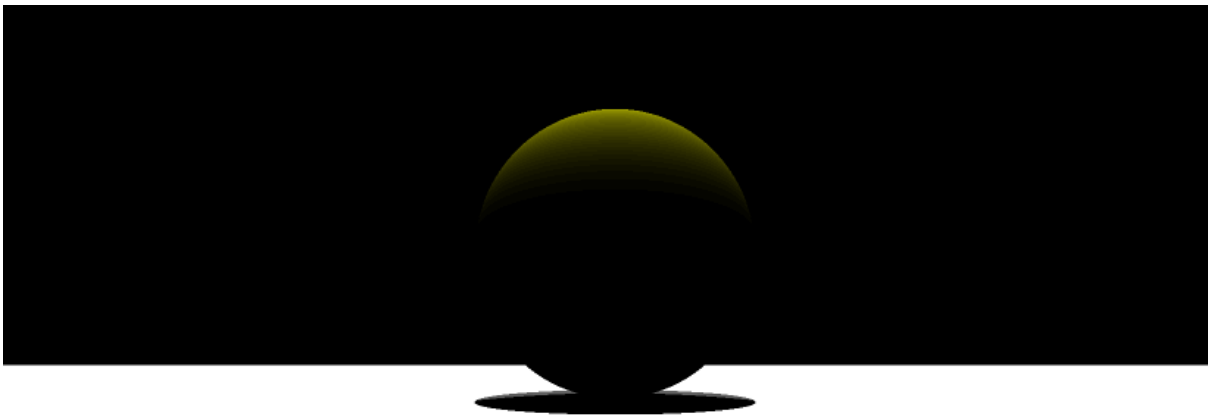
Raytracer

Documentation technique



Réalisé par :

- Axel Fradet **axel.fradet@epitech.eu**
- Mathieu Robert **mathieu1.robert@epitech.eu**
- Kylian Tranchet **kylian.tranchet@epitech.eu**
- Théophile Jérôme-Rocher **theophile.jerome-rocher@epitech.eu**
    - **Promotion Epitech Nantes PGE 2027. 2e année.**

Liens utiles (technique):

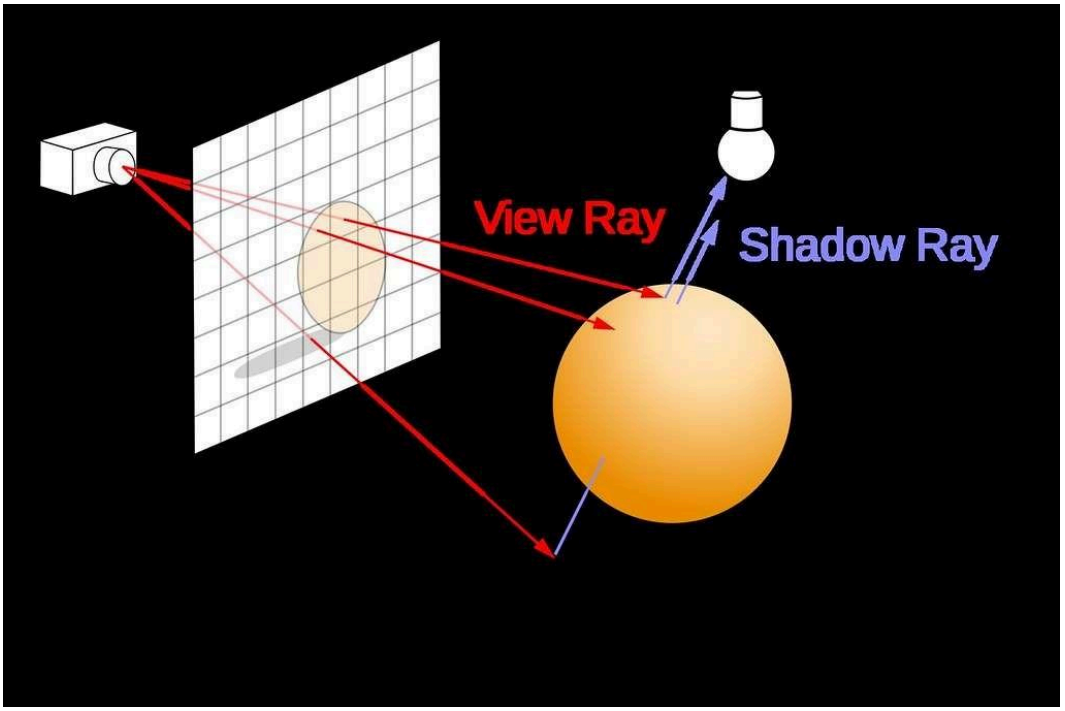
- Github : https://github.com/Njord201/Raytracer
- Github Project : https://github.com/users/Njord201/projects/7
- Norme de commits : https://www.conventionalcommits.org/en/v1.0.0/
- Algorithme Octree Documentation: https://www.geeksforgeeks.org/octree-insertion-and-searching/

