Rapport de Spécifications

Global Illumination with Radiance Regression Functions

Etudiants :
Yannick BERNARD
Pierre GUERINEAU
Kevin MENIEL
Romain MOUTRILLE
Matthias ROVES

Encadrant :
Mathias PAULIN

8 décembre 2016

Table des matières

1 Introduction

L'éclairage indirect est un des problèmes à résoudre lors du rendu physiquement réaliste d'une scène en temps réel. Un effet d'éclairage global détaillé s'obtient grâce à de nombreux rebonds effectués par la lumière, ceci engendrant un temps de calcul coûteux. Pour contourner ce problème, nous utiliserons la « Fonction de Régression de la Radiance » *(RRF : Radiance Regression Function)* nous permettant ainsi de calculer l'éclairage indirect pour chaque point des surfaces en connaissant la direction de la caméra et les conditions lumineuses.

Le projet a pour objectif de pouvoir effectuer le rendu physiquement réaliste d'une scène en temps réel avec calcul de l'éclairage global par une fonction de régression de radiance. Cette fonction sera modélisée par un réseau de neurones qui permettra donc un calcul de l'éclairage indirect en temps réel lors du rendu.

Notre projet sera décomposé en trois parties. Premièrement, pour une scène donnée, nous calculerons la valeur de l'éclairage indirect en chaque point. Ces valeurs constitueront les données d'apprentissage pour notre réseau de neurones. La seconde étape est la création du réseau de neurones ainsi son entraînement au calcul de l'éclairage indirect. Pour finir, nous utiliserons un moteur 3D qui calculera l'éclairage direct, nous intégrerons ensuite notre réseau de neurones pour y additionner l'éclairage indirect. Ceci permettra ainsi un rendu réaliste en temps réel avec éclairage direct et indirect.

2 Cahier des charges

2.1 Parties prenantes

- **Client** :
  - PAULIN Mathias
    - *Contact : mathias.paulin@irit.fr*
- **Equipe** :
  - BERNARD Yannick
    - *Contact : yannick.bernard@univ-tlse3.fr*
  - MOUTRILLE Romain
    - *Contact : romain.moutrille@univ-tlse3.fr*
  - MENIEL Kevin
    - *Contact : kevin.meniel@gmail.com*
  - ROVES Matthias
    - *Contact : matthias.roves@hotmail.fr*
  - GUERINEAU Pierre
    - *Contact : pierre.gue@hotmail.fr*

2.2 Exigences fonctionnelles

Le système doit permettre l'affichage d'une scène avec un éclairage direct et indirect en temps réel. L'implémentation devra se baser sur les méthodes utilisées dans le papier "Global Illumination with Radiance Regression Function".

	Exigences	Priorité
I	**Extraction des données**	
I.1	La fonction de radiance va être apprise par morceaux, les données seront générées pour des sous-régions de la scène.	🔴
I.2	Une version modifiée de PBRT calculant uniquement la composante indirecte devra être réalisée.	🔴
I.3	Des formats de stockage interne et externe devront être définis.	🔴
II	**Apprentissage**	
II.1	Le réseau de neurone doit être créé grâce à la bibliothèque OpenNN.	🟡
II.2	Le réseau de neurone devra calculer l'éclairage indirect d'un point à partir de la position de ce point, de sa normale, de la direction de la lumière et de la direction de la caméra.	🟠
II.3	Le réseau de neurone devra calculer l'éclairage indirect d'un point à partir des paramètres de la BRDF.	🟢
II.4	Le réseau de neurone doit pouvoir calculer l'éclairage indirect d'un point avec une erreur quadratique inférieure à un seuil défini par l'utilisateur.	🔴
III	**Rendu**	
III.1	Le rendu de l'éclairage indirect doit se faire en temps réel grâce l'évaluation des données par un réseau de neurones.	🔴
III.2	Le rendu de l'éclairage indirect de la scène doit être similaire à celui d'un moteur de rendu hors ligne (PBRT, Mitsuba).	🟡
III.3	Le rendu de l'éclairage direct doit être fait en temps réel.	🔴
III.4	Le rendu de l'éclairage direct doit utiliser le même type de BRDF que l'éclairage indirect.	🔴
III.5	Le rendu de l'éclairage direct doit utiliser le même type de source que l'éclairage indirect.	🟡

Priorité : 🟢 faible 🟡 moyenne 🟠 forte 🔴 indispensable

2.3 Tests de recette

Les tests seront effectués sur la scène Cornell Box vide, avec un seuil d'erreur fixé à 0.05.

