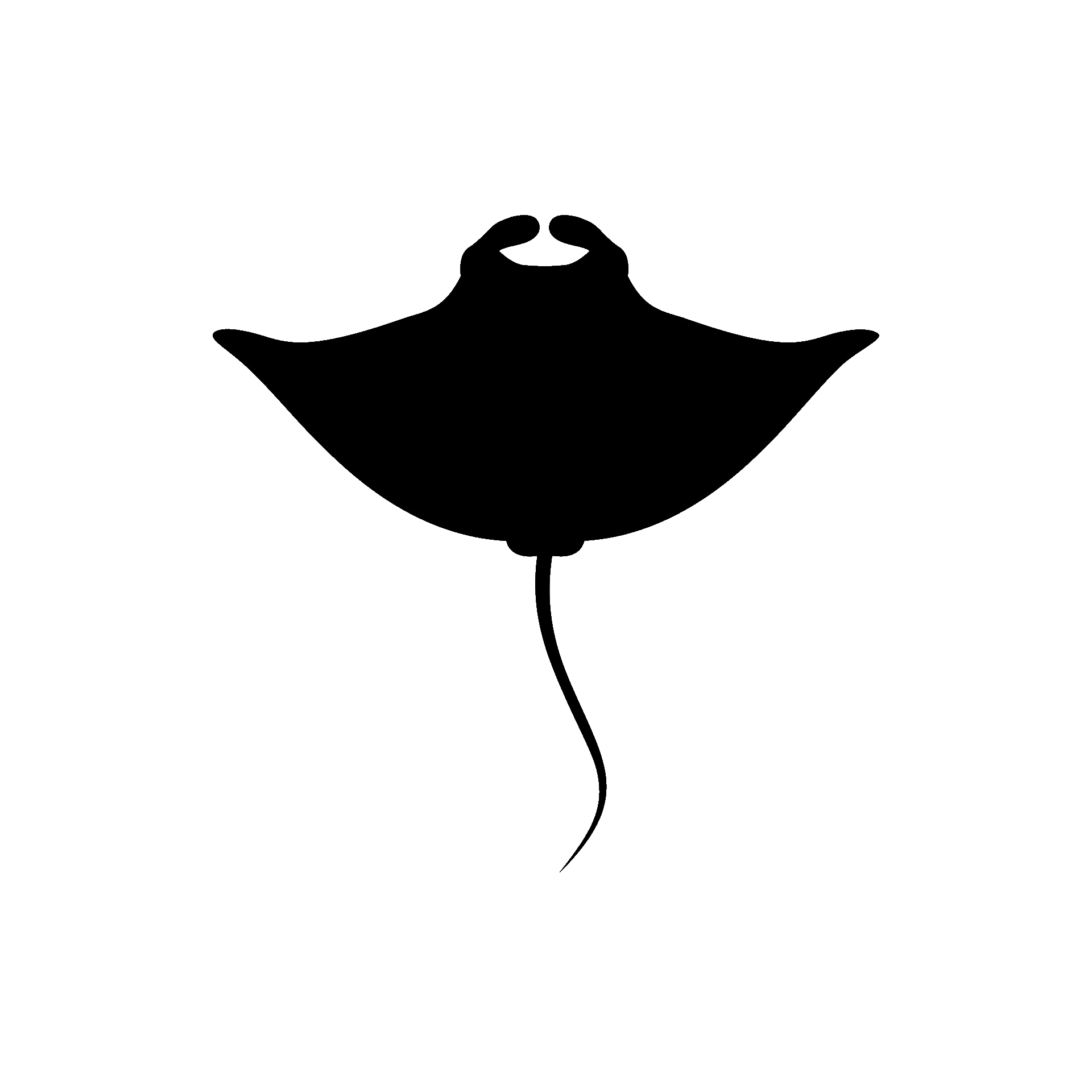
Projet Ray-Tracer

Description des fonctionnalités

#### Par Laurent Louis Lauret



Ray

tracer

Table des matières

[Général 3](#_Toc130895599)

[Fonctionnalités 4](#_Toc130895600)

[Importation de la scène 4](#_Toc130895601)

[Gestion des erreurs dans le fichier de description S.1(Sprint 1) 4](#_Toc130895602)

[Modélisation de la scène S.2 (sprint 1) 4](#_Toc130895603)

[Calcul du rendu 5](#_Toc130895604)

