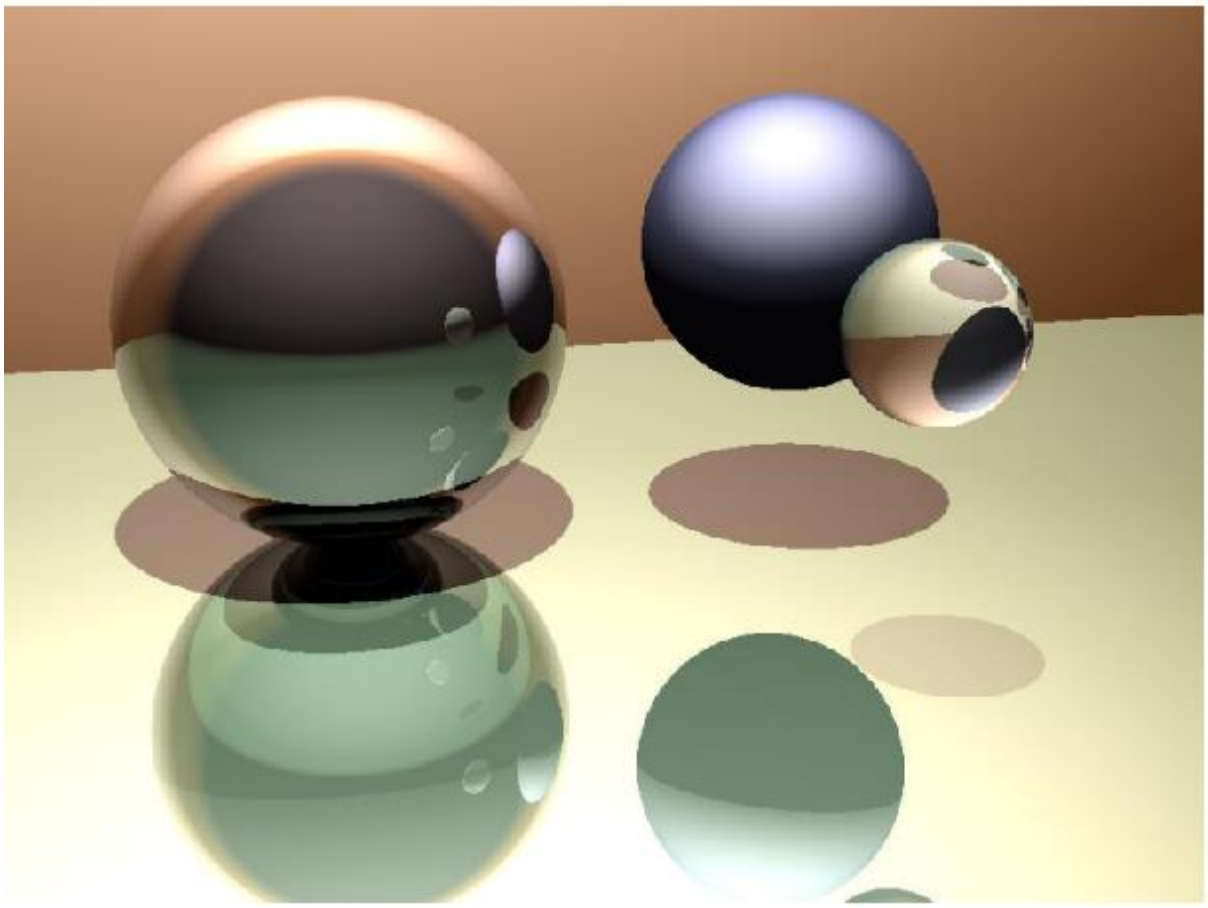
Rapport SAE 3.01A : Ray-Tracing



BOUGE Nicolas
BREIL Thomas
LORIDAN Maïa
POUYADE Samuel

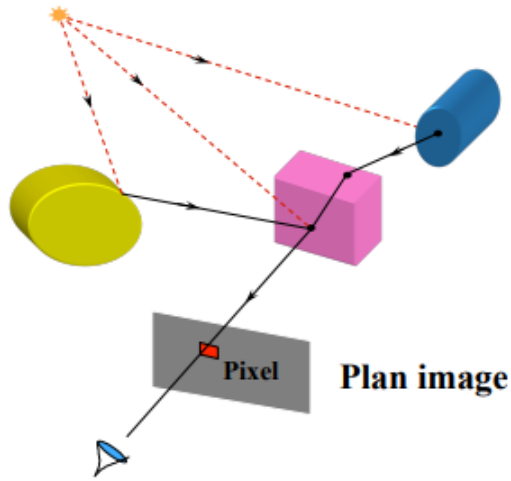
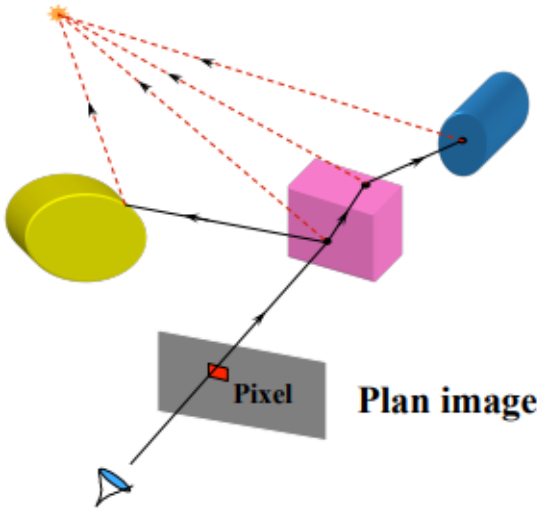
Table Des Matières :

1 - Présentation du travail à réaliser

Le ray tracing, lancer de rayon en français, est une méthode de modélisation du transport de la lumière par ordinateur. Cette méthode utilise des lois de physique, telles que la réflexion ou la réfraction. Elle permet donc de générer des effets de lumière et d'ombre sur des objets numériques, comme par exemple une sphère.

