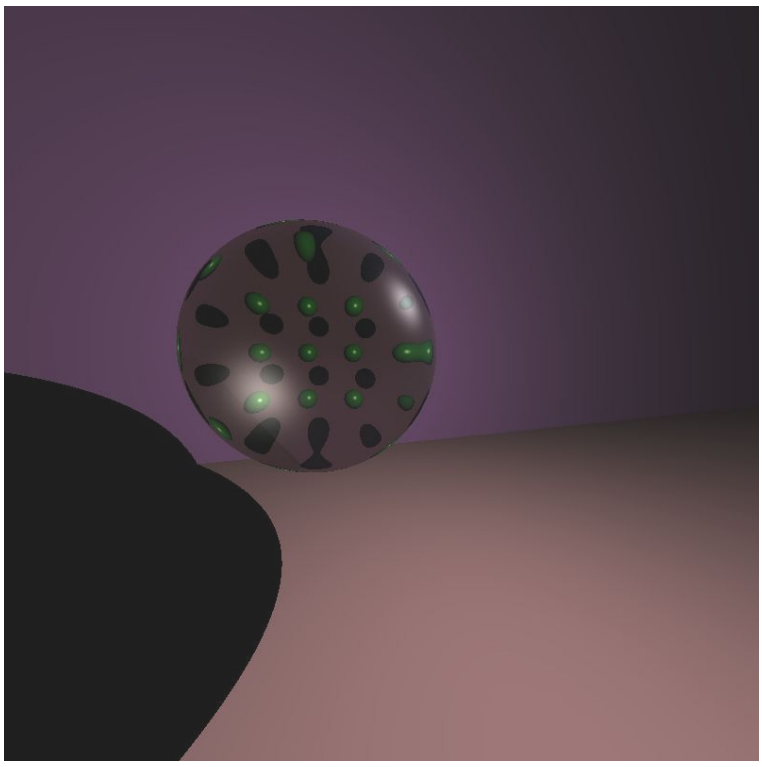
BOROUJERDI Sébastien
JACQUE Jeremy

UNIVERSITÉ DE REIMS
CHAMPAGNE-ARDENNE

Master 1ère année STIC spécialité informatique

Rapport de projet Synthèse d'images



Programmation d'un Raytracer simplifié en C++

Rapport du Lundi 27 février 2006

Contact : sebastien.boroujerdi@etudiant.univ-reims.fr
Jeremy.jacque@etudiant.univ-reims.fr

Enseignant : Mr Rémion

Sommaire :

I – Introduction

Le but de ce projet de synthèses d'images est de réaliser un Raytracer (simplifié). Avant de détailler chacune des parties proposées dans notre plan, nous souhaitons exposer ce que nous possédions (suite aux 2 TP en salle machine) :

- Un lancé de rayons sur une scène, avec gestion d'intersection entre les rayons et les objets de la scène. Ces objets se limitaient exclusivement à un modèle de sphère. Nous avons, suite aux examens, gérer les plans infinis, où toutes les données nécessaires étaient fournies. Nous ne détaillerons donc pas la réalisation du plan, qui est, comme vous pourrez le constater en étudiant notre code source (compilable sous Visual Studio.NET ou bien Visual C++), est relativement simple.
- Une entité Film, permettant la conversion d'un espace en deux dimensions vers un espace en trois dimensions, et vice versa.
- Toutes les autres entités fournies lors des TP.

Pour plus de clarté dans ce rapport, nous avons choisi de n'y ajouter aucun morceau de code en annexe. Nous avons commenté le plus efficacement possible nos fichiers sources.
Nous allons donc pouvoir entamer la première partie, c'est-à-dire la gestion de lampes ponctuelles isotropes.

II – Les lampes ponctuelles isotropes

II.1 – Théorie et implantation

Si l'on se place du point de vue des sources lumineuses uniquement, on constate que le seul besoin est de calculer la puissance lumineuse reçue en un point P de la scène éclairée par ces sources. Lors de son instanciation, une lampe est positionnée dans un repère global, et possède son propre repère local. De même, elle possède un spectre de puissance lumineuse, qui est gérée par la classe SpectrePuiss de notre Raytraycer. Comme la logique le suppose, la couleur reçue en point provenant d'une source lumineuse s'atténue en fonction de la distance entre ce point et la source lumineuse (car elle est ponctuelle). Toutefois, nous nous attarderons quelque peu sur cette atténuation, qui peut être calculée de la façon suivante :

$$\frac{1}{Kc + Kl * d + Kq * d*d}$$

Avec : Kc coefficient d'atténuation constante
Kl coefficient d'atténuation linéaire
Kq coefficient d'atténuation quadratique
d distance source lumineuse / point à éclairer

On remarque 3 facteurs d'atténuation que sont les facteurs d'atténuation linéaire, constante et quadratique. En fonction de ces trois facteurs, la couleur reçue en un point P peut varier fortement. C'est pourquoi nous avons choisi d'attacher à la classe LampeIso (pour lampe ponctuelle isotrope) un champ protégée « att », qui est un objet de type Attenuation, et qui permet, en fonction des trois coefficients précédents et de la distance entre la source lumineuse et le point P, de calculer la valeur du spectre lumineux reçue en P. La définition de la clase atténuation se trouve dans le fichier lampe.h.

II.2 – Exemples

Pour les exemples, veuillez s'il vous plaît vous référer aux exemples de la partie suivante.

III - Le modèle d'éclairement de Phong

III.1 – Théorie et mise en place de ce modèle d'éclairement

Grâce à ce modèle d'éclairement, la classe matière permettant de gérer les propriétés matérielles affectées à un objet de la scène va prendre tout son sens. En effet, que serait une pièce d'argent sans cette tâche spéculaire qui lui est propre ? Comment passer d'une matière plutôt lisse à une matière mate, où la réflexion spéculaire est faible ? Le modèle de Phong nous apporte la solution : selon le point de vue de l'observateur, de la lampe (dans notre cas ponctuelle) et les coefficients relatifs à la gestion de la spécularité d'après Phong. Pour détailler brièvement l'équation fournie dans notre support de cours, on peut modifier l'aspect de cette spécularité en jouant sur les paramètres de taille de la tâche se projetant sur l'objet, et sur la valeur de la fonction de transfert, notée Ks, qui permet de jouer sur un aspect lisse ou mate (métal, plastique) de la matière. Pour la série d'exemples suivants, nous allons faire varier ces différents coefficients, mais également les coefficients d'atténuation de la lampe isotrope. Pour le code source, vous pouvez consulter le fichier scene.cpp, méthode traceRayon de la classe Scene.

