Rapport de stage de L3

Rendu des matériaux participatifs sur GPU

Guillaume Cordonnier

20 août 2013

Résumé

L'effet de dispersion de la lumière dans la matière est un problème récurrent en informatique graphique. Nous cherchons à obtenir le rendu des matériaux participatifs, dans lesquels la lumière transite dans la matière en interagissant. Le rendu est effectué à l'aide du GPU, ce qui apporte l'avantage d'un très faible temps de calcul, moyennant une faible connaissance de la géométrie de l'objet : lors du traitement d'un sommet ou d'un pixel, nous n'avons pas d'information sur les surfaces voisines. Nous utilisons des approximations physiques comme le dipôle pour obtenir un rendu le plus réaliste possible, tout en supposant ne pas avoir d'information particulière sur l'objet (pré-calculs ou découpage de texture spécifique), pour que notre modèle puisse être adapté à toutes sortes d'objets et que ceux-ci soient déformables. Nous proposons une implémentation pour la partie diffuse de l'éclairage, au premier ordre avec le *single scattering*, et aux ordres supérieurs avec l'approximation du dipôle pour le *multiple scattering*. Nous proposerons enfin des techniques d'intégration et d'échantillonnage pour améliorer ce calcul.

Table des matières

Chapitre 1

Introduction

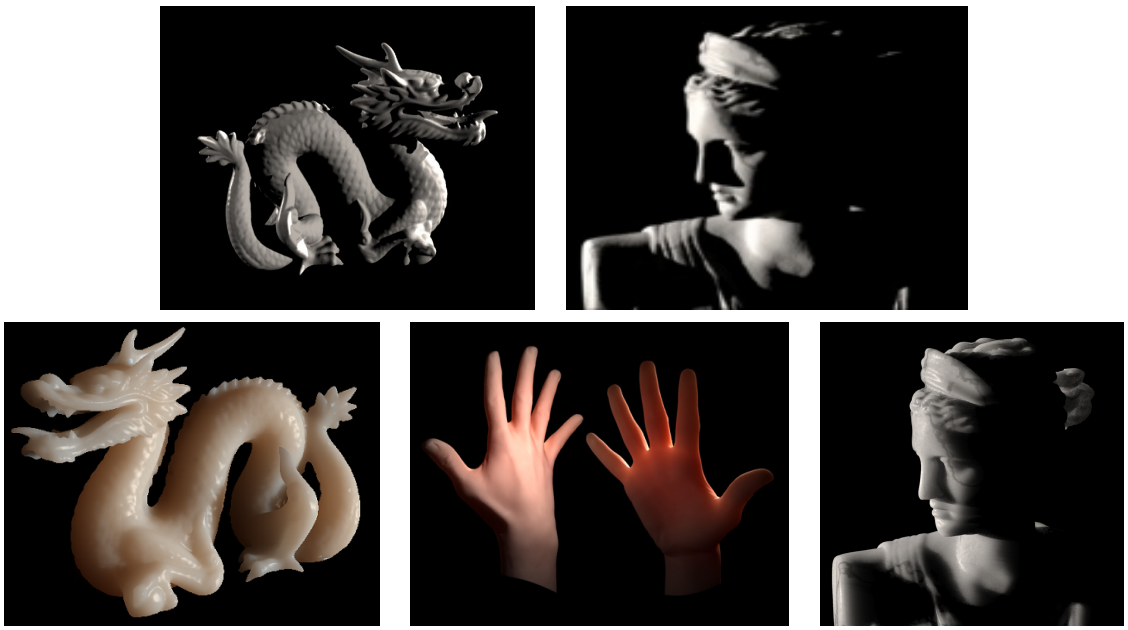


FIGURE 1.1 – Rendu classique (en haut), par rapport à un rendu avec subsrface scattering (en bas).

Dans ce rapport, je vous présente mon travail dans le cadre du stage de L3, dans l'équipe MAVERICK de l'INRIA Grenoble.
L'objectif de ce stage était de chercher comment calculer le rendu des matériaux participatifs sur GPU (le processeur graphique).
Ces matériaux ont une propriété de dispersion (*scattering*) de la lumière : celle ci arrive en un point de l'objet, interagit sous la surface de l'objet, et ressort en un autre point. Ces matériaux sont très courants, comme par exemple la peau, le lait ou le café.
Nous nous plaçons dans le cadre des matériaux participatifs homogènes. Par exemple, notre modèle ne pourra pas gérer des effets de fumée ou les nuages, dont la densité dépend fortement de la position.
L'intégration de ces matériaux dans les rendus de scènes 3D a soulevé de nombreuses questions de recherche, et beaucoup de solutions donnent des résultats très convaincants mais moyennant un important temps de calcul.
Le problème soulevé par ce stage était de trouver comment utiliser les techniques de rendu temps réel pour pouvoir afficher ces matériaux de manière interactive ou s'y rapprochant. Nous cherchons de plus à obtenir un aspect similaire à ceux proposés par les modèles physiques réalistes.
Ce compromis entre le temps de calcul et la précision constituait la difficulté majeure. Le rendu sur GPU rend possible un calcul temps réel grâce à une importante parallélisation, mais en contrepartie elle apporte des contraintes, comme le fait que les accès mémoires (textures et tableaux) sont de loin les plus coûteuses en temps de calcul, et que nous n'avons que des informations locales (par pixel ou par sommet), et non globales (sur l'ensemble de l'objet).
Le fait de pouvoir obtenir le rendu de matériaux participatifs en temps réel a plusieurs intérêts. Tout d'abord il permet de visualiser facilement les variations d'éclairages de ces matériaux en changeant

interactivement les propriétés de l'environnement et de l'objet, ensuite il permet d'améliorer le réalisme et donc l'immersion dans des logiciels de simulations (industries, architecture) ou dans le domaine vidéo-ludique.
Dans notre approche, nous séparons un premier ordre dans la simulation de la dispersion, appelé *single scattering*, des ordres supérieurs (*multiple scattering*), et nous sommons le tout dans le résultat final. Nous verrons tout d'abord dans quel contexte se situe ce travail, puis dans une partie suivante, un travail préliminaire éloigné du problème mais tout de même nécessaire. Nous entrerons ensuite dans le cœur du problème avec une implémentation du *single scattering*, suivi du traitement du *multiple scattering*, et enfin, nous montrerons les améliorations apportées pour rendre le calcul du *multiple scattering* plus précis sur GPU.

1.1 Sources

Moteur de rendu pour les images de référence : *Mitsuba* (http://www.mitsuba-renderer.org/).
Modèles :

— http://graphics.stanford.edu/data/3Dscanrep/
— http://www.cc.gatech.edu/projects/large_models/

Les paramètres des matériaux ont été obtenus dans Jensen *et al.* [7].

Chapitre 2

Contexte

2.1 Éléments de rendu 3D

Le rendu 3D temps réel se fait en discrétisant les surfaces en polygones, principalement des triangles. Ces surfaces sont traitées par le GPU, qui applique certaines transformations à chaque sommet. Le GPU interpole ensuite les données de ces sommets pour calculer la valeur des pixels de chaque polygone.
Les développeurs peuvent contrôler les calculs sur les sommets et les pixels à l'aide de programmes, appelés *shaders*.
Les *Vertex Shaders* reçoivent les sommets et leur donnent leur position géographique finale et les transformations liées à la perspective, puis les *Pixel Shaders* (ou *Fragment Shaders*) effectuent les calculs sur les pixels, notamment pour l'application de textures et l'éclairage.
Lors de ce stage, j'ai travaillé avec le standard graphique *OpenGL*, dont le langage de shaders est *GLSL*, un langage proche du C avec des instructions graphiques et des types géométriques appropriés.
J'ai utilisé des FBO (*frame buffer objects*), qui permettent d'obtenir un premier rendu de la scène dans une texture, qui pourra être réutilisée dans un rendu ultérieur pour avoir des informations plus globales sur la scène, et ce jusqu'au rendu final affiché à l'écran.
La principale contrainte dans l'écriture d'un shader est le temps d'accès en mémoire particulièrement long. Il faut donc minimiser le nombre de lectures sur les textures et l'utilisation des tableaux.

