Université de Montréal

Peinture de lumière incidente dans des scènes 3D

par

Frédérik Rozon

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Mémoire présenté à la faculté des études supérieures et postdoctorales

en vue de l'obtention du grade de

Maître ès sciences (M.Sc.)

en informatique

Août 2009

Université de Montréal
Faculté des études supérieures et postdoctorales

Ce mémoire de maîtrise intitulé

Peinture de lumière incidente dans des scènes 3D

présenté par
Frédérik Rozon

a été évalué par un jury composé des personnes suivantes :

Jean Meunier
président-rapporteur

Pierre Poulin
directeur de recherche

Victor Ostromoukhov
membre du jury

Sommaire

Le design d'éclairage est une tâche qui est normalement faite manuellement, où les artistes doivent manipuler les paramètres de plusieurs sources de lumière pour obtenir le résultat désiré. Cette tâche est difficile, car elle n'est pas intuitive. Il existe déjà plusieurs systèmes permettant de dessiner directement sur les objets afin de positionner ou modifier des sources de lumière. Malheureusement, ces systèmes ont plusieurs limitations telles qu'ils ne considèrent que l'illumination locale, la caméra est fixe, etc. Dans ces deux cas, ceci représente une limitation par rapport à l'exactitude ou la versatilité de ces systèmes. L'illumination globale est importante, car elle ajoute énormément au réalisme d'une scène en capturant toutes les interréflexions de la lumière sur les surfaces. Ceci implique que les sources de lumière peuvent avoir de l'influence sur des surfaces qui ne sont pas directement exposées.

Dans ce mémoire, on se consacre à un sous-problème du design de l'éclairage : la sélection et la manipulation de l'intensité de sources de lumière. Nous présentons deux systèmes permettant de peindre sur des objets dans une scène 3D des intentions de lumière incidente afin de modifier l'illumination de la surface. De ces coups de pinceau, le système trouve automatiquement les sources de lumière qui devront être modifiées et change leur intensité pour effectuer les changements désirés. La nouveauté repose sur la gestion de l'illumination globale, des surfaces transparentes et des milieux participatifs et sur le fait que la caméra n'est pas fixe. On présente également différentes stratégies de sélection de modifications des sources de lumière.

Le premier système utilise une carte d'environnement comme représentation intermédiaire de l'environnement autour des objets. Le deuxième système sauvegarde l'information de l'environnement pour chaque sommet de chaque objet.

Mots clefs :
Design de lumière, rendu inverse, illumination globale, paradigme de peinture.

Abstract

Lighting design is usually a task that is done manually, where the artists must manipulate the parameters of several light sources to obtain the desired result. This task is difficult because it is not intuitive. Some systems already exist that enable a user to paint light directly on objects in a scene to position or alter light sources. Unfortunately, these systems have some limitations such that they only consider local lighting, or the camera must be fixed, etc. Either way, this limitates the accuracy or the versatility of these systems. Global illumination is important because it adds a lot of realism to a scene by capturing all the light interreflections on the surfaces. This means that light sources can influence surfaces even if they are not directly exposed.

In this M. Sc. thesis, we study a subset of the lighting design problem : the selection and alteration of the intensity of light sources. We present two different systems to design lighting on objects in 3D scenes. The user paints light intentions directly on the objects to alter the surface illumination. From these paint strokes, the systems find the light sources and alter their intensity to obtain as much as possible what the user wants. The novelty of our technique is that global illumination, transparent surfaces and subsurface scattering are all considered, and also that the camera is free to take any position. We also present strategies for selecting and altering the light sources.

The first system uses an environment map as an intermediate representation of the environment surrounding the objects. The second system saves all the information of the environment for each vertex of each object.

Keywords :

Light design, inverse rendering, global illumination, painting metaphor.

Table des matières

Table des figures

Remerciements

Je désire remercier mon père, ma mère et Lyne pour leurs conseils, leur écoute et leur soutien inconditionnel dans les moments difficiles. À Alexandre pour son amitié de longue date avec qui je peux décompresser dans des moments peu intellectuels ! À Marie-Ève, pour son amour et son encouragement (car elle avait bien hâte que je termine !).