III.2 – Exemples

Voici quelques exemples, que vous pourrez tester par vous-même avec notre application, en vous référant au fichier AIDE.txt. Nous faisons varier ici la taille de la tâche spéculaire, ainsi que la fonction de transfert de réflexion spéculaire (lampe blanche, sphere rouge):

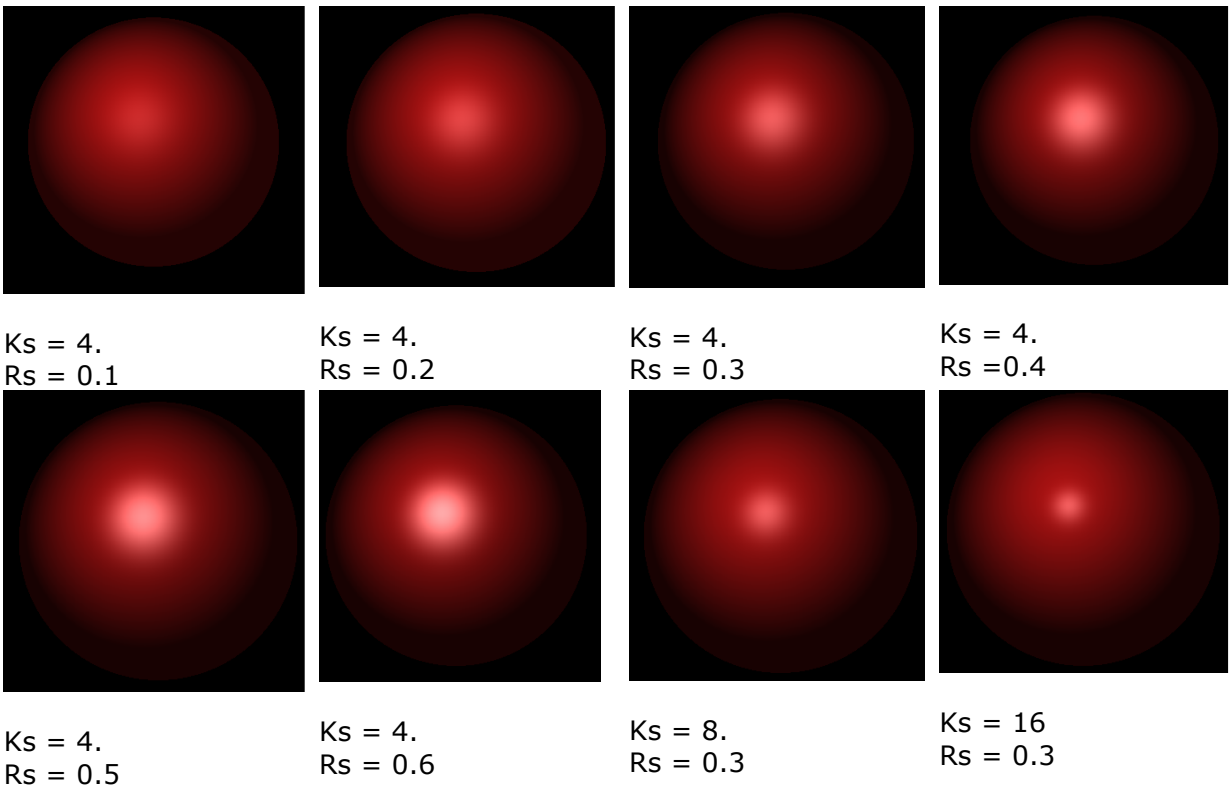


Figure 1: Modèle d'éclairement de Phong avec quelques coefficients différents

Voici d'autres exemples intégrant plusieurs lampes ponctuelles (sphère grise, lumière colorée):

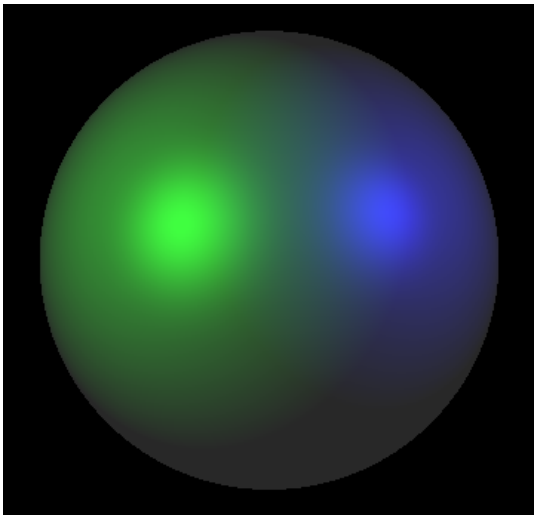


Figure 2: 2 lampes (bleue et verte)
+ ambiante (Ka = 0.2)
Ks = 4.
Rs = 0.4

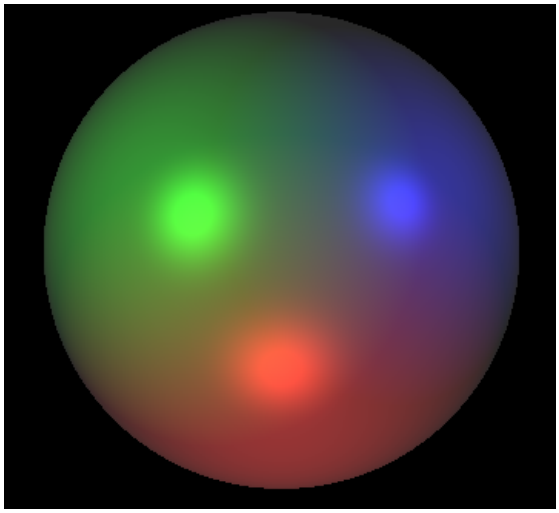


Figure 3: 3 lampes (rouge / bleue / verte)
+ ambiante (Ka = 0.3)
Ks = 8.
Rs = 0.5

IV – Le modèle d'éclairement de Whitted

IV.1 – Théorie et implantation de ce modèle

Ce modèle d'éclairement permet d'ajouter à notre scène la gestion de la réflexion et de la réfraction. Il s'agit donc d'un modèle de Phong amélioré, où les rayons réfléchis et réfractés sont gérés de façon récursive dans notre programme, jusqu'à une profondeur maximale de récursivité (nous avons choisis 6 par défaut).
Pour gérer les rayons réfléchis, nous nous basons sur le schéma suivant :

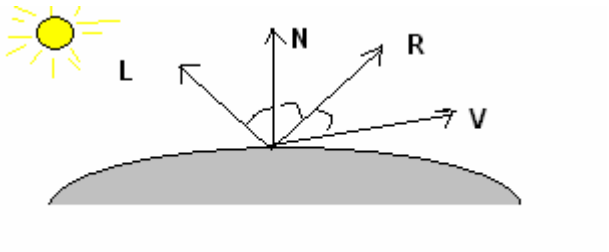


Figure 4: N : normale à la surface, L vecteur de la source lumineuse au point, R rayon réfléchi, V vecteur Observateur / Point.

