Informatique graphique : Introduction au Raytracing

Jean-Thomas BEAUDOIN

Professeur : Nicolas BONNEEL

Année 2019-2020

Préambule

Je n'avais jamais fait de C++ avant de commencer ce cours. Je n'avais déjà pas fait beaucoup d'informatique avant d'arriver en option info en 3A à l'école mais jamais de C++. J'ai donc passé énormément de temps pour réaliser les travaux présentés dans ce rapport (entre 8 et 10 heures par semaine en moyenne, incluant les 4h de cours habituelles).

La vidéo m'a aussi beaucoup aidé, notamment au début lorsque j'apprenais à coder dans ce langage en commençant le projet. Sans cet outil, je ne pense pas que je serais arrivé au quart de ce que j'ai fait (bien que j'ai assisté à tous les cours sauf un pour cause de partiel de master data science à l'université).

Selon moi, il faudrait avoir quelques explications au début du projet sur le langage : comment ça marche, quels sont les résultats qu'on peut obtenir, à quoi ça sert, … Je pense aussi que le cours gagnerait en intérêt pour les étudiants qui découvrent l'infographie et le C++ en passant plus de temps à expliquer étape par étape ce que l'on fait et comment on le fait. Montrer quelques lignes de code via le vidéoprojecteur ne m'a pas du tout servi et m'a plutôt découragé à poser des questions sur des choses très simples mais primordiales pour amorcer le projet (par exemple sur la compilation, la surcharge d'opérateurs, … des choses complètement inconnues pour moi au début de ce projet !).

De plus, je ne pense pas que la seule raison au fait que des élèves ne soient pas à jour dans le projet soit un manque de travail de la part des étudiants (ça peut tout de même l'être pour certains). Lorsqu'on n'a rien compris à ce qu'on fait et à comment ça marche, on a beau passer des heures dessus, ça n'arrangera rien et, encore une fois, ça n'incitera pas à poser des questions sachant que la seule réponse qu'on aura, c'est qu'on n'a pas assez travaillé chez nous.

Sommaire

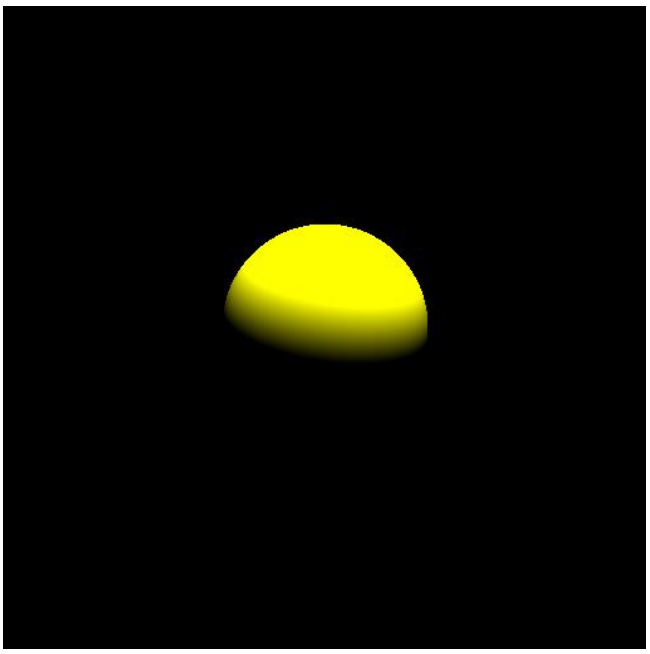
Réalisation de sphères

Avant toute chose, nous avons défini nos classes permettant de créer des objets par la suite. La classe principale est la classe « vector » pouvant définir un vecteur ou un point par ses coordonnées cartésiennes. La classe « sphere » permet de créer des sphères via un centre et un rayon. La classe « scene » permet de rassembler les différents objets que l'on veut ajouter à l'image.

On commence par réaliser une sphère jaune. On éclaire cette sphère avec une source de lumière définie par sa position. On définit aussi l'intensité de la lumière qui permettra d'ajuster le rendu de l'image selon les représentations que nous voudrons faire.

L'image est réalisée en étudiant s'il y a une intersection entre les rayons provenant de la source de lumière et la sphère. Lorsqu'il y a deux intersections (de chaque côté de la sphère), seul le point le plus proche de la source de lumière sera éclairé.