[Modélisation du champ de vision R.1 (Sprint 1) 5](#_Toc130895605)

[Lancer de rayon R.2 (Sprint 1) 5](#_Toc130895606)

[Calcul d’intersection R.3 (Sprint 1) 6](#_Toc130895607)

[Calcul de l’illumination R.4 (Sprint 1 & 2) 6](#_Toc130895608)

[Phong Shading R.4.1 (Sprint 1) 6](#_Toc130895609)

[Réflexion R.4.2 (Sprint 2) 7](#_Toc130895610)

[Transparence R.4.3 (Sprint 2) 7](#_Toc130895611)

[Occlusion ambiante R.4.4 (Sprint 2) 8](#_Toc130895612)

[Multi-threading/Multi-processing R.4.5 (Sprint 2) 8](#_Toc130895613)

[Création du fichier de sortie 9](#_Toc130895614)

[Création du fichier au format BMP F.1 (Sprint 1) 9](#_Toc130895615)

[Aperçu à l’écran F.2 (Sprint 2) 9](#_Toc130895616)

# Général

Ce projet consiste à créer un moteur de rendu de synthèse d’image, en utilisant la technique du Ray-Casting, méthode qui consiste à suivre le chemin inverse de la lumière (de l’objectif vers la source lumineuse) pour pouvoir déterminer la couleur des pixels de l’image. L’application sera utilisée en ligne de commande, et les arguments on retrouve le fichier de description de la scène, ainsi que le nom de l’image souhaitée.

Ex : **raytracer.py** *scene.json* image1.bmp

De manière générale, ce projet nécessite des connaissances :

* Mathématiques telles que les opérations sur les vecteurs (somme, produit vectoriel et scalaire), ainsi que la trigonométrie de base.
* Physiques : avoir des connaissances sur le transport de la lumière dans un milieu homogène, ainsi que lorsqu’il y a un changement de milieu.

À plus long terme, et aux fins d’optimisations des connaissances supplémentaires sont nécessaires :

* Connaissance sur les structures des données pour diminuer le nombre d’intersections requis (segmenter la scène)
* Le multithreading pour diminuer le temps nécessaire pour terminer le rendu, étant donné la quantité de calcul nécessaire pour une résolution acceptable.
* Utiliser des patrons de conception pour pouvoir ajouter des fonctionnalités plus facilement.

# Fonctionnalités

## Importation de la scène

### Gestion des erreurs dans le fichier de description S.1(Sprint 1)

*Priorité : basse*

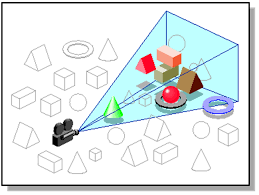
Cette fonctionnalité indiquera à l’utilisateur si le fichier JSON de description de scène contient des erreurs telles que les valeurs inattendues, ou si le fichier est mal formé (clés manquantes). En cas d’erreurs, le programme se termine et indique l’erreur grâce à un message.

Modélisation de la scène S.2 (sprint 1)

*Priorité : haute*

Cette fonctionnalité utilise les informations contenues dans le fichier de description de scène pour configurer la scène. Par exemple, déterminer :

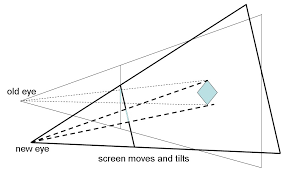
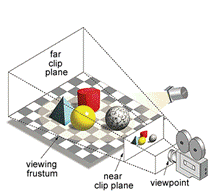
* La résolution souhaitée pour le rendu de la scène
* La position de la caméra ainsi que son orientation
* Les géométries qui y figurent, leur positionnement, leur couleur, leur matériel.
* La qualité souhaitée pour le rendu.



