Projet 2 : Lancer de rayons

Matthieu Gillet, Maé Mavromatis, Mathieu Poirier, Quentin Zanini
ENS Rennes – ISTIC Université Rennes 1
L3 Informatique – parcours SIF

7 décembre 2018

Résumé

Ce rapport expose le travail fourni pour le projet 2 du module de programmation. Il est construit en deux parties : la première traite de la méthode de lancer de rayons classique, la deuxième précise les ajouts que nous y avons fait.

Table des matières

1 Introduction

Le lancer de rayons est une technique de synthèse d'image qui se base sur le suivi de rayons lumineux. L'idée est d'exploiter le principe du retour inverse de la lumière, qui implique que toute lumière reçue sur un écran peut être remontée le long de ses rayons jusqu'à sa source.

L'appui sur ce principe permet de s'affranchir d'un calcul exhaustif des interactions lumineuses de la scène. Remarquons qu'un grand nombre de rayons lumineux n'atteindraient pas la caméra en suivant cette approche, et auraient donc été calculés inutilement.

La méthode du lancer de rayons simule la façon dont la lumière interagit avec les objets du monde virtuel. Elle permet notamment de rendre compte d'un grand nombre de phénomènes optiques tels que la réflexion, la réfraction ou la diffusion et ainsi d'obtenir des scènes réalistes.

2 Lancer de rayons basique

Pour implémenter la méthode basique du lancer de rayons, il nous faut lancer un rayon depuis la position de chaque pixel de l'écran vers la scène. Cela nous permet de savoir quel objet ce rayon intersecte en premier, pour enfin calculer la couleur de cet objet en ce point suivant son éclairage.

2.1 Algorithme

Algorithme 1 : Lancer de rayons

Entrées :
Une scène contenant des objets
Un écran
Corps :
pour tout Pixel de l'écran **faire**
pour tout Objet de la scène **faire**
Calcul des intersections
si Intersection **alors**
Mémoriser la distance à l'origine du rayon
fin si
fin pour
si Aucune intersection **alors**
Pas de lumière
sinon
Illuminer l'intersection la plus proche
fin si
si Surface réfléchissante **alors**
Calcul rayon réfléchi avec les lois de Descartes
fin si
fin pour

2.2 Calcul des intersections

Les intersections entre un rayon et un objet de la scène sont déterminées par la résolution d'équations paramétriques : l'intersection d'une droite avec une boule qui sera explicitée en 3.1.1 ou d'une droite avec un plan que nous verrons en 3.1.2. S'il y a plusieurs intersections entre un rayon et les objets de la scène, nous ne gardons que la plus proche du point d'origine du rayon.[1]

3 Fonctionnalités implémentées

Afin d'obtenir des rendus variés, nous avons implémenté différents objets. Ils sont dotés de plusieurs paramètres tels leur couleur ou leur matériau. Nous avons aussi implémenté certains phénomènes lumineux et enfin permis la création d'animations par déplacement de la caméra.

3.1 Différents objets

3.1.1 Sphères

Une sphère est définie par la position de son centre et par son rayon.

Nous avons donc la possibilité de placer des sphères de différentes tailles dans l'espace, comme le montre la figure 1.

Déterminer s'il y a une intersection $M = (x, y, z)$ entre une sphère de centre $\Omega = (a, b, c)$ et de rayon R, d'équation $(x-a)^2 + (y-b)^2 + (z-c)^2 = R^2$ et un rayon lumineux d'origine $O = (o_x, o_y, o_z)$ et de vecteur directeur $\overrightarrow{v} = (v_x, v_y, v_z)$, c'est résoudre le système d'équation :

$$\begin{cases} x = o_x + tv_x \\ y = o_y + tv_y \\ z = o_z + tv_z \\ (x-a)^2 + (y-b)^2 + (z-c)^2 = R^2 \end{cases}$$