Je désire remercier également le CRSNG pour ma bourse d'étude ainsi que le financement de mon voyage à Toulouse. À l'Université de Montréal pour les bourses d'excellence. Finalement à mes collègues Toulousains Mathias Paulin, Anthony Pajet et Olivier Gourmel pour leur aide et leurs conseils.

Finalement, ce mémoire n'aurait pas été possible sans mon directeur, Pierre Poulin, qui a proposé le sujet, m'a donné des pistes d'idées et m'a encouragé (à plusieurs reprises !) C'est grâce à lui que je m'intéresse à l'infographie en général, par son style d'enseignement "vraiment *cool*". Il m'a également prêté sa maison, ce qui m'a bien dépanné lors d'un moment de crise personnel.

Chapitre 1

Introduction

L'éclairage est un des aspects les plus importants d'une scène lorsqu'un artiste veut communiquer une idée, une émotion. Une pièce très éclairée sera généralement plus chaleureuse qu'une pièce sombre. L'éclairage donne une personnalité à une pièce, invite au repos ou encore rend propice le travail commun d'un groupe ou celui d'un individu solitaire. Pour des objets individuels, la présence de *highlights* ou d'ombres résultant de l'éclairage permet de mieux distinguer leur forme ou leur donner une apparence plus dramatique (p. ex. des oeuvres d'art dans un musée).

Dans les scènes réelles ainsi qu'au cinéma et en photographie, les éclairagistes doivent manuellement contrôler les sources de lumière et utiliser des outils (p. ex. des diffuseurs, des bloqueurs) pour obtenir les effets lumineux désirés. Ceci est habituellement une tâche complexe, qui demande de l'intuition et de l'expérimentation, ce qui peut être fastidieux. Pour les scènes synthétiques, c'est la même chose : l'artiste doit manipuler plusieurs paramètres pour chaque source de lumière. Même dans les cas d'un simple spot de lumière, les quelques paramètres utilisés deviennent rapidement compliqués à gérer lorsque le nombre de sources ainsi que leurs effets indirects se combinent entre eux.

Le problème fondamental reste cependant que l'influence qu'a une source est souvent peu intuitive, à cause de plusieurs phénomènes reliés à l'illumination globale. Par exemple, l'interréflexion de la lumière sur les objets, les concentrations de lumière formant des caustiques, les surfaces semi-transparentes, la diffusion de la lumière dans des milieux participatifs, etc., ne sont que quelques phénomènes qui entravent considérablement notre capacité à prévoir comment la manipulation d'une source de lumière affectera la

scène. En considérant ces effets, lorsqu'une scène contient plusieurs sources de lumière, deviner quelle source manipuler pour modifier l'éclairage d'une certaine zone sans en affecter une autre devient très difficile, car l'éclairage d'une scène est une fonction non linéaire et un petit changement d'un paramètre d'une source de lumière peut affecter toute la scène. Aussi, la visualisation d'un changement peut être très coûteuse si aucun effort d'efficacité n'est déployé.

Dans la littérature, plusieurs méthodes et algorithmes existent déjà pour modifier de manière intuitive l'illumination d'une scène. Certains de ces systèmes se limitent au positionnement de sources de lumière dans le cas d'illumination directe, et dans certains cas, se limitent également à des BRDFs simples, ou à une caméra fixe [PF92, PRJ97, PTG02, PBMF07]. D'autres systèmes incorporent l'illumination globale, mais ne permettent pas de déplacer la caméra et/ou se limitent à la technique de la radiosité [SDS$^+$93, CSF99]. Finalement, un système a été présenté pour modifier l'illumination parvenant d'une carte d'environnement sur un objet [OMSI07], mais sans l'intégrer dans une scène 3D.