On recherche donc le rayon réfléchi représenté par le vecteur R, qui peut être calculé de la façon suivante :

$$\vec{R} = \vec{V} - 2 * (\vec{V} \cdot \vec{N}) * \vec{N}$$

On renverra par la suite, lors des appels récursifs, un rayon ayant pour origine le point d'intersection et pour direction celle obtenue par le calcul précédent. Ainsi deux objets réfléchissant se réfléchiront l'un dans l'autre, comme vous pourrez le constater dans les exemples suivants.

Pour la réfraction, le phénomène est un peu plus compliqué à gérer. En effet, la réflexion se passe en dehors de l'objet, et les calculs d'intersection sont instinctifs. Pour la réfraction, nous devons savoir si l'intersection rayon / objet à lieu à l'extérieur de la sphère ou bien à l'intérieur. Nous avons choisi de vous présenter un petit schéma décrivant le comportement des rayons, par exemple dans une forme de sphère :

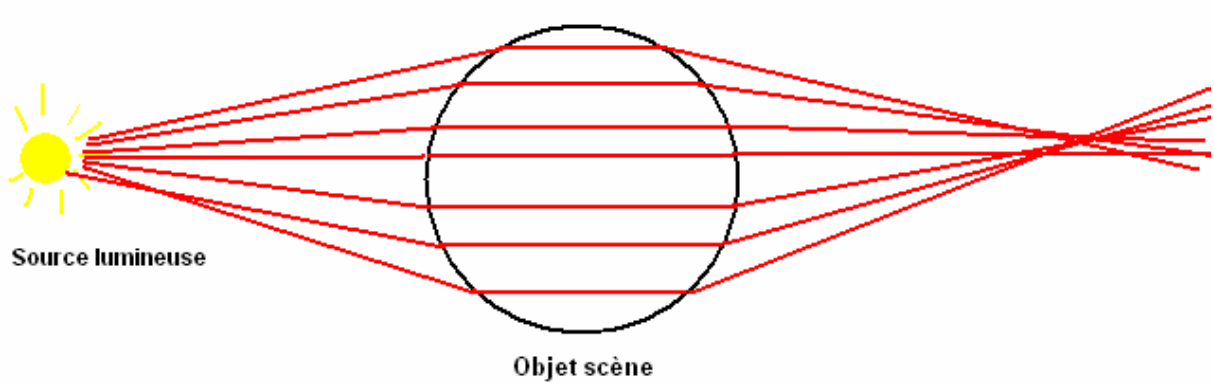


Figure 5: Phénomène de réfraction

Encore une fois, le rayon réfracté dépend de coefficients comme l'indice de réfraction des milieux de la scène (eau, air, etc …). Voici quelques exemples qui reflètent ces quelques explications :

IV.2 – Exemples

1. Phénomène de réflexion

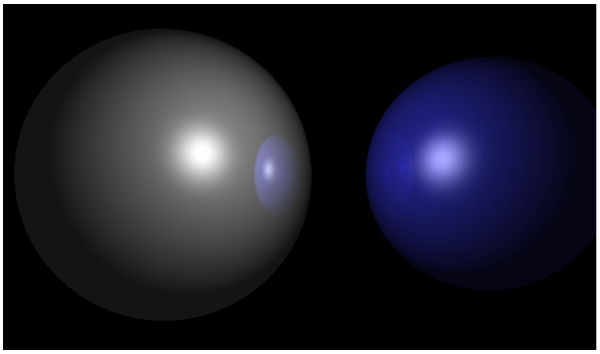


Figure 6: Réflexion récursive sur 2 sphères éclairées par une lampe blanche. On perçoit même la tâche spéculaire de la sphère bleue sur la grise.

Sphère bleue : Kr = 0.3
Sphère grise : Kr = 0.7

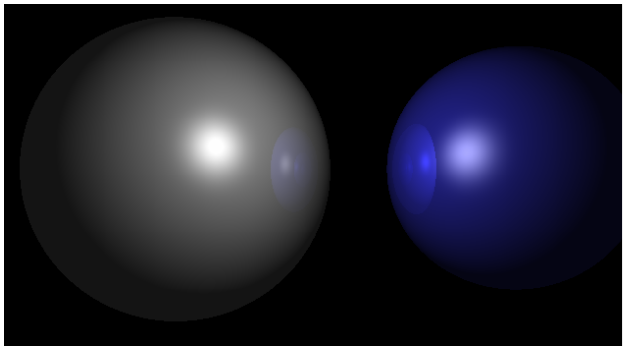


Figure 7: idem figure de gauche mais inversion des coefficients de réflexion.