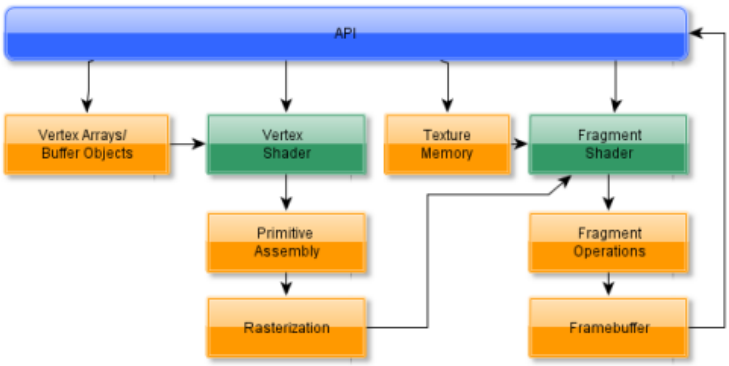


FIGURE 2.1 – Pipeline graphique

2.2 Notations

f_r	BRDF
S	BSSRDF
$\boldsymbol{\omega}_i$, $\boldsymbol{\omega}_o$	Vecteurs incident et sortant de la lumière
$\mathbf{x}_i$, $\mathbf{x}_o$	Points d'entrée et de sortie de la lumière
L_i, L_o	Irradiance lumineuse entrante et sortante
Φ_i	Flux incident de la lumière
$\mathbf{n}_i$, $\mathbf{n}_o$	Normale à la surface au point d'entrée ou de sortie de la lumière
Q	Distribution volumique de la source
p	Fonction de phase
g	Valeur moyenne de l'angle de diffusion
σ_a	Coefficient d'absorption
σ_s	Coefficient de dispersion
σ_t	Coefficient d'extinction
l	Libre parcours moyen
α	albédo
ϕ	Irradiance scalaire
$\mathbf{E}$	Irradiance vectorielle
σ'_s	Coefficient de dispersion réduit
σ'_t	Coefficient d'extinction réduit
l'	Libre parcours moyen réduit
α'	albédo réduit
D	Constante de diffusion
F_{dr}	Terme de Fresnel moyen en réfraction
F_{refr}	Terme de Fresnel en réfraction
F_{refl}	Terme de Fresnel en réflexion
η	Rapport entre les indices de réfraction
k_{spec}	Contribution spéculaire
S_d	Contribution diffuse de la BSSRDF
R_d	BSSRDF diffuse
$S^{(1)}$	Single scattering
σ_{tc}	Coefficient d'extinction combiné
σ_{tr}	Coefficient d'extinction effectif

TABLE 2.1 – Notations utilisées

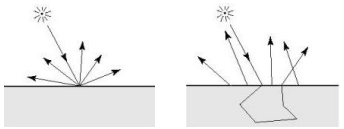
2.3 Modélisation physique des matériaux participatifs

L'approche classique pour définir l'aspect de la surface d'un objet est d'utiliser une fonction, la BRDF (*bidirectional reflectance distribution function*) : $f_r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$, où $\boldsymbol{\omega}_i$ est le vecteur donnant la direction de la lumière entrante, et $\boldsymbol{\omega}_o$ est le vecteur donnant la direction de la lumière sortante.
On peut alors calculer l'irradiance lumineuse sortante du point considéré dL_o en fonction du flux de lumière incident $d\Phi_i$:

$$\mathrm{d}L_o(\boldsymbol{\omega}_o) = f_r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\,\mathrm{d}\Phi_i(\boldsymbol{\omega}_i)$$

Cette fonction est une approximation d'une fonction plus précise, la BSSRDF (*bidirectional surface scattering distribution function*), qui prend aussi en compte un point d'entrée de la lumière $\mathbf{x}_i$, et un point de sortie $\mathbf{x}_o$:

$$\mathrm{d}L_o(\mathbf{x}_o, \boldsymbol{\omega}_o) = S(\mathbf{x}_i, \boldsymbol{\omega}_i, \mathbf{x}_o, \boldsymbol{\omega}_o)\,\mathrm{d}\Phi_i(\mathbf{x}_i, \boldsymbol{\omega}_i)$$



(a) (b)

FIGURE 2.2 – BRDF (a), BSSRDF (b)

Cette fonction nous permet de retrouver l'aspect dispersif de la lumière dans les matériaux participatifs, où l'illumination perçue d'un point dépend de l'éclairage reçu par des points voisins :

$$L_o(\mathbf{x}_o, \omega_o)) = \int_A \int_{\Omega^+} S(\mathbf{x}_i, \omega_i, \mathbf{x}_o, \omega_o) L_i(\mathbf{x}_i, \omega_i) (\mathbf{n}_i \cdot \omega_i) \, \mathrm{d}\omega_i \, \mathrm{d}A(\mathbf{x}_i)$$

L'équation de transfert radiatif exprime la propagation de la lumière dans les matériaux participatifs :

$$(\omega \cdot \nabla) L(\mathbf{x}, \omega) = -\sigma_t L(\mathbf{x}, \omega) + \sigma_s \int_\Omega p(\omega, \omega') L(\mathbf{x}, \omega') \, \mathrm{d}\omega' + Q(\mathbf{x}, \omega)$$

Nous voyons apparaître dans cette équations certains paramètres des matériaux participatifs :

- σ_a, le coefficient d'absorption, correspond à l'atténuation lumineuse lors de la traversée du matériau,
- σ_s, le coefficient de dispersion, correspond à la perte lumineuse liée à la dispersion,
- $\sigma_t = \sigma_a + \sigma_s$ est le coefficient d'extinction.

Nous utiliserons aussi $l = \frac{1}{\sigma_t}$, le libre parcours moyen de la lumière dans le matériau, et l'albédo : $\alpha = \frac{\sigma_s}{\sigma_t}$.
La fonction de phase p est aussi caractéristique et décrit comment la lumière se disperse. Nous retrouverons g, la valeur moyenne du cosinus de l'angle de diffusion :

$$g = \int_\Omega (\omega \cdot \omega') \, p(\omega \cdot \omega') \, \mathrm{d}\omega'$$

Une première approximation est faite ensuite en supposant que la distribution lumineuse dans les matériaux fortement dispersifs devient isotrope (plus de détails dans Jensen *et al.* [7]) :

$$L(\mathbf{x}, \omega) = \frac{1}{4\pi} \phi(\mathbf{x}) + \frac{3}{4\pi} \omega \cdot \mathbf{E}(\mathbf{x})$$

Cette expression fait intervenir l'irradiance scalaire,

$$\phi(\mathbf{x}) = \int_\Omega L(\mathbf{x}, \omega) \, \mathrm{d}\omega$$

et l'irradiance vectorielle,

$$\mathbf{E}(\mathbf{x}) = \int_\Omega L(\mathbf{x}, \omega) \, \omega \, \mathrm{d}\omega$$

Dans le cadre de cette approximation, nous utilisons des coefficients réduits : $\sigma'_s = \sigma_s (1 - g)$ et $\sigma'_t = \sigma'_s + \sigma_a$. En effet, la lumière devenant anisotrope, seule la dispersion vers l'arrière aura une influence. Cette approximation donnera une équation dite équation de diffusion, qui peut être réduite grâce aux conditions limites, à :

$$\phi(\mathbf{x}_s) - 2AD \, (\mathbf{n} \cdot \nabla) \, \phi(\mathbf{x}_s) = 0$$

Avec $D = \frac{1}{3\sigma'_t}$ la constante de diffusion et $A = \frac{1+F_{dr}}{1-F_{dr}}$, où F_{dr} est un terme de Fresnel moyen du à la réfraction :

$$F_{dr} = \int_{\Omega^+} F_{refr}(\eta, \mathbf{n} \cdot \omega') \, \mathrm{d}\omega'$$

Dans le cas général, pour des matériaux de volume fini, l'équation de diffusion n'a pas de solution analytique. Nous verrons plus loin une approximation classique permettant d'obtenir une BSSRDF.