Dans ce mémoire, nous proposons deux systèmes qui permettent de peinturer sur un objet des intentions d'éclairage (plus clair, plus foncé, plus rouge, etc.). Une fois les intentions peinturées, nos systèmes trouvent les sources de lumière qui vont satisfaire ces intentions en augmentant ou en diminuant l'intensité de ces sources. Afin d'être utile, la visualisation de l'éclairage sur un objet en cours de modification doit se faire en temps au moins interactif ; nous expliquerons les structures que nous utilisons pour atteindre cet objectif. Finalement, nos systèmes permettent de déplacer la caméra en tout temps pour permettre une utilisation plus générale et flexible.

Le contenu du mémoire sera divisé comme suit : le chapitre 2 fera une brève révision de quelques techniques permettant de modifier en temps réel l'illumination d'un objet ou d'une scène au complet. Le chapitre 3 fera la revue de la littérature sur le problème du rendu inverse, domaine couvrant la problématique visée par ce mémoire. Le chapitre 4 introduira les différentes stratégies implémentées pour la sélection et la modification des sources de lumière suite à un coup de pinceau. Les chapitres 5 et 6 introduiront les deux systèmes complets implémentés. Le chapitre 7 décrira certains détails d'implémentation. Finalement, le chapitre 8 montrera et discutera les résultats et le chapitre 9 conclura le mémoire ainsi qu'offrira quelques perspectives de travaux futurs.

Une copie électronique avec illustrations en couleur est disponible sur `http://www.iro.umontreal.ca/labs/infographie/theses/rozonfre/`.

Chapitre 2

Ré-éclairage

2.1 Définition

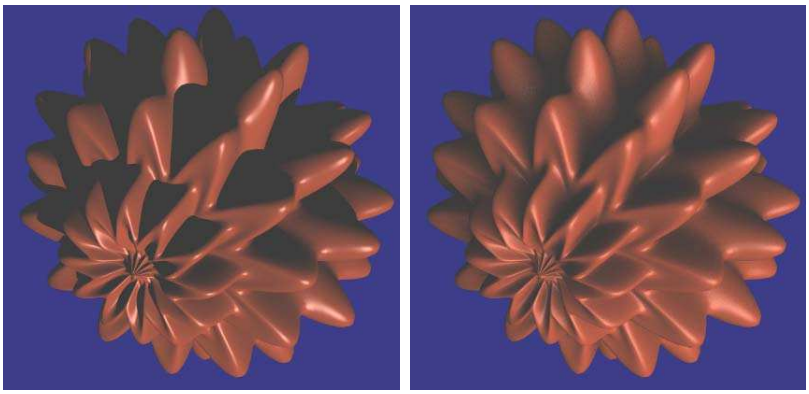
Il y a quelques années à peine, le rendu en temps réel de scènes tridimensionnelles n'utilisait en majorité que des points de lumière et/ou des sources directionnelles, en ne considérant que l'illumination directe. On utilisait également presque exclusivement le modèle de Phong [Pho75] pour la réflectance (*bidirectionnal reflection distribution function* ou BRDF) des surfaces. Dans ce contexte restreint, la lumière transmise vers la caméra à partir des surfaces est simplement la somme du coefficient d'illumination ambiante (qui est une constante d'illumination qu'on ajoute à toutes les surfaces afin d'approximer grossièrement l'illumination globale) et de la contribution directe de chacune des sources de lumière. Selon les propriétés du matériau (p. ex. la rugosité) appliqué sur une surface, la lumière réfléchie par un élément de la surface de façon diffuse est une fonction cosinus de l'angle relatif entre la normale de la surface et de la direction incidente de la lumière. La lumière réfléchie spéculairement est quant à elle souvent estimée par un lobe comme une fonction cosinus d'ordre supérieur de l'angle relatif entre la direction de vue et la direction incidente de la lumière. Ce type de réflectance permet de donner un aspect plus lisse à une surface (p. ex. un métal poli). Ce modèle a beaucoup été utilisé parce qu'il est très simple, permettant de générer très rapidement des images de scènes dynamiques, ce qui est nécessaire lorsqu'une visualisation en temps réel est désirée.

