



INRAE

cirad



UNIVERSITÉ DE MONTPELLIER  
FACULTÉ DES SCIENCES

MASTER 2 INFORMATIQUE  
Parcours IMAGINE

---

## La simulation d'interception de la lumière par Photon Mapping

Rapport de stage

---

effectué au CIRAD  
du 26/02 au 22/08 2024  
par Tuan-Minh NGUYEN

Tuteur en entreprise : M. Thomas ARSOUZE,  
M. Frédéric BOUDON, Mme. Jessica BERTHELOOT  
Tuteur à l'Université : Mme. Noura FARAJ

# Table des matières

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Introduction . . . . .	3
1.2	Présentation des organismes d'accueil . . . . .	3
1.2.1	CIRAD . . . . .	3
L'équipe Phenomen	. . . . .	3
UMR AMAP . . . . .		4
1.3	Problématiques de recherche . . . . .	4
1.3.1	Présentation du projet Physioscope . . . . .	4
1.3.2	Problématiques . . . . .	5
1.3.3	Présentation de la mission . . . . .	5
1.4	Contributions durant le stage . . . . .	6
<b>2</b>	<b>Environnement technique</b>	<b>7</b>
2.1	L-py . . . . .	7
2.2	SEC2 . . . . .	7
2.3	Embree . . . . .	8
2.4	La base du moteur de Photon Mapping . . . . .	8
2.5	Mitsuba 3 . . . . .	8
<b>3</b>	<b>Travaux effectués</b>	<b>9</b>
3.1	Comparaison entre le moteur actuel et le moteur SEC2 . . . . .	9
3.1.1	Définir les propriétés de simulation . . . . .	9
La géométrie de chambre de culture . . . . .		9
Les propriétés optiques de chambre de culture . . . . .		12
Configuration des capteurs . . . . .		13
3.1.2	Résultats . . . . .	14
3.2	Amélioration du moteur de simulation . . . . .	15
3.2.1	Correction du modèle d'illumination . . . . .	15
Le modèle de BRDF de Phong . . . . .		16
Russian Roulette . . . . .		18
Résultat . . . . .		18
3.2.2	Corrections des calculs des valeurs Diffuse et Spéculaire . . . . .	20
Résultat . . . . .		21
3.2.3	Optimiser la distance minimale d'intersection . . . . .	22
Résultat . . . . .		23
3.2.4	Ignorer la face arrière . . . . .	24

## TABLE DES MATIÈRES

---

Résultat . . . . .	25
3.2.5 Modification de géométrie de capteurs . . . . .	26
Résultat . . . . .	26
3.3 Amélioration de la performance . . . . .	28
3.3.1 Optimiser le moteur actuel . . . . .	28
Optimiser le code pour les threads . . . . .	28
Optimiser le générateur de valeurs aléatoires . . . . .	29
Configurer le nombre de threads . . . . .	30
3.3.2 Exploration des outils de calcul parallèle sur GPU . . . . .	30
Embree SYCL . . . . .	30
Mitsuba3 . . . . .	31
3.4 Réaliser la simulation sur une plante . . . . .	32
Résultat . . . . .	32
<b>4 Conclusion</b>	<b>34</b>
<b>A Glossaire</b>	<b>35</b>
<b>B Annexes</b>	<b>36</b>
<b>Bibliographie</b>	<b>37</b>

# Chapitre 1

## Introduction

### 1.1 Introduction

Ce stage de second semestre a eu lieu du 23<sup>er</sup> février 2024 au 22<sup>er</sup> août 2023 dans le cadre du Master 2, parcours IMAGINE. Il a eu lieu au CIRAD au sein de l'institut AGAP dans l'équipe Phenomen et au sein de l'institut AMAP.

Ce stage est la continuation du stage de Aurélien BESNIER effectué avec la même équipe en 2023. L'objectif est d'avoir un outil de simulation de l'interception de la lumière pour les plantes précis et rapide. Pour ce faire, il faut modifier le code existant et implémenter des nouvelles techniques aux outils développé par Aurélien. Les résultats sont évalué par les comparaisons entre nos outils et les anciens outils utilisé par des biologistes.

### 1.2 Présentation des organismes d'accueil

#### 1.2.1 CIRAD

Le Centre de coopération Internationale en Recherche Agronomique pour le Développement (CIRAD) est l'organisme français de recherche agronomique et de coopération internationale pour le développement durable des régions tropicales et méditerranéennes.

Il contribue, à différentes échelles, à la protection de la biodiversité, aux transitions agroécologiques, à la durabilité des systèmes alimentaires durables, à la santé (des plantes, des animaux et des écosystèmes), au développement durable des territoires ruraux et à leur résistance face au changement climatique.

#### L'équipe Phenomen

L'équipe PhenoMEn (Phénotypage et Modélisation des plantes dans leur Environnement agro-climatique), fait partie de l'unité mixte de recherche (UMR) AGAP Institut (Amélioration Génétique et Adaptation des Plantes méditerranéennes et tropicales) du Cirad qui se concentre sur les facteurs du développement des plantes et leur adaptation aux contraintes environnementales.

L'équipe PhenoMen est l'une des équipes du pôle "Développement et fonctionnement des plantes et des peuplements" de l'institut AGAP, tentant d'identifier et de hiérarchiser les traits et les processus associés expliquant la variabilité des performances des plantes à différents niveaux organisationnels. L'équipe est divisée en axes, collaborant sur ce thème commun. Ces trois axes sont nommés :

- Plant plasticity and ideotype : il s'agit de comprendre les mécanismes d'adaptation des plantes, à des évènements stressants isolés ou récurrents.
- Plant interaction and cropping systems : l'objectif est de valoriser la diversité génétique et les interactions biologiques au sein des agrosystèmes, en analysant différentes options des systèmes de cultures.
- Plant and crop modeling : le but est de contribuer au développement de standards de représentations de données biologiques hétérogènes et de méthodes d'analyses et de simulation de ces données. Je fais partie de cette équipe.

## UMR AMAP

L'UMR AMAP (botAnique et Modélisation de l'Architecture des Plantes et des végétations) est une unité interdisciplinaire qui travaille à l'acquisition de connaissances fondamentales sur les plantes et les végétations dans le but de prévoir la réponse des écosystèmes aux forçages environnementaux, en termes de distribution/conservation des espèces et de la biodiversité, production des cultures agronomiques, stockage du carbone dans la biomasse végétale, protection de l'environnement et des services écosystémiques.

### 1.3 Problématiques de recherche

#### 1.3.1 Présentation du projet Physioscope

Le maintien de la performance des plantes dans des conditions de plus en plus stressantes nécessite de comprendre le réseau physiologique complexe impliqué dans le contrôle du développement de la plante par l'environnement. Du fait de la complexité des mécanismes, l'expérimentation seule n'est pas suffisante, et un outil informatique intégrant un modèle explicitant le fonctionnement physiologique de la plante pourrait s'avérer très pertinent pour aider à l'analyse des expérimentations biologiques. En effet, l'idée est d'intégrer dans le modèle à la fois les connaissances et des hypothèses sur le fonctionnement physiologique, et tester la composante hypothétique en comparant le comportement de la plante *in silico* et le comportement réel. Le projet Physioscope vise à développer un tel outil et faire la preuve de concept de son efficacité pour comprendre une problématique biologique spécifique : le contrôle du débourrement des bourgeons par la lumière chez le rosier qui donne naissance à de nouveaux axes sur la plante.

Le projet est subdivisé en 3 axes : (1) Un axe expérimental visant à recueillir des données physiologiques et morphologiques sous différentes conditions expérimentales, (2) un axe "modélisation" visant à créer un modèle intégrant les connaissances sur le fonctionnement

physiologique de la plante, (3) un axe informatique visant à développer un outil intuitif, facilement accessible et manipulable par un biologiste. Ce dernier, dans lequel s'intègre mon stage, doit pouvoir permettre à l'utilisateur de réaliser facilement des expériences *in silico* et d'autre part d'évaluer les conséquences de ces expérimentations sur la physiologie de la plante et son développement.

Pour cela, L-Py [8], un logiciel qui permet de modéliser le fonctionnement de la plante en interaction avec son développement, est couplé à Morphonet[10], un navigateur interactif de jeux de données segmentés 3D et 3D+t qui permet de visualiser des données en 3D, de les explorer et de les enrichir. Sa visée générique et la réponse qu'il apporte à deux verrous scientifiques communs à différentes équipes (interaction fluide avec la plante virtuelle, intégration de processus physiologiques à l'échelle plante) est un levier pour des partenariats internationaux et nationaux, et met en place un continuum entre physiologistes, écophysiologistes, et modélisateurs au sein du métaprogramme Digit-BIO de l'INRAE.

### 1.3.2 Problématiques

Un des principaux objectifs du projet Physioscope est de rendre accessibles des outils de simulation aux biologistes non avertis en termes de programmation informatique, ou aux modélisateurs qui n'ont pas développé le code qui les intéresse. Autrement dit, les utilisateurs peuvent réaliser des simulations concernant le développement des plantes sur la plateforme en ligne MorphoNet. La simulation d'interception de la lumière est l'une des simulations que les biologistes souhaitent vraiment effectuer sur cette plateforme.

Auparavant, pour réaliser cette simulation, ils utilisaient des outils développés dans un ancien projet intitulé SEC2[5]. Bien que les résultats de cet outil soient très bons, le temps de simulation est trop long, environ 6 heures pour terminer une simulation de  $10^{10}$  photons. De plus, cet outil ne supporte pas les simulations multispectrales, obligeant ainsi les biologistes à calculer manuellement les propriétés optiques de chaque matériau pour effectuer la simulation dans une bande de longueur d'onde. Tous ces processus leur prennent environ une semaine pour compléter une simulation.

Pour toutes ces raisons, nous avons décidé de développer un nouvel outil pour réaliser la simulation d'interception de la lumière. Ensuite, nous allons intégrer cet outil sur notre plateforme en ligne MorphoNet.

### 1.3.3 Présentation de la mission

Lors du stage d'Aurélien BENSIER en 2023, il a développé un outil de simulation d'interception de la lumière par Photon Mapping avec la bibliothèque de lancer de rayons Embree 4.0[11]. La performance de ce moteur est très intéressante : le temps de simulation a été réduit de manière spectaculaire, environ deux fois par rapport au moteur SEC2. Cependant, les résultats de la simulation ne sont pas corrects. Il existe encore une grande différence en comparaison avec les résultats obtenus avec le moteur SEC2.

Le but de mon stage est de finaliser et d'optimiser cet outil de simulation pour obtenir des résultats plus précis et plus rapides. De plus, j'ai dû assurer une communication avec les biologistes pour définir les caractéristiques intéressantes à développer pour eux.

## 1.4 Contributions durant le stage

Durant le stage, j'ai réussi à améliorer la précision de notre moteur. Maintenant, le résultat obtenu par notre moteur est très proche de celui obtenu par le moteur SEC2. De plus, en optimisant l'utilisation des threads sécurisés, j'ai amélioré les performances du moteur. En effet, le temps de simulation est réduit de manière significative, rendant notre moteur environ 7 fois plus rapide que le moteur SEC2. Une grande partie de mon travail s'est focalisée sur ces tâches.

D'autre part, j'ai également exploré l'utilisation du moteur de rendu Mitsuba et de la bibliothèque Embree SYCL pour les calculs parallèles sur GPU. Comme la bibliothèque Embree SYCL n'est compatible qu'avec les nouvelles cartes graphiques d'Intel, j'ai décidé d'utiliser Mitsuba. Durant la dernière période du stage, je vais essayer de réimplémenter la simulation sur le moteur Mitsuba pour évaluer les performances sur GPU.

Le code de ce projet est disponible sur ce lien de github ([https://github.com/minhlucky9/photon\\_mapping/tree/main](https://github.com/minhlucky9/photon_mapping/tree/main))

## Chapitre 2

# Environnement technique

### 2.1 L-py

L-Py[8] est un logiciel de simulation qui combine la construction des L-systèmes avec le langage de programmation de haut niveau Python. Les L-systèmes ont été conçus comme un cadre mathématique pour la modélisation de la croissance des plantes. L-Py utilise la bibliothèque de représentations géométriques de données biologiques PlantGL[7].

En plus de ce module logiciel, un environnement de développement visuel intégré a été développé pour faciliter la création de modèles de plantes. En particulier, des outils d'optimisation faciles à utiliser ont été intégrés. Grâce à Python et à son approche modulaire, ce cadre permet d'intégrer une variété d'outils définis dans différents contextes de modélisation, en particulier les outils de la plateforme OpenAlea [6]. De plus, il peut être intégré comme un simple module de simulation de croissance dans des pipelines de calcul plus complexes.