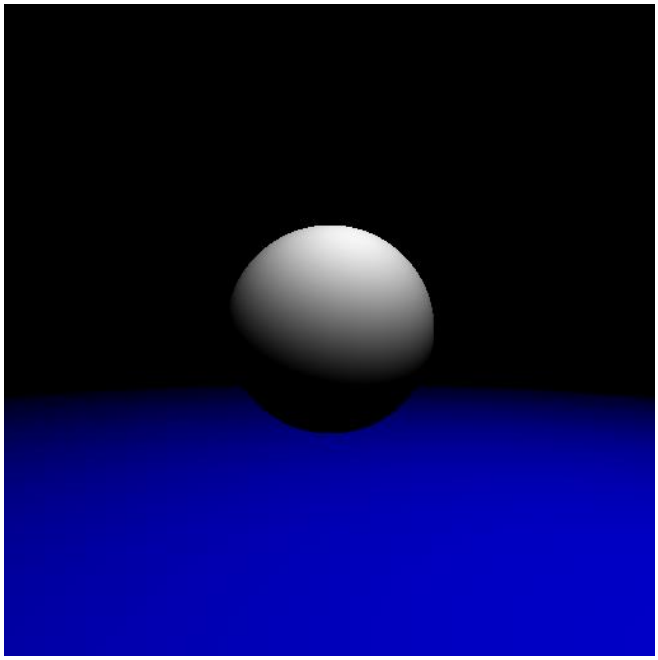
On obtient alors l'image suivante :



Réalisation d'une sphère jaune

Cette image est une simple sphère jaune éclairée par une source de lumière en haut. On peut alors rajouter d'autres éléments dans la scène. On peut notamment rajouter un sol, un plafond et des murs sur les côtés. Pour l'instant nous savons uniquement faire des sphères. Pour obtenir chacun de ces éléments, nous allons donc utiliser des sphères avec un grand rayon pour que les extrémités qui apparaissent sur l'image paraissent plates. Le centre de ces sphères sera alors très éloigné du cadre de l'image.

Dans la figure suivante, on représente une sphère blanche avec un sol bleu.



Réalisation d'une sphère avec un sol

Dans les figures suivantes, on ajoutera de la même manière des murs, un plafond ainsi qu'un fond derrière la figure.

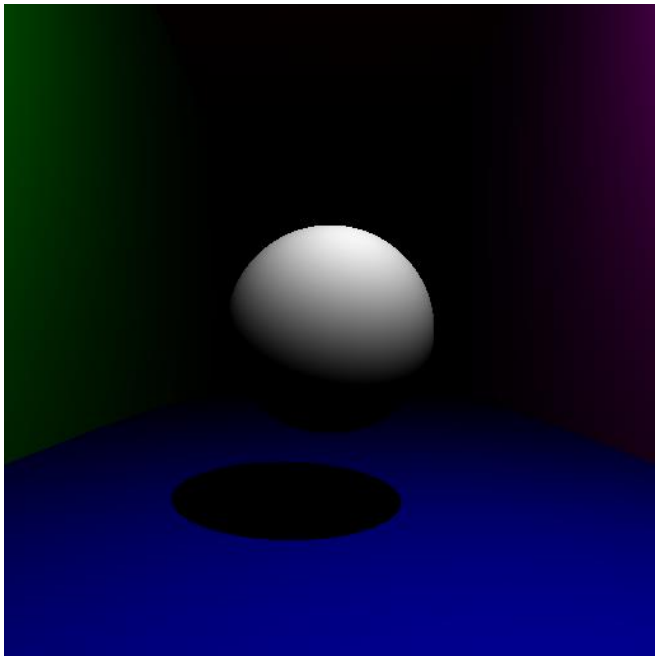
Ombre

Pour l'instant, l'éclairage des sphères par la source de lumière se fait indépendamment des autres sphères présentes dans la scène. Tous les points du sol seront donc éclairés de la même manière bien qu'il y ait la présence d'une sphère au-dessus du sol.

En tenant compte des superpositions des différents objets, nous allons faire apparaître des ombres sur certains des objets de la scène.