2.4 État de l'art

Ce problème se situe dans le contexte plus général du *subsurface scattering*, sur lequel beaucoup de recherches ont été faites.
Le cas particulier des matériaux participatifs a été simulé en premier lieux par des méthodes de type Monte-Carlo, qui donne toujours les meilleurs résultats car sans approximation par rapport aux lois physiques, mais pour un énorme temps de calcul. Une méthode souvent utilisée est l'approximation du dipôle proposée par Jensen *et al.* [7], amélioré par d'Eon sous le nom de *better dipole* [4]. Ce sont ces approximations que nous utiliserons. Jensen a proposé une méthode plus efficace pour l'utilisation du dipôle [6], mais toujours dans un contexte non interactif.
Certaines approches visent un rendu en temps réel, à l'aide de rendu en espace de texture [3], ce qui n'est pas possible pour tous les maillages. Il est aussi possible d'effectuer des pré-calculs sur les modèles [2], ce qui rend difficile l'animation des objets. Dans la lignée de ces recherches, nous cherchons donc une méthode sur GPU se rapprochant du temps réel, sans aucune connaissance sur les modèles.

Chapitre 3

Préliminaires

3.1 Environnement de travail

J'ai commencé par adapter du code existant pour pouvoir effectuer le rendu des objets à l'aide de OpenGL en C++. J'ai utilisé une interface Qt pour pouvoir faire varier aisément les paramètres de l'application. Cet environnement permet de déplacer les objets interactivement et de modifier les propriétés des matériaux. J'ai aussi ajouté une gestion des ombres par la technique classique de la *shadow map*.

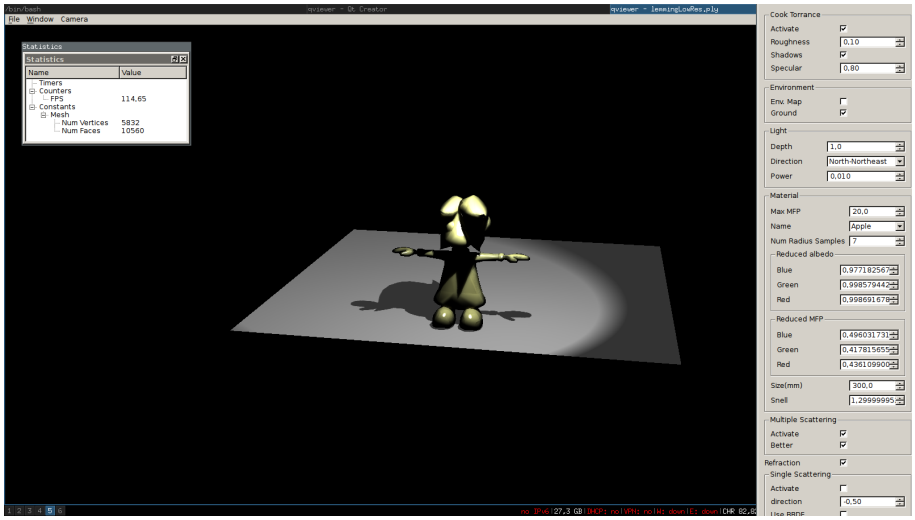


FIGURE 3.1 – Environement de travail

3.2 Éclairage spéculaire

L'éclairage classique utilisé en 3D correspond à la somme de trois couleurs, correspondant à trois aspect différents de la lumière :

- L'éclairage ambiant, résultant d'un parcours chaotique de la lumière, éclairant uniformément toute la scène. Ce terme est une constante, que nous avons fixée ici à 0 pour mieux distinguer l'influence des autres effets de lumière.
- L'éclairage diffus, contribution directe d'une source lumineuse. Elle sera simulée dans la BSSRDF.
- L'éclairage spéculaire, tache lumineuse résultant d'une réflexion de la source lumineuse sur l'objet.

L'éclairage spéculaire a été simulé ici à l'aide du modèle de Cook-Torrance :

$$k_{spec} = \frac{F_{refl}\, D\, G}{4\,(\mathbf{n}\cdot\boldsymbol{\omega}_i)\,(\mathbf{n}\cdot\boldsymbol{\omega}_o)}$$

Ici nous utilisons une BRDF, donc $\mathbf{x}_i = \mathbf{x}_o$ et $\mathbf{n}_i = \mathbf{n}_o = \mathbf{n}$

Nous voyons intervenir :

— Le terme de Fresnel en réflexion, F_{refl}, dont nous parlerons plus loin.

— Une approximation des micro-facettes (distribution de Beckmann) :

$$D = \frac{e^{-\left(\frac{\tan(\delta)}{m}\right)^2}}{\pi m^2 (\cos\delta)^4}$$

Où $\mathbf{h}$ et le demi vecteur entre ω_i et ω_o : $\mathbf{h} = \frac{\omega_i+\omega_o}{\|\omega_i+\omega_o\|}$, et δ l'angle entre $\mathbf{h}$ et $\mathbf{n}$.

— Enfin, l'atténuation géométrique, qui correspond à l'auto-ombrage des micro-facettes :

$$G = \min\left(1, \frac{2(\mathbf{n}\cdot\mathbf{h})(\mathbf{n}\cdot\omega_o)}{(\omega_o\cdot\mathbf{h})}, \frac{2(\mathbf{n}\cdot\mathbf{h})(\mathbf{n}\cdot\omega_i)}{(\omega_o\cdot\mathbf{h})}\right)$$

(a) (b) (c)

FIGURE 3.2 – Influence de l'éclairage spéculaire : (a) éclairage diffus seul avec notre méthode pour l'éclairage, (b) éclairage spéculaire seul, (c) le réalisme est augmenté par la combinaison des deux éclairages.

3.3 Fresnel

Les termes de Fresnel permettent de connaître la proportion de la lumière réfléchie ou réfractée lors d'un changement de milieu. Si nous considérons la lumière comme non polarisée, le terme de Fresnel en réfraction est donné par :

$$F_{refr} = \frac{R_s + R_p}{2}$$

Où R_s et R_p sont les termes polarisés :

$$R_s = \left\|\frac{\cos\theta_i - \eta\cos\theta_t}{\cos\theta_i - \eta\cos\theta_t}\right\|^2$$

$$R_t = \left\|\frac{\cos\theta_t - \eta\cos\theta_t}{\cos\theta_t - \eta\cos\theta_i}\right\|^2$$

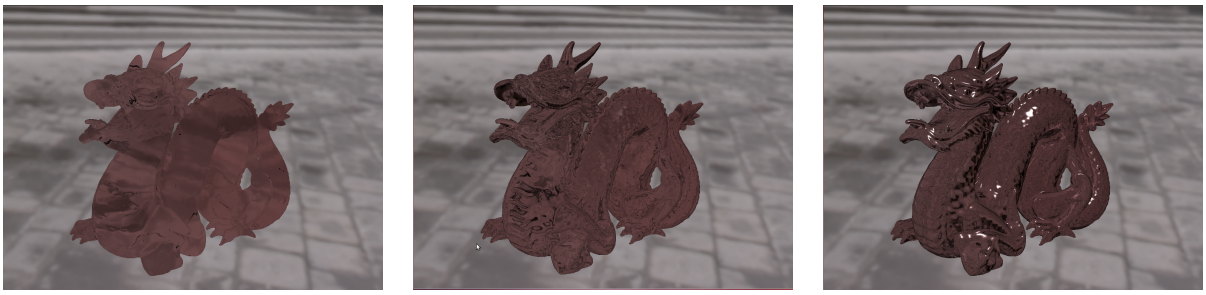
Avec n_1 et n_2 les indices de réfraction des milieux, $\cos\theta_i$ l'angle d'incidence de la lumière, et $\cos\theta_t$ l'angle du rayon réfracté. Nous avons ensuite le terme réfléchi : $F_{refl} = 1 - F_{refr}$