### 2.2 SEC2

SEC2[5] est un programme C++ développé par Christophe Renaud en 2008 au LISIC (<https://www-lisic.univ-littoral.fr/>). Ce programme simule donc l'éclairement reçu par des capteurs et des organes de plantes dans une chambre de culture.

Les utilisateurs peuvent contrôller la simulation grâce à ces trois fichiers entrés :

- Le fichier **.ini** : Configurer le nombre de photons simulé, le profondeur, l'algorithme utilisé, ...
- Le fichier **.rad** : Configurer les géométries et propriétés optiques de chambre de culture.
- Le fichier **-data.txt** : Configurer des capteurs et des organes de plants

Le résultat sorti est un fichier qui contiens le nombre de photons reçu de chaque capteur et chaque organe de plantes dans la simulation.

### 2.3 Embree

Intel Embree (<https://github.com/RenderKit/embree>) est une bibliothèque de ray tracing hautement optimisée, développée par Intel pour des applications de rendu 3D et de simulations physiques. Elle offre des performances exceptionnelles grâce à l'utilisation des instructions SIMD et au support du multi-threading. Embree supporte diverses primitives géométriques telles que les triangles, quadrangles, courbes, sphères et disques, et utilise des structures hiérarchiques BVH pour accélérer les tests d'intersection.

Cette bibliothèque supporte également le lancer de rayons sur les GPU d'Intel via SYCL.

### 2.4 La base du moteur de Photon Mapping

la base de ce moteur s'inspire d'un projet GitHub existant ([https://github.com/yumcyaWiz/photon\\_mapping](https://github.com/yumcyaWiz/photon_mapping)) et a été développé par Aurélien Besnier lors d'un stage en 2023. La mission principale de son stage était de développer une simulation d'interception de la lumière dans une chambre de culture et de compter le nombre de photons reçus par les capteurs situés dans cette chambre. De nombreux outils, ainsi que toute l'architecture du projet, ont été développés lors de ce stage.

### 2.5 Mitsuba 3

Mitsuba (<https://www.mitsuba-renderer.org/>) est un système de rendu orienté recherche pour la simulation de la lumière. Durant la développement de ce moteur, il y a 3 versions principales : Mitsuba 0.6, Mitsuba 2 et Mitsuba 3. Pourtant, en fait, chaque version est un système complètement nouveau qui poursuit un ensemble objectifs différents.

Mitsuba 3 est profondément intégré à Python. C'est-à-dire que tous les algorithme de matériels, textures et rendu peuvent être implémentés en Python. En outre, il supporte du ray tracing sur GPU avec le bibliothèque CUDA/OptiX.

# Chapitre 3

## Travaux effectués

### 3.1 Comparaison entre le moteur actuel et le moteur SEC2

Après avoir communiqué avec Aurélien BESNIER, l'ancien développeur de ce projet, j'ai bien compris le problème actuel de ce projet. En effet, à partir d'un projet de Photon Mapping sur GitHub ([https://github.com/yumcyaWiz/photon\\_mapping](https://github.com/yumcyaWiz/photon_mapping)), Aurélien a déjà ajouté de nombreuses nouvelles fonctions pour adapter la simulation d'interception de la lumière. Il a réussi à remodéliser la chambre de culture et à construire la carte de photons. Cependant, en comparant les résultats de nos outils avec ceux de l'ancienne expérimentation obtenu par le moteur SEC2, il existe encore une grande différence.

Avant de commencer à résoudre ce problème, ma première tâche est de lancer une même simulation sur notre moteur et sur le moteur SEC2 pour identifier la différence entre les résultats des deux moteurs. Les informations de cette simulation sont fournis par des biologistes.

#### 3.1.1 Définir les propriétés de simulation

Afin de lancer la simulation d'interception de la lumière, il est d'abord nécessaire de définir les propriétés de simulation. Cela inclut la géométrie de la chambre, les propriétés optiques des matériaux, et les informations sur les capteurs.

##### La géométrie de chambre de culture

Pour la géométrie de la chambre de culture, j'ai utilisé les données d'une chambre de culture de L'Inrae d'Angers qui nous ont servi de référence. Dans ce projet, la chambre était modélisée avec des fichiers .rad[2][3]. La configuration de la chambre sur laquelle nous voulons valider les simulations de lumière est faite comme suit (Fig 3.1) avec une première représentation virtuelle (Fig 3.2) :

### CHAPITRE 3. TRAVAUX EFFECTUÉS

---

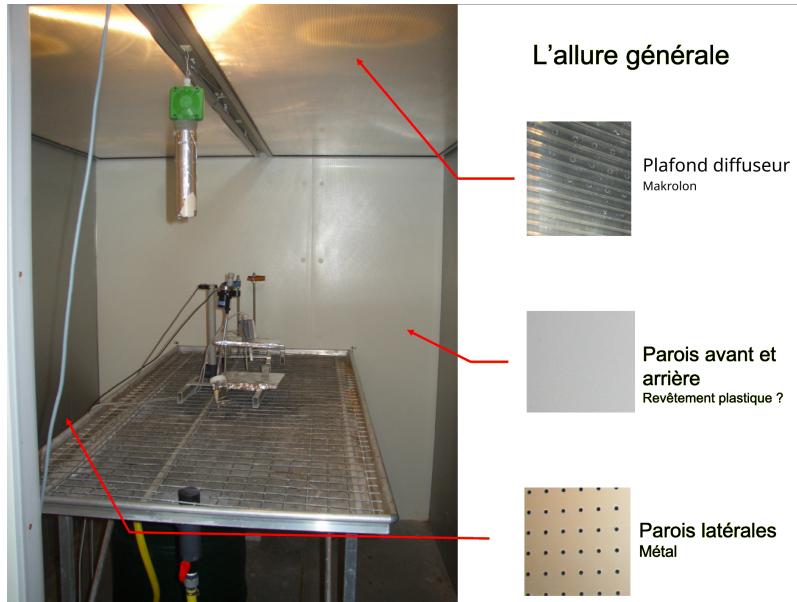


FIGURE 3.1 – La chambre de culture en réalité

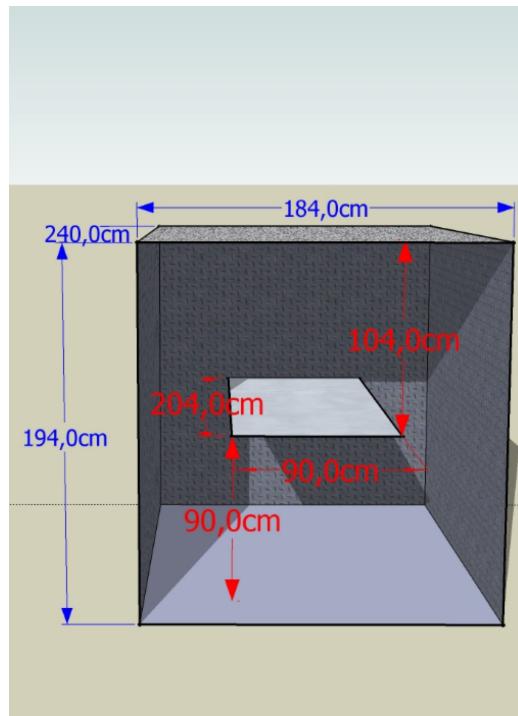


FIGURE 3.2 – Représentation virtuelle de la disposition

## CHAPITRE 3. TRAVAUX EFFECTUÉS

---

Les lampes sont disposées dans quatre quadrants identiques au niveau du plafond 3.3.

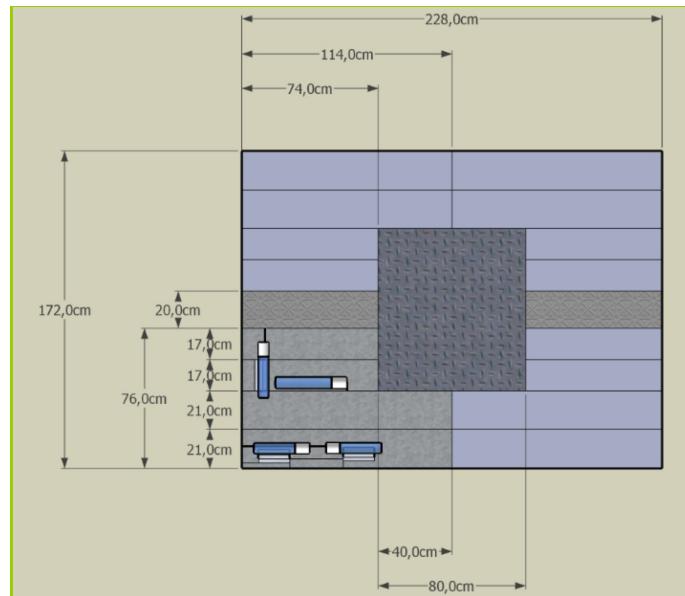


FIGURE 3.3 – Plan du plafond de la chambre

Chaque quadrant possède quatre lampes. Trois de 400 watts (Fig 3.4a et Fig 3.4b) et une de 1000w (Fig 3.4c). Ces lampes sont disposées à proximité de réflecteurs, permettant ainsi de pouvoir réfléchir la lumière plus uniformément dans la chambre.

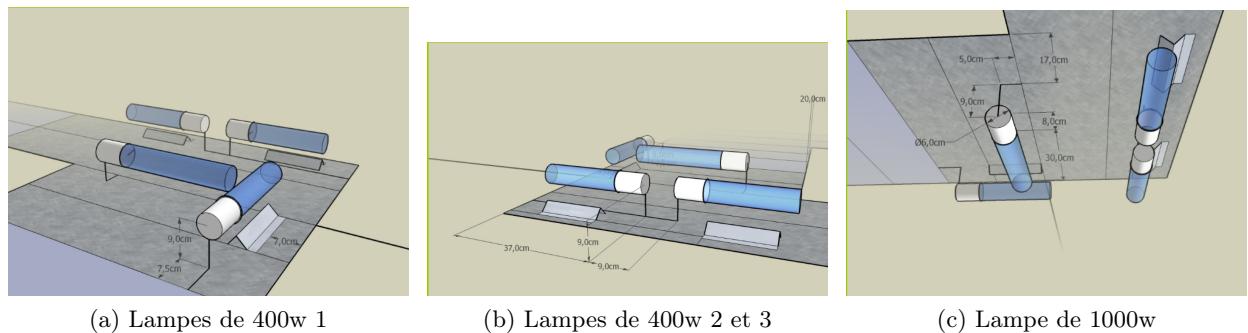


FIGURE 3.4 – Disposition des différentes lampes

## CHAPITRE 3. TRAVAUX EFFECTUÉS

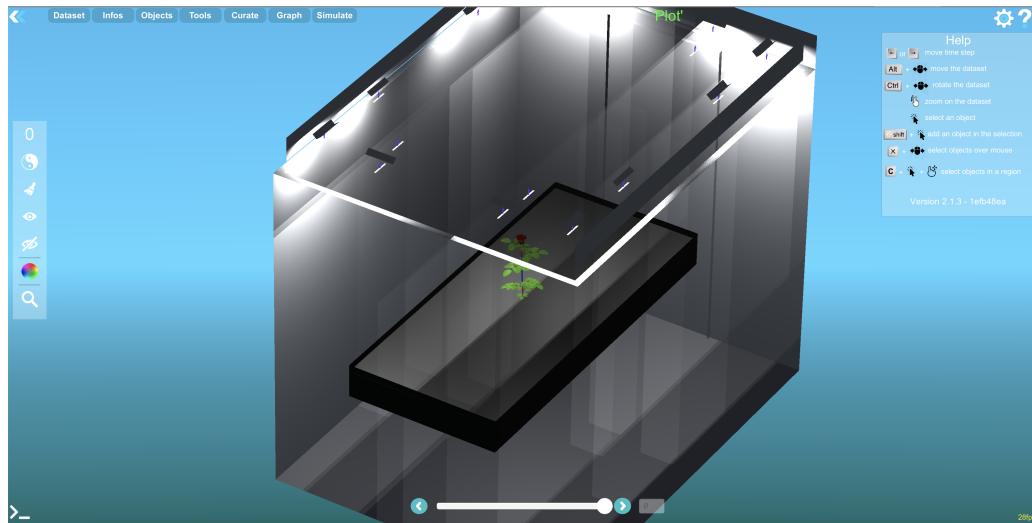


FIGURE 3.5 – La chambre de culture modélisée, représentée dans MorphoNet

### Les propriétés optiques de chambre de culture

Pour construire les matériaux de la chambre de culture, les propriétés optiques (réflectance, transmittance, spécularité et rugosité) de chaque surface ont été mesurées pour toutes les longueurs d'onde du PAR (Photosynthetically Active Radiation).