Le principe est le même que lorsqu'on veut éclairer la partie de la sphère la plus proche de la lumière. On repère s'il y a au moins une intersection entre le rayon provenant de la source de lumière et une sphère (quelle qu'elle soit). Dans le cas où il y a plusieurs intersections entre le rayon et un objet de la scène, on regarde qu'elles sont les distances entre la source de lumière et les points d'intersections et on conserve uniquement le plus petit. Ce sera la distance qui caractérise l'objet qui sera éclairé.

On reprend alors la figure précédente à laquelle on a ajouté deux murs sur les côtés. Etant donné la position de la sphère centrale, on observera uniquement une ombre sur le sol.



Ajout d'une ombre

Correction gamma

Les intensités des couleurs rouge, verte et bleue permettant de réaliser toutes les couleurs des sphères ne sont pas linéaires (ce n'est pas en plaçant le coefficient de l'intensité au milieu du champ possible qu'on aura l'intensité se trouvant au milieu des intensités possibles). Pour rétablir les couleurs, on applique une correction gamma.

La correction gamma consiste uniquement ajouter une puissance à la couleur obtenue via le raytracing. On évalue le coefficient de puissance des différences d'intensité à 2,2. Pour la correction gamma, on met donc toutes les couleurs calculées à la puissance 1/2,2.

Ajout de propriétés aux sphères

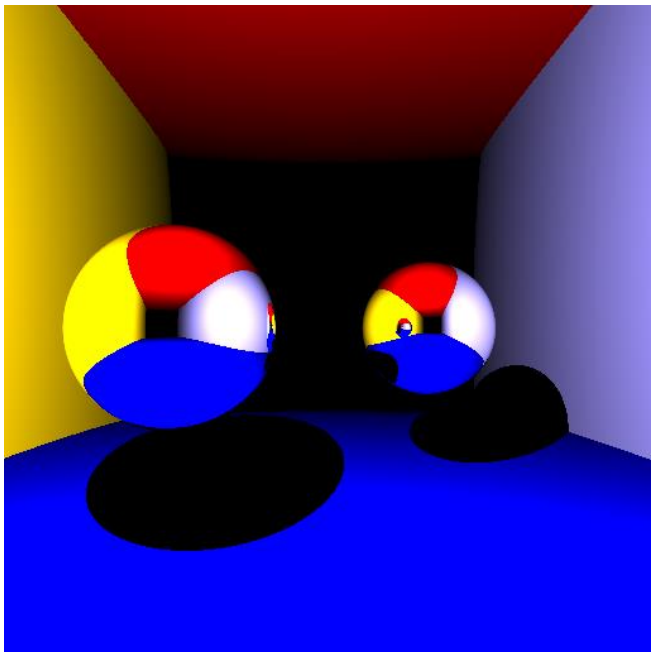
Maintenant que nous avons ajouté des ombres pour que la scène soit plus réaliste, nous pouvons ajouter des propriétés aux sphères.

Nous allons d'abord ajouter la propriété de miroir aux sphères. Pour cela, nous faisons une fonction qui permet d'obtenir la couleur via un rebond du rayon sur la surface miroir. Cette fonction prend en compte l'angle d'incidence du rayon sur la sphère et renvoie la couleur du point sur lequel arrive le rayon réfléchi par le miroir.

Cette fonction sera utilisée uniquement si on caractérise la sphère comme étant un miroir dans ses paramètres.

Dans le cas où le rayon réfléchi arrive sur une autre sphère miroir, on fixe un nombre maximum de rebonds pour que le programme fonctionne indéfiniment à cause d'un rayon faisant des allers-retours entre deux surfaces miroirs. Ce nombre de rebonds est décrémenté à chaque rebond sur une surface miroir.

Dans la figure suivante, on représente deux sphères miroirs avec un sol, deux murs et un plafond. On observe avec les couleurs différentes des parois que les rayons se reflètent correctement, y compris lorsque des rayons se reflètent plusieurs fois sur des surfaces miroirs. La correction gamma a également été appliquée.