Le problème est que ce modèle est trop simple. Dans la réalité, le comportement de la réflexion de la lumière de certains matériaux ne peut pas être approximé assez bien par

ce modèle. De plus, d'autres phénomènes sont complètement ignorés, par exemple les occultations et les interréflexions. Finalement, ce modèle ne peut utiliser que des sources de lumière ponctuelles ou directionnelles. En réalité, les sources surfaciques constituent la majorité des sources existantes. Les techniques susceptibles d'être en mesure de rendre correctement ces effets demeurent très coûteuses, et malgré les progrès impressionnants des CPUs, elles ne sont pas capables de produire des images de scènes complexes assez rapidement. De ces techniques, on compte le lancer de rayons (pour l'illumination directe seulement), la radiosité (pour de l'illumination globale de surfaces diffuses), le lancer de chemins, le lancer de photons, etc.

L'ombrage permet de déterminer si la lumière atteint un élément de surface, et joue donc un rôle important dans l'apparence d'une scène tridimensionnelle. Les techniques pour l'ombrage sont surtout traitées avec des cartes d'ombre (*shadow maps*) ou des volumes d'ombre (*shadow volumes*) [WPF90, HLHS03] et se limitent souvent aux ombres dures provenant de sources de lumière ponctuelles et directionnelles. Le problème est qu'en réalité, les sources de lumière sont souvent surfaciques et que l'illumination provenant de telles sources produit des ombres floues parce que la lumière reçue en un point peut être partiellement bloquée par un objet. Les sources ponctuelles génèrent des images moins réalistes (voir figure 2.1). En général, on perçoit mieux la forme de l'objet quand il est éclairé par une source surfacique. Plusieurs variantes ont été introduites pour adoucir les ombres produites par les techniques temps réel, mais elles sont la plupart du temps des approximations, demeurent plus coûteuses en calculs, et ne traitent que l'illumination directe.

Les interréflexions (limitées) ont récemment été implémentées en (quasi) temps réel. Cependant, ces techniques limitent soit le nombre d'interréflexions ou le nombre de sources de lumière pour que leurs calculs se réalisent en temps réel. En supposant que les paramètres des sources de lumière dans une scène restent fixes, et en utilisant le fait que la lumière réfléchie de façon diffuse est indépendante du point de vue, une technique qui est souvent utilisée dans les systèmes de rendu temps réel (notamment dans les jeux vidéo) est le précalcul de la lumière réfléchie par les surfaces. La quantité de lumière ainsi réfléchie par ces surfaces est stockée dans des textures, qu'on appelle cartes de lumière, utilisées lors du rendu de la scène. Les techniques utilisées pour le précalcul peuvent être n'importe laquelle des techniques d'illumination globale, comme



(a) (b)

FIGURE 2.1 – Un objet éclairé (a) par une source ponctuelle et (b) par une source surfacique. L'éclairage par la source surfacique donne plus de détails sur la forme de l'objet. Source [SKS02a].

la radiosité ou le lancer de photons. L'avantage des cartes de lumière est qu'on obtient une bonne qualité d'image due à une meilleure approximation de l'illumination globale (comparativement à un simple coefficient d'illumination ambiante). Les désavantages sont que ces cartes sont seulement valides pour la configuration des sources de lumière avec laquelle elles ont été calculées : l'illumination dynamique est donc impossible. De plus, seulement les réflexions diffuses peuvent être précalculées ainsi.

Avec la venue des processeurs graphiques (*graphic processing units* ou GPUs), la génération d'images est maintenant devenue très rapide. Par contre, certains effets sont encore très difficiles à accomplir avec ces processeurs, par exemple les réflexions multiples caractéristiques à l'illumination globale. On doit alors recourir à des approximations.

La motivation était donc très grande de visualiser des objets en temps réel en présence d'occultations et d'interréflexions, avec des sources dynamiques de lumière autres que ponctuelles et directionnelles.