Contrairement à la vie réelle, on va générer les rayons depuis la caméra afin d'optimiser les temps de calcul. Grâce à cette méthode, on ne crée que les rayons qui atteignent la caméra.

Cf. Sujet

Vie Réelle	Lancer de Rayons
	

On voit donc que le sens des vecteurs change. Pour ce faire, nous avons créé des vecteurs partant de la caméra vers chaque pixel de l'image. Ensuite, en fonction des caractéristiques des objets rencontrés, on renvoie d'autres rayons selon la réflexion et la réfraction.

Ce projet nous a permis de créer une image constituée d'objets en trois dimensions utilisant le ray tracing. Pour cela, nous devions compléter le squelette d'un code en C++ qui nous a été fourni. Nous avions différents fichiers, dont un principal contenant les données de l'image, et d'autres étant déjà complets.

2 - Analyse du travail

Pour obtenir le résultat attendu, c'est-à-dire l'image finale qui nous a été fixée comme objectif, nous avons d'abord dû regarder quels fichiers nous devions compléter. Nous avons également regardé les fichiers déjà finis pour comprendre leur fonctionnement et leur utilité dans le projet.

Nous avons déterminé que les classes bases3d, sphere3d et camera, qui gèrent respectivement les fonctions liées au vecteur, la fonction d'intersection d'une sphère et la construction de l'image. Il fallait également implémenter rayon, qui détermine la couleur de chaque pixel et lumière, qui calcule l'intensité des couleurs en fonction des lumières.

Pour compléter ces classes, nous avons effectué des recherches sur les différentes lois de physique, nous avons également consulté la documentation des fichiers.hpp des classes correspondantes. Les lois utilisées pour ce projet sont les lois Snell-Descartes, décrivant le comportement de la lumière.

Grâce aux informations obtenus sur les différentes classes, nous avons défini un ordre d'importance pour la complétion des classes. Tout d'abord nous avons ajouté les éléments manquants aux classes bases3d.cpp, sphere3d.cpp et camera.cpp, afin d'obtenir une base pour ensuite finir les classes lumiere.cpp et rayon.cpp.

3 - Méthode de travail

Nous avons commencé notre projet par une phase d'analyse, c'est-à-dire que nous ne devions pas modifier le code avant de comprendre chacune des méthodes à compléter et où elles étaient utilisées.

Après cette analyse du code, nous avons pris la décision de créer un git privé afin que les interactions entre chaque membre soient plus faciles, mais aussi en cas de perte des modifications de la part de l'un des membres. Cela nous évitait de devoir s'envoyer le code à chaque modification.

Une fois notre git créé, nous avons éclairci nos points de blocage sur la non-compréhension de certaines méthodes. Nous avons enfin discuté l'ordre de complétion des classes avant de commencer l'implémentation du code. Nous avons d'abord complété la classe bases3d.cpp qui semblait être la plus essentielle à terminer en premier.

Nous avons par la suite effectué une phase de recherches afin d'avoir suffisamment de ressources et d'informations à notre disposition. Cela nous a permis de ne pas avoir de problèmes trop importants pour remplir les fonctions au début de notre travail. Une fois que nous avions complété ensemble bases3d.cpp, nous avons travaillé par binôme pour la suite du projet afin d'être plus performant et d'éviter d'avoir du retard pour les échéances.

Le travail en binôme consistait à avoir au moins un ordinateur avec les recherches à portée de main et un autre servant à implémenter les méthodes manquantes. Les points sur l'avancée et les difficultés étaient régulièrement discutés entre les binômes durant les heures de travail.

4 - Travail réalisé

Comme indiqué précédemment, nous avons commencé le projet par compléter en priorité les classes bases3d.cpp, sphere3d.cpp et camera.cpp. Cela nous permettait d'obtenir une image en 2D, servant de base pour la suite, car nous pouvions observer les différentes modifications faites au code grâce à une image concrète. Premièrement, dans la classe bases3d.cpp nous avons implémenté les opérations qui nous permettraient d'utiliser et de travailler avec les vecteurs 3D et les points 3D.

Bases3d:

Point3d :

operator +(const Vecteur3D & v)
operator -(const Vecteur3D & v)
operator -(const Point3D & p)
operator ==(const Point3D& p)

Vecteur3d :

Longueur()
Normaliser()
operator +(const Vecteur3D & v)
operator -(const Vecteur3D & v)
operator -()
operator *(float m)
operator ==(const Vecteur3D& v)
operator *(const Vecteur3D & v)
Cross(const Vecteur3D & v)
Reflechir(const Vecteur3D & n)
Refracter(const Vecteur3D & norm, float m1, float m2)

Deuxièmement, dans la classe sphere3d.cpp, nous avons complété la méthode Intersection. Elle vérifie s'il y a une intersection entre un rayon et une sphère afin d'ajouter des effets de lumière.