## Calcul du rendu

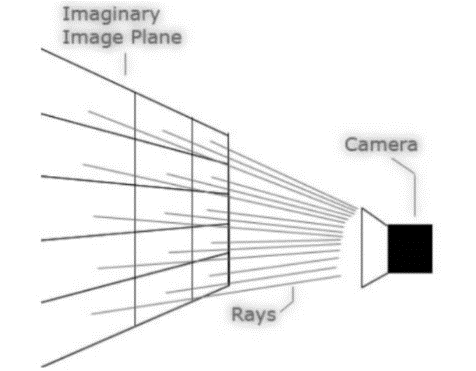
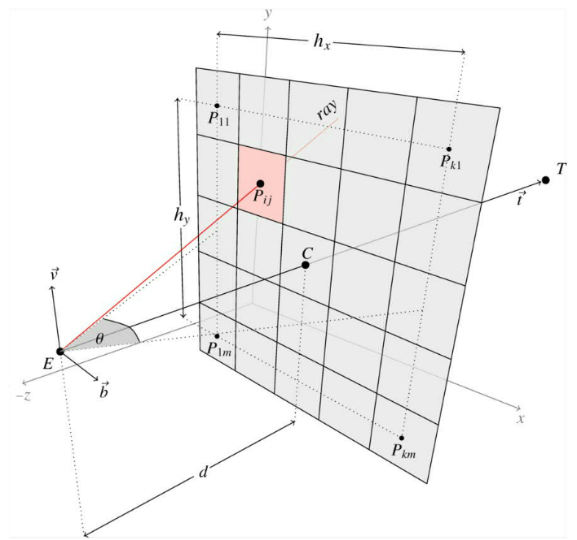
### Modélisation du champ de vision R.1 (Sprint 1)

*Priorité : élevée*

Cette fonctionnalité est le point de départ du rendu. Elle permet de déterminer le plan sur lequel l’image va se projeter. En prenant compte les paramètre de la caméra et de la résolution

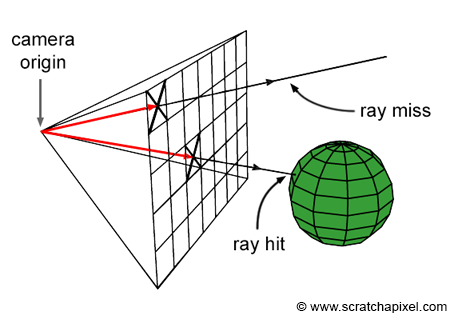
### Lancer de rayon R.2 (Sprint 1)

*Priorité : élevée – Prérequis : Modélisation du champ de vision*

Le lancer de rayon permet de déterminer la direction des de chacun des rayons à travers le champ de vision, en se basant sur la position de la caméra et des coordonnés des points sur le plan de projection.

### Calcul d’intersection R.3 (Sprint 1)

*Priorité : Haute*

Cette fonctionnalité permet d’interpréter un lancer de rayon, à savoir quel sont les paramètres de l’intersection, par exemple, quelles géométries ont été touchées par le rayon, le type de surface, la couleur, etc…

*Point : {5,20,20}*

*Matériel : brillante*

*Normale :…*

### Calcul de l’illumination R.4 (Sprint 1 & 2)

Priorité : haute – prérequis : calcul d’intersection

### Phong Shading R.4.1 (Sprint 1)

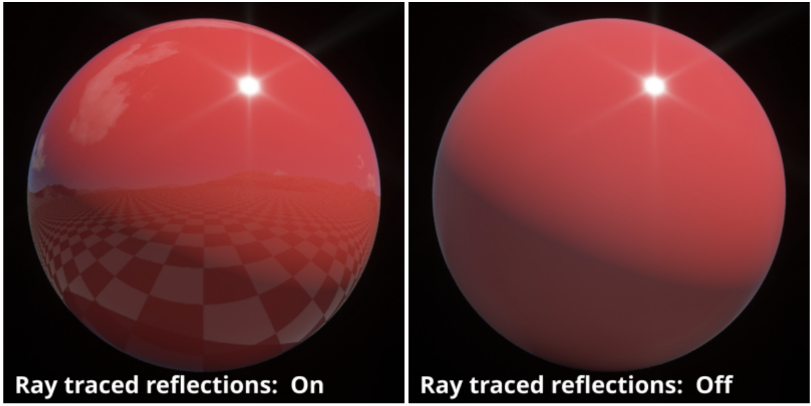
*Priorité : haute*

Cette fonctionnalité permet de déterminer la couleur d’un pixel associé à un rayon en se basant les données d’intersections ainsi que la configuration de la scène. Le modèle de Phong permet de déterminer directement la couleur vue de l’objectif en associant 3 types de luminosité :



### Réflexion R.4.2 (Sprint 2)

*Priorité : Haute – prérequis : Phong Shading*

Cette fonctionnalité permet d’avoir un rendu plus avancé en tenant compte du taux de réflectivité d’une surface.