3.4 Réfraction

Pour accroître le réalisme des matériaux translucides, nous avons ajouté une gestion de la réfraction, en suivant une approche décrite dans Chang *et al.* [9]. Nous effectuons un rendu des faces arrières de l'objet, ce qui nous permet d'obtenir une carte des distances à la camera et des normales pour chaque pixel de ces faces. Ensuite nous estimons la distance parcourue dans l'objet par la lumière réfractée pour en déduire la position du point de sortie de la lumière dans la carte des faces arrières. A l'aide de cette position et de la normale, nous pouvons déterminer la direction du rayon sortant de l'objet, et nous allons chercher la couleur de la lumière dans une *carte d'environnement*, qui simule des objets à une distance infinie. Enfin, nous appliquons la loi de Beer-Lambert pour prendre en compte l'absorption dans le matériau :

$$L_o = L_i \, e^{-\sigma_a \, s}$$

Où s est la distance parcourue dans le materiau.



$\eta = 1.01$ $\eta = 1.10$ $\eta = 1.50$

FIGURE 3.3 – Résultat de la réfraction, pour plusieurs valeur de η

3.5 Fonction de phase

Nous utiliserons dans les chapitres suivants une fonction de phase. Cette fonction indique la proportion de lumière transmise en fonction de l'angle entre la lumière incidente et la lumière dispersée.
Dans notre implémentation, nous avons choisi la fonction de phase de Henyey-Greenstein [5] :

$$p : \theta \mapsto \frac{1}{4\pi} \frac{1 - g^2}{[1 + g^2 - 2\,g\,\cos(\theta)]^{3/2}}$$

Où $\cos(\theta) = \omega_i \cdot \omega_o$, et p est paramétrée par $g \in [-1, 1]$, qui donne la répartition de la direction de dispersion.

Chapitre 4

Single Scattering

4.1 Principe

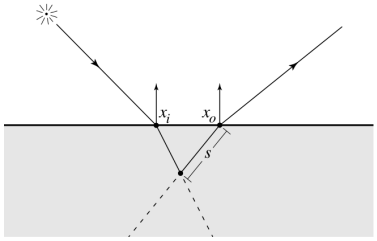


FIGURE 4.1 – *Single Scattering*

Dans notre calcul de la BSSRDF, nous prenons en compte deux termes : l'approximation de diffusion, S_d, dont nous parlerons dans le chapitre suivant, et un terme appelé *single scattering*, $S^{(1)}$:

$$S = S_d + S^{(1)}$$

Ce terme correspond au premier ordre du calcul de la dispersion de la lumière, et est obtenu implicitement comme suit ([7]) :

$$\begin{aligned} L_o^{(1)}(\mathbf{x_o}, \boldsymbol{\omega_o}) &= \sigma_s(\mathbf{x_o}) \int_{\Omega^+} F\, p(\boldsymbol{\omega_i'} \cdot \boldsymbol{\omega_o}) \int_0^\infty e^{\sigma_{tc}\, s}\, L_i(\mathbf{x_i}, \boldsymbol{\omega_i})\, \mathrm{d}s\, \mathrm{d}\boldsymbol{\omega_i} \\ &= \int_A \int_{\Omega^+} S^{(1)}(\mathbf{x_o}, \boldsymbol{\omega_o}, \mathbf{x_i}, \boldsymbol{\omega_i})\, L_i(\mathbf{x_i}, \boldsymbol{\omega_i})\, (\mathbf{n} \cdot \boldsymbol{\omega_i})\, \mathrm{d}\boldsymbol{\omega_i}\, \mathrm{d}A(\mathbf{x_i}) \end{aligned}$$

Avec F le produit des deux termes de Fresnel en réfraction, et σ_{tc} le coefficient d'extinction combiné :

$$\sigma_{tc} = \sigma_t(\mathbf{x_o}) + G\, \sigma_t(\mathbf{x_i})$$

Où G est un terme d'atténuation géométrique.

4.2 Implémentation

Concrètement, nous devons intégrer sur le trajet de la lumière :

$$\frac{\sigma_s(\mathbf{x_o})\, F\, p(\boldsymbol{\omega_i} \cdot \boldsymbol{\omega_o})}{\sigma_{tc}} e^{-s_i' \sigma_t \mathbf{x_i}}\, e^{-s_o' \sigma_t \mathbf{x_o}}\, L_i(\mathbf{x_i}, \boldsymbol{\omega_i})$$

Où s_o' est la distance du point de sortie de la lumière au point d'échantillonnage, et s_i' la distance du point d'échantillonnage au point d'entrée de la lumière. Connaissant un point dans le matériau et la position de la lumière, il est difficile de connaître le trajet exact de la lumière sans information précise sur la géométrie, à cause d'une réfraction à l'entrée dans le matériau. Il est possible d'approximer le *single scattering* en négligeant cette réfraction et en estimant la distance réelle parcourue par la lumière grace aux lois de Descartes ([7]) :

$$s_i' = s_i \frac{|\mathbf{n} \cdot \boldsymbol{\omega_i}|}{\sqrt{1 - \left(\frac{1}{\eta}\right)^2 (1 - |\mathbf{n} \cdot \boldsymbol{\omega_i}|^2)}}$$

Nous avons charché à adapter cette méthode au calcul sur GPU. Nous connaissons le point de sortie de la lumière puisqu'il s'agit du pixel à éclairer, la direction de la lumière sortante (qui va vers la caméra), et la normale à la surface en ce point. A l'aide des lois de Descartes, nous en déduisons la direction de la lumière dans le matériau, et nous choisissons quelques points d'échantillonnage dans cette direction. Il nous manque donc seulement la distance entre le point d'entrée de la lumière et le point d'échantillonnage pour calculer la valeur de la formule précédente, et la somme de ces valeurs pour tous les points donnera la couleur du pixel.
Pour obtenir cette distance, nous effectuons un premier rendu de l'objet du point de vue de la lumière, et nous stockons les distances de chaque pixel dans une texture que nous pouvons utiliser lors du rendu final. Il suffit alors de projeter le point d'échantillonnage dans le repère de la lumière, ce qui permet de trouver le pixel correspondant dans la texture. La distance cherchée est alors obtenue en faisant la différence entre la valeur de ce pixel et la distance du point d'échantillonnage à la lumière.

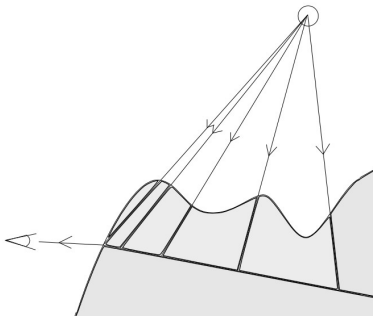


FIGURE 4.2 – Implémentation

Pour optimiser les performances tout en gardant une qualité acceptable, nous constatons que le terme exponentiel est prédominant. Nous allons donc placer les points d'échantillonnage en suivant une distribution exponentielle :

$$s'_o(k) = -\log\left((e^{c_l} - 1)\,\frac{k}{k_{\max}}\right) l$$

La précision dépendra donc du nombre de points déchantillons, $k_{\max}$, et du facteur multiplicatif devant le libre parcours moyen : c_l, qui détermine alors la distance maximale d'échantillonnage : $c_l\, l$. Nous multiplions enfin chaque valeur d'échantillonnage par la distance avec le précédent pour intégrer la fonction en escalier associée. Expérimentalement, nous arrivons à des résultats convaincants avec 30 points échantillonnage et $c_l = 7$.