Sphere3d :

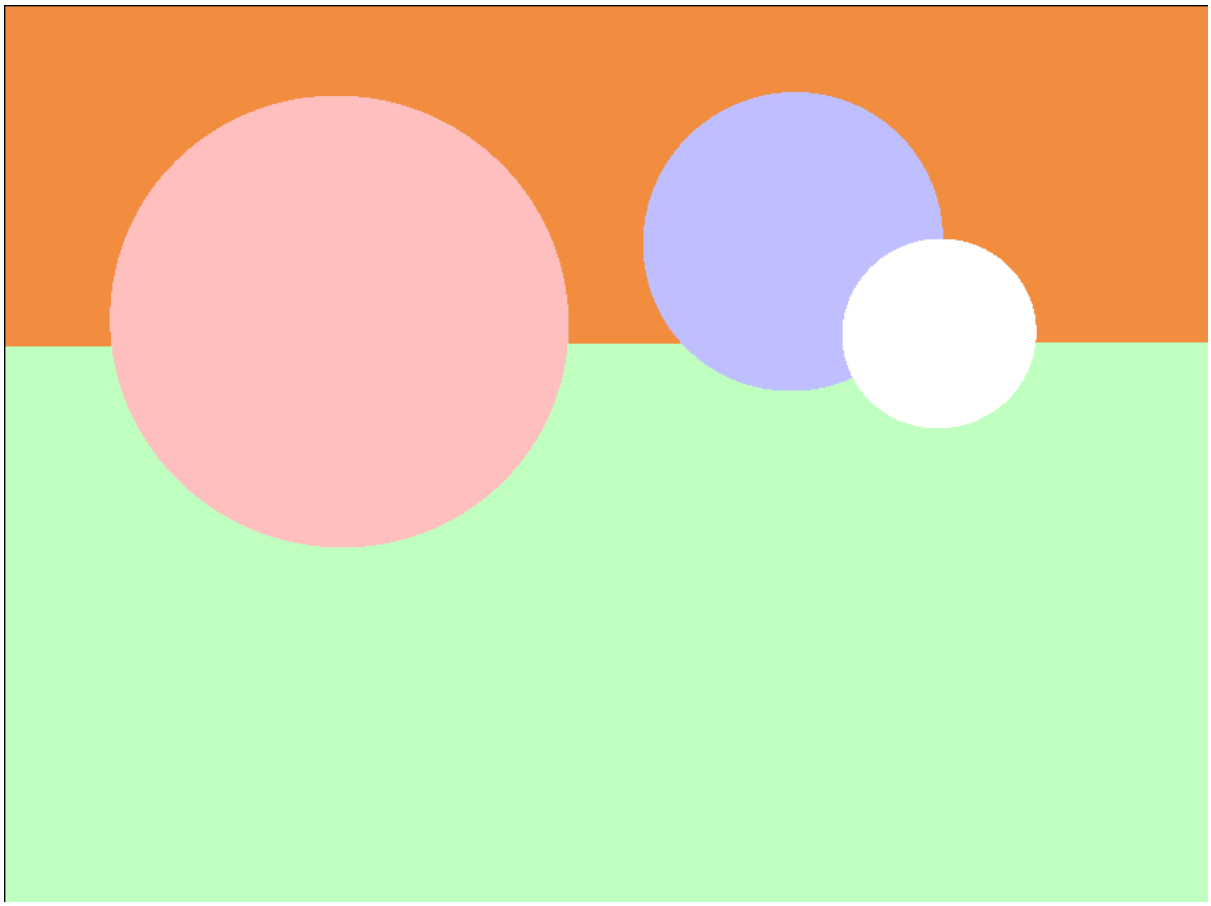
Intersection(const Rayon & r, C_Liste_Intersection & l)

Troisièmement, dans la classe camera.cpp, qui permet la visualisation de l'image contenant les sphère, nous avons implémenté la méthode Calculer_image.

Camera :

Calculer_image(Pixelmap & pm, Liste<Objet3D> & lo, Liste<Lumiere> & ll, int complexite)

Voici le jeu d'essai effectué qui nous a permis de valider la pleine implémentation des méthodes réalisées auparavant :



Nous avons choisi de compléter dans le même temps les classes rayon.cpp et lumiere.cpp, puisqu'elles permettaient d'apporter les effets de réflexion et de réfraction.

Rayon :

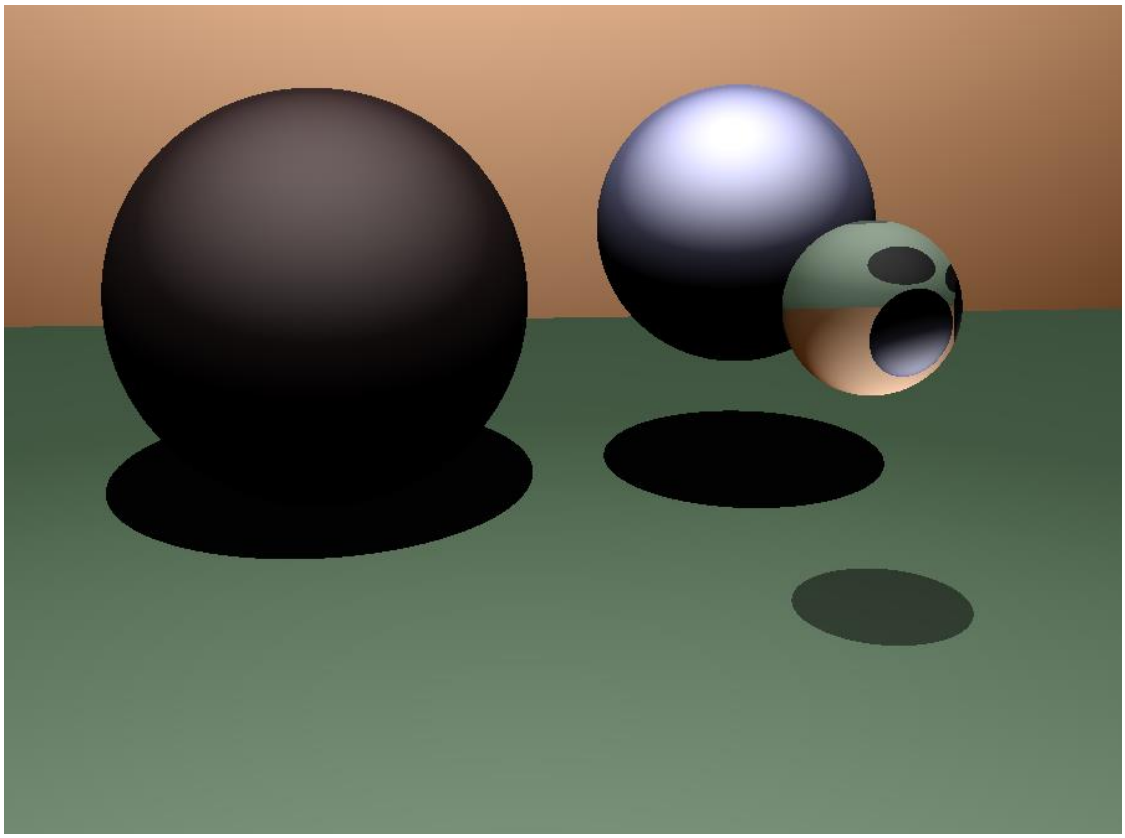
Lancer(Liste<Objet3D> & lo, Liste<Lumiere> & ll, int recur)