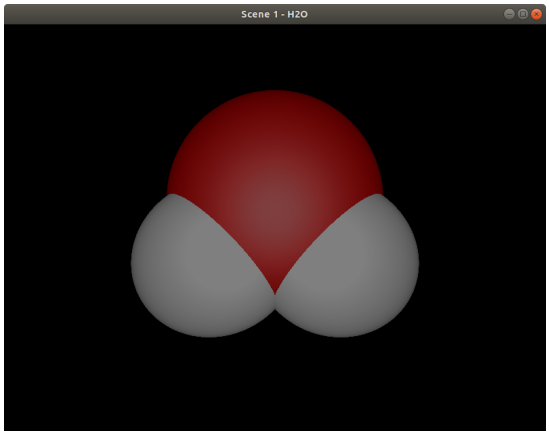
$$\text{i.e. } ((o_x + tv_x) - a)^2 + ((o_y + tv_y) - b)^2 + ((o_z + tv_z) - c)^2 = R^2$$

$$\text{donc en posant } \begin{cases} A = v_x^2 + v_y^2 + v_z^2 \\ B = 2(v_x(o_x - a) + v_y(o_y - b) + v_z(o_z - c)) \\ C = (o_x - a)^2 + (o_y - b)^2 + (o_z - c)^2 - R^2 \end{cases}$$

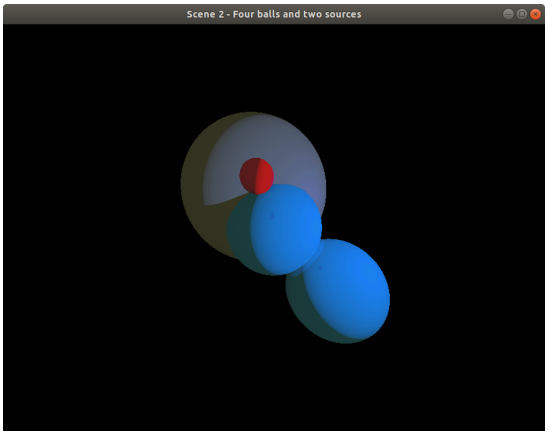
$$\text{on cherche } t > 0 \text{ tel que } At^2 + Bt + C = 0$$

3.1.2 Plans

Nous avons souhaité ajouter des plans à nos scènes. Les plans sont définis par une position et deux vecteurs.



(a) Trois sphères intersectées



(b) Quatre sphères

FIGURE 1 – Différentes scènes représentant des sphères

Nous avons rencontré des problèmes au niveau de la représentation des plans dans un premier temps. En effet, les calculs sur les flottants entraînent des imprécisions. En particulier, un point appartient au plan s'il est légèrement au dessus ou légèrement en dessous de ce dernier.

En prenant en compte ce paramètre, nous avons obtenu des résultats concluants, comme sur la figure 2 par exemple.

Pour déterminer s'il y a une intersection entre le rayon lumineux d'origine $O = (o_x, o_y, o_z)$ et de vecteur directeur $\overrightarrow{d} = (d_x, d_y, d_z)$ et la section de plan en position $P = (p_x, p_y, p_z)$ et définie par les vecteurs $\overrightarrow{u} = (u_x, u_y, u_z)$ et $\overrightarrow{v} = (v_x, v_y, v_z)$, nous avons commencé par calculer (si elle existe) l'intersection entre notre rayon et le plan infini induit par la section, puis nous avons vérifié l'appartenance de ce point (s'il existe) à la section définie ci-dessus.

Il s'agit donc dans un premier temps de calculer l'intersection $M = (x, y, z)$ entre le rayon et le plan infini.
Posons $n = \overrightarrow{u} \wedge \overrightarrow{v} = (a, b, c)$ une normale au plan. Si une telle intersection existe, alors $\overrightarrow{d} . \overrightarrow{v} = ad_x + bd_y + cd_z \neq 0$. L'équation du plan infini est $ax + by + cz + d = 0$ où $d = -n_x p_x - n_y p_y - n_z p_z$, et nous résolvons le système :

$$\begin{cases} x = o_x + tv_x \\ y = o_y + tv_y \\ z = o_z + tv_z \\ ax + by + cz + d = 0 \end{cases}$$

$$\text{i.e. } t = -\frac{ao_x + bo_y + co_z + d}{ad_x + bd_y + cd_z}$$

Ensuite, il reste à tester l'appartenance de $M = (x, y, z)$ à la section de plan définie plus haut. Nous résolvons donc :

$$\begin{cases} \| \overrightarrow{u} \|^2 x + \overrightarrow{u} . \overrightarrow{v} y = \overrightarrow{PM} . \overrightarrow{u} \\ \overrightarrow{u} . \overrightarrow{v} x + \| \overrightarrow{v} \|^2 y = \overrightarrow{PM} . \overrightarrow{v} \end{cases}$$

On a enfin que $(x, y) \in [0, 1]^2$ si et seulement si M est dans le parallélogramme.