(a) (b)

FIGURE 4.3 – Résultats : (a) sans *single scattering*, (b) avec *single scattering*, 15 FPS pour une résolution de 1024 par 768.

Chapitre 5

Première approche pour le multiple scattering

5.1 Approximations : Dipôle et *better dipole*

Pour approximer l'équation de diffusion, nous avons choisi d'utiliser l'approximation du dipôle, qui donne de bons résultats pour des matériaux très dispersifs et qui est relativement peu coûteux en terme de calculs. Pour chaque position $\mathbf{x_i}$, on ajoute deux sources de lumière ponctuelles, au-dessus et en dessous de la surface, dans l'alignement de la normale au point considéré. Le point au-dessus est nommé lumière réelle, et est située à $z_r = \frac{1}{\sigma'_t}$ au-dessus de $\mathbf{x_i}$. La lumière virtuelle est elle située à $z_v = z_r + 4AD$ au dessous de $\mathbf{x_i}$, ceci pour respecter les condition de bord de l'équation de diffusion. On a alors, pour x_i fixé,

$$\phi(x) = \frac{\Phi}{4\pi D}\left(\frac{e^{-\sigma_{tr}dr}}{dr} - \frac{e^{-\sigma_{tr}dv}}{dv}\right)$$

Où dr est la distance de x à la lumière réelle et dv la distance de x à la lumière virtuelle. On en déduira (voir tableau ci dessous) la valeur de R_d à l'aide de :

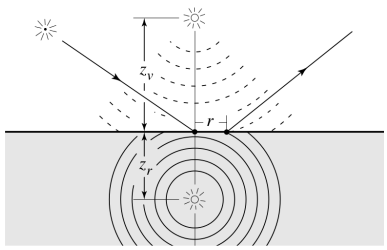
$$R_d(r) = -D\frac{\mathbf{n}\cdot\nabla\phi(x)}{\mathrm{d}\Phi_i}$$

Ce qui nous permet d'obtenir la BSSRDF de diffusion en ajoutant les termes de Fresnel en réfraction, à l'entrée et à la sortie du matériau :

$$S_d(\mathbf{x_i}, \omega_i, \mathbf{x_o}, \omega_o) = \frac{1}{\pi}F_t(\eta, \omega_i)R_d(||\mathbf{x_i} - \mathbf{x_o}||)F_t(\eta, \omega_o)$$

Nous avons implémenté la méthode du dipôle, ainsi que l'amélioration *better dipole* par d'Eon, qui ajoute des précisions physiques au calcul. Nous présentons les deux méthodes de calcul de R_d comme dans son article :

Terme	Dipôle	*Better Dipole*
$D =$	$\frac{1}{3\sigma'_t}$	$\frac{2\sigma_a + sigma'_s}{3(\sigma_a+\sigma'_s)^2}$
$A =$	$\frac{1+2C_1}{1-2C_1}$	$\frac{1+3C_2}{1-2C_1}$
$z_b =$	$2AD$	$2AD$
$z_r =$	$\frac{1}{\sigma'_t}$	$\frac{1}{\sigma'_t}$
$z_v =$	$-z_r - 2z_b$	$-z_r - 2z_b$
$d_r =$	$\sqrt{r^2 + z_r^2}$	$\sqrt{r^2 + z_r^2}$
$d_v =$	$\sqrt{r^2 + z_v^2}$	$\sqrt{r^2 + z_v^2}$
$\sigma_{tr} =$	$\sqrt{\frac{\sigma_a}{D}}$	$\sqrt{\frac{\sigma_a}{D}}$
C_ϕ	0	$\frac{1}{4}(1 - 2C1)$
$C_\mathbf{E} =$	1	$\frac{1}{42}(1 - 3C2)$
$C_n =$	$\frac{\alpha'}{4\pi}$	$\frac{(\alpha')^2}{4\pi}$



Le rayon entrant est transformé en source dipolaire.

$$R_d(r) = C_n\left[\left(C_\mathbf{E}\frac{z_r(\sigma_{tr}d_r + 1)}{d_r^2} + \frac{C_\phi}{D}\right)\frac{e^{-\sigma_{tr}d_r}}{d_r} - \left(C_\mathbf{E}\frac{z_v\sigma_{tr}d_v + 1)}{d_v^2} + \frac{C_\phi}{D}\right)\frac{e^{-\sigma_{tr}d_v}}{d_v}\right]$$

Ce calcul fait intervenir des approximations pour les moyennes de Fresnel :

$$2C_1 \approx \begin{cases} 0.919317 - 3.4793\eta + 6.75335\eta^2 - 7.80989\eta^3 + 4.98554\eta^4 - 1.36881\eta^5, & \eta < 1 \\ -9.23372 + 22.2272\eta - 20.9292\eta^2 + 10.2291\eta^3 - 2.54396\eta^4 + 0.254913\eta^5, & \eta \geq 1 \end{cases}$$

et

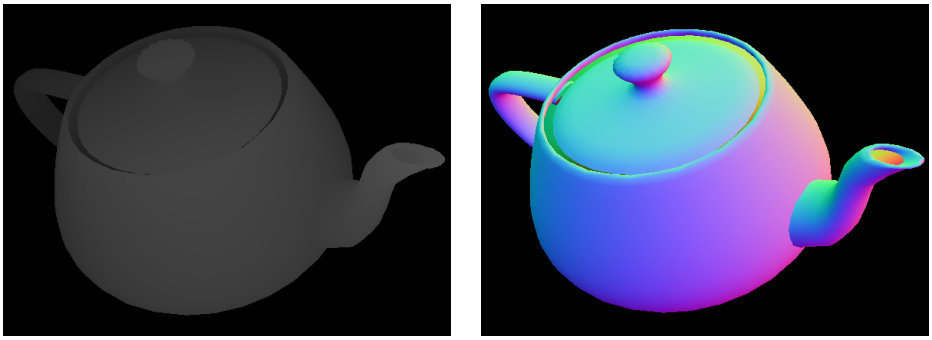
$$3C_2 \approx \begin{cases} 0.828421 - 2.62051\eta + 3.36231\eta^2 - 1.95284\eta^3 + 0.236494\eta^4 + 0.145787\eta^5, & \eta < 1 \\ -1641.1 + \frac{135.926}{\eta^3} - \frac{656.175}{\eta^2} + \frac{1376.53}{\eta} + 1213.67\eta - 568.556\eta^2 + 164.798\eta^3 - 27.0181\eta^4 + 1.91826\eta^5, & \eta \geq 1 \end{cases}$$

Enfin, cet article ajoute un coefficient de normalisation manquant dans la BSSRDF donnée ci-dessus :

$$S_d(\mathbf{x_i}, \omega_i, \mathbf{x_o}, \omega_o) = \frac{1}{\pi} F_t(\eta, \omega_i) R_d(\|\mathbf{x_i} - \mathbf{x_o}\|) \frac{F_t(\eta, \omega_o)}{1 - 2C_1 \frac{1}{\eta}}$$

5.2 Implémentation

A l'aide de la formule générale ci-dessus, nous avons implémenté une fonction permettant aussi bien d'évaluer le dipôle que *better dipole*. Le problème est donc d'intégrer cette formule sur une surface proche du pixel dont on veut connaître la couleur, alors que nous n'avons pas d'information sur la géométrie de l'objet. Puisque l'intensité lumineuse reçue en un point dépend de $\mathbf{n} \cdot \omega_i$, il apparaît que nous pouvons intégrer sur une distribution de points seulement visibles à partir de la source lumineuse. Nous effectuons donc un premier rendu de la scène à partir de la source lumineuse, pour obtenir des textures contenant les distances à la source et les normales pour tous les pixels de la scène visibles de la source.
Lors du rendu final, pour un point donné de l'objet dont nous voulons connaître la couleur, $\mathbf{x_o}$, nous choi-



(a) (b)

FIGURE 5.1 – Rendu sur texture, du point de vue de la lumière : (a) profondeurs, (b) normales.

sissons une distribution de points sur des portions de disques centrés en $\mathbf{x_o}$, et nous intégrons la formule du dipôle en multipliant chaque valeur obtenue par l'aire de la portion de disque correspondant. Nous avons donc comme paramètres le nombre de couronnes (différence de deux demi-disques concentriques), et le nombre de points d'échantillons par couronnes. Les disques choisis sont alignés orthogonalement à la direction de la lumière. Pour chaque point d'échantillonnage, nous avons besoin de sa position, $\mathbf{x_i}$, pour obtenir $\|\mathbf{x_i} - \mathbf{x_o}\|$, et de la normale à la surface en ce point, pour calculer la proportion de lumière reçue. Ces informations s'obtiennent en projetant les points d'échantillonnage sur les textures calculées précédemment.

