

Lancer de rayon

1 Informations pratiques

Ce projet est à effectuer en binômes. Le code rendu sera écrit en C++ pour le développement. **Tout projet ne compilant pas se verra attribuer la note de 0.** Mieux vaut rendre un projet incomplet mais compilant, qu'un projet ne compilant pas. Si plusieurs binômes ont des codes trop similaires, leur note sera divisée par le nombre de binômes impliqués.

Le code devra être compilé en ligne de commande ainsi :

```
g++ -g -Wall -Wextra -o projet *.cpp `pkg-config --cflags --libs sdl2`
```

Pas de Makefile, de projet Eclipse, de projet Visual Studio, de projet Xcode ou quoi que ce soit d'autre.

Le code devra être indenté de manière uniforme. La définition des méthodes et des fonctions ne devront pas dépasser un nombre de lignes raisonnable (au plus 50 lignes)

Le code devra être commenté en utilisant la syntaxe de Doxygen.

Le rapport sera écrit en \LaTeX et inclura les choix, les problèmes techniques qui se posent et les solutions trouvées (la conception (dont un diagramme UML complet) et réalisation). Le soin apporté à la grammaire et à l'orthographe sera largement pris en compte.

Le projet sera envoyé sous forme d'archive tar.xz ayant pour nom

NOM1_NOM2_projet_rayon.tar.xz avant le Lundi 2 janvier 2023 à 23h59 à l'adresse vincent.torri@gmail.com, où *NOM1* et *NOM2* sont les noms de famille des deux binômes (pas de prénom, et en majuscule).

Le non respect du nom de l'archive sera pénalisé par 5 points en moins.

L'archive contiendra le rapport en \LaTeX et PDF, ainsi que les fichiers C++.

Et **UNIQUEMENT** : une archive contenant un fichier exécutable sera refusé par Gmail si vous utilisez cette messagerie par exemple.

Création de l'archive :

1. créer le répertoire *NOM1_NOM2_projet_rayon*,
2. mettre dans ce répertoire les fichiers source, en-tête, le fichier \LaTeX ainsi que le fichier PDF,

3. aller dans le répertoire parent,
4. exécuter les commandes suivantes :
 - (a) `tar cvf NOM1_NOM2_projet_rayon.tar NOM1_NOM2_projet_rayon`
 - (b) `xz -9 NOM1_NOM2_projet_rayon.tar`

Tout projet rendu en retard se verra attribuer la note de 0. Donc ne pas attendre le dernier moment pour l'envoyer. Un mail de confirmation sera envoyé.

2 Le tracé de rayon

L'obtention de la représentation d'une scène en 3D sur un écran peut s'obtenir de deux manières :

- La *rasterisation* : on crée un monde en 3D et on le projette sur l'écran à l'aide de formules de projection classiques. Cette technique est rapide (animation en temps réel possible) mais les effets réalistes tels les ombres doivent être simulés, d'où un rendu quelques fois peu réalistes de ce point de vue.
- Le *tracé de rayon* : on crée aussi un monde en 3D, mais au lieu de projeter le monde 3D sur l'écran, on suit le trajet d'un rayon partant de notre œil, passant par un pixel de l'écran. Il suffit alors de suivre le rayon qui "rebondit" sur chacun des objets de la scène. Les images sont très réalistes, mais le calcul est plus coûteux et donc une animation en temps réel est difficile, malgré la puissance des ordinateurs actuels. Néanmoins, en disposant, soit de Windows 10 ou plus, soit un autre OS comme linux, les API Direct3D 12 et Vulkan (resp.), ainsi qu'avec une carte graphique adaptée, le lancé de rayon commence à devenir possible en temps réel.

On peut voir sur la Figure 1 (issue de Wikipedia) le principe du tracé de rayon.

L'algorithme général du tracé de rayon est le suivant :

1. Construire la scène avec les objets, la source de lumière et la caméra (*i.e.* l'œil).
2. Calculer l'image.
3. Sauver l'image.

Un algorithme basique (qui devra être amélioré) pour le calcul de l'image peut être le suivant :