Remarques : Afin de gérer correctement le modèle de Whitted, les phénomènes de réflexion et de réfraction, notre méthode Rendu de la classe scène renvoie le premier objet intercepté par le rayon lancé. De ce fait, nous arrêtons le parcours de la liste d'objets de la scène dès qu'une intersection se produit (ce qui n'empêchera pas par la suite de renvoyer d'autres rayons). Cet arrêt prématuré permet de gérer implicitement un tampon de profondeur ou ZBuffer, sans avoir à stocker en mémoire une zone

supplémentaire réservée uniquement au ZBuffer. Vous pouvez le constatez sur l'image suivante :

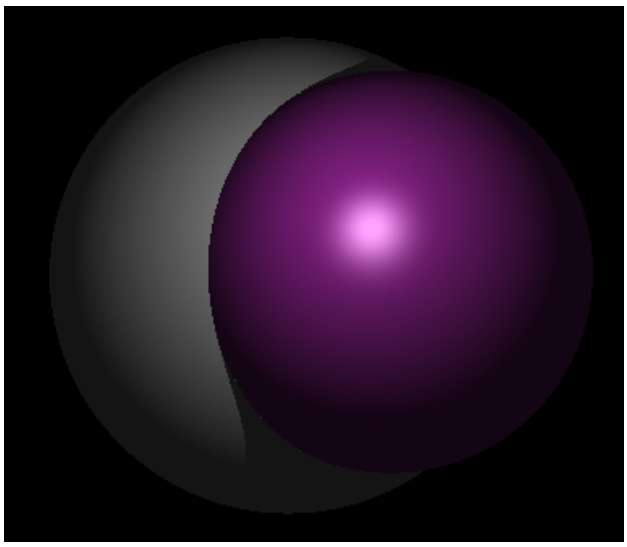


Figure 8: La petite sphère est devant la grosse (on voit l'ombre portée de la violette sur la grise)

2. Phénomène de réfraction

Nous considérerons dans nos exemples l'indice de réfraction du verre pour une sphère transparente. Le rayon réfracté est également renvoyé de façon récursive. La figure suivante présente une scène composée de 2 plans, de plusieurs petites sphères qui vont servir de « témoins » pour la réfraction. Voici notre résultat (notre préféré) :

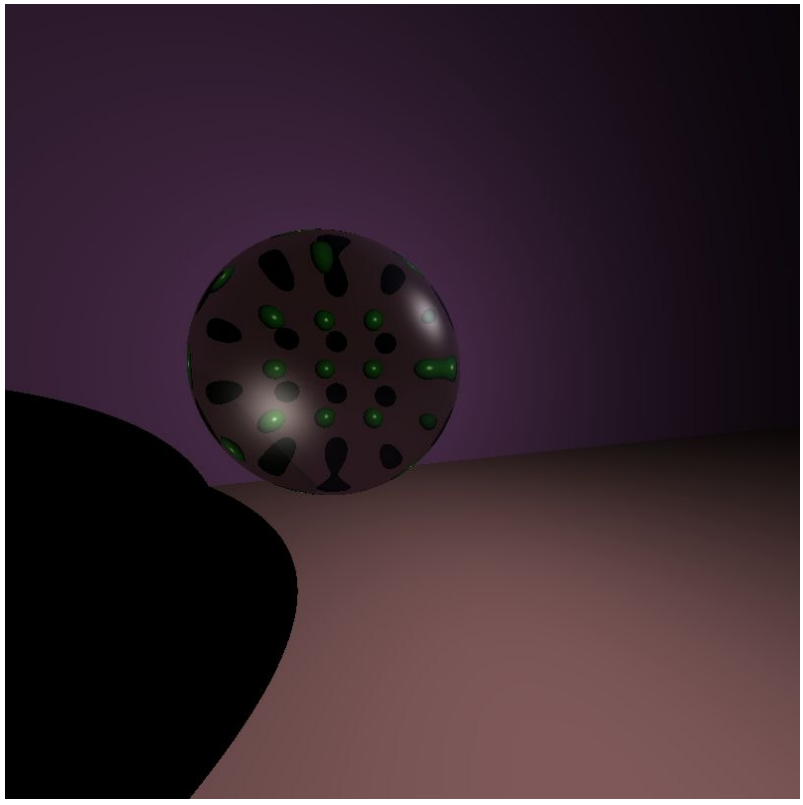


Figure 9: On constate que le phénomène de réfraction déforme les petites sphères que l'on perçoit à travers (vertes). L'indice de réfraction pour l'intérieur de la sphère est celui du verre, soit 1.5.

V – Caméra trou d'aiguille ou Pinhole

V.1 – Théorie et mise en place

La majorité des calculs sont détaillés dans le support de cours, nous ne détaillerons donc pas cette partie. Toutefois, il faut veiller, lors de l'implantation de la caméra, à gérer le pas d'échantillonnage lors du parcours de l'image destination, afin de ne pas obtenir de déformation. Cette caméra nous permet d'obtenir une vue en « perspective », et de pouvoir placer l'observateur en divers endroit de la scène.

V.2 – Exemples

Voici les résultats que nous obtenons après implantation de la caméra trou d'aiguille (comparaison avec la scène d'origine) :

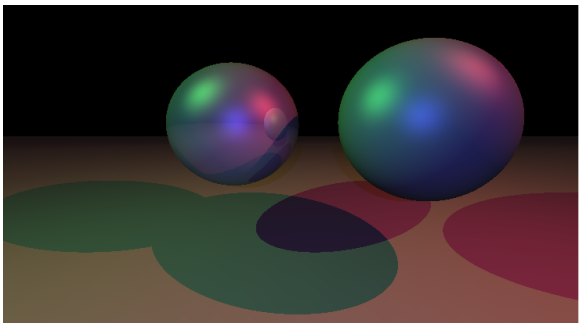


Figure 10: Sans caméra pinhole

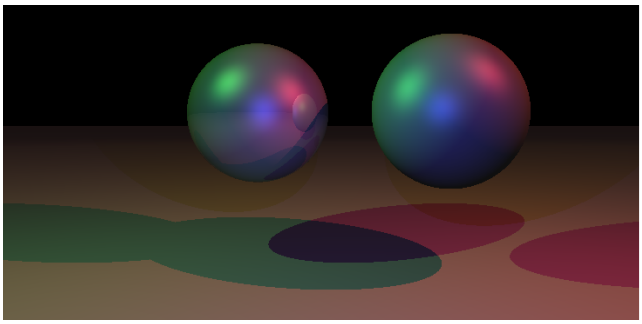


Figure 11: Avec caméra pinhole

VI – Les quantificateurs

Les quantificateurs permettent un traitement post opératoire du film obtenu. Une fois le rayon lancé, réfléchi, etc … nous récupérons la valeur du spectre de puissance au point d'intersection rayon / scène. Ce spectre est transmis à l'opérateur () d'un objet Quantificateur, qui va ensuite, selon son type d'instanciation (template et abstrait), nous permettre d'obtenir divers effets visuels comme le noir et blanc (luminance), ou encore l'effet sépia bien connu pour représenter des scènes anciennes. Nous allons tout d'abord étudier le quantificateur noir et blanc, qui nous permettra d'obtenir facilement celui de l'effet sépia.