Chapitre 6

Améliorations pour le calcul du multiple scattering

6.1 Problème

(a) (b)

FIGURE 6.1 – Résultat peu convainquant (a) par rapport à la référence (b).

Le modèle proposé n'est pas efficace et il faut un très grand nombre d'échantillons pour s'approcher des résultats de référence (dipôle calculé à l'aide de Mitsuba). Nous avons cherché comment améliorer les calculs sur du marbre, et nous avons pour cela dessiné les courbes de la valeur du dipôle, multipliée par l'aire associée, en fonction la distance à $\mathbf{x_0}$. Ces courbes présentent des courbures importantes, une partie importante de l'information est donc omise avec un échantillonnage uniforme. Les courbes du deuxième graphique représentent la somme des ces valeurs, et montre des dominances de couleurs différentes suivant la distance. Il est donc important d'avoir un calcul précis pour que les changements de couleur se fassent pour les bonnes distances. Deux améliorations sont donc envisageables, améliorer la méthode d'intégration et trouver une meilleure distribution d'échantillonnage.

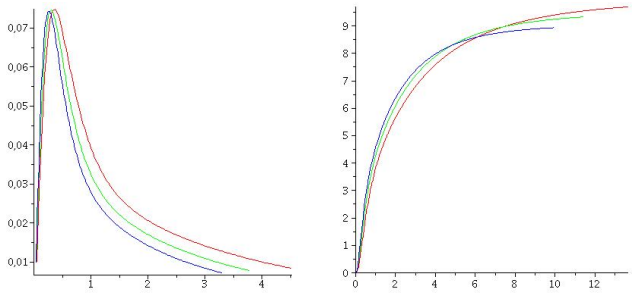
6.2 Intégration

Pour améliorer la méthode d'intégration, au lieu de sommer la valeur en un point, multipliée par l'aire alentour, nous avons supposé que la valeur en un point était donnée par l'interpolation bilinéaire des valeurs aux quatre points délimitant nos surfaces d'intégration. Nous interpolons tout d'abord selon la première variable d'intégration, θ :

$$C_1 : \theta \mapsto \lambda\, c_4 + (1 - \lambda)\, c_3$$
$$C_2 : \theta \mapsto \lambda\, c_1 + (1 - \lambda)\, c_2$$

Avec *lambda* tel que :

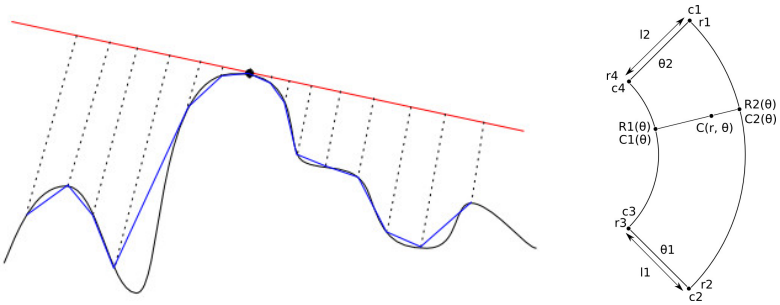
$$\theta = (1 - \lambda)\, \theta_1 + \lambda\, \theta_2$$



(a) (b)

FIGURE 6.2 – Valeur du dipôle multipliée par l'aire associée en fonction de la distance à $\mathbf{x_o}$ (a), et somme de ces valeurs (b) pour du marbre. Chaque courbe correspond à un canal de couleur différent (rouge, vert, bleu).

Les deux valeurs sont ensuite interpolées entre R_1 et R_2.
L'intégrale de cette interpolation donne un résultat exact qu'il est facile de sommer. Une autre amélioration a été de constater que notre disque d'intégration n'est qu'une projection de la surface de l'objet sur un plan orthogonal à la direction de la lumière. A l'aide des textures de rendu en espace lumière, nous pouvons connaître la position exacte de chacun des points, et délimiter ainsi plus précisément l'aire à intégrer. Nous pouvons donc avoir des rayons différents à l'intérieur ou à l'extérieur de la surface à intégrer (par exemple r_1 et r_2), car le rayon est maintenant associé à la distance entre le point d'origine du calcul et la position réelle du point à calculer, sur la surface.



(a) (b)

FIGURE 6.3 – (a) A l'aide de la projection du disque d'intégration (en rouge), nous déduisons la surface sur laquelle nous intégrons (en bleu). (b) Schéma de dessus de la surface d'intégration.

Nous devons donc aussi interpoler le rayon entre les 4 points :

$$R_1 : \theta \mapsto \lambda\, r_4 + (1-\lambda)\, r_3$$

$$R_2 : \theta \mapsto \lambda\, c_1 + (1-\lambda)\, c_2$$

Ce qui nous donne pour la valeur du dipôle en un point de coordonnées polaires (r, θ) :

$$C : (r, \theta) \mapsto \rho\, C_2(\theta) + (1-\rho)\, C_1(\theta)$$

Avec ρ tel que :

$$r = (1-\rho)\, R_1(\theta) + \rho\, R_2(\theta)$$

Nous pourrons intégrer simplement sur r et θ, mais pour prendre en compte l'aire de la surface nous devons en plus interpoler l_1 et l_2 :

$$l : \theta \mapsto \lambda\, l_l + (1-\lambda)\, l_1$$

Il ne nous reste plus qu'à intégrer sur cette surface, pour obtenir une approximation de l'intégrale du dipôle entre ces quatre points :

$$I = \int_{\theta_1}^{\theta_2} \frac{l(\theta)}{R_2(\theta) - R_1(\theta)} \int_{R_1(\theta)}^{R_2(\theta)} r\, C(r, \theta)\, \mathrm{d}r\, \mathrm{d}\theta$$

Soit

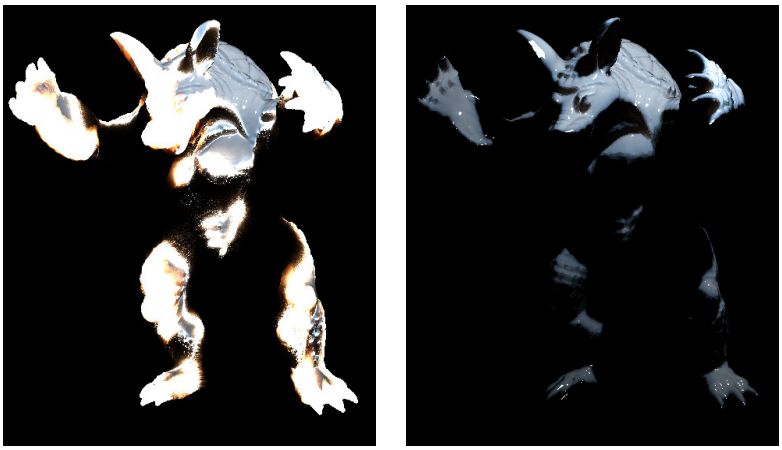
$$I = \mathbf{r}_v^t \left(l_1 \mathbf{M}_1 + l_2 \mathbf{M}_2\right) \mathbf{c}_v$$