### Transparence R.4.3 (Sprint 2)

*Priorité : Moyenne – prérequis : le multithread*

Cette fonctionnalité permet d’avoir un rendu plus réaliste et plus varié en mettant en scènes des géométries qui ne serait pas totalement opaques. Cependant, celle-ci implique également de comprendre la réfraction et les changements de milieu. Aussi, elle nécessite plus de calculs que le Phong Shading, en augmentant le nombre de rayons secondaires

Une image contenant graphique

Description générée automatiquement

Figure : transparence sans réfraction

Figure : transparence et réfraction

### Occlusion ambiante R.4.4 (Sprint 2)

*Priorité : Moyenne – prérequis : le multithread*

Cette fonctionnalité permet d’avoir un rendu plus réaliste des coins de la scène ou encore de l’intersection de deux géométries. En outre, elle donne plus de cohérence au rendu, et facilite l’interprétation de la scène. En contrepartie, cette fonctionnalité nécessite aussi de relancer des rayons ce qui augmente le nombre de calculs à effectuer.

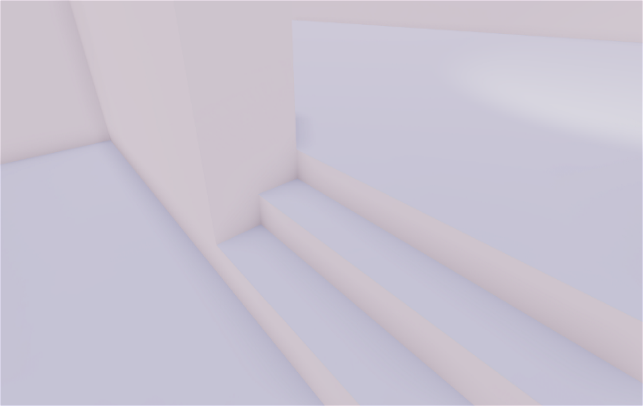


Figure : exemple d'occlusion ambiante

### Multi-threading/Multi-processing R.4.5 (Sprint 2)

*Priorité : Haute*

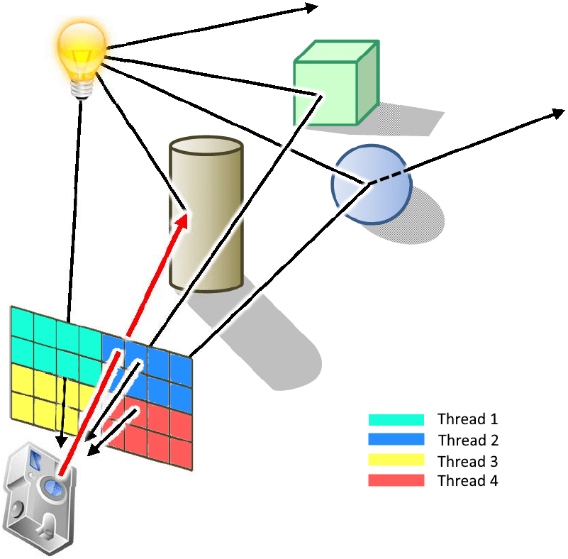
Cette fonctionnalité permet d’avoir un rendu plus rapide en utilisant un traitement parallèle, bénéficiant de la multitude de cœurs disponibles dans les processeurs. En répartissant le traitement des rayons en 4 parties parallèles, on pourrait balayer l’ensemble des pixels en un temps réduit. Cependant, en utilisant Python, le **multithread** pourrait être un risque à cause du *Global Interpreter Lock* (méchanisme de sécurité). On serait alors obligé d’utiliser le **multiprocessing**, un autre type de traitement parallèle.

Figure : exemple de division pour le parallélisme

## Création du fichier de sortie

### Création du fichier au format BMP F.1 (Sprint 1)

*Priorité : Basse*

Cette fonctionnalité permet d’exporter le rendu vers un fichier au format Bitmap, dont le nom est passé en paramètre au programme. Avec Python, son implémentation devrait être assez simple.

### Aperçu à l’écran F.2 (Sprint 2)

*Priorité : Moyenne*

Cette fonctionnalité permet d’afficher directement l’image une fois le rendu traité. Aucun fichier n’est sauvegardé. Cela peut être pratique pour avoir un aperçu visuel de la scène et aux fins de tests. L’implémentation de cette fonctionnalité dans Python devrait est aussi simpliste que l’exportation vers un fichier.