DÉPARTEMENT DU GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL

INF8702
INFOGRAPHIE AVANCÉE

TP3 :
LANCEUR DE RAYONS

OBJECTIFS DU LABORATOIRE

L'objectif du TP3 est de vous familiariser avec la technique du lancer de rayons. Historiquement, parce qu'ils sont très gourmands en calculs vectoriels, les engins de lancer de rayons étaient conçus pour rouler sur le CPU de manière à produire une image hors ligne (enregistrée sur le disque). Même si l'avènement des cartes graphiques programmables (GPU) permet dorénavant de lancer des rayons « en temps réel », nous nous limiterons pour l'instant à implémenter un engin de lancer de rayons sur CPU afin de bien comprendre tous les mécanismes internes.

Le présent laboratoire permettra donc à l'étudiant de construire son propre engin de lancer de rayons.

MATÉRIEL

On met à votre disposition les choses suivantes :

- Une solution Visual Studio intégrant un projet intitulé « TP3 ».
- Une série d'articles disponible dans Moodle.
    - « Modélisation des quadriques.pdf » portant sur une technique d'expression matricielle des quadriques.
    - « Intersection des quadriques.pdf » traitant de la mathématique de l'intersection entre droites paramétriques et quadriques.
    - « Intersection_rayon_plan.pdf » traitant de la mathématique de l'intersection entre droites paramétriques et plans.
- Un répertoire « Solutionnaire_TP2 » qui contient les différentes modifications commentées et annotées du dernier TP. N'hésitez donc pas à les comparer à votre dernier projet pour vous corriger !

DESCRIPTION DES TÂCHES

1. Se familiariser avec la structure du lanceur de rayons

Ceci n'est pas un requis au barème de correction, mais bien un encouragement à prendre le temps de bien comprendre avant de commencer la structure de code du lanceur de rayons sur CPU que l'on vous a remis.

En plus, pour utiliser correctement le lancer de rayons, vous aurez à indiquer à Visual Studio quel fichier de scène utiliser. Pour ce faire, inscrivez par exemple « Scenes\scene1.dat » dans le menu suivant :

- <Projet>
    - <Propriétés de TP3v1...>
        - <Propriétés de configuration>
            - <Déboguage>
                - <Arguments>

Comportement observable :

Une fois ce requis complété, vous devriez obtenir le résultat visuel suivant :

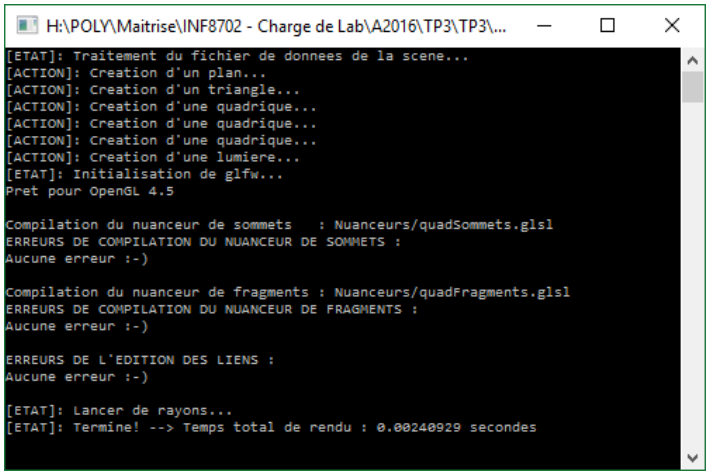

Figure 1 : Sortie initials sur la console.

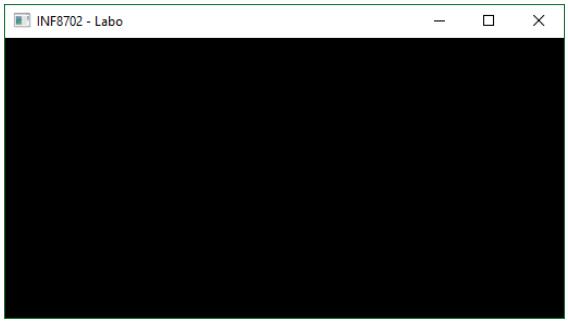

Figure 2 : Affichage initial.