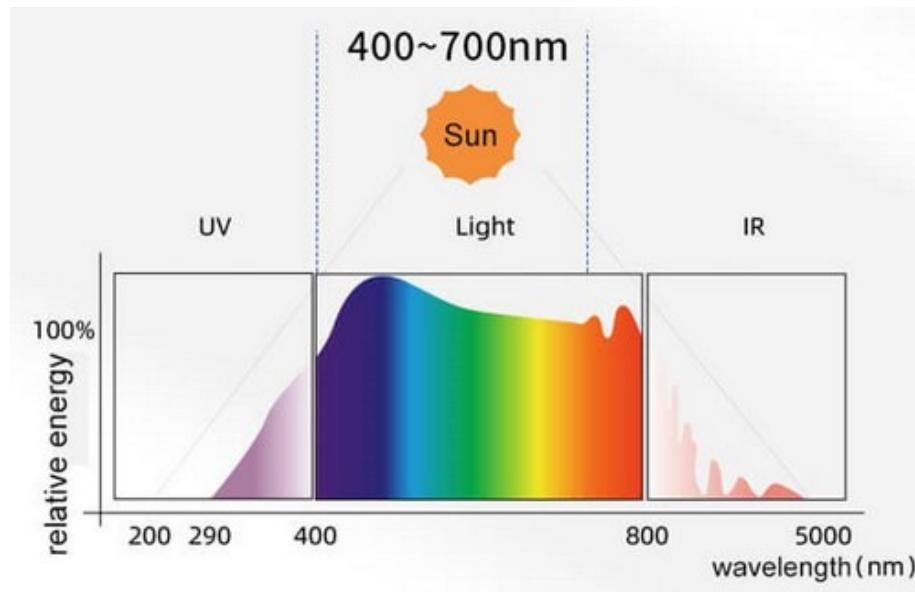


FIGURE 3.6 – PAR (Photosynthetically Active Radiation)

Le PAR désigne la bande d'ondes du rayonnement solaire de 400 à 700 nanomètres que

## CHAPITRE 3. TRAVAUX EFFECTUÉS

---

les organismes photosynthétiques sont capables d'utiliser dans le processus de photosynthèse. Dans les simulations pour l'étude de la croissance des plantes, cette gamme spectrale est divisée en 12 parties. Pour chaque partie, les propriétés optiques moyennes de chaque surfaces sont calculées et utilisées pour les simulations.

Pour diminuer le temps de débogage, j'ai lancé la simulation sur seulement une de ces 12 parties, de 655 à 665 nanomètres.

### Configuration des capteurs

Dans le moteur SEC2, il existe plusieurs types de capteurs : des quads solides/transparents, des capteurs circulaires solides/transparents et des organes de plantes. Pour commencer, je vais tester notre moteur sur le cas le plus simple. La simulation sera réalisée uniquement avec des capteurs circulaires transparents. Dans cette simulation, les capteurs seront positionnés sur deux plans dans un quart de la chambre de culture, à 10 et 50 cm au-dessus de la table. Les données réelles pour une telle configuration ont été acquises dans une expérimentation précédente.

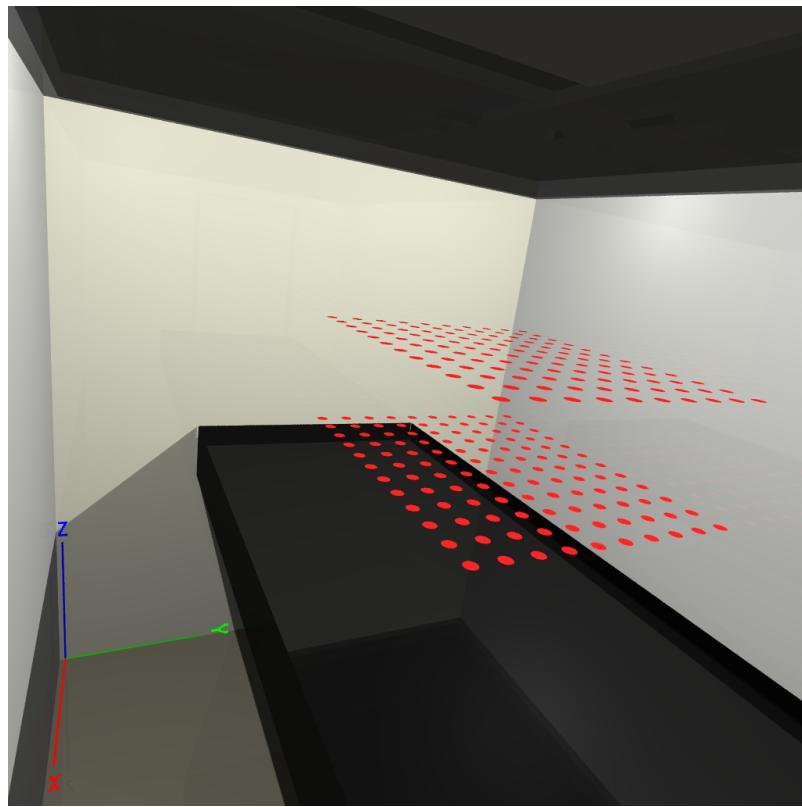


FIGURE 3.7 – Les positions des capteurs

### 3.1.2 Résultats

Dans un premier test, une simulation avec  $10^9$  photons et une profondeur maximale de rebond de 50 a été lancé. La simulation a pris 1264 secondes. Ci-dessous, voici la comparaison des distributions de photons interceptés par chaque capteur obtenues par le moteur SEC2 et notre moteur.

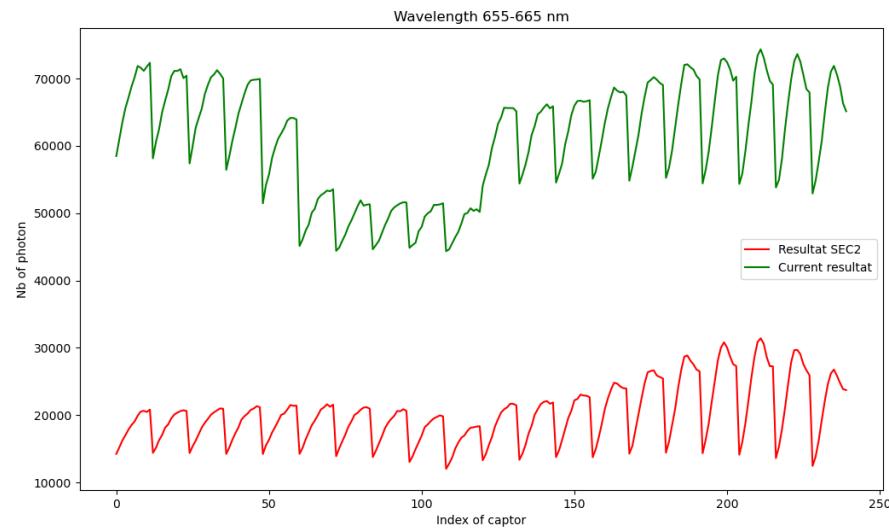


FIGURE 3.8 – La distribution de photons sur les capteurs

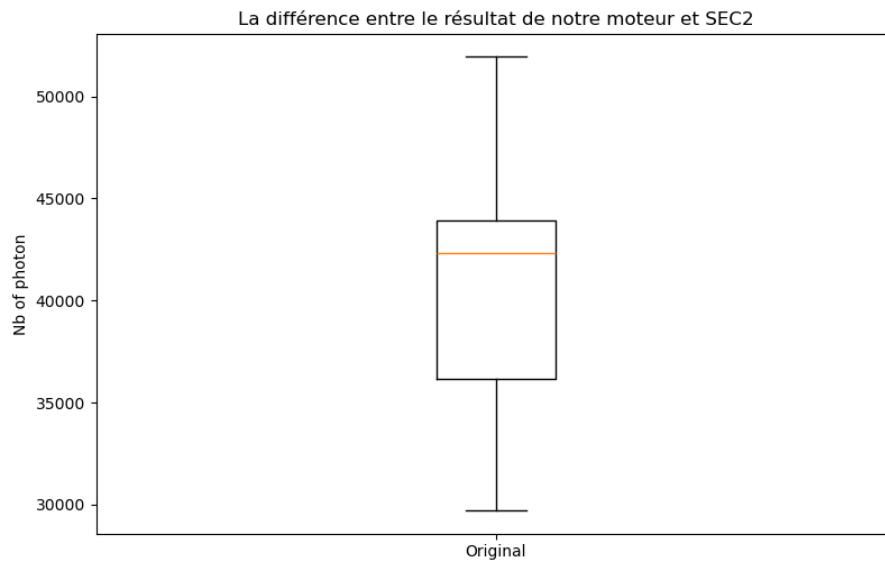


FIGURE 3.9 – La différence entre les deux résultat

Après avoir comparé les résultats obtenus par les deux moteurs, je peux noter quelques points similaires entre les deux résultats :

- Le nombre de photons sur le capteur augmente lorsque la coordonnée x de la position augmente.
- Plus le capteur est situé en haut de la chambre, plus de photons le capteur reçoit.

Pourtant, il existe encore une grande différence par rapport aux résultats obtenus avec le moteur SEC2 :

- Notre moteur enregistre plus de photons que le moteur SEC2, environ 35000 à 45000 photons de plus sur chaque capteur.
- Les cinq premières rangées de capteurs reçoivent plus de photons, même plus de photons que les capteurs de la deuxième hauteur.

## 3.2 Amélioration du moteur de simulation

Après avoir évalué les résultats obtenus par notre moteur, nous avons constaté de nombreux problèmes affectant la précision des simulations. Ma tâche suivante consiste à identifier et résoudre ces problèmes afin d'améliorer la précision de notre moteur.

Dans les parties suivantes, je vais identifier chaque problème de notre moteur, proposer des solutions, et présenter les résultats obtenus après les corrections.

### 3.2.1 Correction du modèle d'illumination

La première raison est venue de le modèle d'illumination. En effet, dans le moteur SEC2, quand un rayon de lumière touche une surface, il y a cinq possibilités de comportement :

- Réflexion spéculaire
- Réflexion diffuse
- Réfraction spéculaire
- Réfraction diffuse
- Absorption

Cependant, dans notre moteur, toutes les surfaces sont considérées comme des surfaces de Lambert, autrement dit, ce sont des surfaces complètement diffuses. Ainsi, quand un rayon de lumière touche une surface, il subit toujours une réflexion diffuse et n'est jamais absorbé. C'est une des raisons pour lesquelles notre moteur enregistre plus de photons.