2.2 Transfert de lumière précalculé

Le transfert de lumière précalculé (*Precomputed radiance transfer* ou PRT [SKS02b, SKS02a]) a été introduit pour visualiser, en temps réel, un objet éclairé par une carte d'environnement en précalculant des fonctions qui transforment la lumière reçue par l'objet en lumière réfléchie. Ces fonctions étant indépendantes de la carte d'environ-

nement, elle peut donc changer. Il s'agit d'une des premières techniques qui permet de modifier en temps réel l'illumination d'un objet et d'en faire un rendu de qualité.

La carte d'environnement, qui définit l'illumination incidente, est d'abord projetée dans une nouvelle base qui permet la compression. Différentes bases sont décrites à la section 2.4. À l'origine, les harmoniques sphériques ont été utilisées.

Les objets à illuminer sont d'abord divisés en éléments de surface. Pour avoir de meilleurs résultats, les surfaces sont subdivisées en polygones d'aires sensiblement identiques. Ceci implique que même si un objet contient de grandes surfaces planes, qui pourraient être représentées par un grand polygone, elles devront être subdivisées en plusieurs polygones. L'illumination de la surface se calcule seulement à ses sommets et ensuite la couleur est interpolée entre eux. Donc si les polygones sont trop gros, l'illumination ne sera calculée qu'aux coins de ceux-ci, et possiblement certains phénomènes pourraient être totalement ignorés, par exemple un *highlight* fin.

Un ensemble de fonctions de transfert d'illumination sont précalculées et stockées pour chaque sommet de l'objet. Ces fonctions convertissent l'illumination arbitraire incidente en illumination sortant de l'objet, incluant des effets comme les ombres et les interréflexions de l'objet sur lui-même.

L'idée principale de la technique est de précalculer la manière dont l'objet réfléchit la lumière incidente sur lui-même ou sur l'environnement autour de lui. Autant la lumière incidente que les fonctions de transfert sont représentées dans la même base. L'intégrale du rendu devient un simple produit scalaire des coefficients des bases pour les surfaces diffuses, et une multiplication de matrices et un produit scalaire pour les surfaces spéculaires. La technique est limitée à de l'illumination incidente distante. On présente maintenant les grandes lignes de l'algorithme.

2.2.1 Précalcul des objets

Une fonction de transfert pour un sommet donné transforme l'illumination incidente en lumière réfléchie. Pour un objet convexe éclairé par une carte d'environnement (illumination distante), la fonction de transfert est simplement l'intégrale de la lumière en chaque sommet, définie sur l'hémisphère positif et pondérée par le cosinus de l'angle entre la direction de la lumière et la normale au sommet. Une fonction de transfert plus complexe est nécessaire si l'objet est concave, car elle doit prendre en compte les

occultations. Dans ce cas, l'intégrale de la lumière est multipliée par un facteur additionnel représentant la visibilité dans chaque direction à partir d'un point de l'objet. Une fonction encore plus complexe prendra en compte la lumière que l'objet réfléchit sur lui-même.

Les occultations, l'intégration de la lumière et les interréflexions ne peuvent pas être calculées en temps réel au moment du rendu. Comme la quantité de mémoire est devenue beaucoup moins une restriction, comparativement à la puissance de calcul des ordinateurs de nos jours, l'approche consiste à précalculer les fonctions de transfert pour chaque sommet et à les sauvegarder.

La lumière incidente et les fonctions de transfert sont représentées dans la même base. Pour chacune des bases décrites dans la section 2.4, l'intégrale entre la lumière incidente et les fonctions de transfert est facilement calculable et dans certains cas, la carte vidéo peut en accélérer davantage le calcul.