2. Implémenter le lancer de rayons

Le coeur de l'engin de lancer de rayons est cette boucle principale qui s'affaire à lancer des rayons vers chaque pixel de la grille virtuelle pour en récupérer les couleurs finales. On vous demande ici de compléter la fonction suivante :

- `Scene :: LancerRayons()`

... afin d'implémenter correctement le lancer de rayons et la récupération des couleurs de chaque pixel.

Comportement observable :

Figure 3 : Affichage d'un plan après lancer de rayon.

3. Implémenter l'intersection des quadriques

Comme vous l'avez vu à la Figure 3, le programme peut déjà afficher des plans, car la fonction d'intersection CPlan::Intersection est déjà fournie. Cependant, la « scene1 » comporte d'autres objets dont des quadriques. Vous pouvez aller à ce moment-ci lire le fichier « Scenes\scene1.dat » pour consulter la liste des objets intégrés à la scène. On vous demande ici de compléter la fonction

- `CQuadrique :: Intersection()`

... afin d'implémenter correctement l'intersection rayon-quadrique.

Comportement observable :

Une fois ce requis complété, vous devriez obtenir le résultat visuel suivant :

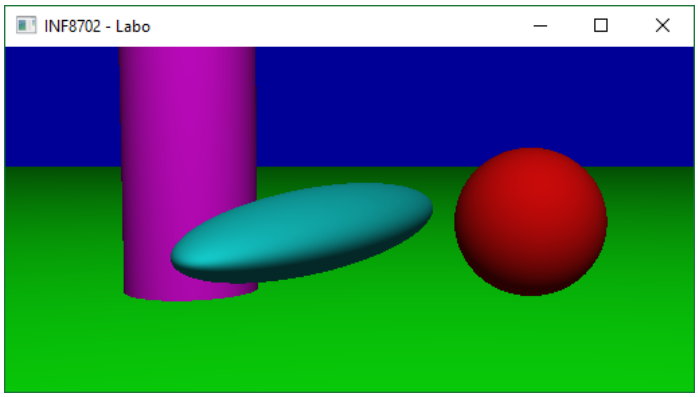

Figure 4 : Affichage de quadratiques, scene1.dat

4. Implémenter l'intersection des triangles

Les scènes 1, 2 et 3 comportent également des triangles, dont l'intersection n'est pas encore prise en charge. On vous demande ici de compléter la fonction

- `CTriangle:: Intersection()`

... afin d'implémenter correctement l'intersection rayon-triangle.

Comportement observable :

Une fois ce requis complété, vous devriez obtenir le résultat visuel suivant :

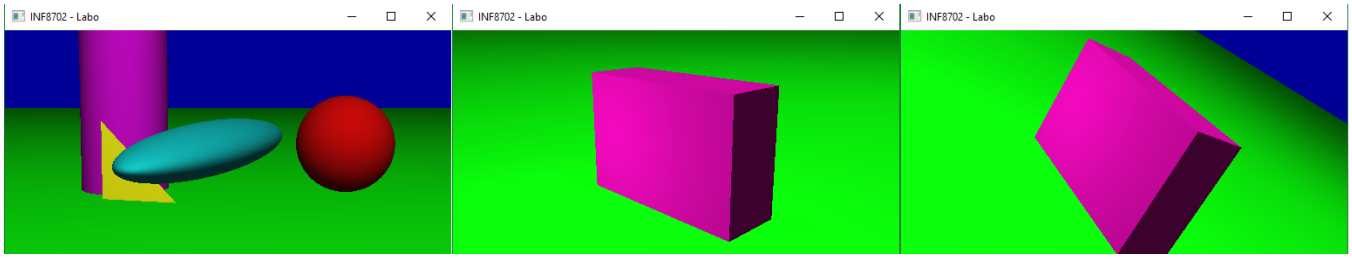

Figure 5 : Affichage de quadratiques et de triangles dans scene1.data (gauche) scene2.dat (centre), scene3.dat (droite).