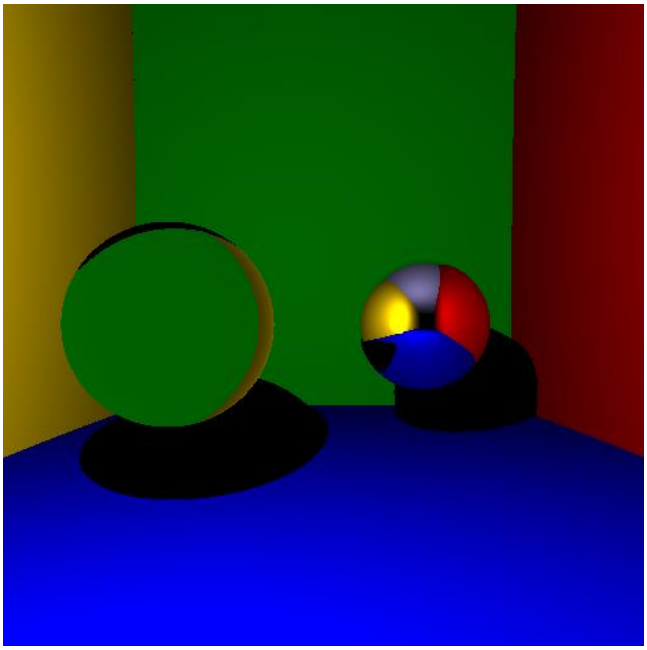


Sphères miroirs

On va alors ajouter une propriété de transparence à l'une des sphères. On ajoute un paramètre de transparence aux objets sphères de la même manière que pour la propriété miroir. Pour cela, si la surface est transparente, le rayon qui arrive de la caméra va être réfracté (suivant les lois de réfraction classique, il faudra donc ajouter un indice de réfraction pour le matériau utilisé pour faire la sphère transparente (par la suite, on prendra 1,3).

En appliquant les lois de la réfraction à l'entrée et à la sortie de la sphère, le rayon sera légèrement modifié pour arriver sur un point qui lui donnera sa couleur.

On obtient l'image suivante avec une sphère miroir et une sphère transparente (après avoir aussi ajouté un fond à l'arrière de l'image, on peut observer le plafond grâce à la sphère miroir).



Sphère transparente et sphère miroir

Eclairage indirect

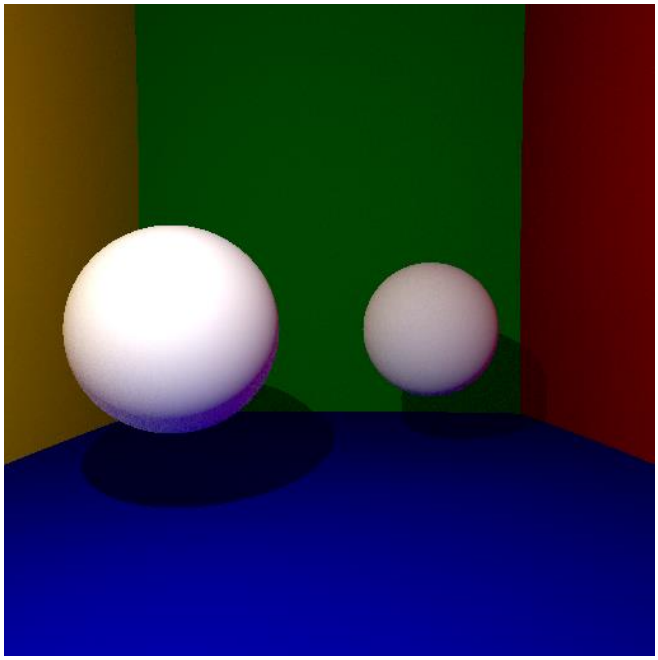
On revient à une image avec deux sphères qui ne sont ni des miroirs, ni transparentes.

On va alors rajouter de l'éclairage indirect. L'éclairage indirect est le fait que les rayons provenant de la source de lumière se reflètent pour avoir la couleur de la surface sur laquelle arrive leur rayon réfléchi. La couleur provenant de la surface sur laquelle arrive le rayon réfléchi sera atténuée selon la distance que parcourt le rayon réfléchi. Cette couleur sera alors ajoutée à la couleur de base de l'objet.

L'éclairage indirect a donc le même fonctionnement que les objets miroirs. Les seuls changements seront l'atténuation de l'intensité de la couleur avec la distance parcourue par le rayon réfléchi et le fait que le rayon ne se réfléchisse une seule fois.