Objets diffus

Pour calculer la lumière réfléchie à partir d'un sommet de l'objet dans une certaine direction $\mathbf{v}$, l'intégrale suivante doit être évaluée :

$$R(\mathbf{v}) = \int_{\Omega} L(\mathbf{s})V(\mathbf{s})f(\mathbf{s},\mathbf{v})H_{\mathbf{n}}(\mathbf{s})d\mathbf{s}. \tag{2.1}$$

Le terme $L(\mathbf{s})$ représente l'illumination provenant de la direction $\mathbf{s}$ sur l'hémisphère Ω sur lequel on intègre. $V(\mathbf{s})$ est une fonction binaire de visibilité, dont la valeur est 1 si la partie de l'environnement dans la direction $\mathbf{s}$ est visible, et 0 autrement. $V(\mathbf{s})$ peut contenir toute l'information sur les occultations de l'objet. $f(\mathbf{s},\mathbf{v})$ représente la BRDF du matériau de la surface. $H_{\mathbf{n}}(\mathbf{s})$ est le produit scalaire entre la normale $\mathbf{n}$ de la surface au sommet et la direction $\mathbf{s}$.

Quand la lumière incidente est représentée par des coefficients l_i dans une base B_i quelconque, on peut alors remplacer $L(\mathbf{s})$ de l'équation 2.1 et on obtient

$$R(\mathbf{v}) \approx \int_{\Omega} \left(\sum_i l_i B_i(\mathbf{s}) \right) V(\mathbf{s})f(\mathbf{s},\mathbf{v})H_{\mathbf{n}}(\mathbf{s})d\mathbf{s}. \tag{2.2}$$

Maintenant, la lumière incidente est la somme pondérée des bases $B_i(\mathbf{s})$. En utilisant la propriété de la linéarité du transport de la lumière, on obtient la somme des coefficients

de l'illumination incidente multipliée par l'intégrale qui dépend seulement des bases et de la BRDF :

$$R(\mathbf{v}) \approx \sum_i l_i \int_\Omega B_i(\mathbf{s}) V(\mathbf{s}) f(\mathbf{s}, \mathbf{v}) H_{\mathbf{n}}(\mathbf{s}) d\mathbf{s}. \tag{2.3}$$

Si la BRDF est diffuse, on a déjà précalculé $f(\mathbf{s}, \mathbf{v})$: les BRDFs diffuses ne dépendent pas de la direction de la vue. L'intégrale entière peut être évaluée et remplacée par un scalaire t_i. Maintenant, la lumière réfléchie peut être exprimée comme :

$$R(\mathbf{v}) \approx \sum_i l_i t_i. \tag{2.4}$$

Cette dernière équation ne dépend que des coefficients de l'illumination incidente et des propriétés de l'objet au sommet. La lumière réfléchie par l'objet peut être calculée avec un simple produit scalaire.

Comme précalcul, l'intégrale de l'équation 2.3 est évaluée pour chaque sommet de l'objet et pour chaque base. Ceci donne autant de coefficients qu'il y a de bases pour représenter la lumière incidente, pour chaque sommet. Pour évaluer l'intégrale, un ensemble de 10,000 à 30,000 vecteurs de direction aléatoires est premièrement créé. Ensuite, pour chaque sommet, on détermine s'il y a occultation pour chacune de ces directions. L'intégrale devient alors une grosse somme sur chaque vecteur de direction.

Pour inclure les interréflexions, on doit modifier l'équation. La lumière réfléchie en présence d'occultations et d'interréflexions correspond à

$$R_{ir}(\mathbf{v}) = R(\mathbf{v}) + \int_\Omega \bar{L}_p(\mathbf{s}) H_{\mathbf{n}}(\mathbf{s}) (1 - V_P(\mathbf{s})) d\mathbf{s}, \tag{2.5}$$