Lumiere :

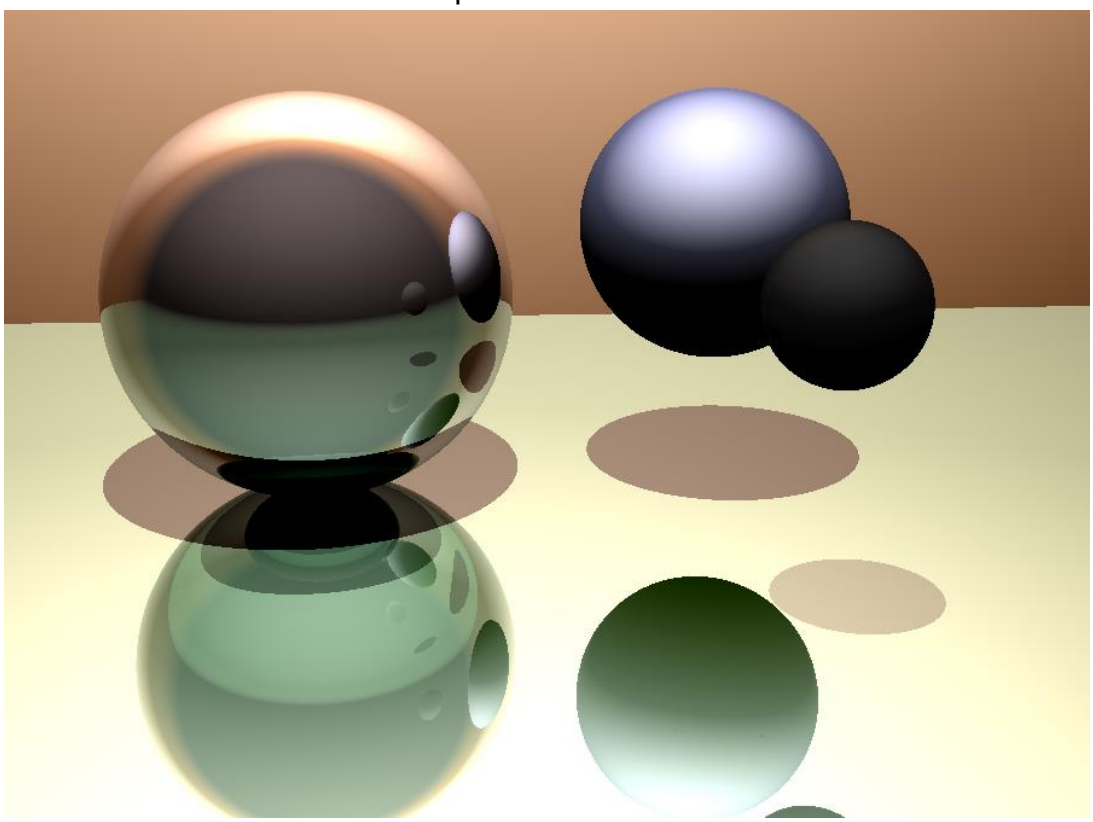
Illumination(const Rayon& r, const Intersection3D& i, const Point3D& p, Liste<Objet3D>& lo)

Afin d'être sûr de nos implémentations nous avons effectué les jeux d'essais suivant :

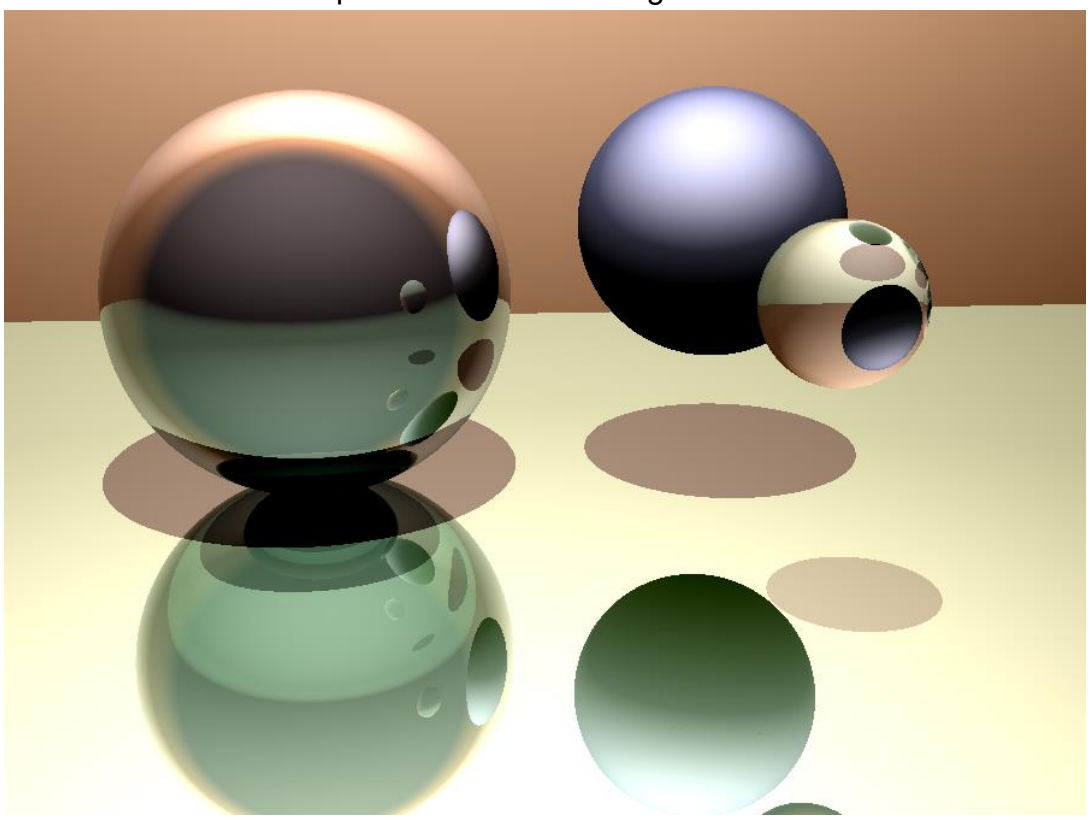
Premièrement vous pouvez observer la réfraction seule sur la sphère en verre