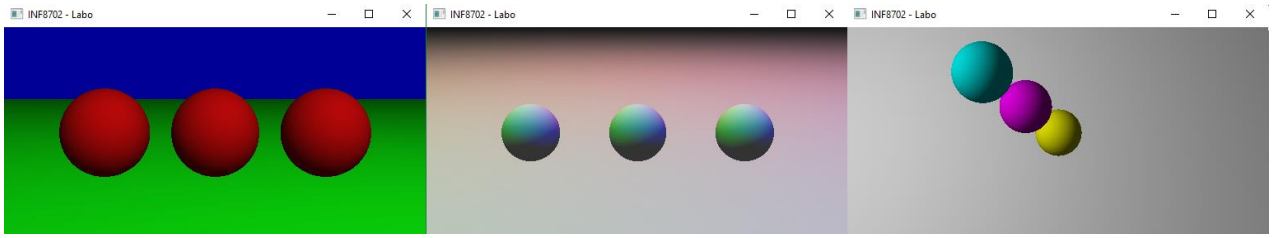

Figure 6 : Affichage de quadratiques dans scene4.data (gauche) scene5.dat (centre), scene6.dat (droite).

5. Implémentation la réflexion vectorielle

Vous l'avez vu en classe, les contributions spéculaires font appel à des vecteurs réfléchis (« vue-réfléchie »). Afin de faciliter votre travail dans ce volet, on vous demande ici de compléter la fonction utilitaire suivante :

- `CVecteur3:: Reflect()`

... afin de calculer correctement un vecteur réfléchi autour d'une normale, à partir d'un vecteur quelconque et d'une normale passés comme arguments d'entrée.

Aucun comportement observable n'est associé à ce requis. Vous devrez déboguer le fonctionnement de ce requis à l'aide des comportements observables des requis suivants utilisant la réflexion.

6. Implémenter l'éclairage spéculaire (modèle de Phong)

Il s'agit ici pour l'étudiant de modifier le code fourni afin d'implémenter un rendu spéculaire (dit de Phong) qui est fonction, pour chaque fragment, du vecteur vue, du vecteur lumière et du vecteur de vue réfléchi.

Tout d'abord, on vous demande de compléter la fonction suivante :

- `CScene:: ObtenirCouleurSurIntersection()`

... afin de calculer et d'ajouter la contribution spéculaire au point d'intersection.

<u>Comportement observable :</u>

Une fois ce requis complété, vous devriez obtenir le résultat visuel suivant :

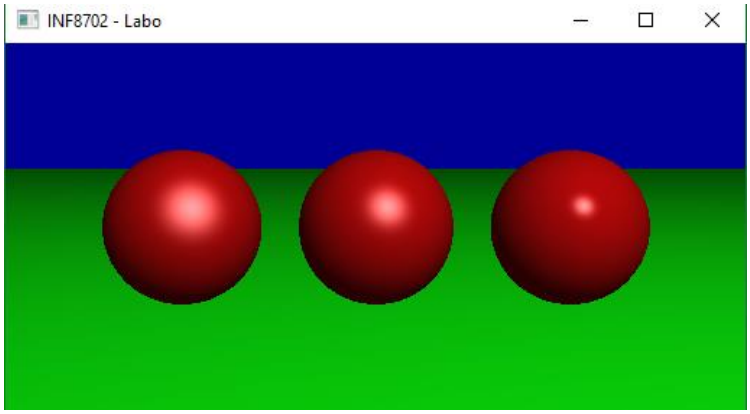

Figure 7 : Affichage avec éclairage de Phong, scene3.dat.

7. Implémenter les ombres portées

Tel que fourni, le nuanceur de fragments ne tient pas compte des ombres portées par les objets de la scène. On vous demande donc de compléter la fonction :

- `CScene:: ObtenirFiltreDeSurface()`

... afin d'implémenter les effets suivants :

- Les objets opaques projettent une ombre (Figure 8)
- Les objets translucides projettent une ombre d'intensité ajustée par rapport au coefficient de réfraction de l'objet (Figure 9),
- Les objets translucides colorés teintent la lumière blanche (Figure 10)

Comportement observable :

Une fois ce requis complété, vous devriez obtenir le résultat visuel suivant :