FIGURE 2 – Une sphère intersectée avec un plan

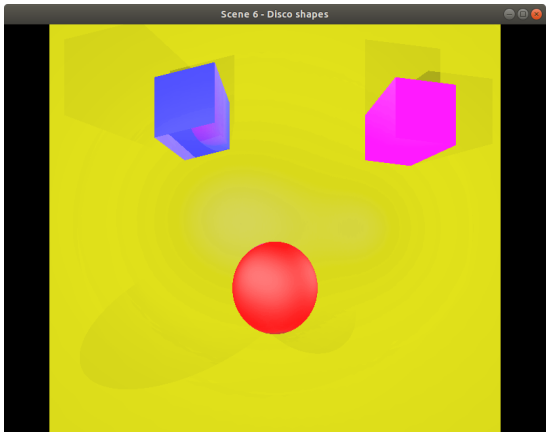


FIGURE 3 – Différentes formes, incluant des parallélépipèdes

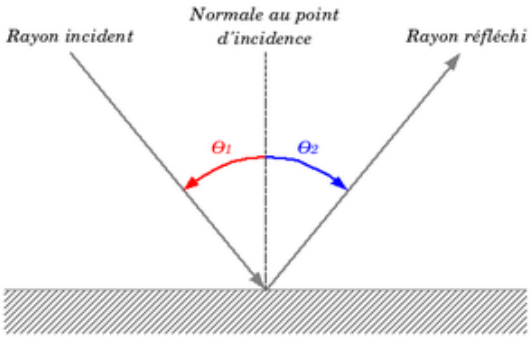


FIGURE 4 – La loi de Snell-Descartes[2]

3.1.3 Parallélépipèdes

Nous avons ensuite souhaité implémenter les parallélépipèdes. Ils sont définis par une position et trois vecteurs.

Pour gérer les intersections avec les rayons de lumière, nous en considérons les six faces, et nous utilisons les intersections définies pour les plans.

La figure 3 présente une scène avec des parallélépipèdes.

3.2 Réflexions

Les réflexions sont obtenues avec les lois de Snell-Descartes. La figure 4 en rappelle le principe physique.

En pratique, il nous suffit de relancer le calcul de la couleur obtenue par un rayon lumineux en prenant comme nouveau rayon celui partant du point d'intersection et dirigé par le vecteur réfléchi donné par la loi physique citée ci-dessus.

Pour assurer la terminaison de l'algorithme et limiter la complexité en temps de calcul, nous avons choisi de permettre aux rayons de se réfléchir au plus trois fois. Cette borne fut fixée après quelques essais, qui ont révélé que les reflets obtenus par plus de trois réflexions n'étaient plus perceptibles. La troisième réflexion est elle-même tout juste observable sur la figure 6.

La prise en compte de la couleur obtenue finalement est détaillée en partie 3.4.2.

3.3 Différentes surfaces

En plus d'une couleur, chaque objet de la scène dispose d'un matériau. Ce matériau détermine à quel point l'objet est mat ou réfléchissant. Un coefficient entre 0 et 1 donne l'importance de la réflexion. Le complément à 1 de cette valeur donne le coefficient d'absorption, correspondant en fait à la matification de l'objet.

3.3.1 Miroirs parfaits

Les miroirs parfaits ont un coefficient de réflexion égal à 1. Un miroir qui n'est pas éclairé est donc invisible. En pratique, nous avons finalement choisi 0.9 et non 1 comme coefficient de réflexion, afin de voir où les miroirs sont grâce à la lumière ambiante, et donc mieux se représenter la scène.

3.3.2 Réflecteurs

Les réflecteurs sont les surfaces que nous avons le plus souvent utilisées. Elles réfléchissent la moitié du rayon qu'elles reçoivent, et en absorbent l'autre moitié. Ainsi, on peut distinguer leur couleur tout en appréciant leurs reflets. Nous pourrons en effet remarquer le reflet du parallélépipède magenta de la figure 3 dans le parallélépipède bleu à sa gauche.