## CHAPITRE 3. TRAVAUX EFFECTUÉS

---

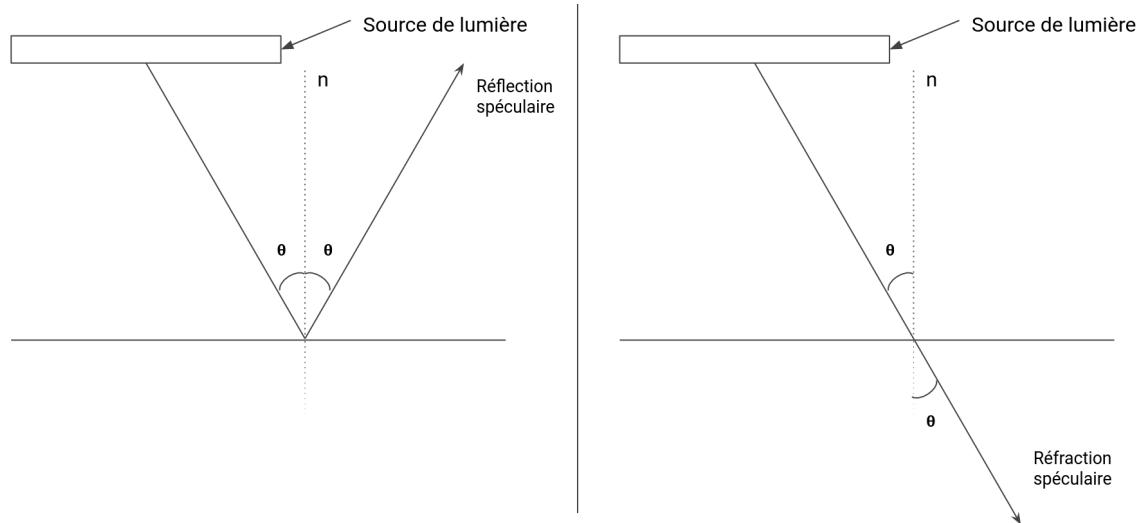


FIGURE 3.10 – Réfraction / réflexion spéculaire

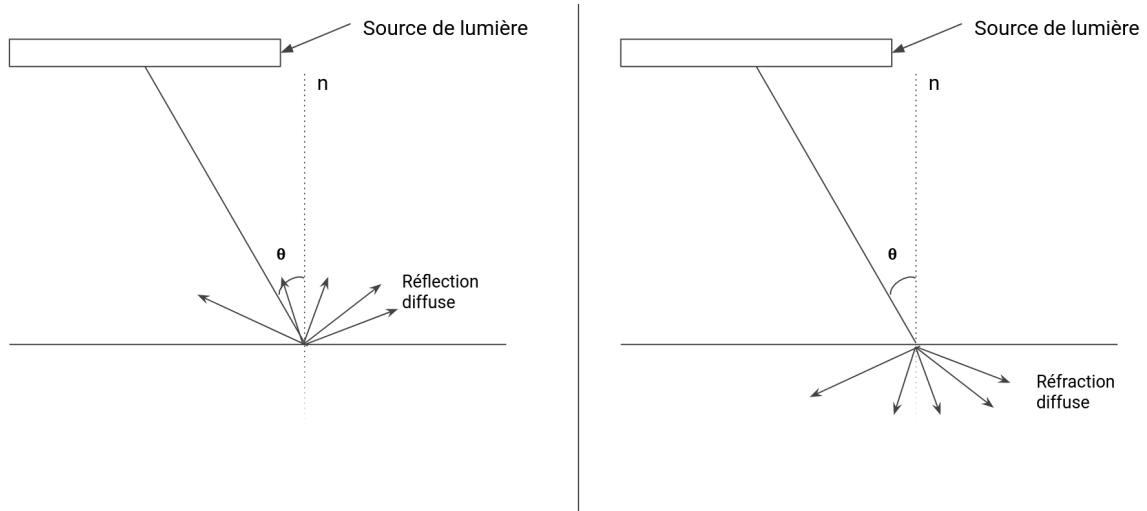


FIGURE 3.11 – Réfraction / réflexion diffuse

Afin de résoudre ce problème, j'ai implémenté le modèle d'illumination de Phong dans notre moteur. Ce modèle est très adapté pour représenter des matériaux complexes, comme dans notre cas.

### Le modèle de BRDF de Phong

La fonction de distribution de la réflectance bidirectionnelle (BRDF) décrit la réflexion d'une onde lumineuse sur une surface. En effet, pour une direction d'éclairement et une

direction de réflexion données, la BRDF est le rapport de l'énergie lumineuse réfléchie en un point d'une surface infinitésimale à l'éclairement incident sur celle-ci.

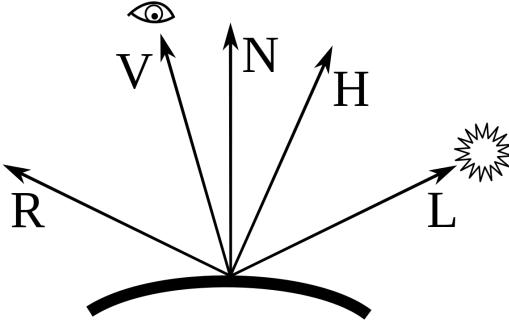


FIGURE 3.12 – Les vecteurs utilisés dans la modèle de Phong<sup>1</sup>

La BRDF de Phong[1] est représenté par cette formule :

$$f(\vec{L}, \vec{V}) = k_d(\vec{N} \cdot \vec{L}) + k_s F_s(\vec{L}, \vec{V}) \quad (3.1)$$

où :

- $\vec{L}$  est la direction à la lumière.
- $\vec{V}$  est la direction de réflexion.
- $\vec{N}$  est la normale de surface.
- $k_d$  est la coefficient (couleur) diffuse de l'objet.
- $k_s$  est la coefficient (couleur) spéculaire de l'objet.

Le terme  $F_s(\vec{L}, \vec{V})$  est appelé lobe spéculaire. Il peut être exprimée de plusieurs manières. Originalement, Phong avait proposé l'expression suivant :

$$F_s^P(\vec{L}, \vec{V}) = \begin{cases} (\vec{R} \cdot \vec{V})^n & \text{si } \vec{R} \cdot \vec{V} > 0 \\ 0 & \text{sinon} \end{cases} \quad (3.2)$$

où :

- $\vec{V}$  est la direction de réflexion en réelle.
- $\vec{R}$  est la direction de réflexion définie par la première loi de Descarte ; et exprimée par :  $\vec{R} = 2(\vec{L} \cdot \vec{N})\vec{N} - \vec{L}$ .
- $n$  est la rugosité de la surface. Plus  $n$  est grand, plus la surface apparaît lisse.

À partir de la BRDF, on peut facilement calculer l'énergie de la lumière sortant d'une surface. Idéalement, cette énergie se retrouve totalement dans un seul rayon sortant. En pratique, une partie de l'énergie peut être absorbée, diffusée ou réfractée au niveau de l'interface. Pour chaque possibilité, il faut générer un nouveau rayon avec une partie de l'énergie

---

1. [https://en.wikipedia.org/wiki/Phong\\_reflection\\_model](https://en.wikipedia.org/wiki/Phong_reflection_model)

pour calculer la carte de photons. Cela rend le moteur plus compliqué à implémenter et augmente le temps de simulation.

La méthode Russian Roulette permet de résoudre le problème de la détermination du type de comportement de la lumière lors d'un contact avec une surface. Le moteur SEC2 intégrait une telle approche. Je l'ai implémenté dans notre nouveau moteur.

### Russian Roulette

La technique de la roulette russe est souvent utilisée pour réduire le temps de calcul dans le processus de lancer de rayons [4]. L'idée générale de cette technique est que, lorsqu'un rayon de lumière touche une surface, on génère un rayon de sortie correspondant. Pour ce faire, on effectue un test de probabilité avec les valeurs de diffusion, de réflexion spéculaire et de transmission pour déterminer l'état du rayon suivant. L'énergie de ce rayon est également l'énergie totale sortante de la surface.

Ci-dessous, c'est le pseudo-code de la technique Russian Roulette.

#### Algorithme 1 : RussianRoulette

```
Données : diff : float, spec : float, trans : float
Début
    //Générer un valeur aléatoire de 0 à 1
    p ← rand() / RAND_MAX;
    //Test de probabilité
    si p < diff alors
        | return DIFFUSSION;
    finsi
    si p < diff + spec alors
        | return REFLEXION_SPECULAIRE;
    finsi
    si p < diff + spec + trans alors
        | return TRANSMISSION;
    finsi
    sinon
        | return ABSORPTION;
    finsi
Fin
```

### Résultat

Après d'effectuer ces corrections, j'ai relancé la simulation dans la section 3.1. Voici le résultat.

## CHAPITRE 3. TRAVAUX EFFECTUÉS

---

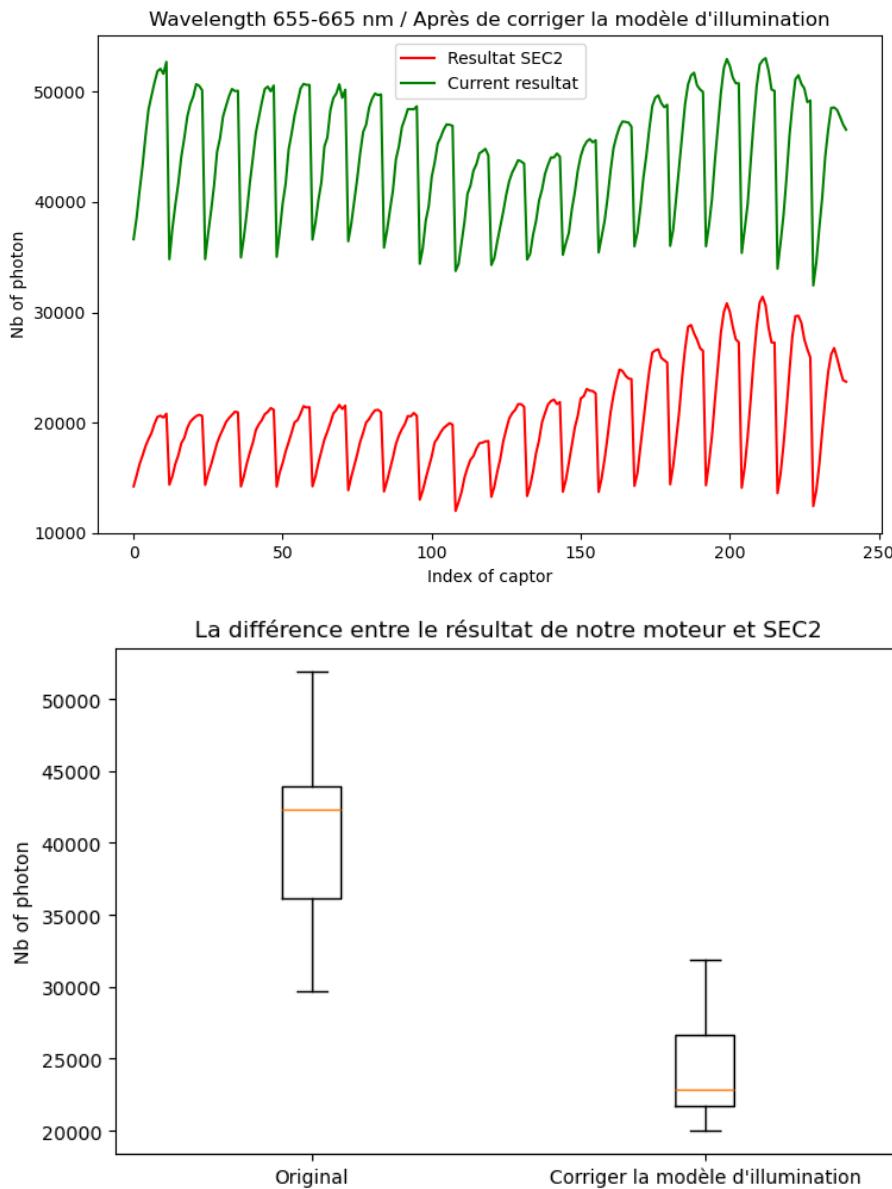


FIGURE 3.13 – Les résultats après la correction du modèle d'illumination

Après l'intégration de ce modèle d'illumination plus complet, on peut voir que la différence entre les deux résultats est amélioré, de [35000 - 45000] à [22000 - 27000] photons.

### 3.2.2 Corrections des calculs des valeurs Diffuse et Spéculaire

Le problème suivant vient de la manière dont notre moteur calcule les valeurs de diffusion et de réflexion spéculaire à partir des valeurs de réflectance (albédo) et de spécularité fournies par les biologistes. En effet, puisque le projet de base n'utilise que des surfaces lambertiennes (diffusion complète), le module de chargement des propriétés optiques de notre moteur considère directement l'albédo comme la valeur de diffusion. Cela n'est pas correct dans notre cas, où le matériau est plus complexe.

En fait, quand un rayon de lumière est réfléchi par une surface, il y a deux scénarios de réflexion suivant :

- Si la surface est lisse comme un miroir, la lumière se reflétera dans la direction opposée et nous pourrons voir un reflet ou un point lumineux si nous nous tenons sur le chemin de ce faisceau lumineux. Ce phénomène s'appelle la réflexion spéculaire.
- Si la surface est rugueuse, la lumière est diffusée dans des différentes directions et on peut voir la couleur originale des objets. Ce phénomène s'appelle la diffusion.

