2.2 La correction gamma

On observe sur l'image que les intensités lumineuses affichées ne semblent pas varier linéairement avec les valeurs données. En effet, les écrans appliquent une puissance 2,2 à la luminosité. Il faut donc compenser cette transformation dans notre code pour observer les couleurs attendues.

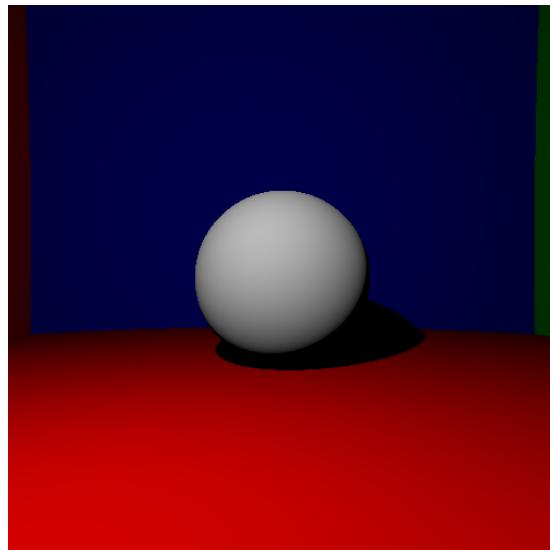


FIGURE 5 – Première scène avec la correction gamma

## 2.3 Les surfaces "miroir"

Cette fois-ci, au lieu de renvoyer un rayon en direction de la source de lumière, on veut réfléchir le rayon incident et renvoyer la couleur de ce qui est réfléchi. Ce nouveau rayon pourrait alors être réfléchi par un nouvel objet et ainsi de suite ...

Pour réaliser un tel procédé, il faut créer une fonction en charge de déterminer la couleur d'un rayon vers la scène. Elle prendra en compte le cas décrit ci-dessous via une méthode récursive. Par ailleurs, il convient d'ajouter une propriété à la classe "Sphere" pour indiquer si l'objet est un miroir ou non.

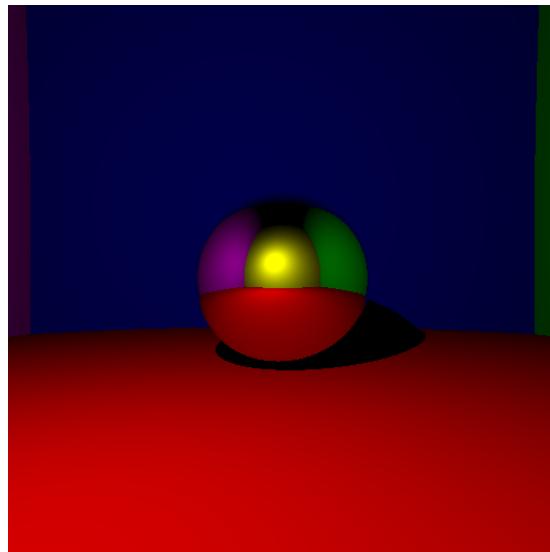


FIGURE 6 – Cas d'une sphère "miroir"

## 2.4 Les surfaces transparentes

On s'intéresse maintenant au cas de sphères transparentes et à la gestion de rayons réfractés. En effet, si l'indice de réfraction de la sphère et du milieu sont différents, alors les lois de Descartes-Snell prouvent l'existence de rayons réfractés. Les calculs mathématiques permettent d'obtenir les formules des composantes tangentielles et normales pour modéliser ce phénomène.