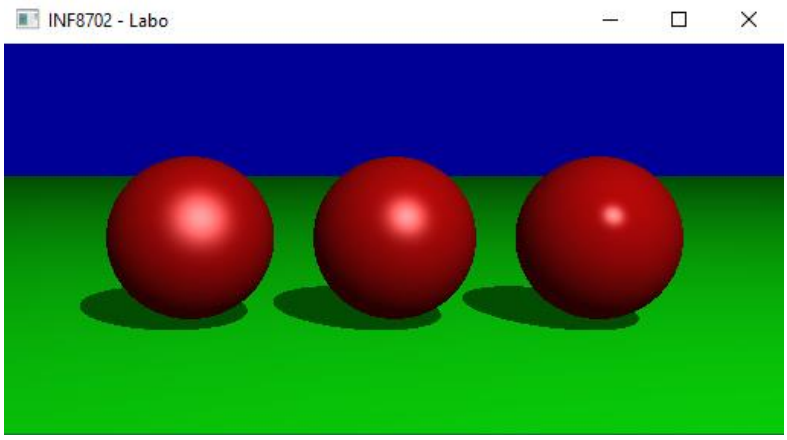

Figure 8 : scene4.dat avec ombre portées.

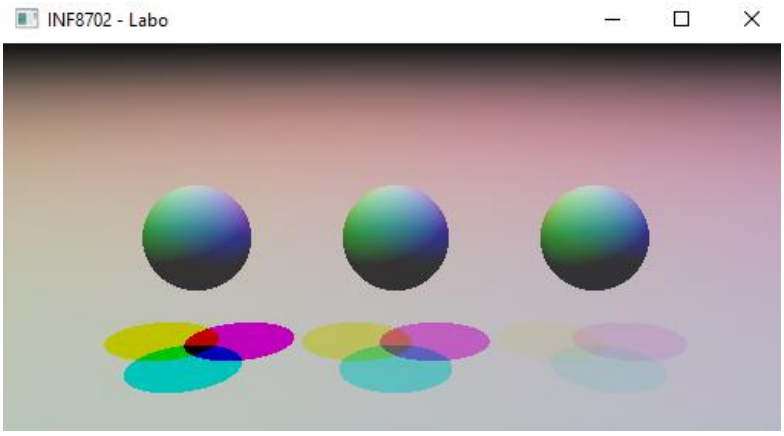

Figure 9: scene5.dat avec ombre portées.

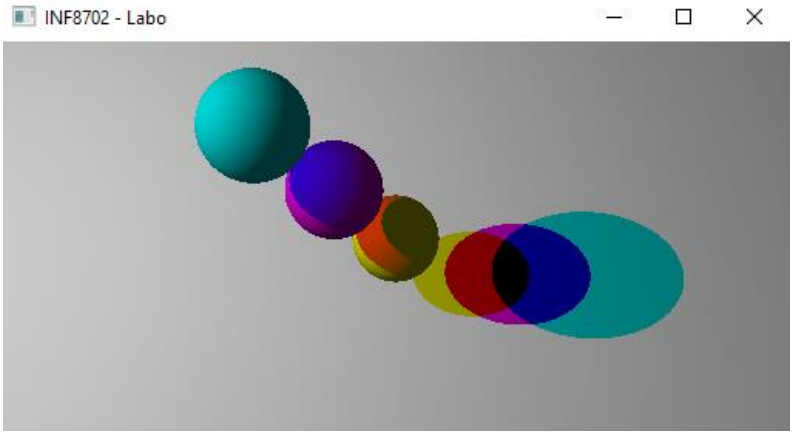

Figure 10: scene6.dat avec ombre portées.

8. Implémenter la réflexion

Il s'agit ici de modifier le code afin de permettre la réflexion des rayons primaires sur les objets réfléchissants. On vous demande de modifier la fonction :

- `CScene:: ObtenirCouleurSurIntersection()`

… afin d'appeler correctement votre fonction « reflect » que vous avez codée plus tôt, et d'être en mesure de rendre correctement les effets réflexifs dans la scène.