3.3.3 Absorbants

Une surface absorbante ne réfléchit pas du tout la lumière. Ainsi, aucun reflet n'est visible à sa surface. La sphère supérieure bleue en figure 5 est donc plus lumineuse que l'autre, car aucun rayon ne se perd en étant réfléchi.

3.4 Synthèse des couleurs

De nombreux phénomènes étant pris en compte, la manipulation des couleurs a composé une part importante de notre travail pour des rendus réalistes.

L'addition de deux couleurs a été implémentée comme la racine de la somme des carrés, composante par composante. Ce choix tient du fait qu'une caméra renvoie une valeur proportionnelle à la racine de l'intensité lumineuse reçue.

Pour la soustraction de couleur, nous souhaitions minimiser les cas où la couleur de retour devient noire. Ainsi chaque composante de la couleur à réduire est réduite proportionnellement à la grandeur de la seconde.

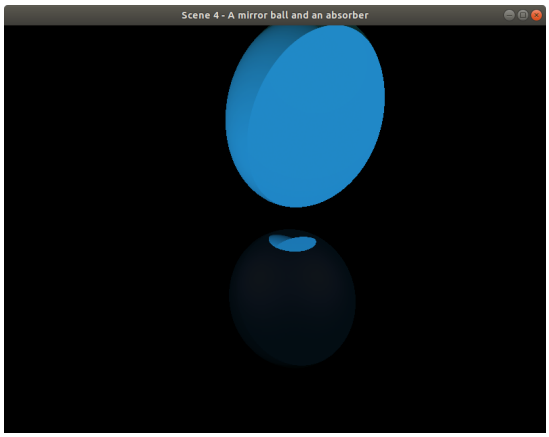


FIGURE 5 – La sphère supérieure est absorbante, l'autre est un miroir

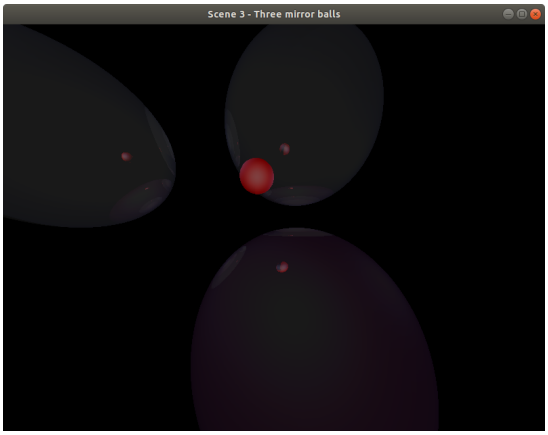


FIGURE 6 – Une petite sphère rouge reflétée dans trois sphères miroir

3.4.1 Sources et objets

Nous avons implémenté la possibilité pour les sources et pour les objets d'être colorés.

Pour synthétiser la couleur propre d'un objet, c'est-à-dire la couleur obtenue considérant la couleur de l'objet étant données les sources l'éclairant, nous avons procédé à la somme de toutes les couleurs l'éclairant. Cette somme est amenuie par la proportion des sources éclairant le point.

Pour éviter que la couleur soit unie sur une surface éclairée par les mêmes sources, nous avons réduit la lumière reçue par la racine du cosinus entre la normale en le point d'intersection et le vecteur vers la source.

Ainsi, la sphère blanche représentée en figure 7 montre un léger dégradé de couleurs là où les sources se rencontrent. Au centre la somme des couleurs donne un blanc grisâtre.

De même pour garantir qu'un objet est toujours visible, nous avons implémenté des lumières ambiantes, éclairant les objets quelle que soit leur position. Ainsi l'ombre à gauche des objets de la figure 1b n'est pas totale.

3.4.2 Prise en compte de la couleur réfléchie

Pour prendre en compte la lumière réfléchie, nous considérons que celle-ci n'est pas altérée au contact de la surface réfléchissante. La couleur réfléchie et la couleur propre de l'objet sont donc sommées, avec pour coefficients les constantes de réflexion et d'absorption du matériau de l'objet.