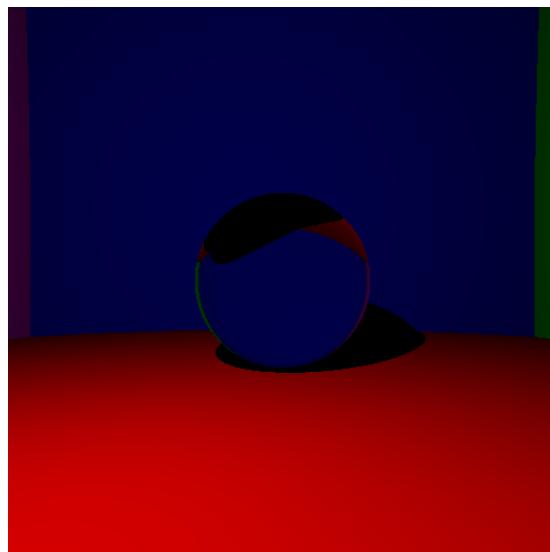


FIGURE 7 – Cas d'une sphère transparente

Sur la figure obtenue, on remarque que la couleur de la sphère correspond à la couleur située sur le mur opposé, le plafond etc ... On remarque d'ailleurs sur le sommet de la sphère que l'on retrouve l'ombre portée.

## 3 L'éclairage indirect

### 3.1 Résumé théorique

La prise en charge de l'éclairage indirect fonctionne de manière très similaire à celle des surfaces spéculaires. Cette fois-ci, les surfaces diffuses font rebondir le rayon de lumière de manière aléatoire, contrairement aux surfaces spéculaires qui renvoient le rayon dans la direction dite "miroir".

Pour modéliser ce comportement, on introduit l'équation du rendu. Il s'agit de l'équation définissant la couleur d'un pixel. Elle s'exprime comme ceci :

$$L_o(x, \vec{o}) = E(x, \vec{o}) + \int f(\vec{i}, \vec{o}) L_i(x, \vec{i}) \cos \theta_i d\vec{i}$$

Avec :

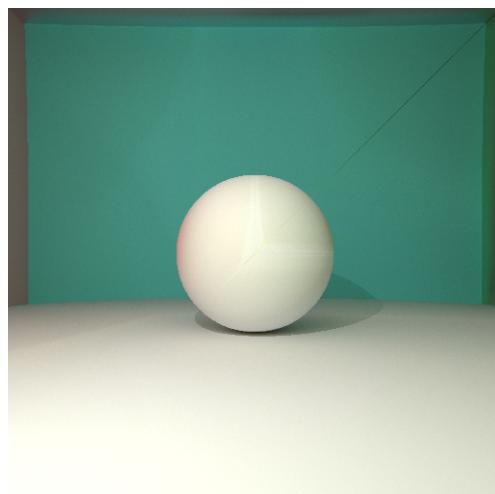
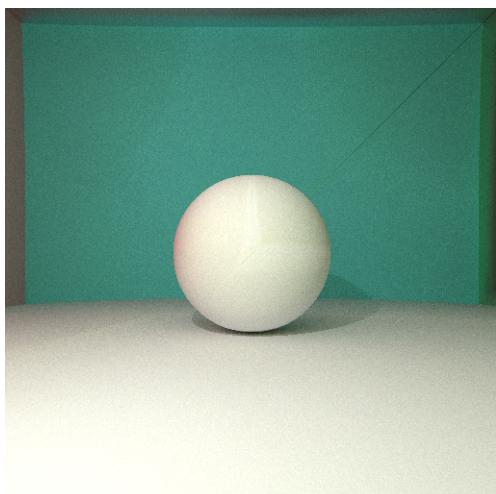
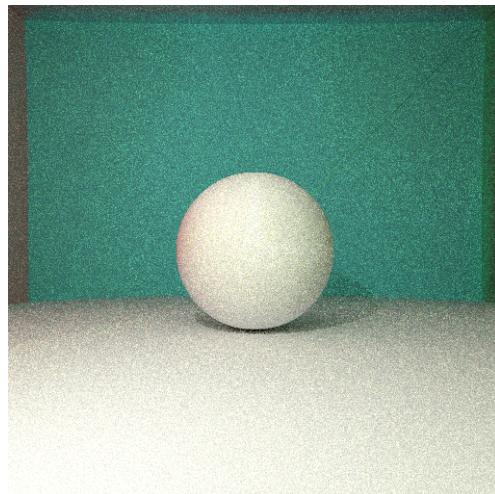
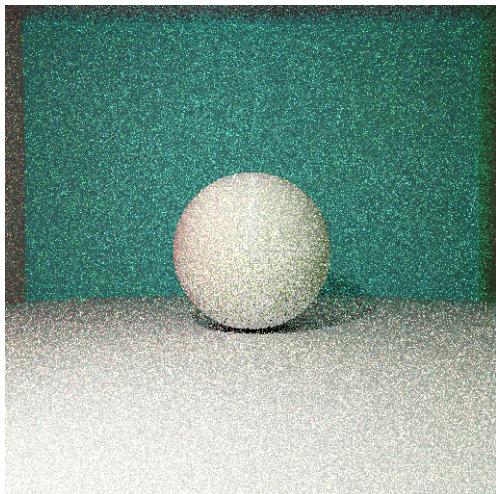
- $L_i(x, \vec{i})$  l'intensité lumineuse qui sort du point d'intersection ;
- $L_o(x, \vec{o})$  l'intensité lumineuse arrivant à ce point ;
- $E(x, \vec{o})$  l'émissivité de la surface ;
- $f(\vec{i}, \vec{o})$  la BRDF ;
- $\cos \theta_i$  le cosinus de l'angle entre le rayon incident et la normale à la surface.