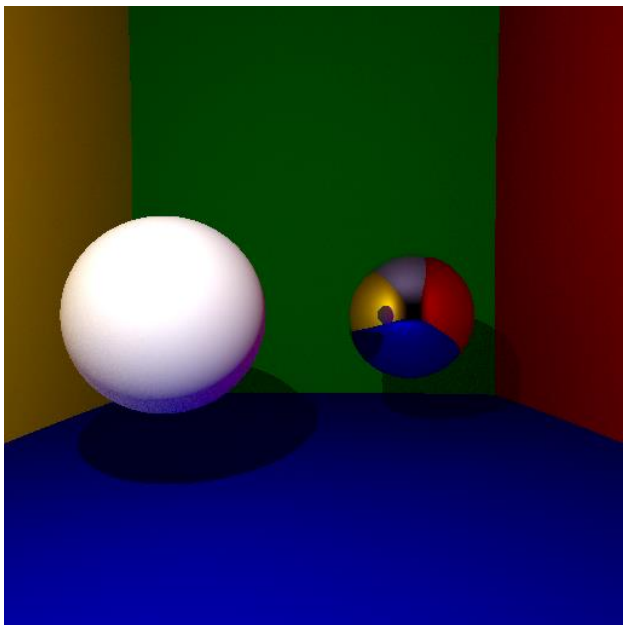
On implémente donc l'éclairage indirect en reprenant le code de la surface miroir en divisant l'intensité de la lumière ajoutée par la distance qu'il y a entre un objet et la source de lumière secondaire (en général une des parois de la pièce que l'on représente).

Pour pouvoir observer l'éclairage indirect, on réalise deux sphères blanches sur la figure suivante :



Ajout de l'éclairage indirect

On peut alors faire la même image avec la sphère en arrière étant un miroir. On observe alors bien l'éclairage indirect à l'arrière de la sphère avant.



Réalisation d'une image avec l'éclairage indirect et une sphère miroir

Anti-aliasing

L'image précédente semble être satisfaisante lorsque qu'on la voit. Or, lorsque l'on agrandit cette image, on observe un crénelage sur les frontières de la sphère blanche comme ci-dessous (on observe la même avec la sphère en miroir). On appelle cela l'aliasing.

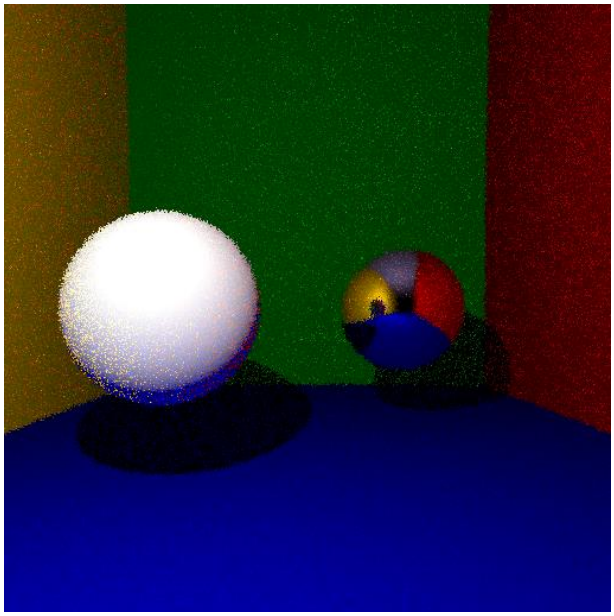
Représentation de l'aliasing sur la sphère blanche

L'aliasing est dû au fait que les couleurs sont uniformes par pixel et que tous les rayons sont envoyés de la caméra vers les centres des pixels. Pour remédier à ce problème, nous allons envoyer les rayons vers des endroits différents des pixels. En envoyant plusieurs rayons par pixel, on pourra obtenir des couleurs plus uniformes pour chaque pixel.

La répartition de l'arrivée des rayons sur un pixel se fait selon une gaussienne centrée au centre du pixel et d'écart-type la longueur du pixel.