3.4.3 Reflet direct d'une source

En fonction des positions des sources, nous pouvons observer des zones où la couleur de l'objet s'efface devant celle de la source de lumière. Pour implémenter ce phénomène, nous avons comparé la direction venant de la source par rapport à celle du rayon incident, considérant la normale.

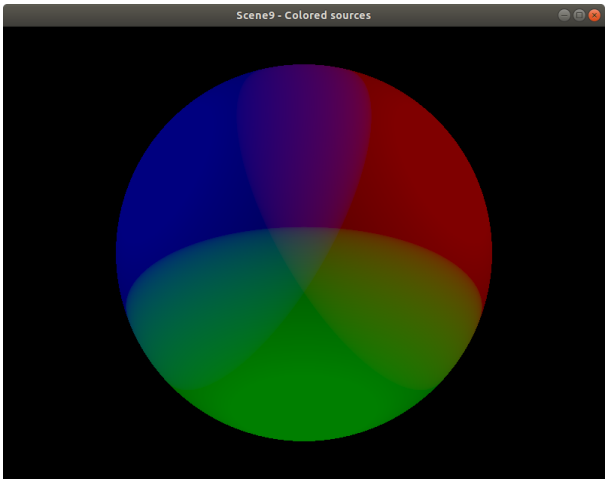


FIGURE 7 – Une sphère blanche éclairée par des sources colorées en trois endroits

Pour cela, en notant $\overrightarrow{i}$ le vecteur directeur du rayon incident et $\overrightarrow{s}$ le vecteur en direction de la source lumineuse, nous considérons leur somme $\overrightarrow{v} = \overrightarrow{i} + \overrightarrow{s}$.

En notant $\overrightarrow{n}$ la normale au point d'impact, l'angle $\theta = (\overrightarrow{v}, \overrightarrow{n})$ nous donne par le carré de son cosinus la proportion de la lumière de la source qui sera considérée pour ce halo lumineux[3].

Cette information est prise en compte de le calcul de la couleur propre de l'objet. La figure 1a présente un exemple de ce phénomène par le halo blanc au milieu de la sphère rouge.

3.5 Animation

La position de la caméra étant donnée par deux points, nous avons naturellement eu l'idée de la déplacer, puisqu'un vecteur permet d'effectuer une translation. Pour créer une animation, il nous est apparu nécessaire de précalculer chaque image avant de les rendre successivement à l'écran étant donné le temps de calcul de chaque image.

Ainsi, une classe dédiée à la création de scènes animées, héritant de la classe des scènes, dispose d'une implémentation nouvelle pour le précalcul ainsi que pour le rendu. Celle-ci conserve dans ses attributs une collection d'images fournies ensuite à la classe chargée de l'affichage.

La classe s'occupant de l'affichage a elle-même été enrichie afin de pouvoir afficher en boucle l'animation. La gestion des événements est alors légèrement différente car une boucle doit pouvoir continuer d'animer la scène tout en attendant un événement pour quitter l'affichage.

3.6 Conclusion

Ces extensions à notre travail ont permis d'améliorer dans tous les cas les rendus obtenus, soit en enrichissant les scènes que nous pouvions créer, par la diversité dans la géométrie des objets et leurs propriétés ; soit en rendant plus vivante la scène, par les réflexions, les reflets directs des sources ou encore l'animation de la scène.

4 Conclusion

Finalement, malgré les difficultés que nous avons rencontrées en début de projet, nous avons réussi à avancer. Une bonne répartition des tâches, et la relecture croisée de nos codes nous ont permis de rebondir face à ces échecs car de l'avance avait été prise sur certains points d'implémentation malgré le blocage sur d'autres points cruciaux.

Ce projet a été l'occasion de se confronter au paradigme de la programmation orientée objet avec le langage C++. Malgré le peu d'expérience avec ce langage, nous avons su tirer parti de son paradigme et ainsi produire un code robuste.

5 Bibliographie

Références

[1] W. J. Matt Pharr and G. Humphreys, "Physically based rendering :from theory to implementation." http://www.pbr-book.org, 2004.

[2] "https://upload.wikimedia.org/wikipedia/commons/thumb/9/91/Reflexion_fr.png/300px-Reflexion_fr.png]."

[3] "Ray-casting formulas." http://www.ccs.neu.edu/home/fell/CS4300/Lectures/Ray-TracingFormulas.pdf.