VI.1 - Quantificateur Noir et Blanc

Pour ce quantificateur, on effectue simplement une moyenne des composantes du spectre de puissance (luminance). On affecte à chaque composante RGB du résultat la valeur de cette moyenne (multipliée par 255, pour être dans l'intervalle [0, 255]). Voici ce que nous obtenons après implantation de ce quantificateur :

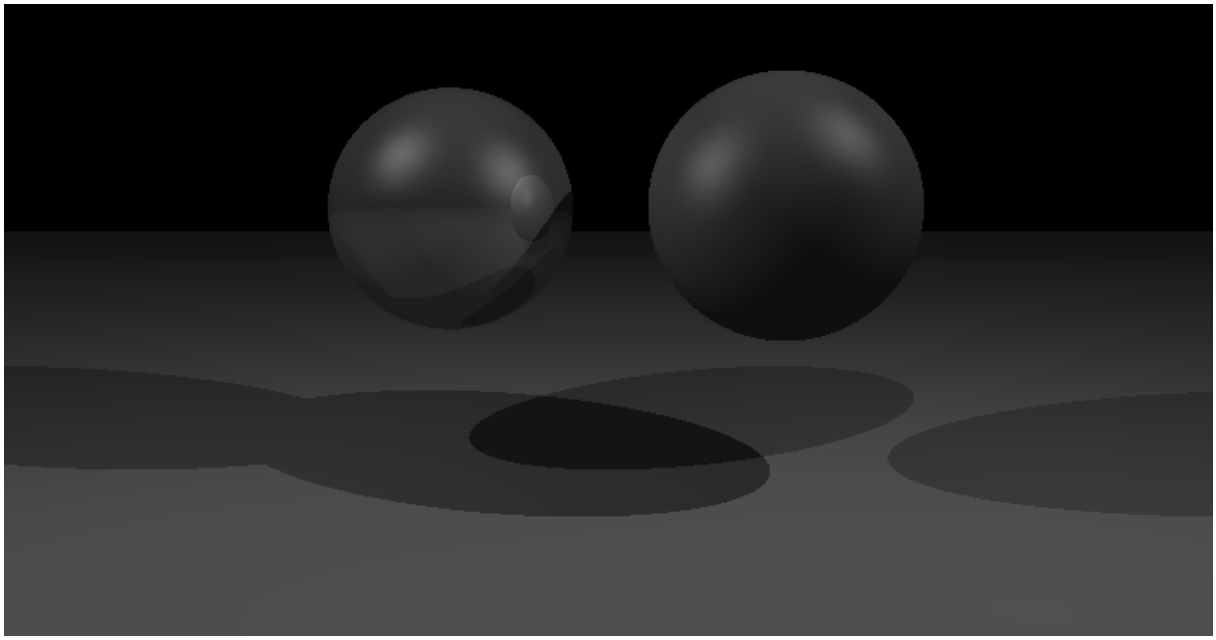


Figure 12: Application du quantificateur noir et blanc (cf scène précédente)

IV.2 – Quantificateur « Sépia »

D'après les informations que nous avons pu obtenir sur ce quantificateur, il s'agit d'une fonction de luminance sur le spectre de puissance, avec des pourcentages pour chaque composante du spectre de puissance. Nous avons retenus les coefficients suivants :

100% pour le rouge (par rapport à la luminance du spectre)
89.2% pour le vert
69.4% pour le bleu.

Nous obtenons les résultats suivants :

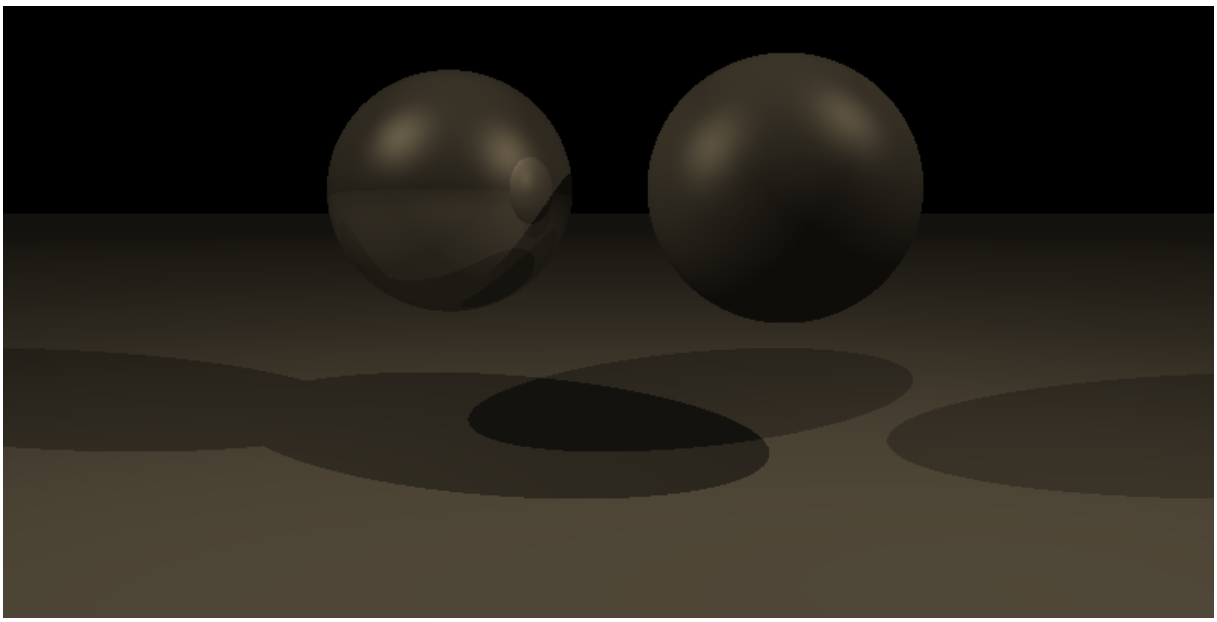


Figure 13: Application du quantificateur sépia (cf scène précédente)

VII – Les ombres portées

VII.1 - Implantation

Le principe de l'ombre portée est assez simple ; en effet, il nous suffit, lors de l'interaction d'un rayon avec un objet, de savoir si un objet est déjà devant. Si oui, il est dans l'ombre (ou la partie de l'objet concerné), sinon il est pleinement visible. Il nous suffit donc, dans notre méthode traceRayon de la classe Scene de savoir si il y a un objet entre un autre objet et le rayon. Voici ce que nous obtenons :