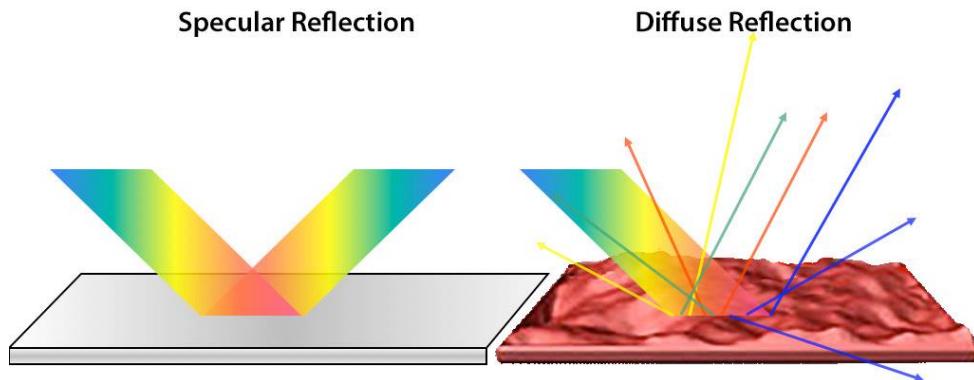


FIGURE 3.14 – La diffusion et la réflexion spéculaire<sup>2</sup>

Dans notre cas, la valeur d'albédo inclut la valeur de diffusion ainsi que la valeur de réflexion spéculaire. Alors, il faut recalculer ces valeurs avec ces formules suivantes :

$$\begin{aligned} diffuse &= albedo * (1 - spec) \\ speculaire &= albedo * spec \end{aligned} \tag{3.3}$$

---

2. <https://coolcodea.wordpress.com/2013/09/28/115-lighting-in-3d-diffuse-light/>

## Résultat

Après d'effectuer ces corrections, j'ai relancé la simulation dans la section 3.1. Voici le résultat.

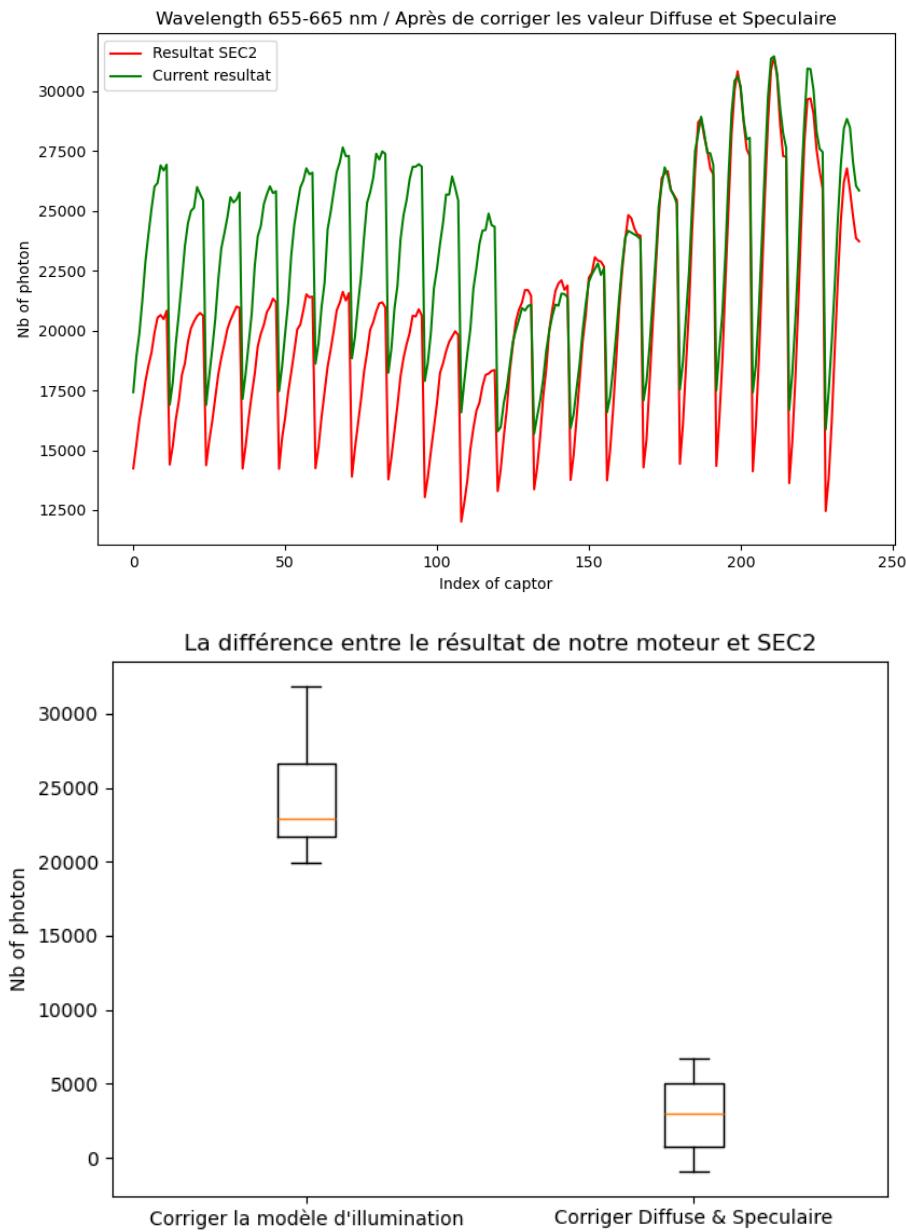


FIGURE 3.15 – Les résultats après les corrections des valeurs Diffuse et Spéculaire

Après les corrections des valeurs Diffuse et Spéculaire, on peut voir que la différence

entre les deux résultats est encore minimisée, de [22000 - 27000] à [1000 - 5000] photons.

### 3.2.3 Optimiser la distance minimale d'intersection

Le problème suivant que j'ai trouvé dans notre moteur actuel concerne la distance minimale d'intersection. Afin de mieux comprendre ce problème, je voudrais d'abord rappeler le principe de l'algorithme de calcul de l'intersection entre un rayon et un triangle. Dans cet algorithme, on calcule la distance ( $t$ ) entre l'origine du rayon et le point d'intersection. Grâce à cette distance, on peut facilement déterminer la position du point d'intersection. Pour éviter des intersections inattendues, telles que des intersections avant l'origine du rayon ou trop loin de l'origine du rayon, il faut définir une distance minimale ( $t_{near}$ ) et une distance maximale ( $t_{far}$ ) d'intersection. Par défaut, dans notre moteur, cette plage est fixée entre  $10^{-5}$  et l'infini. Ces valeurs utilisées par défaut dans notre moteur ne donnent pas des valeurs de simulation satisfaisantes.

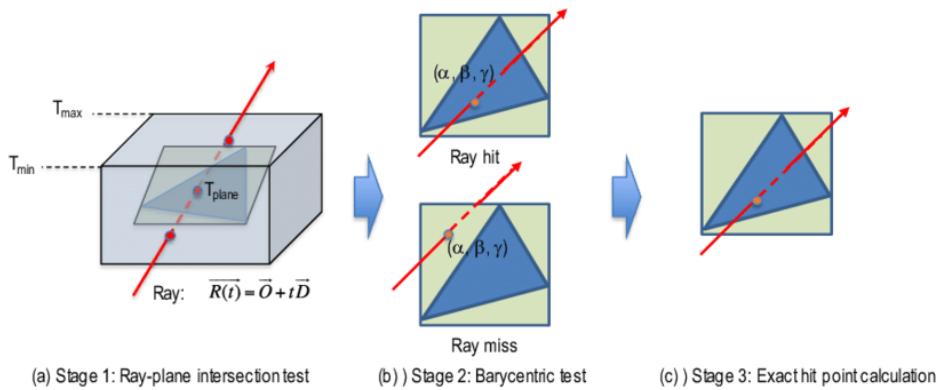


FIGURE 3.16 – Les étapes de calculs l'intersection entre un rayon et un triangle<sup>3</sup>

En effet, dans le processus de tracer le rayon, chaque fois que le rayon touche une surface, on génère un nouveau rayon à partir du point d'intersection et on continue les calculs. En théorie, si la distance minimale d'intersection est plus grande que zéro, il est impossible qu'un rayon touche la même surface d'où il est parti. Cependant, à cause des erreurs d'arrondi des ordinateurs, les calculs ne sont pas toujours exacts, et on peut rencontrer des intersections inattendues si la valeur de la distance minimale est trop petite. Dans notre cas, certains capteurs interceptaient plusieurs fois le même rayon à cause d'erreur d'arrondi et recevaient ainsi trop de photons.

Pour résoudre ce problème, il faut choisir une nouvelle valeur de  $t_{near}$  qui soit plus grande que la valeur par défaut, mais qui ne dépasse pas la distance minimale entre les objets dans la scène. Après avoir testé plusieurs valeurs, j'ai obtenu le meilleur résultat avec une valeur

3. [https://www.researchgate.net/figure/Three-stages-of-ray-triangle-intersection-test\\_fig8\\_319442281](https://www.researchgate.net/figure/Three-stages-of-ray-triangle-intersection-test_fig8_319442281)

de  $t_{near}$  de  $10^{-1}$ . De plus, j'ai également permis aux utilisateurs de contrôler les valeurs de  $t_{near}$  et  $t_{far}$  en configurant les données d'entrée.

## Résultat

Après d'effectuer ces corrections, j'ai relancé la simulation dans la section 3.1. Voici le résultat.

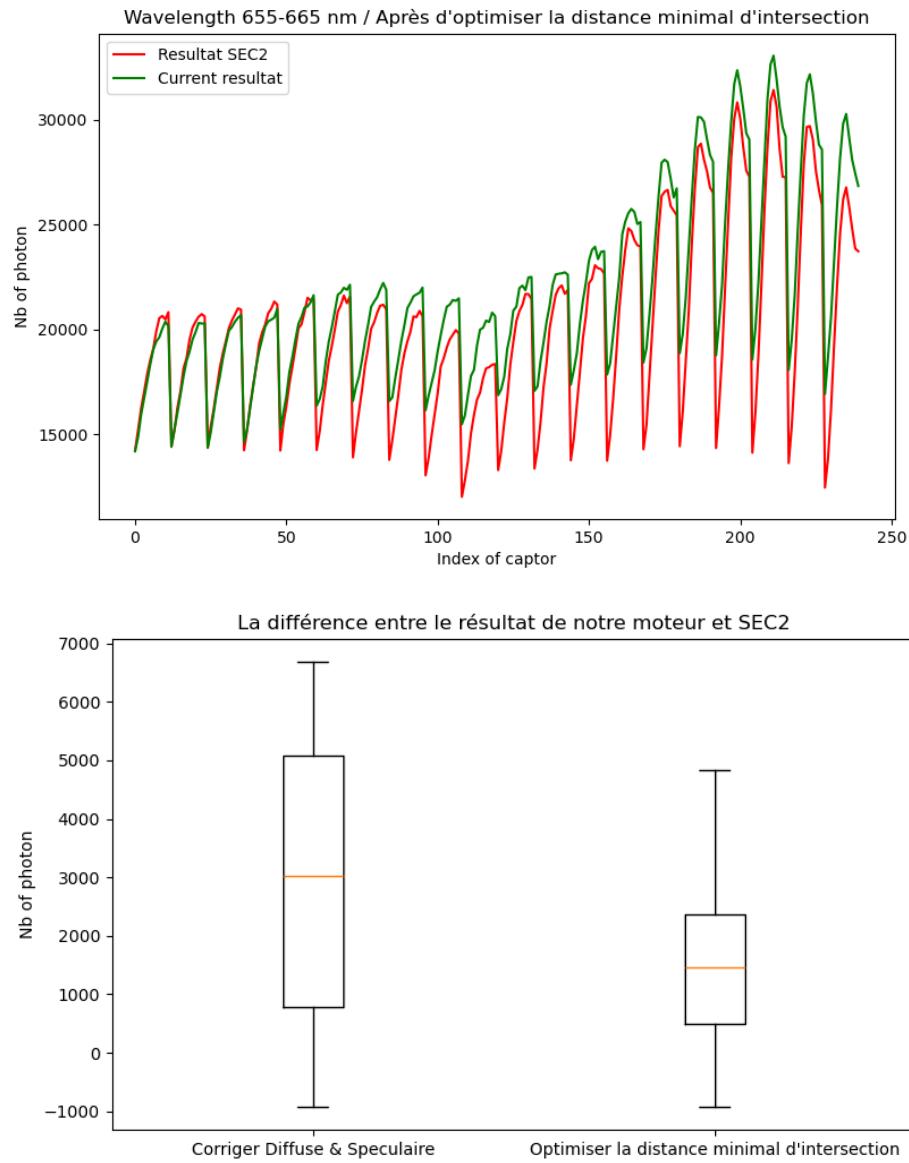


FIGURE 3.17 – Les résultats après l'optimisation de la distance minimal d'intersection

Après l'optimisation de la distance minimal d'intersection, on peut voir que l'erreur faite sur le nombre de photons reçu pour chaque capteur est minimisée.