Cette équation exprime le fait que la lumière sortante d'un point est la somme de toutes les lumières incidentes multipliées par la BRDF. Par ailleurs, la BRDF est une fonction qui exprime la probabilité pour un photon arrivant à la surface d'un objet avec une direction incidente  $\vec{i}$  de rebondir dans la direction  $\vec{o}$ .

Pour le calcul, on utilisera les méthodes de Monte-Carlo et l'algorithme de Box-Muller pour déterminer la valeur de l'intégrale.

### 3.2 Résultat et améliorations

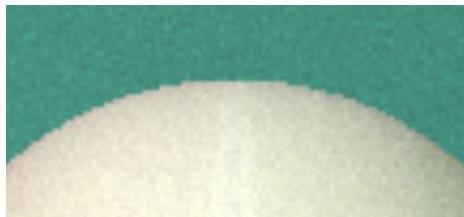
On obtient alors les résultats suivants, avec un nombre de rayons égal à 1, 10, 100, 1000 :



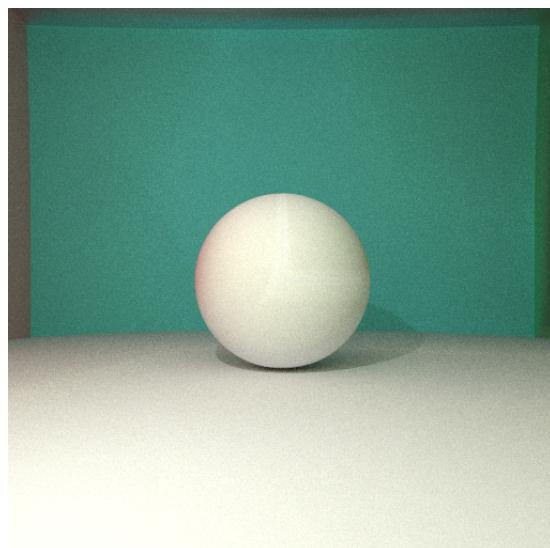
On observe sur l'image un éclairage étrange avec des teintes de blanc plus claires sur certaines zones de la boule et une sorte de halo de lumière au-dessus de celle-ci. Par ailleurs, on remarque également l'apparition d'un trait en diagonale de l'image.

Dans un premier temps, on souhaite accélérer les temps de calcul du programme. En effet, pour obtenir l'image avec 1 000 rayons, entre 15 et 16 minutes sont nécessaires. En activant la prise en charge d'OpenMP sur Visual Studio et en ajoutant une directive *pragma omp*, le temps de calcul passe à 7 minutes pour 1 000 rayons. On ajoute ensuite à l'image On observe un trait qui disparaît avec l'anti-aliasing.