2.3.1 Similarité de rendu

Pour tester la similarité entre deux rendus fait par deux moteurs différents, on va calculer la distance euclidienne pixel par pixel dans l'espace LAB, puis calculer la RMSE sur le Δ_E de l'image résultat, et les considérer comme similaire si RMSE < 1.

2.3.2 Test du temps réel

Il sera vérifié lors de l'exécution du programme sur une machine de la salle U3-112 que le nombre d'images calculées par seconde restera supérieur à 15.

3 Découpage du projet

3.1 Vue générale du système

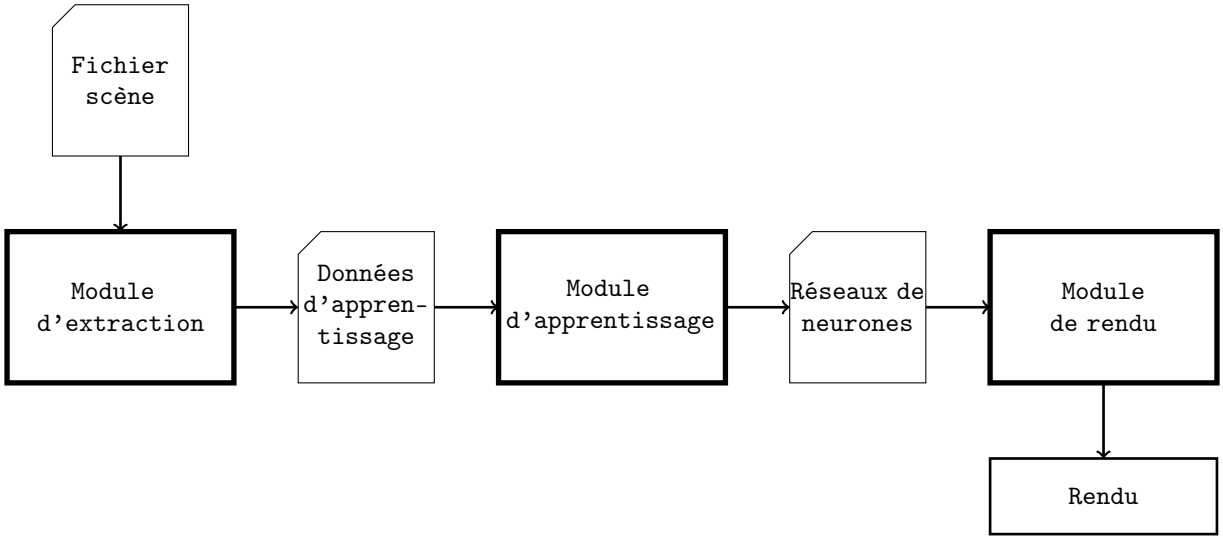


FIGURE 1 – Vue générale du système

3.1.1 Fonctions

→ Module d'extraction

Le module d'extraction va permettre à partir d'un fichier scène de récupérer des données d'apprentissage. Celles-ci seront constituées des paramètres d'entrée du réseau de neurones *(voir* FIGURE 2*)* de chaque configuration *(paires (position_caméra, position_source))* ainsi que l'éclairage indirect calculé par le moteur pour ces configurations.

→ Module d'apprentissage

Le module d'apprentissage va effectuer l'apprentissage du réseau de neurones avec les données précédemment extraites. Le réseau de neurones va donc apprendre quel éclairement doit être obtenu en fonction des paramètres des données extraites, nous obtiendrons donc un réseau de neurones opérationnel et vraisemblablement capable de déterminer l'éclairement de n'importe quel point de la scène apprise.

→ Module de rendu

Le module de rendu va utiliser le réseau de neurones construit précédemment pour déterminer l'éclairage indirect de chaque fragment lors de la phase de rendu.

3.2 Tests unitaires

Les tests unitaires seront effectués grâce à Google Test.