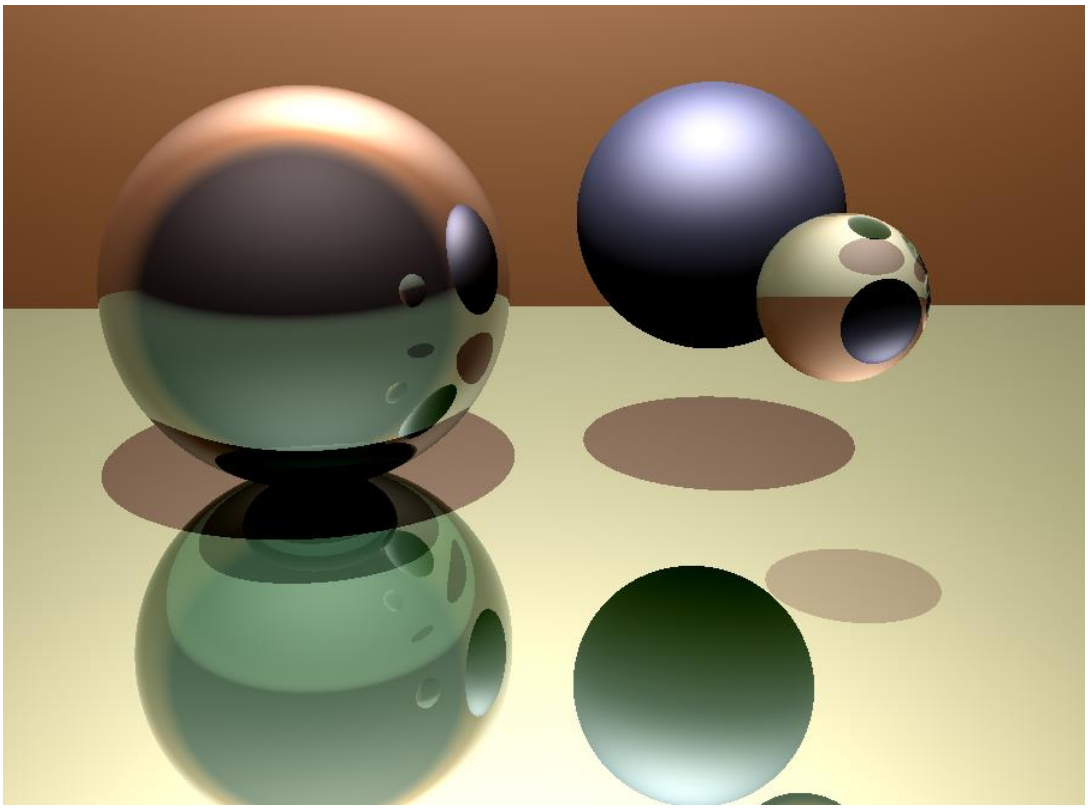
Deuxièmement vous pouvez observer la réflexion seule



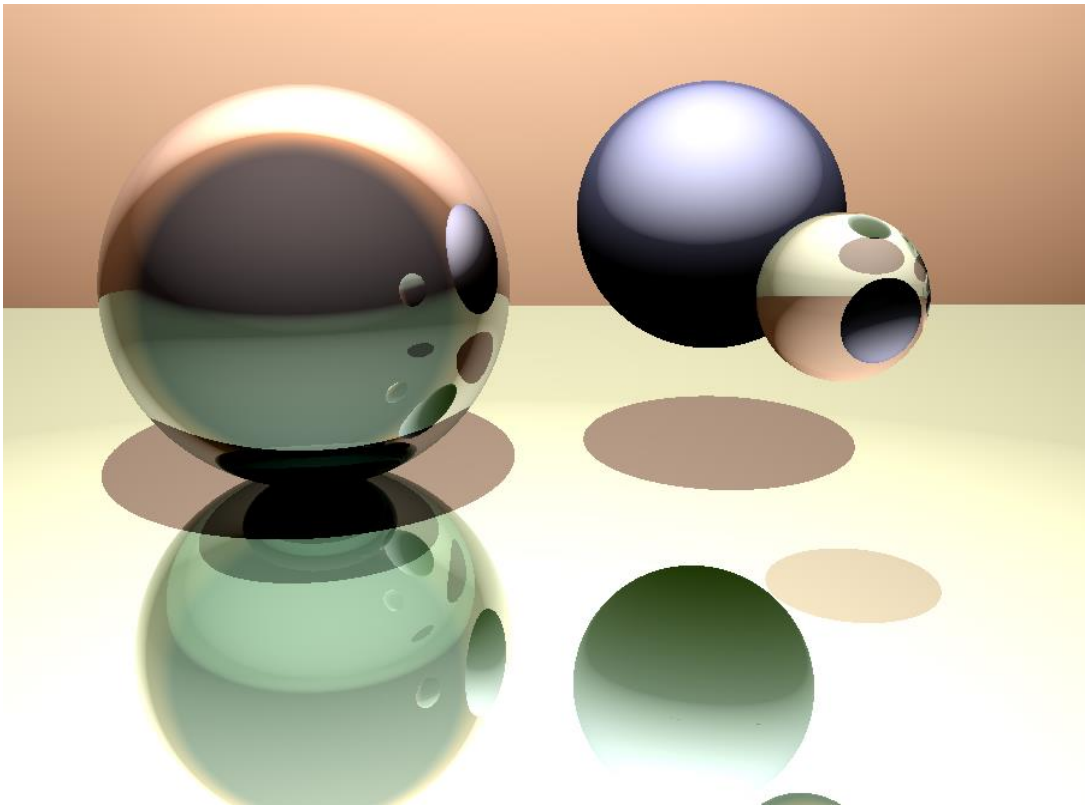
Troisièmement vous pouvez observer l’image finale