VII.1 – Exemples

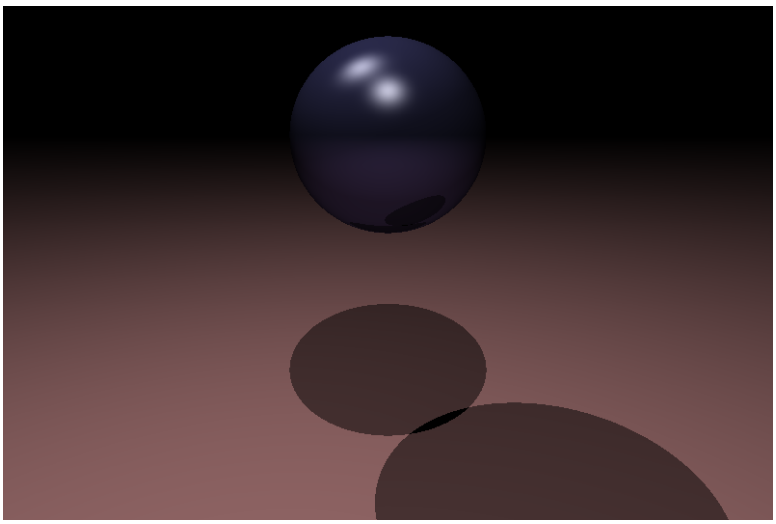


Figure 14: Ombres portées (avec réflexion sur la sphère)

VIII – Les textures

VIII.1 – Implantation

A elle seule, l'implantation de textures occuperait les 5 pages de ce rapport. Nous décrirons donc brièvement cette implantation, qui concerne uniquement les textures dites « textures 2D ».
Avant de commencer, nous devons préciser que le repère associé aux coordonnées de textures est un repère en deux dimensions, obtenu par rapport aux dimensions de l'image et de l'objet où la texture doit être appliquée. Nous allons donc, d'après un point de R^3 (objet pR3), calculer les coordonnées correspondantes dans l'image de texture. Nous irons ensuite chercher la valeur du pixel sur l'image (texel), pour la multiplier par la valeur du spectre de puissance obtenu par le modèle d'éclairement et la matière de l'objet. Comme pour le calcul de normale, l'affectation d'une coordonnée de texture dépend de l'objet à « recouvrir » de la texture. Nous détaillerons ici la sphère, mais vous pourrez consulter le code relatif à chacun des types de forme dans form.cpp (*cf* également la méthode PropOpt de la classe Objet).
Soit u et v les coordonnées de texture ; nous allons devoir utilisé un système de coordonnées polaires, où r représente le rayon et ζ la coordonnée angulaire.
On obtient donc :

r = arcos(-Vn . Vp) où . représente le produit scalaire, Vn un vecteur dirigée vers le pôle nord de la sphère, et Vp le vecteur ayant pour origine le centre de la sphère et dirigée vers le point d'intersection. Par la suite, nous pouvons donc calculer v telle que :

v = r / Pi

On obtient ensuite :

ζ = (arcos(Vn . Vp) / sinus(r)) / (2*Pi)

avec finalement:

u = ζ (ou bien 1. – ζ)

Les exemples suivants seront certainement plus parlants :

VIII.2 – Exemples

1. Application de texture de marbre su une sphère, et de bois sur le plan

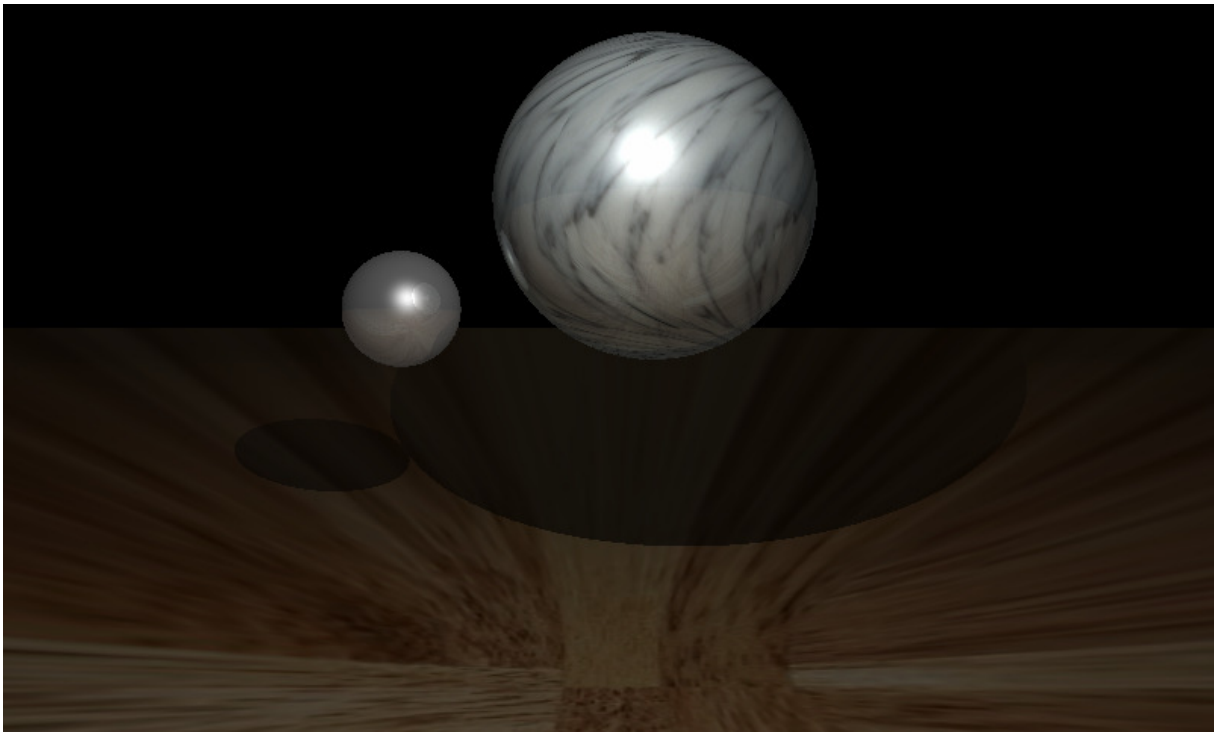


Figure 15: Sphère texturée de marbre, plan texturé de bois

On change maintenant la matière de la sphère, qui était blanche :