3.2.1 Module d'extraction

Vérification de l'exactitude des données

- Execution du module sur une scène très simple et vérification des résultats. *scène simple : trois plans perpendiculaires, monochromes et de couleurs différentes, avec une source placée de manière à éclairer les 3 plans.*
- Modification d'un paramètre *(ex :couleur, position lumière...)* et vérification des résultats.

Capacité de traiter de n'importe quelle scène

— Execution du module sur différentes scènes (Cornell Box avec différents objets plus ou moins complexes), de configurations différentes.

3.2.2 Module d'apprentissage

Vérification de la déduction

— Création de 3 données d'apprentissage, dont 2 très proches n_1 et n_2, et une très éloignée des 2 autres n_3. Apprentissage du réseau de neurones avec n_1 et n_3, puis donnée en entrée n_2, vérification de la déduction de n_1.

Vérification de l'exactitude des données

— Application de la cross-validation : Apprentissage avec 70% des données, et test sur les 30% restants, comparaison des résultats.

3.2.3 Module de rendu

Vérification de l'exactitude des données

— Comparaison avec/sans ajout de la composante indirecte.

— Vérification visuelle.

— Comparaison d'une frame du rendu avec un rendu équivalent effectué avec PBRT.

3.3 Tâches

3.3.1 Module d'extraction

Le module d'extraction prend en entrée une scène et renvoie en sortie les données d'apprentissage *(voir* FIGURE 2*)*. Il est découpé en 3 tâches :

1 - Définition des échantillons

La première tâche consiste à récupérer un certain nombre de configurations de scènes *(paires* `(position_caméra, position_source)` *)* de la scène avec un échantillonnage par l'hypercube latin. Le nombre de positions de caméra et de positions des sources sera défini par l'utilisateur.

2 - Modification de PBRT

Nous devons modifier le moteur de rendu PBRT de manière à utiliser les informations précédentes pour effectuer les rendus et récupérer les données d'apprentissage *(voir* FIGURE 2*)*.
Nous allons donc créer une caméra sphérique pour pouvoir échantillonner les directions visibles depuis la caméra, grâce à l'échantillonnage de Fibonacci. Nous ajouterons une heuristique afin de ne pas considérer les directions inutiles (intérieurs d'objets, faces non visibles...).

3 - Calcul de l'éclairage indirect

Nous allons créer un nouvel intégrateur PBRT se basant sur du path tracing, qui devra échantillonner la luminance sur le point d'intersection avec le rayon plutôt que sur le pixel. On utilisera un échantillonnage par importance.

4 - Stockage

Nous allons utiliser un système de fichiers binaires pour stocker les données d'apprentissage que nous allons compresser.

3.3.2 Module d'apprentissage

Le module d'apprentissage prendra en entrée les données d'apprentissage, et donnera en sortie une structure de la scène divisée en régions avec un réseau de neurones par région.

1 - On créé le réseau de neurones selon la structure suivante :