1. Déterminer la grille correspondant à l'image que l'on désire sauver

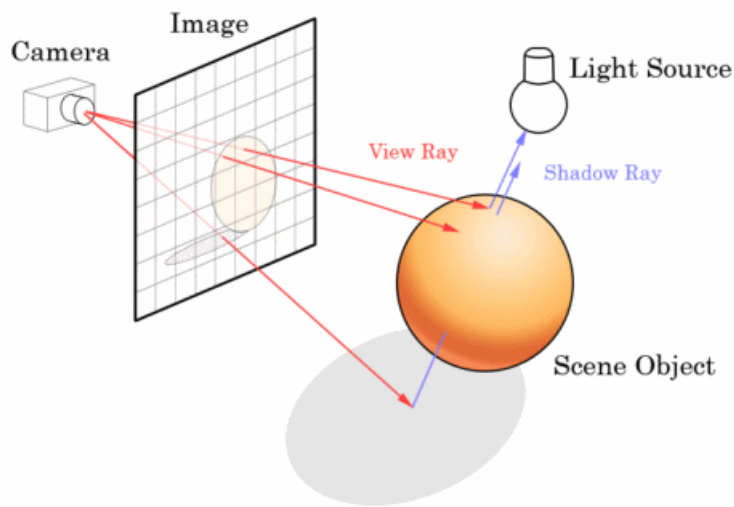


FIGURE 1 – Principe du lancé de rayon

2. Pour chaque point de la grille :
 - (a) Déterminer l'équation de la ligne qui passe par l'œil et le point de la grille en question.
 - (b) Pour chaque objet de la scène :
 - i. Déterminer si la ligne intersecte l'objet.
 - ii. S'il y a intersection et si ce point est le plus proche de l'œil, on sauve ce point. L'intensité de la lumière sur le point d'intersection se fera grâce à la position de source de lumière.
 - iii. Si l'objet possède un indice de réflexion, on continue de suivre le rayon avec la ligne réfléchi.
 - (c) On calcule la couleur et on l'assigne au point de la grille considéré.

La Figure 2 page 4 présente un diagramme de classe simplifié.

On présente les diverses classes :

- **Vector3f** : c'est un vecteur 3D de float.
- **Ray3f** : un rayon avec une origine et une direction
- **Camera** : la caméra ou œil d'où on regarde la scène.
- **Material** : la couleur (uniforme) et un coefficient de luminosité entre 0 et 1 (0 signifie pas de réflexion (objet mat) et 1 signifie que le rayon est entièrement réfléchi (un miroir)).
- **Shape** : classe abstraite. La méthode *is_hit* teste si le rayon intersecte l'objet et la méthode *reflect* renvoie le rayon réfléchi.
- **Cube/Quad** : un cube ou un rectangle, défini par une origine (le centre) et la taille.

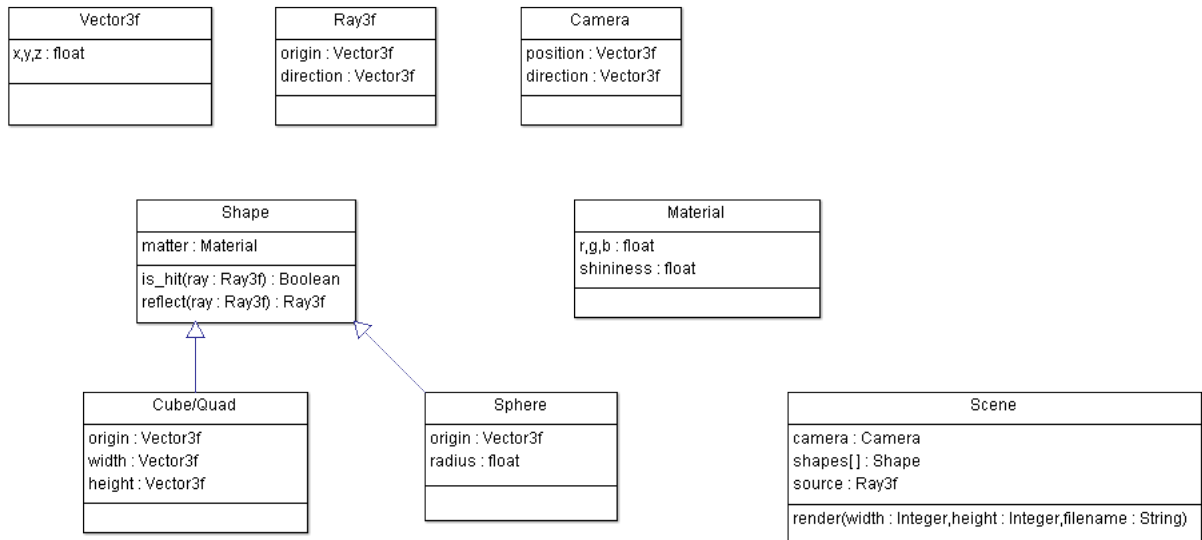


FIGURE 2 – Diagramme de classes

- **Sphere** : une sphère définie par une origine et un rayon.
- **Scene** : la scène qui comprend la caméra et les objets et la source de lumière. La méthode *render* définit la taille de la grille (donc de l'image) ainsi que le nom du fichier dans lequel on sauve l'image.

3 Sujet

La scène sera composée de 5 quadrilatères définissant une boîte, la caméra se trouvant au niveau du côté manquant (voir Figure 3).

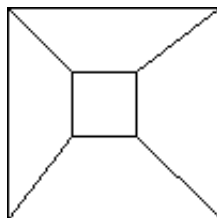


FIGURE 3 – Scene vide

On placera la source de lumière au niveau du quadrilatère supérieur, ainsi qu'une sphère et un cube.

Ecrire un programme en C++ correctement structuré, sans l'aide d'une quelconque bibliothèque extérieure au C++ qui affichera dans une fenêtre la scène calculée.

Pour afficher dans une fenêtre les caractères du travail ci-dessus (une fenêtre pour chacune de ces polices), on utilisera la bibliothèque SDL, dont l'utilisation sera faite grâce à une classe `Sdl` facilitant son usage.