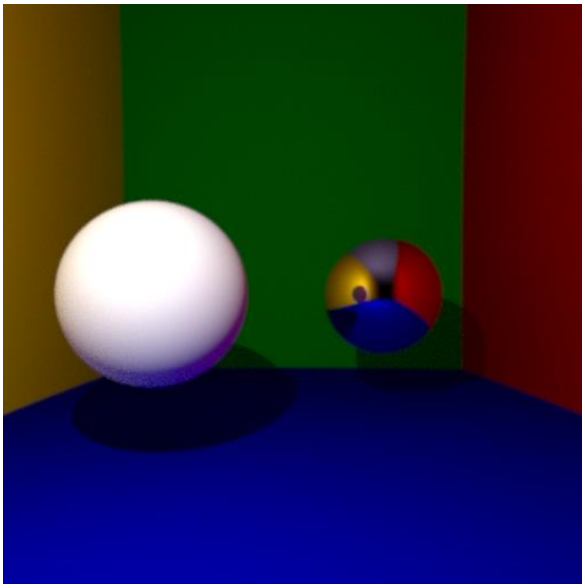
L'exécution met alors beaucoup plus de temps. Plus on envoie des rayons, plus le résultat sera précis et inversement.

On obtient le résultat suivant avec peu de rayons :



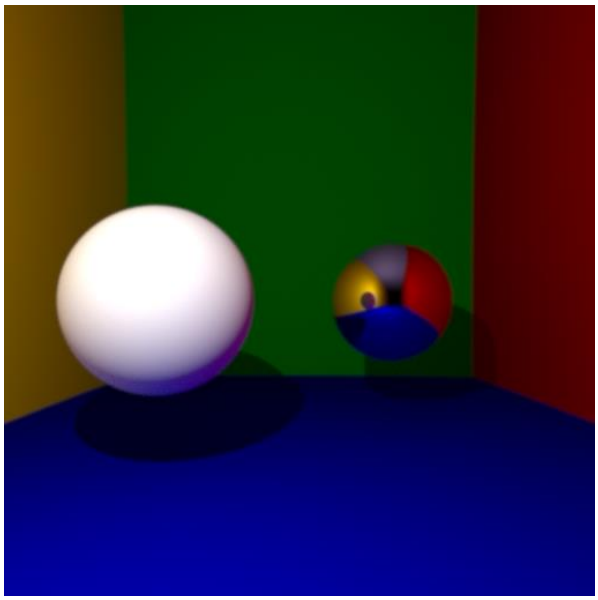
Anti-aliasing avec peu de rayons

On obtient une image très bruitée. On prend alors 100 rayons par pixel et on obtient l'image suivante :



Anti-aliasing avec 100 rayons

On obtient une image très satisfaisante, on peut essayer d'ajouter des rayons et on obtient l'image suivante :



Anti-aliasing avec 500 rayons

En agrandissant l'image, on voit qu'elle n'est pas crénelée :

Représentation de la sphère sans crénelage

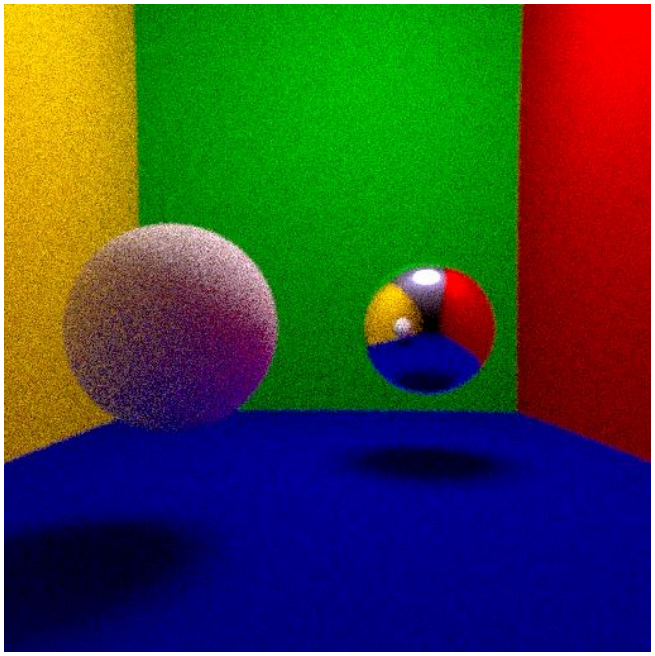
L'image obtenue avec 500 rayons est un peu moins bruitée que celle obtenue avec 100 rayons mais le temps d'exécution du programme est beaucoup plus long. Pour l'obtention des images suivantes, nous utiliserons donc 100 rayons.