Exécution:

Le makefile à la racine propose make, make re, make fclean, make clean, make doc.

Point technique:

Pour ce raytracer, nous avons tout simplement un rayon (origine + vecteur) émit par une caméra, passant par un “rectangle”, et en fonction de si ce rayon intersecte avec une primitive, nous mettons de la couleur sur le rectangle. Tout simplement des calculs de vecteurs, d’intersection...



Point technique sur l’algorithme d’optimisation:

Afin d’optimiser la génération de l’image nous avons dû mettre en place un algorithme. En l'occurrence, l’algorithme “Octree” (cf. liens utiles). Nous disposons également d'un anticrénelage par suréchantillonnage, “anti-aliasing”, pour obtenir des rendus plus lisses.

Technologies:

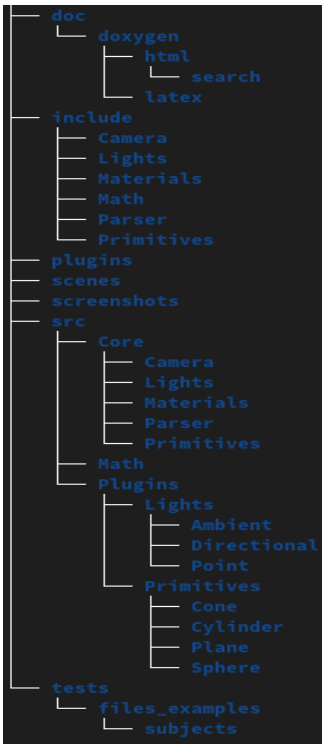
- C++ (Langage), Make (Outil), SDL2 (Interface graphique)
- Pour la documentation: Doxygen, LaTeX, GraphViz
    - Exécutez “**sudo yum install doxygen doxygen-latex graphviz"**

Contraintes développeur:

- Suivre la norme de commit (Conventional Commits).
- Suivre la norme pour les variables, noms de fichiers... --> Camel case.
- Ouvrir une issue, créer une branche depuis celle-ci, et push sur main uniquement à travers des pull requests. Minimum 2 reviews approved.
- Faire du code rigoureux (norme / propreté).
- Suivre l’architecture actuelle du projet .
- Utiliser et mettre à jour le github project.

Petit point sur la structure:

- Les primitives, les lumières... doivent être des “plugins”, donc des .so ou shared object, qui sont par la suite ré-injectés en les chargeant. Nous possédons donc un Makefile compilant le Core et les plugins en deux étapes, eux-mêmes possédant leurs propres Makefile (makefile dans Plugins/ ou Core/)...



include/ tous les fichiers hpp.

src/ tous les fichiers .cpp.

scenes/ contient les fichiers .cfg pour les scènes.

screenshots/ des images proposées par les développeurs de ce Raytracer.

plugins/ les .so des plugins compilés.

tests/ accueil les tests.

doc/ la documentation.

Créer une nouvelle primitive (développeurs):

- Créer une nouvelle primitive dans include/Primitives, suivant la logique des primitives déjà existantes, basées sur l'interface IPrimitive.
- De même pour le code C++ dans src/Plugins/Primitives.
- Pensez à bien mettre à jour le Makefile de Plugins avec les bonnes références.

Accéder à la documentation technique:

Si vous souhaitez plus d'informations sur les classes et différentes structures utilisées dans notre programme. Vous avez accès à **refman.pdf** dans **/doc**. Vous pouvez exécuter "**make doc"** pour re-générer la documentation, ce pdf et également /doc/doxygen/html qui contiendra par la suite un site que vous pourrez lancer afin d'accéder à la documentation doxygen du Raytracer, proposant des graphiques.