Ensuite, on souhaite faire disparaître l'effet de crénelage observé sur les différentes images. Cet effet provient du fait qu'actuellement tous les rayons tirés passent par le centre des pixels. Pour réaliser cet *anti-aliasing*, on choisit d'ajouter deux variables aléatoires gaussiennes indépendantes aux coordonnées x et y du vecteur directeur du rayon tiré. On obtient la comparaison avant-après suivante, pour 100 rayons :



Par ailleurs, on remarque également que le trait en diagonale disparaît avec cette étape.

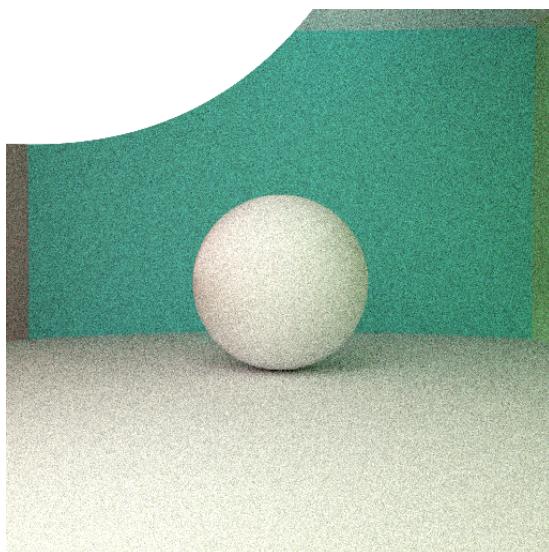
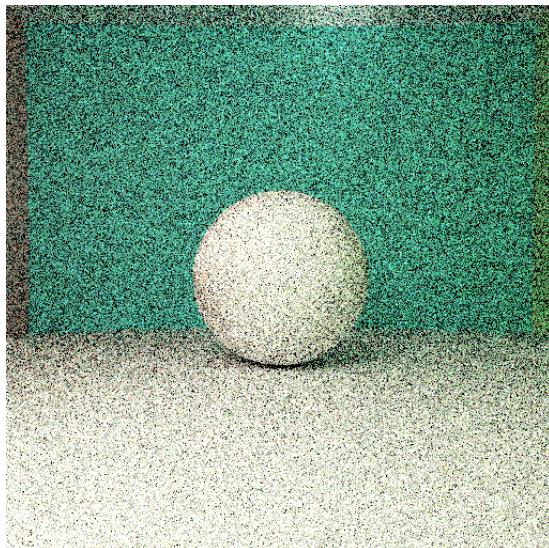


## 4 Les ombres douces

Le concept des ombres douces repose sur la modification de la source de lumière. A partir de maintenant, la lumière ne sera plus supposée ponctuelle mais sphérique.

### 4.1 Une première méthode naïve

Une première méthode pour appliquer ces lumières étendues est de dire que lorsqu'un rayon tiré depuis la caméra arrive par hasard sur la sphère de lumière, on lui assigne son émissivité avec une BRDF nulle.

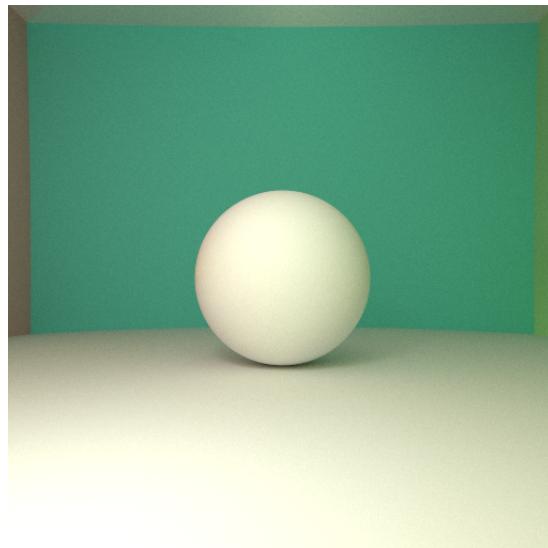


On remarque que cette méthode présente un bruit très important lorsque le rayon de la sphère de lumière est faible, comme sur la première image où il vaut 5 contre 15 sur la seconde. En effet, la probabilité pour un rayon d'intersecter la lumière devient elle aussi faible. Une autre méthode est donc nécessaire.