Comportement observable :

Une fois ce requis complété, vous devriez obtenir le résultat visuel suivant :

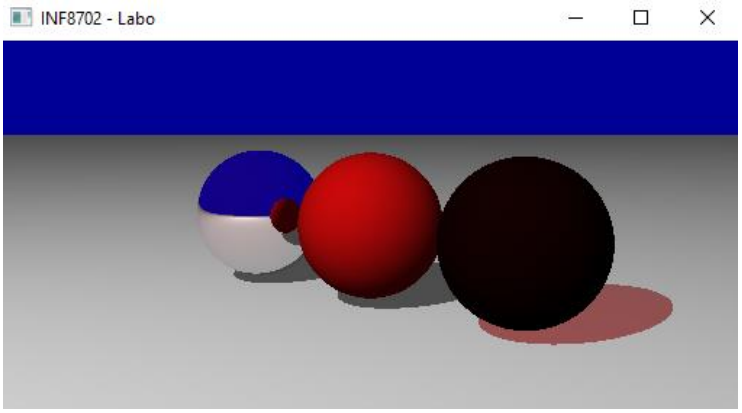

Figure 11 : scene7.data avec réflexion.

9. Implémenter la réfraction

On doit ici permettre la réfraction d'une part des rayons primaires vers l'intérieur des modèles 3D réfringents et d'autre part des rayons secondaires internes vers l'extérieur des modèles 3D réfringents. On vous demande pour ce faire de modifier la fonction :

- `CVecteur3::Refract()`

… afin d'implémenter correctement la réfraction d'un vecteur autour d'une normale, à partir d'un vecteur quelconque, d'une normale et d'un ratio d'indices de réfraction.

Pour que ce requis fonctionne, il est également nécessaire d'appeler correctement cette fonction utilitaire. On vous demande donc également de modifier la fonction :

- `CScene::ObtenirCouleurSurIntersection()`

… afin d'ajouter la réfraction au compte des effets pris en compte pour obtenir la couleur finale sur l'intersection.

Comportement observable :

Une fois les deux parties de ce requis complétées, vous devriez obtenir le résultat visuel suivant :

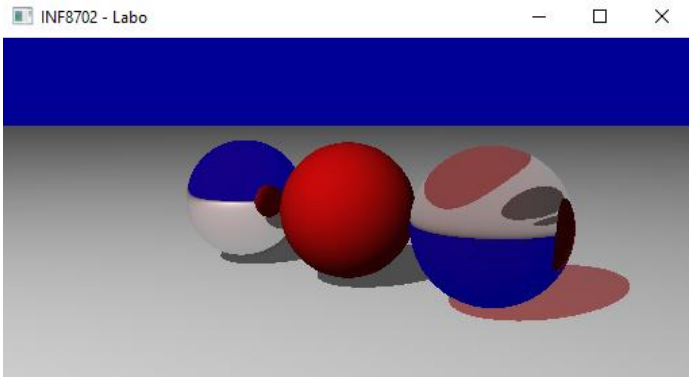

Figure 12: scene7.dat avec réfraction

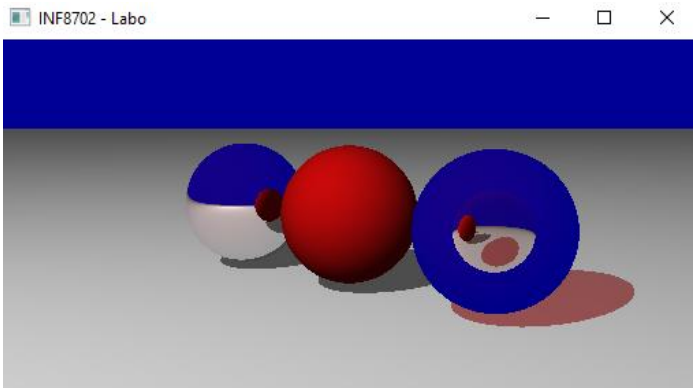

Figure 13: scene8.dat avec réfraction

10. Implémenter l'angle critique de la réfraction

Selon le deuxième volet de la loi de Snell-Descartes sur la réfraction, lorsque l'angle réfracté est éloigné de la normale de plus de 90 degrés, le phénomène se transforme est réflexion totale interne. Dans ce travail pratique, on vous demande d'implémenter cet effet dans les conditions d'angle critique décrites plus haut. On vous demande pour ce faire de modifier la fonction :

- `CVecteur3::Refract()`

… afin d'ajouter l'effet de réflexion totale interne au compte des effets visuels pris en charge par votre engin de lancer de rayons