Avec :

$$\mathbf{r}_v = \begin{pmatrix} r1 & r2 & r3 & r4 \end{pmatrix} \quad \mathbf{c}_v = \begin{pmatrix} c1 & c2 & c3 & c4 \end{pmatrix}$$

$$\mathbf{M}_1 = \begin{pmatrix} 2 & 2 & 1 & 1 \\ 2 & 6 & 3 & 1 \\ 1 & 3 & 6 & 2 \\ 1 & 1 & 2 & 2 \end{pmatrix} \quad \mathbf{M}_2 = \begin{pmatrix} 6 & 2 & 1 & 3 \\ 2 & 2 & 1 & 1 \\ 1 & 1 & 2 & 2 \\ 3 & 1 & 2 & 6 \end{pmatrix}$$

L'approximation de l'interpolation bilinéaire donne de bons résultats en temps normal, mais est trop inexacte lorsque l'aire devient grande. Ceci correspond à des conditions limites, lorsqu'on intègre entre deux surfaces distinctes ou sur un bord de l'objet, et donne une trop grande illumination des contours. Pour corriger cela, nous calculons la courbure entre deux surfaces consécutives et lorsque celle-ci dépasse un certain seuil, nous n'incluons pas la valeur considérée dans l'intégrale.



(a) (b)

FIGURE 6.4 – Résultat obtenu après intégration (a), amélioré en analysant la courbure de la surface à intégrer (b).

6.3 Échantillonnage

Une deuxième manière de rendre le calcul plus précis est dans le choix de la distribution des points d'échantillonnage. Nous considérons cette distribution comme une fonction $f : [0, 1] \to [0, 1]$. Nous lui donnerons en argument des valeurs réparties linéairement, et nous multiplierons sa valeur de sortie par le rayon maximal pour obtenir le rayon d'échantillonnage.
Nous devons trouver f pour que la courbure de la courbe obtenue en composant la fonction du dipôle avec f soit le plus proche possible de 0. Nous n'avons pas trouvé de solution analytique, mais nous constatons que le terme en r^2 prédomine pour r petit, puis que le terme exponentiel devient plus important pour r grand.
La combinaison linéaire

$$f_\beta : x \mapsto \frac{\beta \sqrt{x}\,(1 - x)}{1 - \beta} + \frac{x\,(e^x - 1)}{e - 1}$$

donne de bons résultats en pratique, à condition de trouver une bonne valeur pour β. Pour ceci, nous cherchons la valeur de β qui minimise $\int |g''(r)|\,\mathrm{d}r$, où g représente la composition de la fonction du dipôle avec f_β. Cette valeur ne dépend que des propriétés du matériau, nous pouvons donc la calculer numériquement en pré-calcul.
Un autre perfectionnement de l'échantillonnage a été de séparer les trois canaux de couleur (rouge, vert, et bleu), car les distances représentatives ne sont pas les mêmes suivant la couleur.

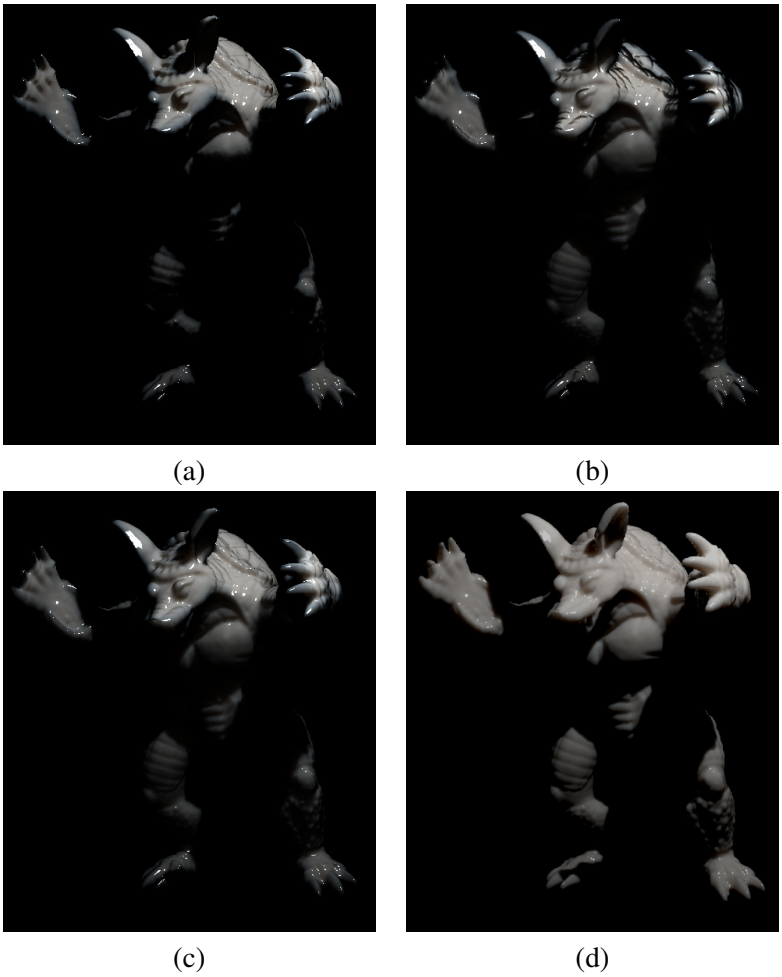
6.4 Précision

Notre méthode souffre d'un dernier problème : la résolution des textures dans lesquels nous effectuons les rendus intermédiaires est limitée. Ceci se voit surtout lorsque la lumière est presque parallèle à la surface, car dans ce cas l'information contenue dans un pixel correspondra à une large surface. Nous avons donc adapté notre modèle pour qu'il puisse effectuer le même calcul à l'aide d'informations obtenues par un premier rendu du point de vue de la caméra. Si $R_{d,light}$ est la contribution du point de vue de la lumière, et $R_{d,cam}$ la contribution de la caméra, nous interpolons les deux à l'aide d'une sigmoïde :

$$s = \frac{1}{1 + e^{-\lambda\,((\mathbf{n}\cdot\boldsymbol{\omega}_i)-(\mathbf{N}\cdot\boldsymbol{\omega}_o))}}$$

$$R_d = s\,R_{d,light} + (1 - s)\,R_{d,view}$$

Les différentes modifications apportées ont été testées séparément, et ne sont pas satisfaisantes. Seule la



(a) (b)

(c) (d)

FIGURE 6.5 – Résultat après calcul du point de vue de la lumière (a), du point de vue de la caméra (b), et résultat final (c) par rapport à la référence, pour du marbre sur un objet de 200 mm.

combinaisons de toutes donne un résultat proche de Mitsuba, chacune d'elles visant à corriger un défaut différent présent dans l'implémentation initiale.