Ombre douce

On va maintenant essayer d'ajouter de la lumière différemment. Jusqu'à présent, la source de lumière était ponctuelle. On va alors éclairer la scène avec une lumière répartie sur un surface plus grande. Cela permettra de créer des ombres douces : les ombres ne seront plus aussi marquées que sur les images précédentes. La transition entre une partie à l'ombre et une partie éclairée sera moins directe.

Pour réaliser cela, on ajoute une sphère qui sera la sphère émissive (remplaçant la source de lumière). Cette sphère est repérée en étant la première sphère contenue dans la classe « Scène ». La couleur de cette sphère dépendra alors de l'intensité de la lumière que l'on souhaite avoir. Le principe de la création de l'image reste alors le même (rebond des rayons, …).

L'intensité de la lumière sur l'image sera alors un peu altérée du fait qu'elle soit renvoyée par une sphère pouvant avoir une certaine couleur. Il faut alors augmenter l'intensité de la lumière que l'on définit dans notre programme. Cela nous donne alors l'image suivante :

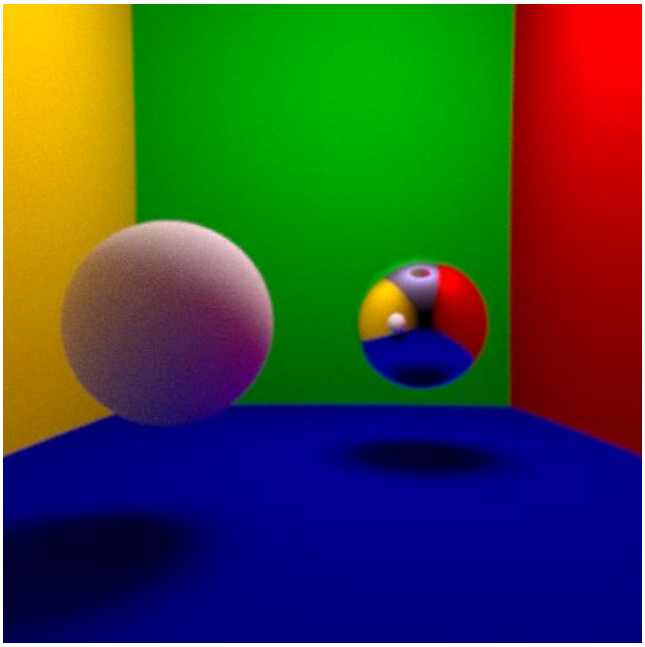


Ajout d'une sphère émissive

Cette image est un peut bruitée, on pourrait alors augmenter le nombre de rayons envoyés pour obtenir une image plus convenable. Cependant, le temps d'exécution est déjà assez long pour obtenir cette image. Ajouter des rayons rallongerait encore ce temps d'exécution.

Pour éviter cela, nous complétons la routine permettant d'obtenir la couleur d'une sphère qui n'est ni un miroir, ni transparente. Cet ajout permettra de diriger les rayons aléatoires renvoyés vers la source de lumière plutôt que dans une direction totalement aléatoire.

Nous obtenons alors le résultat suivant :