Les deux prochaines images montrent les résultats d’une diminution ou d’une augmentation de la lumière spéculaire.
On voit donc que la tache lumineuse sur les sphères diminue.



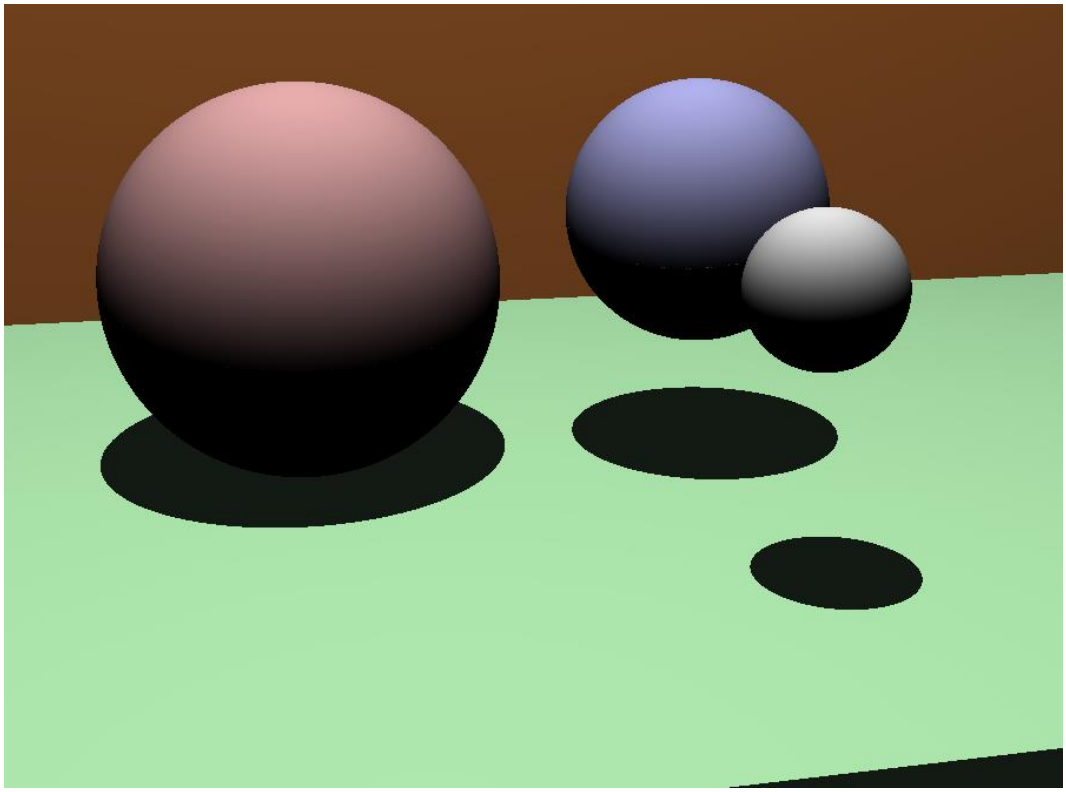
Ici, la tache spéculaire est plus importante.