## 4.2 Modification de la méthode d'échantillonnage

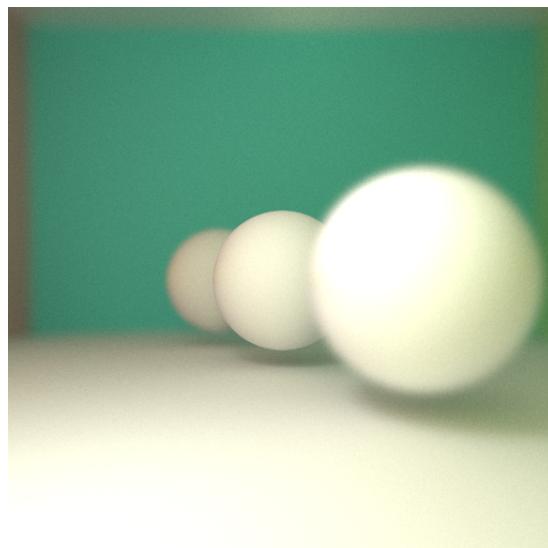
Pour obtenir de meilleurs résultats, on choisit de séparer en deux parties l'hémisphère d'intégration des sphères. La première contient l'éclairage direct, à savoir la partie des sphères où la source de lumière se projette, obtenue en tirer des rayons vers la dite sphère. La seconde partie correspond au reste des surfaces : on tire alors des rayons aléatoires dans tout l'hémisphère, et leur contribution devient nulle s'ils intersectent la sphère de lumière.

Le résultat obtenu pour une sphère de lumière de rayon 5 et 1000 rayons est illustré ci-dessous. On remarque notamment que les problèmes d'éclairage de la partie précédente semblent avoir disparus.



### 4.3 Amélioration du modèle de caméra

On souhaite enfin ajouter du réalisme à notre modèle de caméra en simulant une profondeur de champ. Il s'agit donc de rendre compte de l'effet de flou sur les images selon le placement de l'objet observé. Physiquement, on modifie notre modèle de caméra pour y intégrer le comportement d'une lentille convergente et d'un obturateur d'appareil photo.



## 5 Les maillages

Dans la suite du projet, afin de diversifier les images réalisées, on utilisera des maillages pour simuler les objets de la scène.

## 5.1 Création d'un premier maillage

Un maillage est représenté par une collection de triangles. Précisément, les maillages étudiés seront composés de :

- une liste de sommets ;
- une liste de triangles représentés par un triplet d'indices de sommets ;

En pratique, pour utiliser les maillages, il faut définir l'intersection entre un triangle et un rayon. Pour cela, on considère en premier lieu l'intersection entre un rayon et un plan, puis on vérifie l'appartenance du point d'intersection au triangle à l'aide d'un calcul de coordonnées barycentriques.

Le maillage considéré dans les prochaines étapes du projet est celui d'un berger australien. Par ailleurs, les temps de calcul sont considérablement augmentés. Une baisse de qualité est nécessaire pour obtenir des rendus en des temps acceptables. Par exemple, pour une image de 128 pixels par 128 et 2 rayons par pixels, le temps de calcul est de 5 min 26. Pour 5 rayons et la même dimension (image ci-dessous), on passe à 15 min 06.