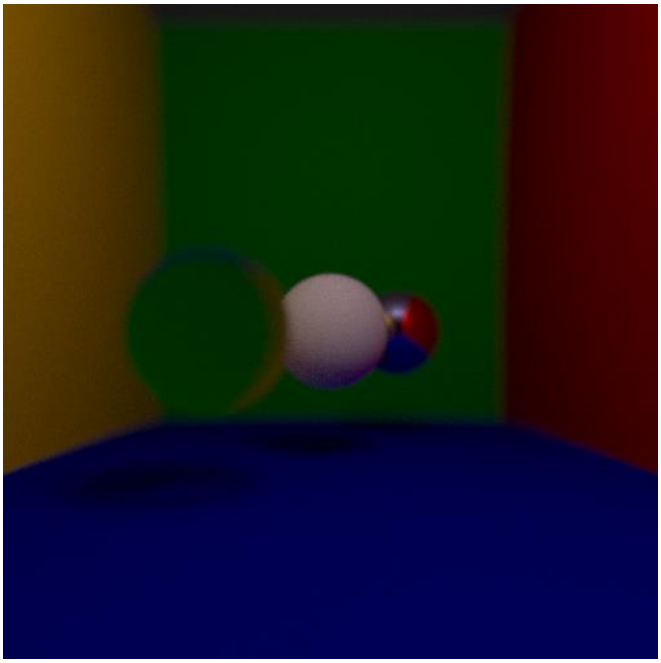
Amélioration de la sphère émissive

Profondeur de champ

On peut alors ajouter une profondeur de champ. Cet aspect est réalisé grâce à un plan focal. Ce plan focal est calculé selon une distance que l'on choisit. On va alors calculer la couleur des différents points en forçant les rayons à se diriger vers les points de ce plan focal.

En pratique, les rayons allant vers chaque point seront focalisés sur le plan focal à l'endroit où le rayon allant vers ce point traverse le plan focal.

Pour pouvoir mieux observer notre profondeur de champ, nous avons ajouté une sphère blanche entre nos deux sphères initiales (transparente et en miroir). Nous faisons en sorte que la profondeur de champ arrive sur la sphère centrale et nous obtenons alors le résultat suivant :



Profondeur de champ

Maillage

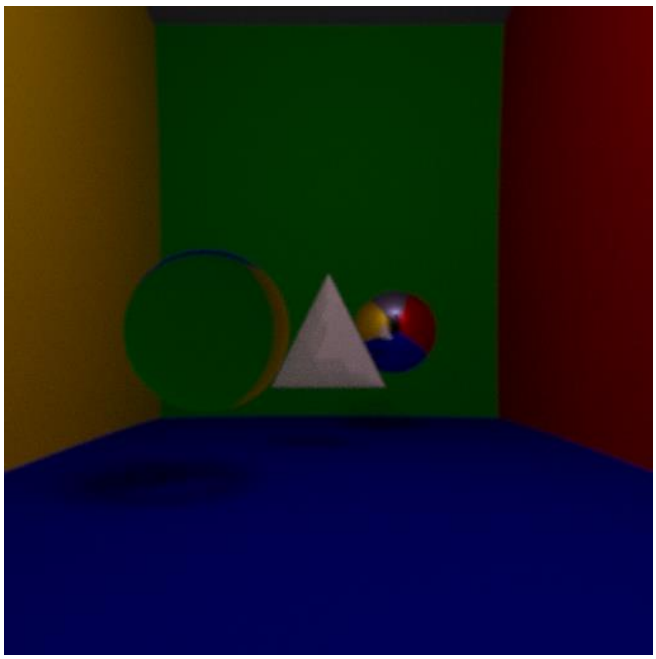
On va maintenant essayer de faire d'autres formes que des sphères.

Pour les sphères, l'équation de la sphère suffisait pour trouver son intersection avec le rayon car cette équation était simple. Pour d'autres objets, il peut y avoir des équations beaucoup plus complexes voire pas d'équation pour un objet que l'on veut implémenter. Nous allons donc utiliser des maillages.

Le principe est de diviser l'image en pixels via un maillage. On va alors procéder comme avec chaque pixel comme nous le faisions pour la sphère. Les couleurs sont affectées pixel par pixel selon les objets que l'on veut représenter.

Grâce aux maillages, nous allons représenter un triangle. Pour cela, nous créons une classe « Triangle » définissant les différents paramètres dont nous avons besoin pour dessiner un triangle ainsi que la routine d'intersection entre un rayon et un triangle. Nous avons aussi défini une classe « Object » dont hériteront les classes « Triangle » et « Sphere » afin de définir une bonne fois pour toutes les paramètres communs aux différents objets que nous serons amenés à représenter.

Nous avons alors représenté un triangle au milieu de nos deux sphères dans l'image suivante :



Réalisation d'un triangle via le maillage

Conclusion

Dans ce travail, je suis arrivé jusqu'au maillage simple (réalisation d'un triangle). Je n'ai pas réussi à implémenter d'autres éléments (objets plus complexes, textures) du fait que j'avais beaucoup de mal à comprendre le code fourni et à l'adapter à mon travail.

Néanmoins, je pense que les aspects principaux du raytracing ont été réalisés : réalisation de sphères, ombres, propriétés de miroir et de transparence, correction gamma, éclairage indirect, anti-aliasing, ombre douce, profondeur de champ et maillage simple.