Comportement observable :

Une fois ce requis complété, vous devriez obtenir le résultat visuel suivant :

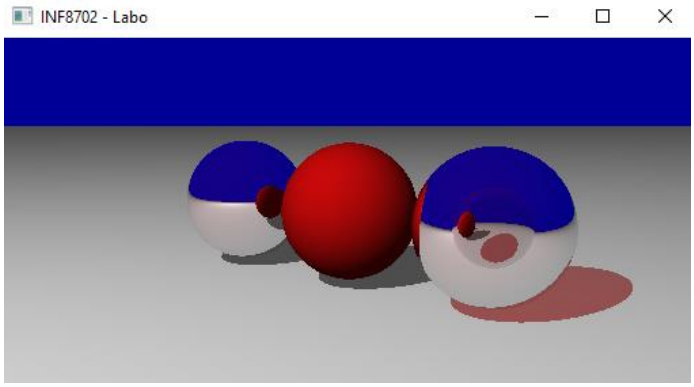

Figure 14: scene8.dat avec refraction et réflexion totale interne.

QUESITONS

1. **Dans le code, vous utilisez un coefficient de réfraction et un indice de réfraction. En quoi leurs utilités sont-elles différentes ?**
2. **Si vous vouliez implémenter un lanceur de rayon sur GPUs quelles seraient les étapes nécessaires pour y arriver ?**
3. **Une fois cela fait, comment pourriez-vous appliquer un éclairage réaliste (lanceur de rayon) sur seulement quelques parties de la scène (comme les yeux d'un personnage) ?**

BARÈME DE CORRECTION

Description des requis	Points alloués
Lancer des rayons	1.50
Intersection des quadriques 1.0	1.50
Intersection des triangles	1.00
Calcul d'un vecteur réfléchi	0.50
Éclairage spéculaire correct	1.00
Ombres portées	
Ombres projetées par objets opaques	0.50
Ombres projetées par objets translucides	0.50
Ombres colorées par objets translucides colorés	1.00
Réflexion	1.00
Réfraction	1.50
Angle critique	1.00
Questions	
Q1	0.50
Q2	1.00
Q3	0.50
Lisibilité du code	2. 00
Total	**15.00**

REMISE

- On demande une seule remise par équipe.
- **Ne remettez que les fichiers modifiés** durant le TP.
- Remettez un seul fichier .ZIP contenant :
  - Les fichiers modifiés (.cpp, .h, .glsl)
  - Un document (.txt, .pdf) contenant les réponses aux questions posées.
  - Un document facultatif (.txt, .pdf) contenant des commentaires sur un ou des aspects du TP (instructions, code fourni, bogues trouvés, etc.) dans le but de rendre les TP plus intéressants dans le futur.
- Nommez le fichier avec vos matricules et le nom du TP
  - Ex : « *MATRICULE1_MATRICULE2_TP3.zip* »

RESSOURCES UTILES

- Intersection rayon-plan :
  - http://www.groupes.polymtl.ca/inf8702/moodle/Articles_et_lectures/Lecon_03/Intersection_rayon_plan.pdf
- Intersection rayon-triangle :
  - http://www.graphics.cornell.edu/pubs/1997/MT97.pdf
- Intersection rayon-quadratique :
  - http://www.groupes.polymtl.ca/inf8702/moodle/Articles_et_lectures/Lecon_03/Intersection%20des%20quadriques.pdf
- Modéliser des quadratiques :
  - http://www.groupes.polymtl.ca/inf8702/moodle/Articles_et_lectures/Lecon_03/modelisation_des_quadriques.pdf
- Lois de Snell-Descartes pour la réfraction (Voir leçon 5):

$$\mathbf{R} = \mathbf{I} - 2(\mathbf{I}\cdot\mathbf{n})\mathbf{n}$$

$$\mathbf{Z} = \frac{n_1}{n_2}(\mathbf{I} - (\mathbf{I}\cdot\mathbf{n})\mathbf{n})$$

$$\mathbf{T} = \mathbf{Z} - \left(\sqrt{1 - |\mathbf{Z}|^2}\right)\mathbf{n}$$