### 3.2.4 Ignorer la face arrière

La géométrie d'un objet est définie par une liste de triangles. Chaque triangle a un vecteur normal qui permet de définir la face avant et arrière de ce triangle. Après avoir lu le code du moteur SEC2, j'ai découvert que ce moteur ne considère que les intersections entre le rayon et les faces avant. Cependant, notre moteur accepte toutes les intersections car les APIs d'Embree n'ignorent pas les faces arrière lors du calcul de l'intersection. Du a cela, les capteurs reçoivent plus de photons que prévu.

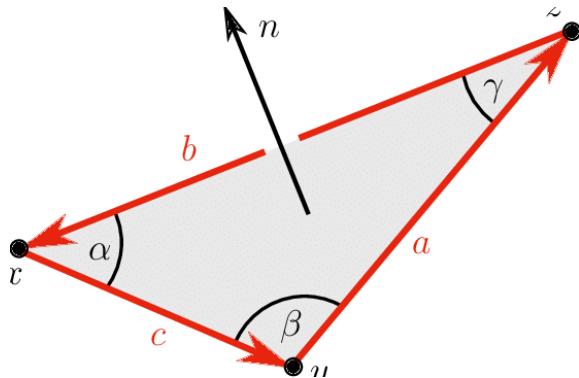


FIGURE 3.18 – La face avant d'un triangle<sup>4</sup>

Pour résoudre ce problème, chaque fois que notre moteur trouve une intersection, un test de direction de face est réalisé.

#### Algorithme 2 : TestFaceDirection

Données : RayHit rayhit

Début

```

normal ← rayhit.hit.normal ;
ray_dir ← rayhit.ray.direction ;
n_dot_r ← dotproduct(normal, ray_dir) ;
si n_dot_r < 0.00001 alors
    | return FRONTFACE ;
finsi
sinon
    | return BACKFACE ;
finsi

```

Fin

<sup>4</sup>. [https://www.researchgate.net/figure/Three-stages-of-ray-triangle-intersection-test\\_fig8\\_319442281](https://www.researchgate.net/figure/Three-stages-of-ray-triangle-intersection-test_fig8_319442281)

## Résultat

Après d'effectuer ces corrections, j'ai relancé la simulation dans la section 3.1. Voici le résultat.

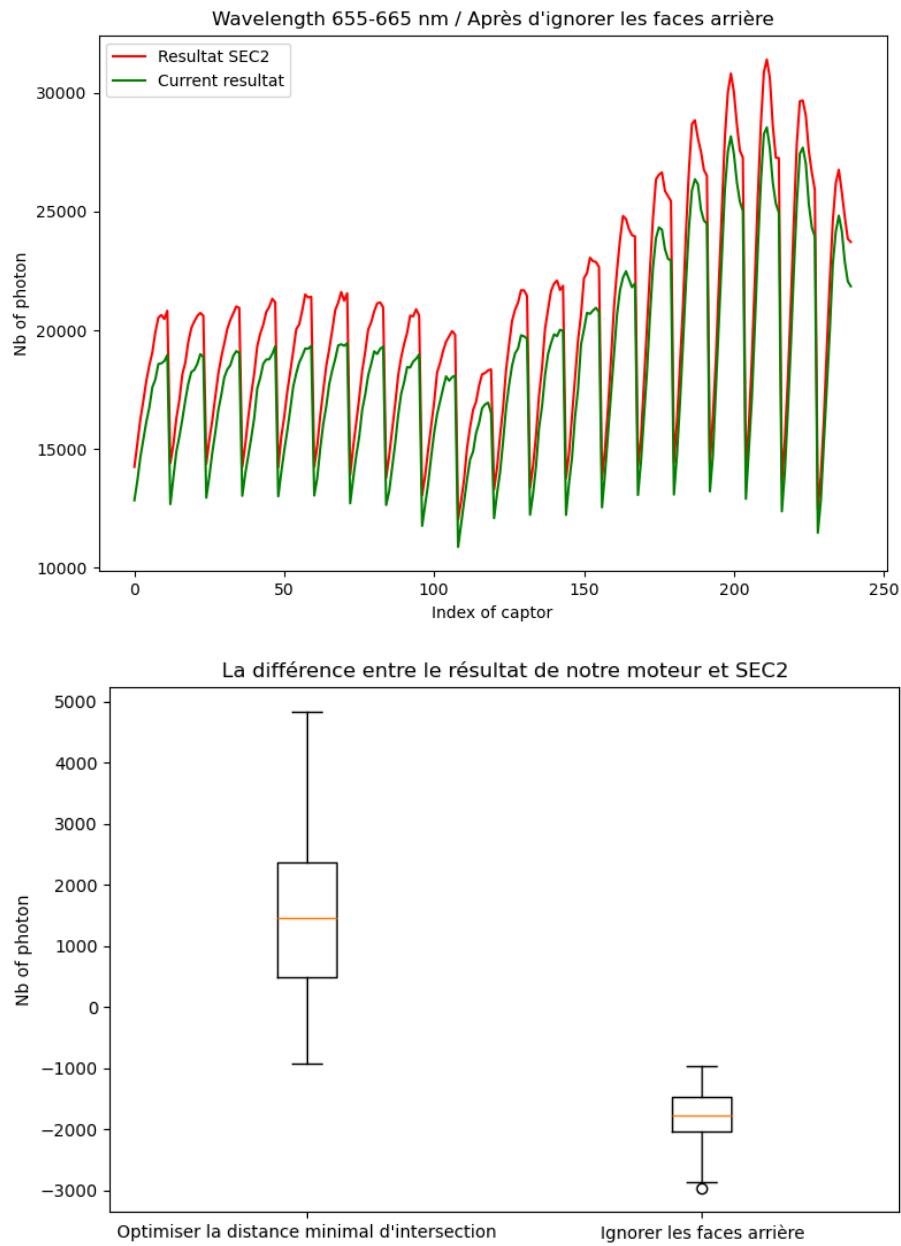


FIGURE 3.19 – Les résultats après d'ignorer les faces arrière

Après avoir ignoré les faces arrière, la distribution des photons sur les capteurs obtenue par notre moteur est très proche de celle obtenue par SEC2, mais avec moins de photons.

### 3.2.5 Modification de géométrie de capteurs

Le dernier problème que j'ai rencontré concerne la modélisation des capteurs dans notre moteur. Dans le moteur SEC2, chaque capteur est représenté sous forme de cercle, et il peut calculer l'intersection entre un rayon et un cercle. Cependant, par défaut, Embree ne supporte que l'intersection avec des triangles. Bien qu'il soit possible de définir une fonction d'intersection avec des géométries personnalisées dans Embree, cela affecterait fortement la performance de notre système car ces géométries ne sont pas prises en charge par la structure d'accélération de lancer de rayons d'Embree. C'est la raison pour laquelle, dans notre moteur, nous utilisons des triangles pour représenter les capteurs.

Auparavant, en raison des préoccupations de performance, Aurélien utilisait seulement 8 triangles pour représenter le capteur. Cette discréétisation simple entraîne un nombre de photons reçus moins important que prévu. Pour résoudre ce problème, nous avons décidé d'utiliser une discréétisation des capteurs plus fine.

Après plusieurs tests, j'ai constaté qu'avec 32 triangles ou plus, nous obtenons les meilleurs résultats. Le temps de simulation ne change pas car Embree utilise une structure BVH (Bounding Volume Hierarchy) qui évite de tester inutilement des triangles trop éloignés des rayons considérés et accélère ainsi les calculs d'intersection [9].

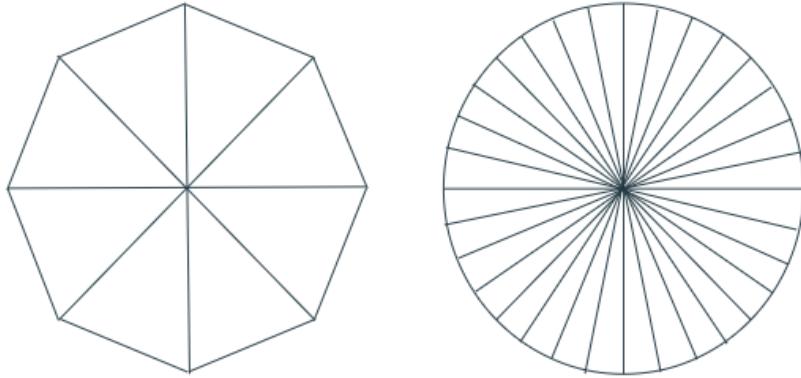


FIGURE 3.20 – Le géométrie des capteurs avec 8 triangles et 32 triangles

### Résultat

Après d'effectuer ces corrections, j'ai relancé la simulation dans la section 3.1. Voici le résultat.

## CHAPITRE 3. TRAVAUX EFFECTUÉS

---

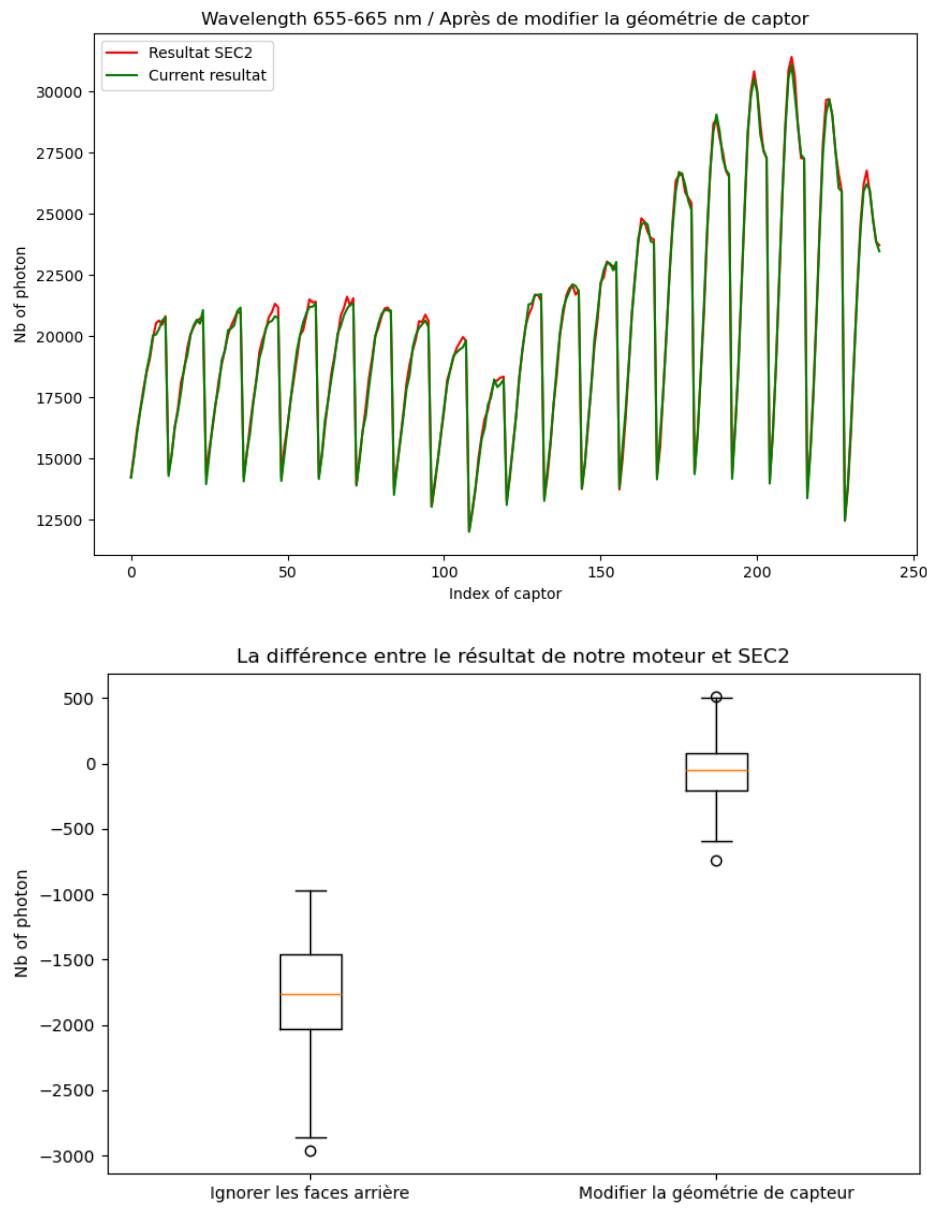


FIGURE 3.21 – Les résultats après de modifier la géométrie de capteurs

Après avoir modifié la géométrie des capteurs, les résultats des deux moteurs deviennent très similaires. La différence est proche de zéro.