où $R(\mathbf{v})$ est l'intégrale de l'équation 2.1. L'intégrale additionnelle représente la lumière qui provient des interréflexions. $\bar{L}_p(\mathbf{s})$ représente la lumière provenant de l'objet lui-même dans la direction $\mathbf{s}$. Si la lumière incidente varie peu sur l'objet (p. ex. si la lumière provient de l'infini), $\bar{L}_p(\mathbf{s})$ est bien approximée comme si l'objet est éclairé par $L(\mathbf{s})$ pour chaque sommet pour toute direction $\mathbf{s}$. Par contre, si la source de lumière incidente n'est pas distante, alors l'illumination directe varie sur l'objet et $\bar{L}_p(\mathbf{s})$ ne peut pas être connue à partir de la lumière incidente au sommet parce qu'il dépend de la lumière réfléchie des autres sommets, qui eux-mêmes, dépendent des changements d'illumination directe. Donc, pour être capable de précalculer les interréflexions, on doit supposer que la lumière incidente est invariante sur tout l'objet.

Le terme d'occultation $(1 - V_p(\mathbf{s}))$ filtre toutes les directions autres que celles qui ne sont pas cachées dans l'intégrale. Il ne peut y avoir d'interréflexions qu'à partir des directions où la lumière atteint l'objet.

La lumière réfléchie de l'équation 2.5 dépend linéairement de la lumière incidente et peut être factorisée dans un vecteur de coefficients de la même façon que l'ombrage.

Pour inclure les interréflexions dans le précalcul, on procède d'abord comme indiqué plus haut. Ensuite, on évalue les interréflexions en effectuant plusieurs passes. Chaque passe évalue seulement les chemins qui proviennent des sources de lumière et qui sont réfléchis au moins une fois sur l'objet avant d'arriver au sommet en cours d'évaluation. Ces contributions sont ensuite ajoutées au sommet, après être transformées dans la base.

Objets spéculaires

Comme la lumière réfléchie par une surface spéculaire dépend de la direction de vue (au lieu d'être une fonction qui dépend seulement de $\mathbf{s}$), la fonction de transfert pour chaque sommet devient une matrice, au lieu d'un vecteur. Maintenant, le calcul de l'illumination doit dépendre de la direction de vue pendant la visualisation et le terme $f(\mathbf{s}, \mathbf{v})$ de l'équation 2.3 ne peut pas être précalculé.

La matrice de transfert transforme la lumière incidente en lumière réfléchie pour chaque sommet de l'objet. Elle prend en compte les occultations et les interréflexions, mais ne peut plus inclure la BRDF car la lumière réfléchie par une surface spéculaire dépend de la direction de vue.

Étant donné que la fonction de transfert dépend de la réflexion $\mathbf{r}$ de la direction de vue au lieu d'une normale fixe $\mathbf{n}$, le terme $H_{\mathbf{n}}(\mathbf{s})$ dans l'équation 2.1 devient une fonction plus complexe $G(\mathbf{s}, \mathbf{r})$.

Le précalcul dans le cas des objets spéculaires est d'évaluer les coefficients pour la matrice de transfert pour chaque sommet. La version de l'intégrale qui ne contient que les ombres est

$$R_S(L, \mathbf{v}) = \int L(\mathbf{s}) G(\mathbf{s}, \mathbf{v}) V_p(\mathbf{s}) d\mathbf{s} \tag{2.6}$$

et la version avec les ombres et les interréflexions est

$$R_I(L, \mathbf{v}) = R_S(L, \mathbf{v}) + \int \bar{L}_p(\mathbf{s}) G(\mathbf{s}, \mathbf{v}) (1 - V_p(\mathbf{s})) d\mathbf{s}. \tag{2.7}$$

Ces équations donnent la lumière réfléchie pour la direction de vue $\mathbf{v}$ comme une fonction de la lumière incidente L et $\mathbf{v}$. Ces quantités sont toutes les deux inconnues au moment du précalcul. Avec quelques manipulations mathématiques, les intégrales de l'équation 2.7 peuvent être factorisées comme des coefficients de la matrice de transfert. Les coefficients peuvent être calculés en intégrant un triple produit des bases en utilisant les formules de récursion de la série de Clebsch-Gordan.