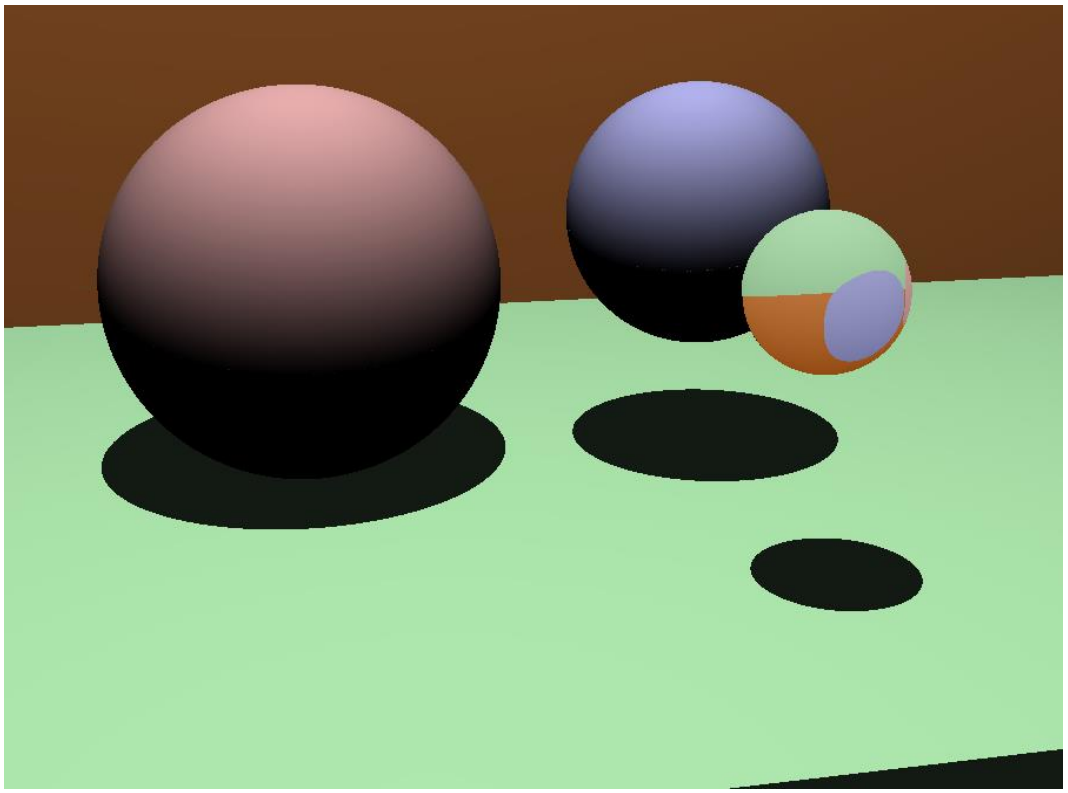
5 - Présentation des difficultés rencontrées

Au cours du projet nous avons rencontré quelques difficultés que nous allons illustrer par des captures d'écran commentées.

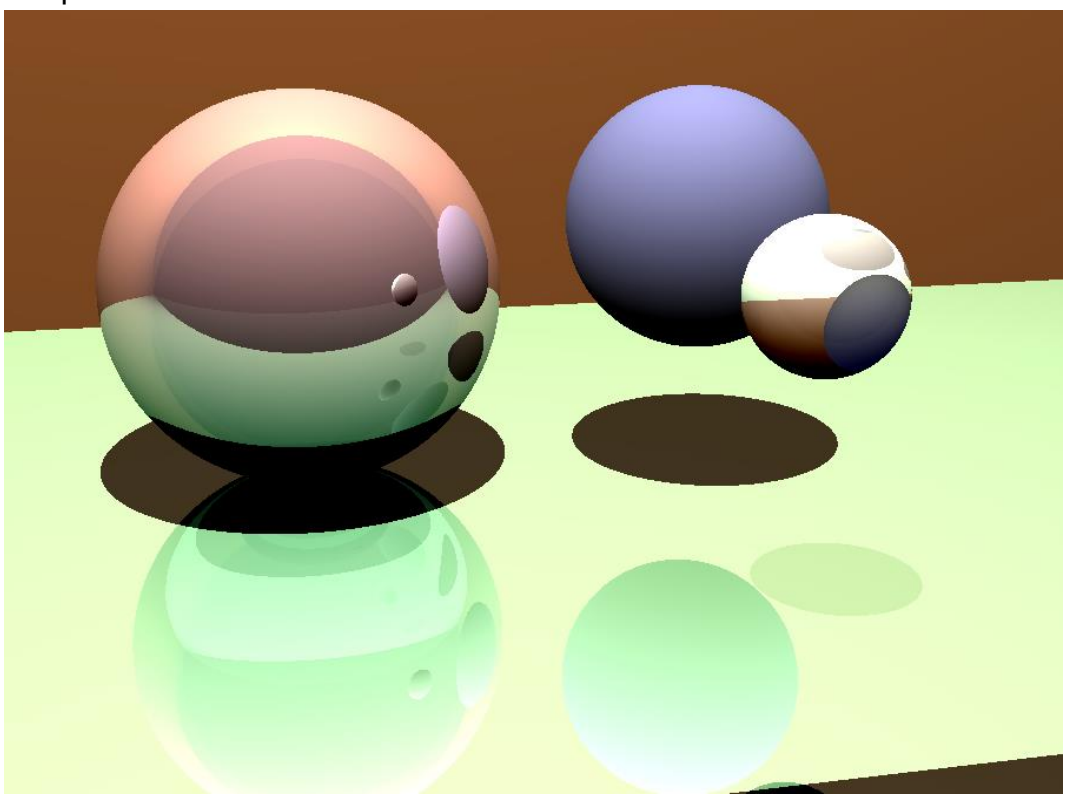
Par exemple pour cette capture d'écran, nous avions eu un problème au niveau du bas de l'image car une bande noire apparaissait.