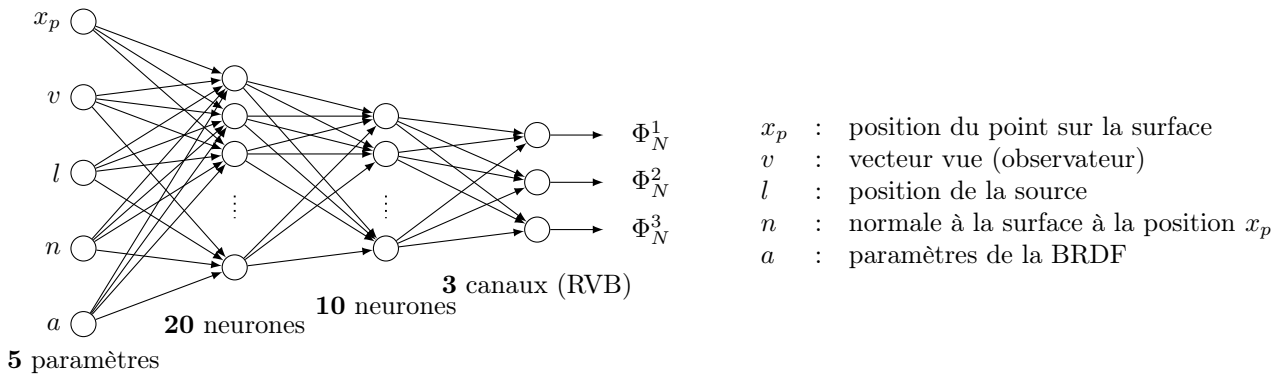


FIGURE 2 – Schéma de la structure du réseau de neurones

2 - Nous allons récupérer les données d'apprentissage grâce à la bibliothèque `stxxl`, et nous allons les organiser en arbre, les plans de coupe étant définis par une minimisation de l'erreur du réseau de neurones (Cross-Validation), les feuilles ayant chacune leur propre réseau de neurones. On ne récupèrera que les réseaux de neurones.

3 - Les données des réseaux de neurones seront stockées en binaire.