### 3.3 Amélioration de la performance

Un de mes objectifs de ce stage est d'améliorer la performance de nos outils. Pour ce faire, il est nécessaire d'exploiter la capacité de calcul en parallèle sur CPU et GPU, ainsi que d'optimiser les algorithmes et les structures de données.

Dans notre moteur, Aurélien a déjà implémenté les calculs parallèles sur CPU. Ma prochaine tâche consistera à optimiser la performance du moteur actuel et à implémenter les calculs parallèles sur GPU sur notre système.

#### 3.3.1 Optimiser le moteur actuel

Dans cette partie, je vais présenter les travaux que j'ai déjà effectués concernant la performance de notre moteur.

##### Optimiser le code pour les threads

Après avoir lu le code du moteur, j'ai constaté que, dans le module de construction de la carte de photons, de nombreuses sections de code ont été sécurisées pour les threads sans que cela soit nécessaire. Cela nuit à l'optimisation des calculs parallèles sur CPU dans notre moteur.

Après avoir optimisé ces sections de code, le temps de simulation de notre moteur a été réduit à environ 290 secondes, soit environ 7 fois plus rapidement que le moteur SEC2.

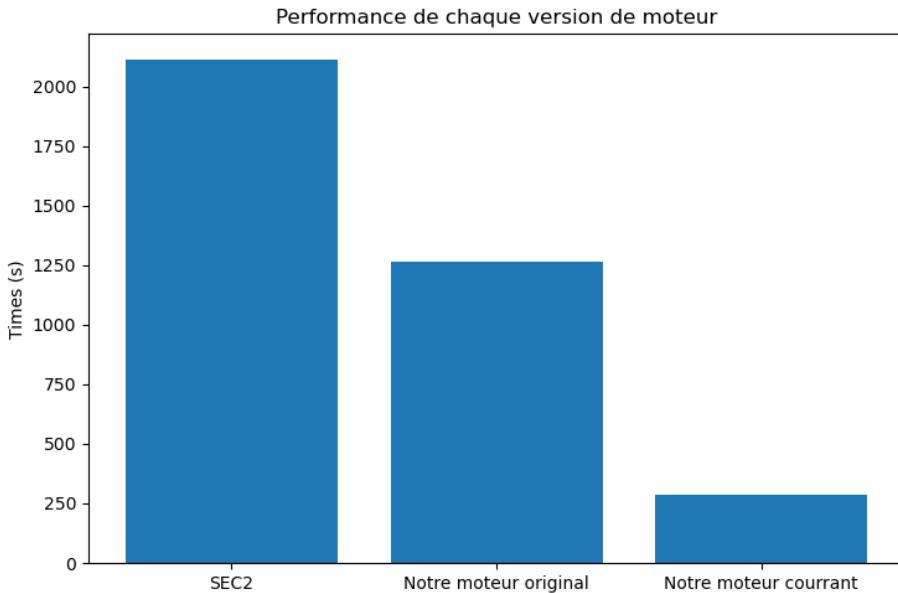


FIGURE 3.22 – Comparaison de performance entre les deux moteurs

### Optimiser le générateur de valeurs aléatoires

En optimisant le code du moteur, j'ai trouvé un problème très intéressant concernant la performance de moteur. C'est qu'utiliser la fonction `rand()` de C++ pour générer une valeur aléatoire dans notre moteur augmente considérablement le temps de simulation, presque jusqu'à 10 fois plus que le temps original.

Après plusieurs test, j'ai découvert la raison de ce phénomène. C'est que lorsqu'on utilise la fonction `rand()`, un seul générateur est utilisé pour plusieurs threads, ce qui entraîne des blocages dus à la gestion de la sécurité des threads (thread-safe). Alors, il est nécessaire d'avoir plusieurs générateurs correspondant à chaque thread pour optimiser la performance.

D'autre part, j'ai aussi fait un test de performance avec les différents algorithmes de génération de valeurs aléatoires. Les algorithmes que j'ai considéré sont :

- PCG
- LCG
- SplitMix
- Xoroshiro

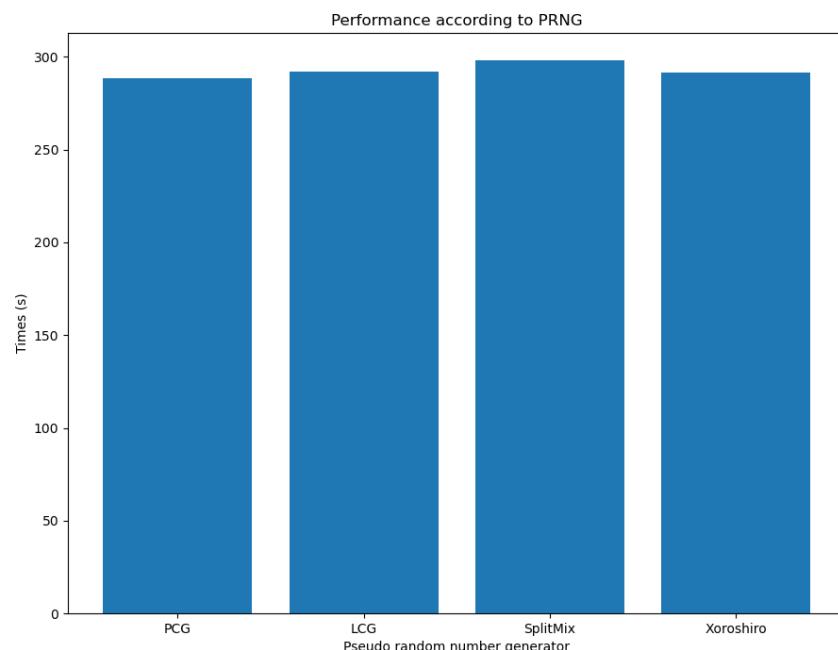


FIGURE 3.23 – Performance avec les différents générateurs des valeurs aléatoires

On peut voir que l'algorithme pour générer des valeurs aléatoires n'influence pas trop à la performance d'algorithme.

### Configurer le nombre de threads

Par défaut, notre moteur choisit automatiquement le nombre maximal de threads et utilise toutes les ressources CPU pour lancer la simulation. Par conséquent, les utilisateurs ne peuvent pas effectuer d'autres tâches sur l'ordinateur pendant que la simulation fonctionne. C'est pourquoi j'ai permis aux utilisateurs de choisir le nombre de threads utilisés. Voici les temps de simulation pour  $10^9$  photons avec des différents nombres de threads :

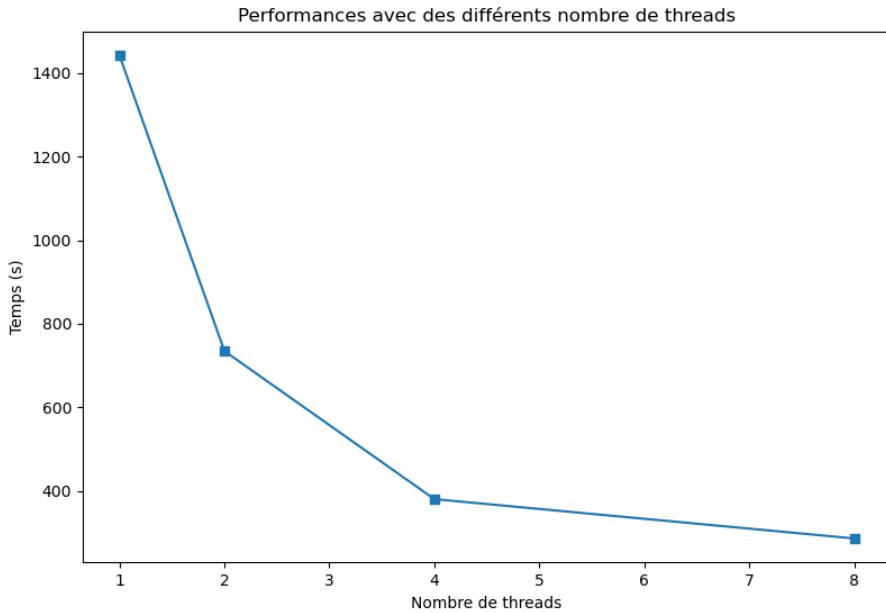


FIGURE 3.24 – Performances avec des différents nombre de threads

Il est facile de constater qu'à partir de 4 threads, le temps de simulation ne diminue pas beaucoup.

### 3.3.2 Exploration des outils de calcul parallèle sur GPU

Après avoir obtenu de très bons résultats avec le calcul parallèle sur CPU, j'ai commencé à explorer comment effectuer le calcul parallèle sur GPU sur notre système. Pour ce faire, j'ai déjà travaillé avec Embree SYCL et Mitsuba3. Voici les informations que j'ai trouvées sur ces outils.

#### Embree SYCL

Comme vous le savez, notre moteur utilise la bibliothèque Embree pour le lancer de rayons, et cette bibliothèque supporte également les GPU avec le module SYCL. C'est la raison pour laquelle j'ai décidé d'utiliser cette bibliothèque pour les calculs GPU au lieu de CUDA/OptiX et d'autres bibliothèques.

Après quelques jours à tester les exemples d'Embree SYCL, j'ai réussi à compiler le code d'exemple. Malheureusement, lorsque j'ai lancé l'exemple, cela n'a pas fonctionné car mon ordinateur n'est pas compatible avec Embree SYCL. Comme je l'ai découvert sur le site d'Embree, cette bibliothèque ne supporte que les cartes graphiques Intel de dernière génération, ce sont :

- Intel® Arc, sortie en 2022
- Intel® Data Center GPU, sortie en 2022
- Intel Meteor Lake, sortie en 2023

Après une discussion avec mes encadrants, nous avons décidé de ne pas utiliser cette bibliothèque car il est trop difficile de trouver un ordinateur compatible. Pour continuer à faire calculs sur GPU, on a décidé d'utiliser le moteur Mitsuba.

### Mitsuba3

Mitsuba est un système de rendu orienté vers la recherche pour la simulation de la lumière. D'après ce que j'ai découvert sur ce moteur, Mitsuba 0.6 est la seule version à inclure le module de Photon Mapping. Cependant, à partir de Mitsuba 3, ce moteur commence à supporter le calcul sur GPU. Ainsi, pour tester les performances du GPU, il faudrait créer un module de Photon Mapping similaire à celui de notre moteur basé sur la version de Mitsuba 3.

Après avoir lancé tous les exemples fournis par Mitsuba, je trouve que cette idée est réalisable car ce moteur nous offre de nombreux outils nécessaires pour réaliser le Photon Mapping, tels que l'intersection entre le rayon et la scène, l'échantillonnage des directions aléatoires, la génération d'un nouveau rayon en fonction des interactions de surface, etc.

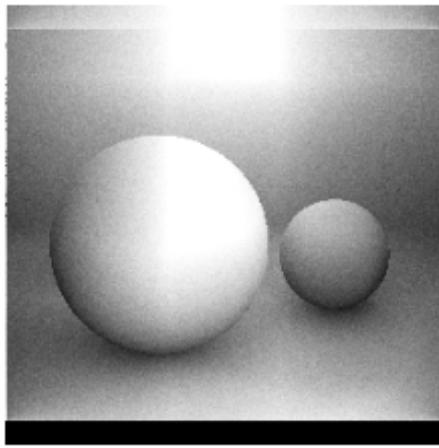


FIGURE 3.25 – Le résultat d'un exemple de Mitsuba

Le prochain travail consiste à implémenter notre moteur de simulation sur Mitsuba. Étant donné que cette tâche est longue et compliquée, avant de la réaliser, je vais me concentrer sur la finalisation de notre moteur sur CPU et sur la documentation détaillée de mes travaux précédents, ainsi que sur la rédaction de guides d'utilisation du moteur.

### 3.4 Réaliser la simulation sur une plante

Actuellement, dans notre moteur, toutes les simulations sont réalisées seulement sur les capteurs. C'est la raison pour laquelle on voudrait faire un module à réaliser la simulation sur une modèle de plant complète. En discutant avec les biologistes, il semble qu'ils sont très intéressé par ce module.

Pour ce faire, j'ai utilisé les L-systèmes pour représenter le modèle de plante. D'une part, les L-systèmes permettent aux utilisateurs de modéliser n'importe quelle structure de plante qu'ils souhaitent avec les outils L-Py 2.1. D'autre part, ils permettent de simuler la croissance de la plante en fonction des conditions de lumière futures. Ensuite, j'ai créé un module pour compter les photons touchant les organes de cette plante.