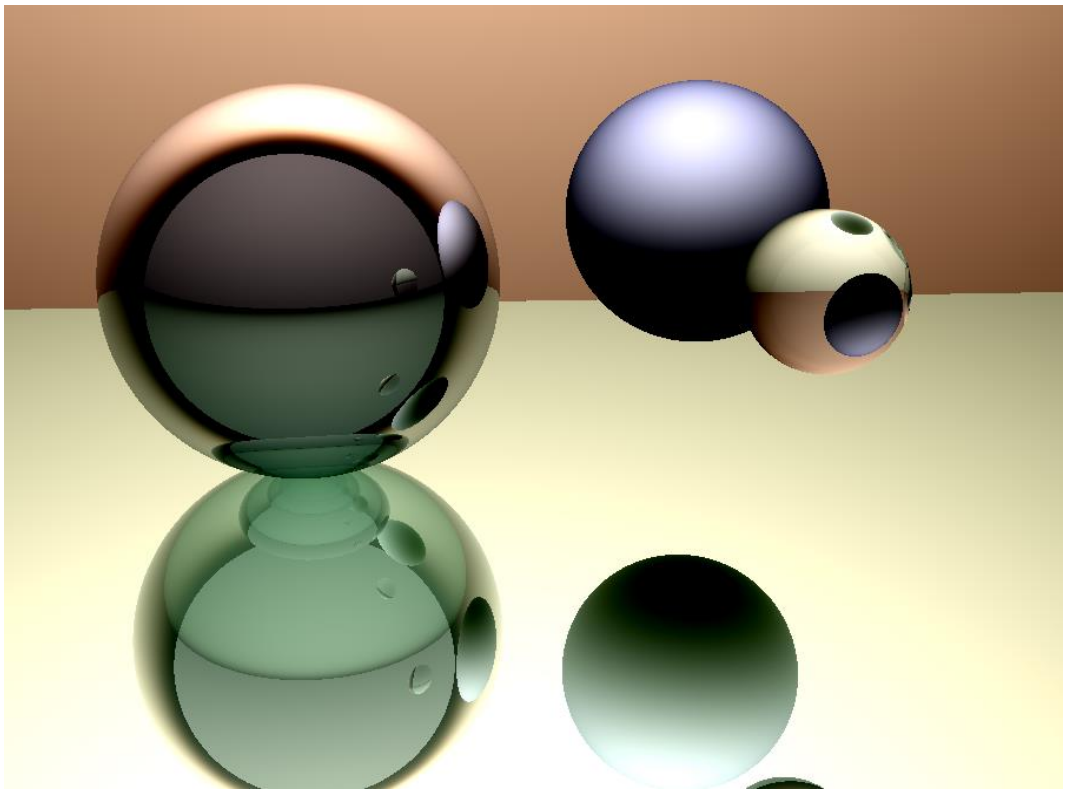
Pour celle-ci la réfraction ne convenait pas, de plus la manière dont nous l'avions implémentée n'était pas la plus adaptée.



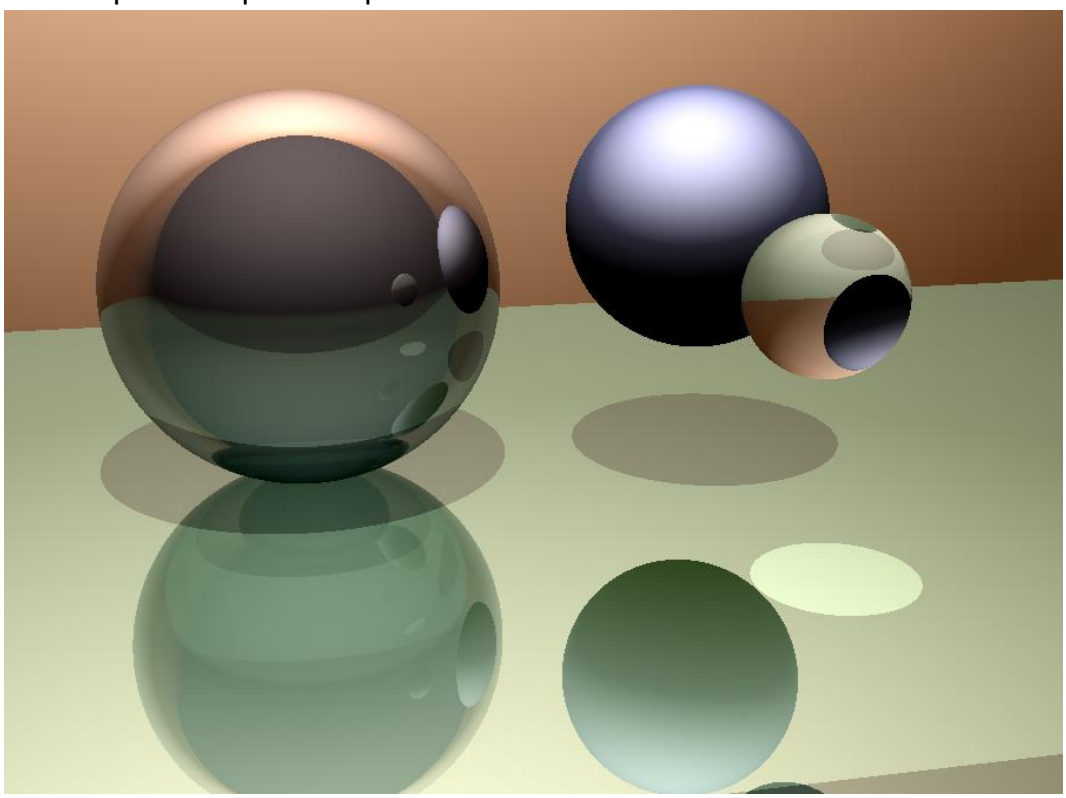
Ici nous avions trop de lumière pour la sphère en verre. De plus, la réflexion n'était pas acceptable.



Pour cette capture d’écran, il nous manquait les ombres principales. Cependant nous commencions à avoir une réfraction convenable.



Pour terminer nous avons eu de nouveaux obstacles à franchir avec les ombres. Qui ici étaient trop claires pour la sphère en verre.



6 - Conclusion

Pour conclure, nous avons trouvé ce sujet intéressant à réaliser : en effet dans un premier temps cela nous a permis de consolider nos bases en C++. Dans un second temps, nous avons apprécié le fait de partir d'un code pour obtenir une image en trois dimensions.

De plus, ce projet nous a motivé à aller plus loin dans nos recherches afin d'ajouter de nouveaux éléments par la suite. Cela donne aussi pour les personnes du groupe souhaitant s'orienter vers des études dans le domaine vidéoludique ou de l'animation une première approche de ceux-ci.

Enfin, pour ce qui est des pistes d'amélioration de notre projet, nous avons pensé aux éléments suivants:

- Eau
- Gestion de plusieurs sources de lumières
- Néon
- Autres formes géométriques
- Nouvelles textures