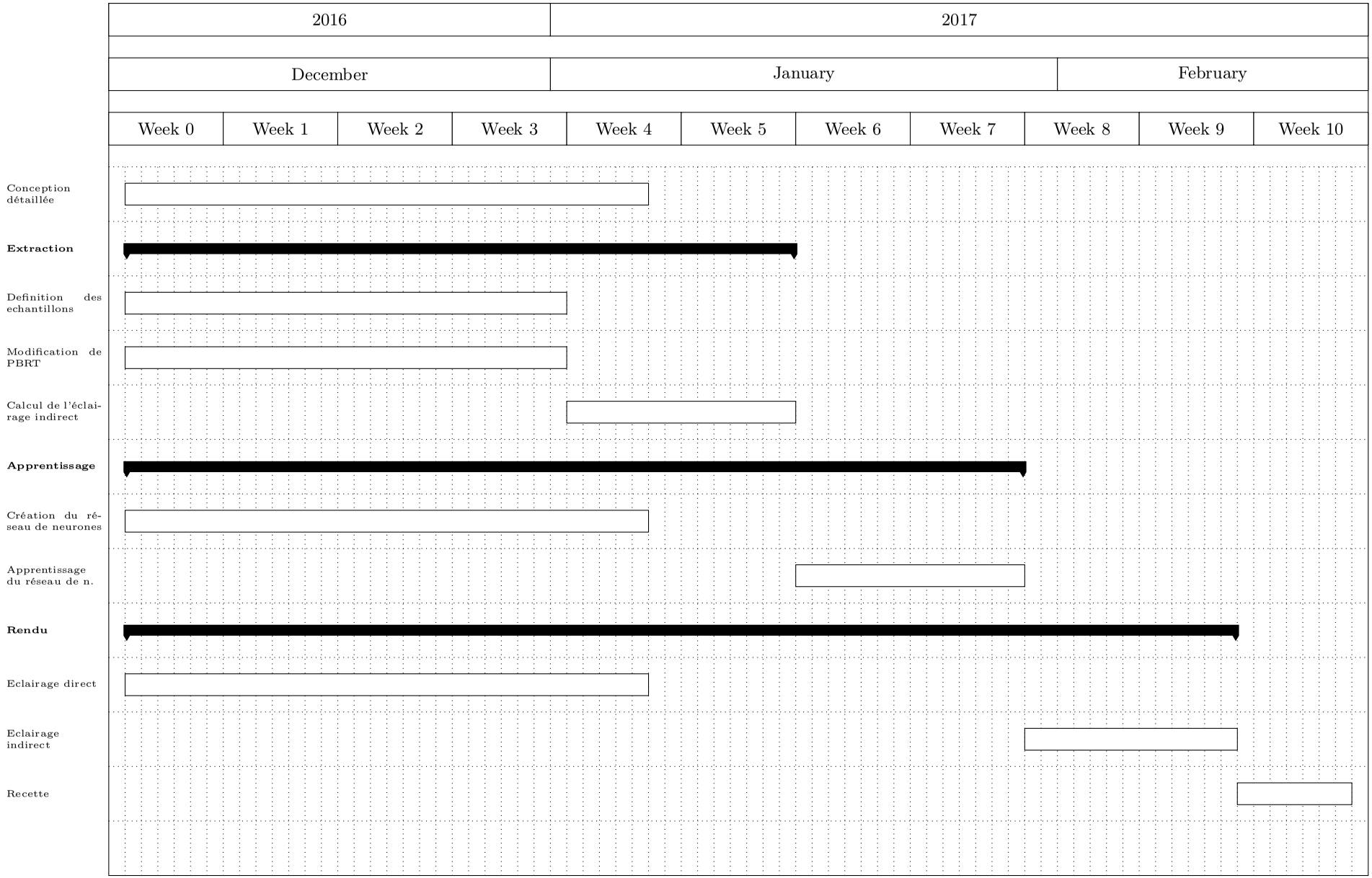
3.3.3 Module de rendu

Le module de rendu prend en entrée les réseaux de neurones créés à l'étape précédente, et fournit un rendu.

1. Nous devrons dans un premier temps prendre comme base un moteur 3D que nous avons déjà créé.
2. Il faudra ensuite intégrer l'évaluation du réseau de neurones grâce aux 5 paramètres d'entrée du réseau récupérés lors du rendu *(voir* FIGURE 2*)*, et en utilisant GLSL.
3. Nous devrons ensuite ajouter la composante indirecte dans l'évaluation de l'éclairage.

4 Planning prévisionnel

NOM	DATE DÉBUT	DATE FIN
Conception détaillée	06/12/16	06/01/17
Extraction	06/12/16	15/01/17
Définition des échantillons	06/12/16	01/01/17
Modification de PBRT	06/12/16	01/01/17
Calcul de l'éclairage indirect	02/01/17	15/01/17
Apprentissage	06/12/16	29/01/17
Création du réseau de n.	06/12/16	06/01/17
Apprentissage du réseau de n.	16/01/17	29/01/17
Rendu	06/12/16	11/02/17
Eclairage direct	06/12/16	06/01/17
Eclairage indirect	30/01/17	11/02/17
Recette	12/02/17	18/02/17


2016
2017
December
January
February
Week 0
Week 1
Week 2
Week 3
Week 4
Week 5
Week 6
Week 7
Week 8
Week 9
Week 10
Conception détaillée
Extraction
Definition des echantillons
Modification de PBRT
Calcul de l'éclairage indirect
Apprentissage
Création du réseau de neurones
Apprentissage du réseau de n.
Rendu
Eclairage direct
Eclairage indirect
Recette

4.1 Dépendances

L'apprentissage nécessite qu'il y ait eu préalablement un échantillonnage de la scène et une extraction des données d'apprentissage.
Le rendu de la scène nécessite qu'il y ait eu préalablement un apprentissage du réseau de neurones.
Extraction > Apprentissage > Rendu
Le développement de chaque module peut se faire en parallèle malgré ces dépendances.

4.2 Analyse des risques

n°	Type	Description	Proba.	Impact	Solutions	
					Evité	Accepté
1	–	Mise hors tension des machines lors de l'extraction ou de l'apprentissage	🟡	🟠	Réaliser des sauvegardes toutes les 10 minutes de l'état du programme d'extraction ou d'apprentissage	Recommencer l'extraction ou l'appprentissage
2	–	Pertes de données (facteur multiples) lors de l'extraction ou de l'apprentissage	🟡	🟠	Réaliser des sauvegardes toutes les 10 minutes de l'état du programme d'extraction ou d'apprentissage	Recommencer l'extraction ou l'apprentissage
3	–	Durée trop grande d'apprentissage	🟠	🟠	Commencer l'apprentissage le plus tôt possible et en connaître le temps estimé	On n'exploite que les données valides
4	–	Difficultés d'utilisation de PBRT	🟢	🔴	Se documenter et se former au plus tôt sur PBRT	Utiliser Mitsuba
5	–	Ne pas avoir une base fonctionnelle de notre moteur	🟢	🔴	Réaliser le module d'évaluation au plus tôt	Utiliser le Radium Engine
6	–	Non respect du planning	🔴	🟢	Respecter le planning	Revoir le planning
7	–	Risque de perte d'un membre de l'équipe	🟢	🟠	Organiser une réunion	Ré-attribuer les tâches au sein de l'équipe
8	+	Risque de changement des ordinateurs de la salle 112, pour des configurations plus puissantes	🟡	🟡	-	Sauter de joie

Probabilité & Impact : 🟢 faible 🟡 moyen 🟠 fort 🔴 très fort