Afin de tester ce module, j'ai ajouté une modèle de plant de rose créé par L-système dans la chambre de culture 3.1.1, puis lancé une simulation avec  $10^9$  photons. Voici le modèle utilisé dans la simulation.

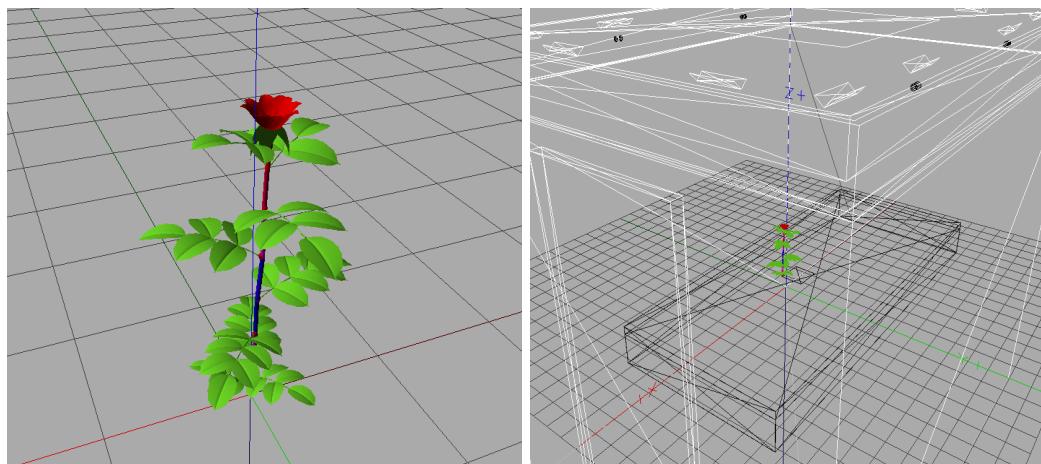


FIGURE 3.26 – Le modèle de plant de rose et la chambre de culture

#### Résultat

Voici la distribution de photons sur des organes de ce plant.

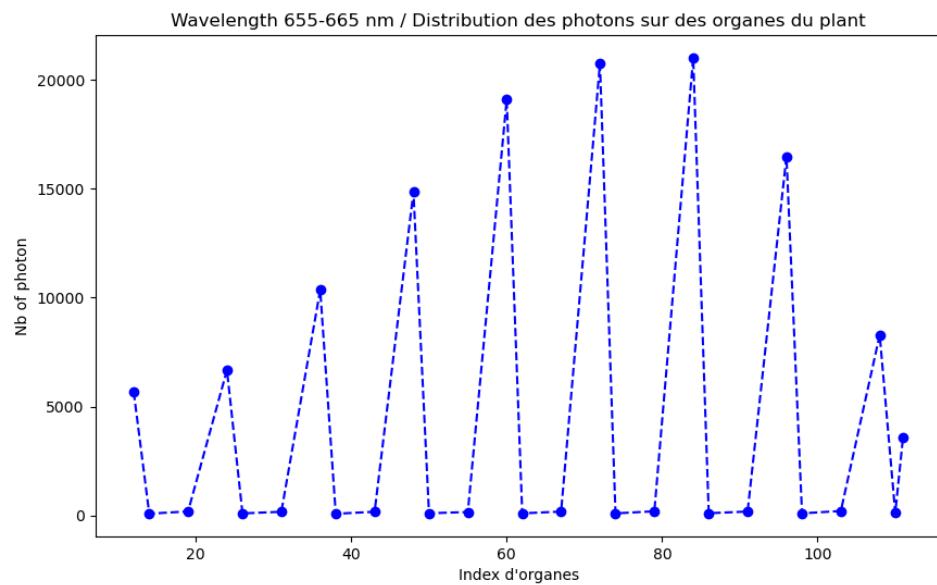
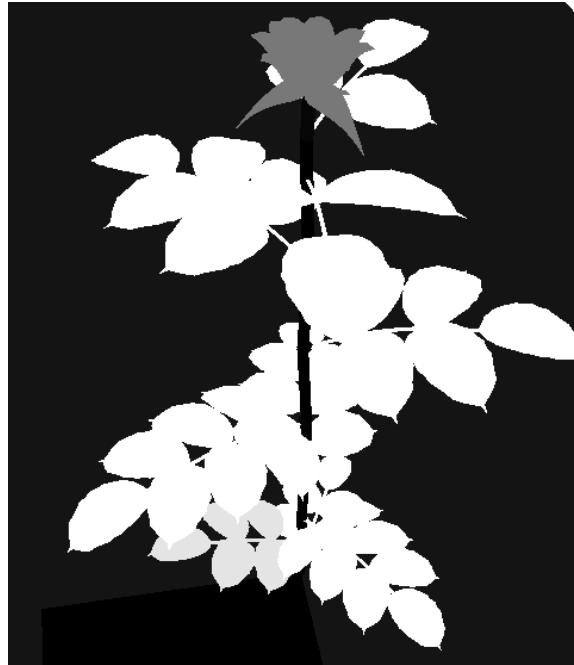


FIGURE 3.27 – La distribution des photons sur des organes du plant

Il reste à évaluer le résultat de ce module avec les données des biologistes.

# Chapitre 4

## Conclusion

La plupart des objectifs du stage ont été atteints. Nous disposons maintenant d'un moteur de simulation d'interaction de la lumière qui fonctionne très bien sur le CPU. Les résultats de ce moteur sont très proches de ceux du moteur SEC2, et le temps de simulation est considérablement réduit, environ 7 fois plus rapide.

Cependant, il reste encore à évaluer notre moteur sur des simulations avec d'autres types de capteurs (tels que des quads et des organes de plantes). De plus, il faut implémenter le module Photon Mapping sur Mitsuba pour évaluer les performances. D'autre part, nos fichiers d'entrée sont assez nombreux et compliqués, donc il est nécessaire de définir une bonne structure de données d'entrée qui soit facile à utiliser pour les utilisateurs et permette un débogage efficace. Dans la dernière période de stage, je vais essayer de résoudre ces problèmes.

Les perspectives pour la poursuite du stage incluent l'utilisation de la structure de données de l'arbre BSP (Binary Space Partitioning) pour accélérer le temps de simulation. Par défaut, Embree utilise l'arbre BVH (Bounding Volume Hierarchies) pour calculer l'intersection entre un rayon et des triangles. L'avantage de cette structure est qu'elle ne prend pas beaucoup de temps pour reconstruire l'arbre dans une scène dynamique. Cependant, l'inconvénient est que les boîtes englobantes peuvent se chevaucher, ce qui peut augmenter le temps nécessaire pour calculer l'intersection. En revanche, avec le BSP, les boîtes englobantes ne peuvent pas se chevaucher, ce qui nécessite plus de temps pour reconstruire l'arbre, mais réduit le temps de calcul des intersections. Dans notre simulation, la scène est immuable, donc la structure de l'arbre BSP pourrait être plus appropriée.

En somme, ce stage m'a permis de découvrir le travail en équipe et m'a également enseigné comment m'intégrer dans un projet informatique en développement depuis plusieurs années. De plus, j'ai acquis beaucoup de nouvelles connaissances telles que le modèle d'illumination, la BRDF, le photon mapping, etc., ce qui m'a aidé à approfondir ma compréhension des bases de la réalité virtuelle. Ces expériences seront bénéfiques pour ma carrière professionnelle dans les années à venir.

## Annexe A

# Glossaire

- **Cirad** : Centre de coopération internationale en recherche agronomique pour le développement.
- **Ray tracing** : Terme générique qui recouvre les techniques de rendu qui trace des rayons entre une scène et la caméra.
- **Photon Mapping** : En imagerie numérique, le photon mapping ou placage de photons est un algorithme d'illumination globale fondé sur le lancer de rayon (ray tracing) utilisé pour simuler l'interaction de la lumière avec différents objets de manière réaliste.
- **BRDF** : Bidirectional Reflectance Distribution Function. Fonction permettant de calculer la lumière réfléchit par une surface.
- **PAR (Photosynthetically Active Radiation)** : désigne la gamme spectrale (bande d'ondes) du rayonnement solaire de 400 à 700 nanomètres que les organismes photosynthétiques sont capables d'utiliser dans le processus de photosynthèse.
- **CPU (Central Processing unit)** : Un composant matériel d'ordinateur qui constitue l'unité centrale de traitement. Les ordinateurs convertissent les données en signaux numériques et y effectuent des opérations mathématiques.
- **GPU (Graphics Processing Unit)** : Une puce informatique disposée sur la carte graphique, qui vise à optimiser le rendu d'images, l'affichage 2D et 3D, ou encore les vidéos.
- **BSP (Binary Space Partitioning)** : Une structure de données utilisée pour accélérer le ray tracing
- **BVH (Bounding Volume Hierarchy)** : Une structure de données utilisée pour accélérer le ray tracing.

## Annexe B

# Annexes

36



FIGURE B.1 – Diagramme de Gantt

# Bibliographie

- [1] B. T. PHONG, « Illumination for computer generated pictures, » *Communications of the ACM*, t. 18, p. 311-317, juin 1975. DOI : 10.1145/360825.360839.
- [2] S. CRONE, « Radiance users manual, » *Architectural Dissertation*, t. 2, 1992.
- [3] G. J. WARD, « The RADIANCE lighting simulation and rendering system, » in *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, 1994, p. 459-472.
- [4] J. ARVO, A. K. PHIL DUTRE, A. O. HENRIK WANN JENSE et P. S. MATT PHARR, « Monte Carlo Ray Tracing, » *Siggraph 2003 Course 44*, juill. 2003. adresse : [https://www.researchgate.net/profile/Peter-Shirley-2/publication/340769537\\_Monte\\_Carlo\\_Ray\\_Tracing\\_Siggraph\\_2003\\_Course\\_44/](https://www.researchgate.net/profile/Peter-Shirley-2/publication/340769537_Monte_Carlo_Ray_Tracing_Siggraph_2003_Course_44/).
- [5] M. CHELLE, C. RENAUD, S. DELEPOULLE et D. COMBES, « Modeling light phylloclimate within growth chambers, » in *5. International Workshop*, sér. Functional-Structural Plant Models. Abstracts of papers and posters, Napier, New Zealand, nov. 2007, np. adresse : <https://hal.science/hal-01191959>.
- [6] C. PRADAL, S. DUFOUR-KOWALSKI, F. BOUDON, C. FOURNIER et C. GODIN, « OpenAlea : a visual programming and component-based software platform for plant modelling, » déc. 2008, p. 751-760. DOI : 10.1071/FP08084.
- [7] C. PRADAL, F. BOUDON, C. NOUGUIER, J. CHOPARD et C. GODIN, « PlantGL : A Python-based geometric library for 3D plant modelling at different scales, » *Graphical Models*, t. 71, n° 1, p. 1-21, 2009, ISSN : 1524-0703. DOI : <https://doi.org/10.1016/j.gmod.2008.10.001>. adresse : <https://www.sciencedirect.com/science/article/pii/S1524070308000143>.
- [8] F. BOUDON, C. PRADAL, T. COKELAER, P. PRUSINKIEWICZ et C. GODIN, « L-Py : An L-System Simulation Framework for Modeling Plant Architecture Development Based on a Dynamic Language, » *Frontiers in plant science*, t. 3, p. 76, mai 2012. DOI : 10.3389/fpls.2012.00076.
- [9] I. WALD, S. WOOP, C. BENTHIN, G. S. JOHNSON et M. ERNST, « Embree : a kernel framework for efficient CPU ray tracing, » *ACM Trans. Graph.*, t. 33, n° 4, juill. 2014, ISSN : 0730-0301. DOI : 10.1145/2601097.2601199. adresse : <https://doi.org/10.1145/2601097.2601199>.
- [10] B. LEGGIO, J. LAUSSU, A. CARLIER, C. GODIN, P. LEMAIRE et E. FAURE, « MorphoNet : an interactive online morphological browser to explore complex multi-scale data, » *Nature Communications*, t. 10, p. 2812, juin 2019. DOI : 10.1038/s41467-019-10668-1.

## BIBLIOGRAPHIE

---

- [11] A. BESNIER, « Rapport de stage, » 2023. adresse : [https://github.com/minhlucky9/photon\\_mapping/blob/main/Rapport\\_de\\_Stage\\_M2\\_BESNIER\\_Aurelien.pdf](https://github.com/minhlucky9/photon_mapping/blob/main/Rapport_de_Stage_M2_BESNIER_Aurelien.pdf).