Chapitre 7

Conclusion

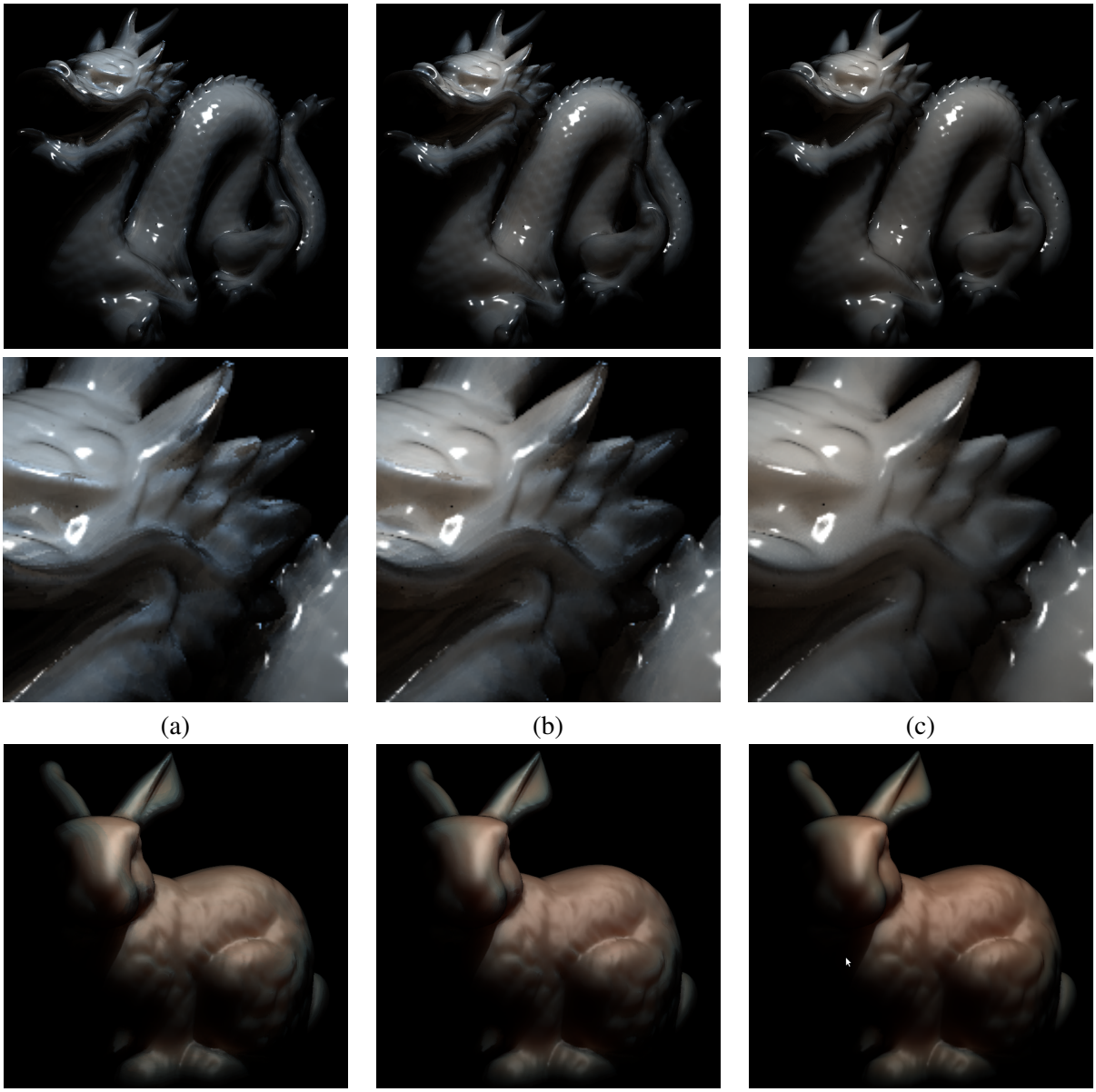
Après avoir effectué une partie de travail préliminaire pour pouvoir intégrer la BSSRDF de diffusion des matériaux participants dans un modèle complet d'éclairage, nous nous sommes penchés sur deux aspects de cette BSSRDF. Le premier est le *Single Scattering*, terme du premier ordre obtenu en intégrant l'influence lumineuse le long de son trajet direct dans le matériau. L'implémentation sur GPU a été rendue possible par l'utilisation d'un premier rendu dans une carte de distances à la lumière. Les termes d'ordres supérieurs sont obtenus par l'approximation du dipôle, consistant à intégrer l'influence lumineuse d'un ensemble de dipôles autour du point à éclairer. Ici encore, nous utilisions des rendus de distances et de normales du point de vue de la lumière pour effectuer cette intégration. Ce calcul ne donnant pas de résultat satisfaisant, nous avons amélioré la technique d'intégration et trouvé un échantillonnage donnant des images proches de la référence (calcul du dipôle par ray-tracing).

7.1 Résultats

L'aspect de nos rendus donne bien l'effet de dispersion attendu. La comparaison aux images de référence montre quelques différences, principalement l'intensité lumineuse, qui peuvent être expliquées par les imprécisions liées à l'approche sur GPU. Nous éliminons le *Single Scattering* de nos comparaisons, celui-ci n'étant pas géré par Mitsuba, et finalement négligeable pour les ordres de grandeur utilisés. Les effets de couleur suivent bien ce qui était prévu théoriquement par les courbes, mais sont un peu accentués sur les contours et sur les bords des ombres à cause de la correction sur l'intégrale présentée à la fin de la section 4.2. Une difficulté de l'utilisation de notre technique est de régler les paramètres (nombre de points d'échantillonnage, distance maximale d'échantillonnage), pour avoir le meilleur résultat pour un temps de calcul optimal.
Les performances dépendent de la résolution de notre fenêtre de rendu et de la proportion de pixels occupés par l'objet à traiter. Avec un GPU haut de gamme (Nvidia Quadro 6000), nous arrivons à rendre des images utilisant 168 points d'échantillonnage à 10 fps, pour un objet occupant un carré de 500 pixels de côté, soit 4000 fois plus rapidement que *Mitsuba*, moteur de rendu par raytracing qui affiche les images de référence en 7 min.

Marbre crème ketchup

FIGURE 7.1 – Quelques rendus pour différents matériaux.



(a) (b) (c)

(d) (e) (f)

FIGURE 7.2 – Influence des différents paramètres : dragon puis zoom sur un détail avec 168 échantillons répartis sur 10 mfp, rendu à 7 fps (a), 336 échantillons répartis sur 30 mfp, rendu à 2 fps (b), 1200 échantillons répartis sur 30 mfp, rendu à 0.3 fps (c), et enfin le lapin de Stanford, d'une dimension de 100 mm et avec un matériau représentant de la peau, avec comme paramètres : 168 échantillons répartis sur 10 mfp, rendu à 10 fps (d), 336 échantillons répartis sur 15 mfp, rendu à 3 fps (e), et 1200 échantillons répartis sur 30 mfp, rendu à 0.5 fps (f).

7.2 Travail futur

Il est encore possible d'optimiser le programme pour obtenir une meilleure performance, et de chercher d'autres distributions d'échantillonnage pour pouvoir utiliser moins de points d'échantillons. Nous pouvons chercher plus en détail du côté de l'*importance sampling*, qui est moins lourd en calcul mais que nous avons rejeté car les rendus sont trop bruités.
Enfin, une publication basée sur ces travaux pourrait être envisagé.

Bibliographie

[1] Antoine Bouthors, Eric Bruneton, Fabrice Neyret, and Nelson Max. Real-time subsurface scattering on the gpu, 2007.

[2] Nathan A. Carr, Jesse D. Hall, and John C. Hart. Gpu algorithms for radiosity and subsurface scattering. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, HWWS '03, pages 51–59, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[3] Chih-Wen Chang, Wen-Chieh Lin, Tan-Chi Ho, Tsung-Shian Huang, and Jung-Hong Chuang. Real-time translucent rendering using gpu-based texture space importance sampling. *Comput. Graph. Forum*, 27(2) :517–526, 2008.

[4] Eugene D'Eon. A better dipole. http://www.eugenedeon.com/papers/betterdipole.pdf, November 2012.

[5] L. G. Henyey and J. L. Greenstein. Diffuse radiation in the Galaxy. *ApJ*, 93 :70–83, January 1941.

[6] Henrik Wann Jensen and Juan Buhler. A rapid hierarchical rendering technique for translucent materials. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, New York, NY, USA, 2005. ACM.

[7] Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 511–518, 2001.

[8] Bruce Walter, Shuang Zhao, Nicolas Holzschuch, and Kavita Bala. Single scattering in refractive media with triangle mesh boundaries. *ACM Trans. Graph.*, 28(3) :92 :1–92 :8, July 2009.

[9] Chris Wyman. An approximate image-space approach for interactive refraction. *ACM Trans. Graph.*, 24(3) :1050–1053, July 2005.