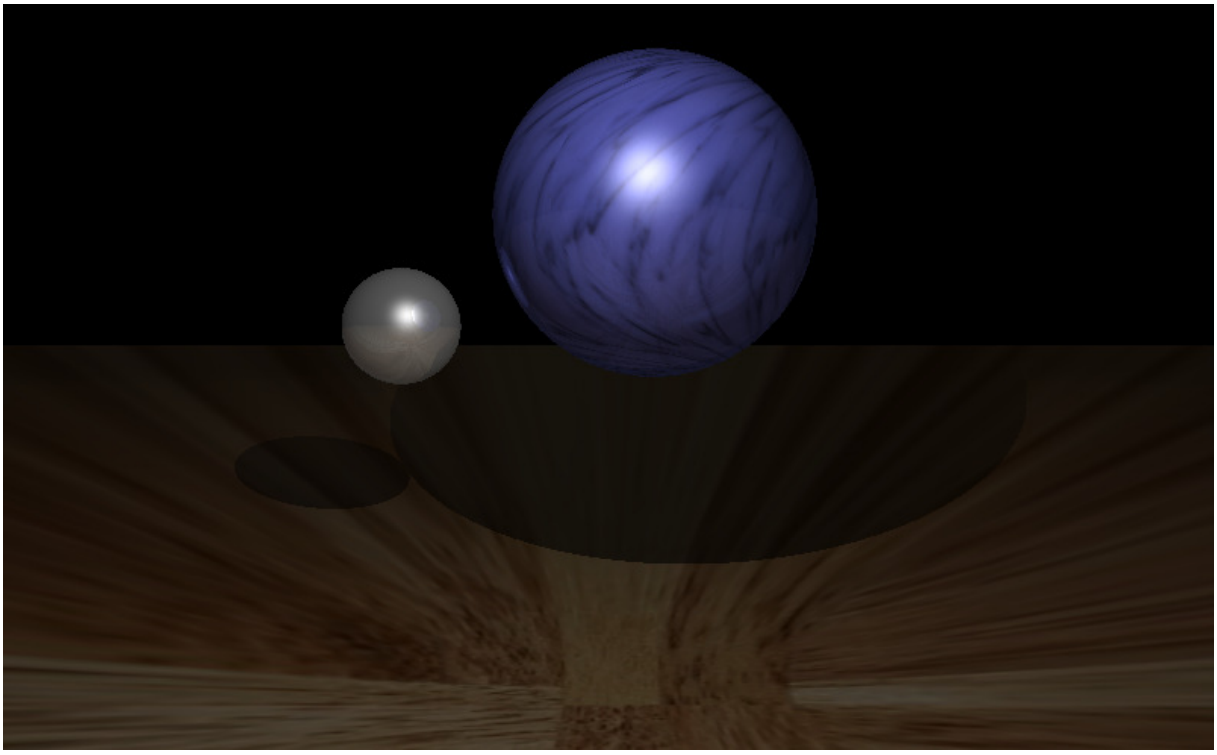
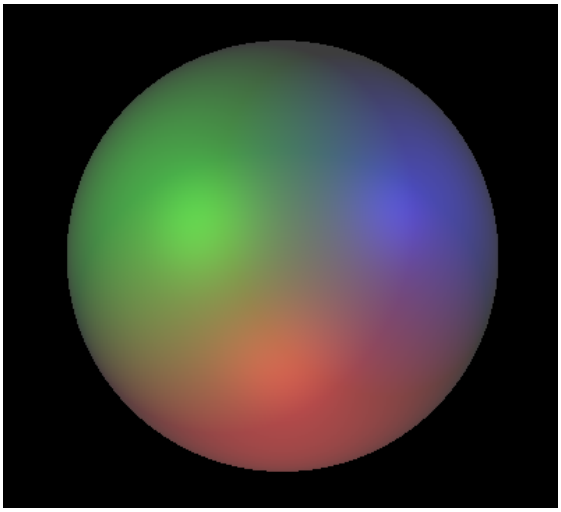
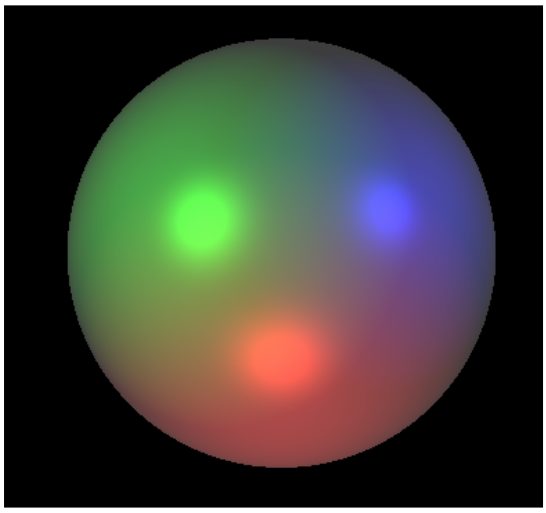


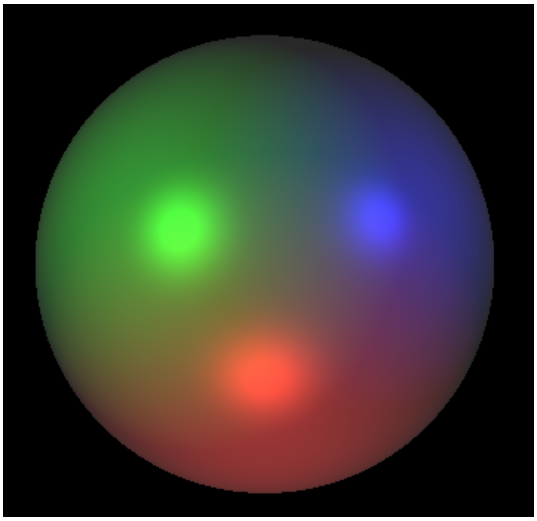
Figure 16: Application de texture et prise en compte de la matière