## 5.2 Ajout d'une *Bounding Box*

Pour accélérer les temps de calcul, on se propose d'utiliser la méthode de la boîte englobante. Une boîte englobante se définit comme le volume qui englobe l'ensemble des triangles du maillage de l'objet étudié. Cette boîte est définie par 3 paires de 2 plans selon les axes X, Y et Z. L'accélération consiste à déterminer si le rayon à l'étude intersecte cette boîte englobante. Si c'est le cas, alors seulement on réalise la routine d'intersection rayon-triangles. On économise donc un nombre important de tests d'intersection.

Pour déterminer si une intersection avec la boîte englobante existe, il faut dans un premier temps calculer les coordonnées des points d'intersection avec chacune des paires de plans. Ces paires d'intersection définissent 3 intervalles : il y a intersection avec la boîte englobante si ces 3 intervalles ont une intersection non nulle.

On obtient alors les vitesses de calcul suivantes pour une image de 128 pixels par 128 :

- 2 rayons : 44 sec
- 5 rayons : 1 min 50
- 20 rayons : 6 min 10
- 50 rayons : 13 min 51

L'image ci-dessous correspond au résultat pour 50 rayons :



### 5.3 *Bounding Volume Hierarchies*

Pour améliorer encore plus les temps de calcul, on peut travailler récursivement avec les *Bounding Box*. En premier lieu, on décompose la grande boîte englobante autour de l'objet en deux sous-boîtes de mêmes dimensions. Il faut alors tester laquelle des deux sous-boîtes interseecte le rayon tiré.

Récursivement, chaque sous-boîte peut elle aussi être divisée en deux sous-boîtes et ainsi de suite ... Au final on obtient un arbre de décomposition de la grande boîte englobante de l'objet complet, dont les feuilles finales sont des boîtes ne contenant que quelques triangles. C'est une fois arrivé à ce niveau de granularité qu'on effectue les tests d'intersection rayon-triangles.

Les résultats sur les vitesses de calcul sont remarquables. On passe de plus de 13 min pour une image 128x128 et 50 rayons à moins de 10 secondes ! Il est donc maintenant possible d'obtenir des images de plus grande dimension avec plus de rayons tirés. L'image ci-dessous est en 512x512 avec 200 rayons tirés ; elle a été obtenue en seulement 5 minutes.

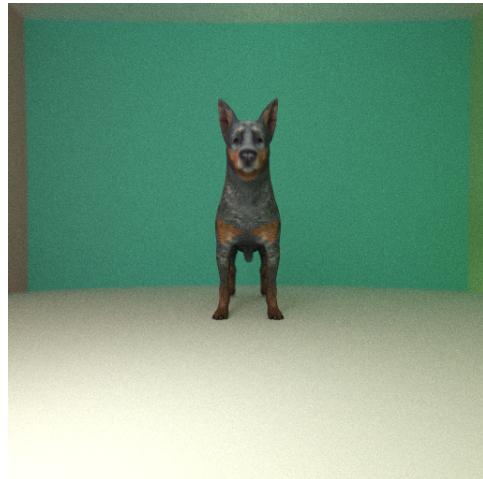


## 6 Les textures

On aimerait maintenant ajouter des textures aux objets. Pour cela, on dispose en ligne de textures bitmap : elles sont composées d'une image stockée comportant des pixels

colorés et une paramétrisation du maillage indiquant la correspondance entre chaque pixel de l'image colorée et les sommets du maillage.

L'exemple du berger australien disponible en ligne donne cette image pour 200 rayons et 512 pixels de côté, obtenue en 8 minutes 30 :



## 7 Déplacement de la caméra

Pour terminer ces travaux, on cherche à modéliser la possibilité d'un mouvement de l'objet ou de la caméra pendant la prise de l'image. Par exemple, on souhaite pouvoir observer la transition de l'objet d'un point A à un point B. Pour cela, on considère que le rayon tiré fait le mouvement inverse de l'objet. On génère alors des vecteurs de direction de rayon modifiés par des rotations/translations.

Les images ci-dessous illustrent deux prises de vue différentes du berger australien, l'une en contre-plongée et l'autre en plongée.