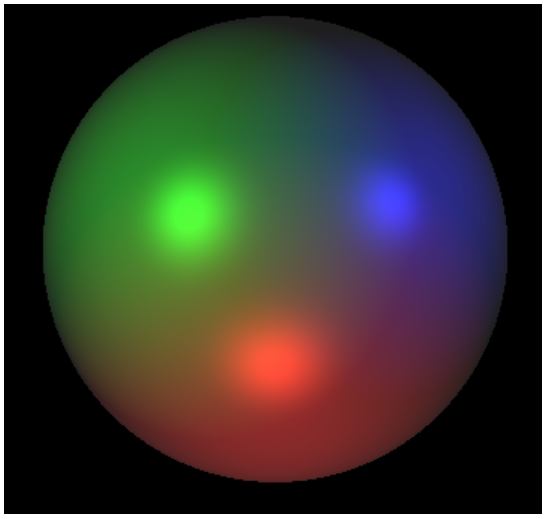
IX – D'autres exemples et résultats

Et voici une série d'images reflétant les points précédemment traités.
Ici, on joue avec les propriétés matérielles des objets (ambiant et spéculaire) :

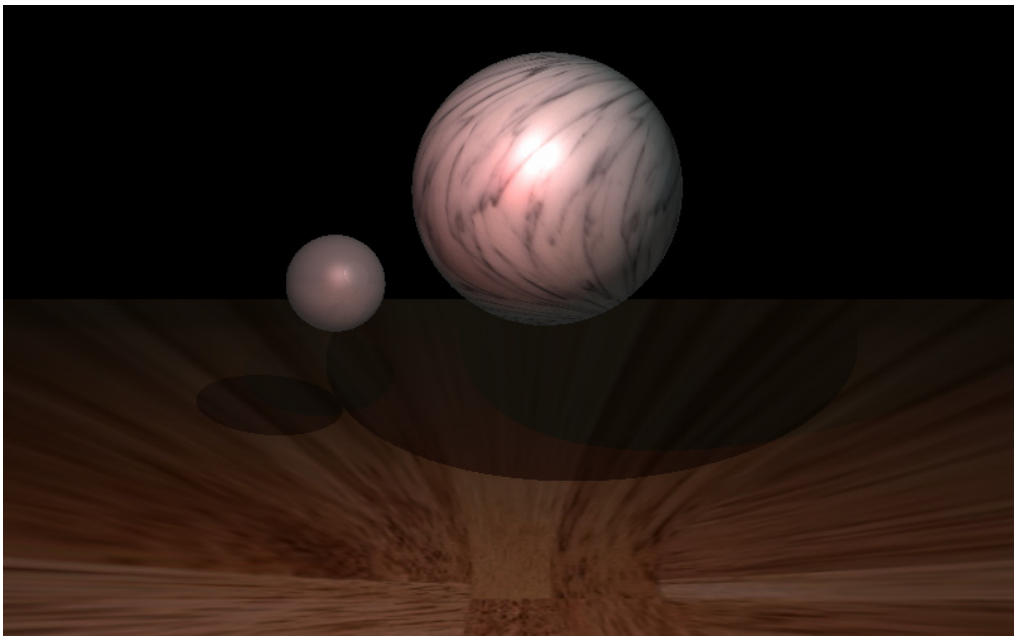




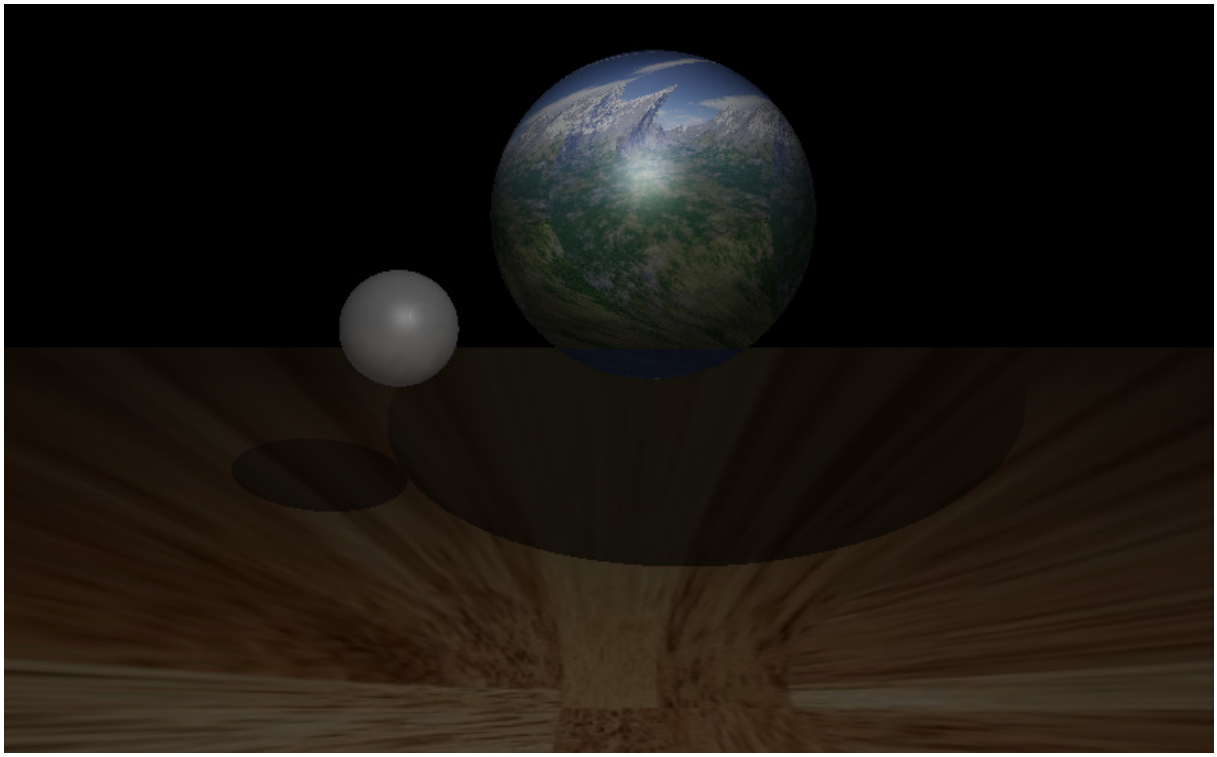




On peut également appliquée différentes propriétés aux objets texturés, avec prise en compte des lampes :



Et une jolie texture pour finir :



X – Bibliographie

1) http://www.linuxgraphic.org/section3d/articles/raytracing/reflexion.html
2) http://www.limsi.fr/Individu/jacquemi/IG-TR-7-8-9/menu-eclairt.html
3) http://lab.erasme.org/3d/light.html