2.2.2 Rendu

Le processus général pour déterminer la couleur finale de chaque sommet consiste en les étapes suivantes :

1. La lumière incidente L est calculée en échantillonnant les directions de la lumière incidente à partir d'un point (p. ex. le centre de l'objet). On transforme cet échantillonnage dans la base choisie, ce qui donne un vecteur de coefficients. Si on ignore les interréflexions de l'objet, on peut aller chercher la lumière incidente locale pour quelques points échantillonnés sur la surface de l'objet, ce qui donnera plusieurs vecteurs de coefficients. On utilise ensuite ces vecteurs pour déterminer la lumière incidente pour chaque sommet de l'objet en interpolant les points échantillonnés.
2. La lumière incidente est replacée dans l'espace de l'objet.
3. Seulement une transformation linéaire est nécessaire à chaque sommet pour obtenir la couleur. Ceci correspond à un produit scalaire pour les objets diffus, et un produit matrice-vecteur pour les objets spéculaires.
4. Pour les objets spéculaires, on doit faire la convolution entre la lumière transférée et la BRDF pour chaque sommet, pour ensuite évaluer le résultat dans la direction $\mathbf{v}$.

2.3 Ré-éclairage direct vers indirect

Hašan et al. [HPB06] présentent une technique pour ré-éclairer une scène arbitrairement complexe, dans un contexte d'illumination globale. On décrit ici les grandes lignes de l'algorithme.

La technique suppose certains faits pour qu'elle fonctionne :

– La caméra fixe permet de réduire la complexité géométrique de la scène à ce qui est visible dans l'image.
– Le calcul de l'illumination directe à partir de sources ponctuelles d'une scène peut être réalisé efficacement, grâce aux avancées des cartes vidéo.
– L'illumination provient surtout d'interréflexions de surfaces diffuses et donc, en particulier, les caustiques ne sont pas gérées.

La technique introduit deux ensembles de points : un ensemble V d'échantillons de la caméra, qui sont entièrement visibles à partir du point de vue, et un ensemble G d'échantillons de collecte qui sont distribués dans toute la scène. Le problème du ré-éclairage est alors de recalculer un vecteur $\mathbf{v}_i$ de la lumière indirecte réfléchie sur les échantillons de la caméra V, qui correspondent en fait aux pixels de l'image générée. Étant donné que la caméra est fixe, ce calcul peut être défini comme une transformation linéaire $\mathbf{T}$ de la lumière directe réfléchie par les échantillons de collecte $\mathbf{g}_d$

$$\mathbf{v}_i = \mathbf{T} \cdot \mathbf{g}_d. \tag{2.8}$$

Pour obtenir une image de bonne qualité, on doit distribuer beaucoup d'échantillons de la caméra (ce qui correspond ultimement à la résolution de l'image) et d'échantillons de collecte. En supposant que le nombre d'échantillons de la caméra $n_v = 640 \times 480 \approx 300\text{K}$ et que le nombre d'échantillons de collecte $n_g = 64\text{K}$, alors la matrice $\mathbf{T}$ de la transformation linéaire a n_v lignes et n_g colonnes. Donc la plus importante partie de la technique est de compresser une telle matrice tout en restant capable d'évaluer l'équation 2.8 efficacement. La technique projette l'illumination directe sur les échantillons de collecte $\mathbf{g}_d$ et les lignes de la matrice $\mathbf{T}$ dans une base d'ondelettes de Haar 2D (voir section 2.4.2), tout en éliminant les coefficients les moins significatifs.

2.3.1 Précalculs

Les précalculs à faire dans cette technique est la distribution des échantillons de collecte et de la caméra, et le calcul de la matrice de transfert $\mathbf{T}$.

Le choix des échantillons de collecte peut être fait de différentes façons. Deux de ces façons sont :

Échantillonnage uniforme : Si on a un budget de n_g échantillons et un ensemble de triangles dans la scène, alors on peut distribuer les échantillons sur les triangles

proportionnellement à l'aire de ceux-ci.

Échantillonnage par importance : Si on utilise la technique du lancer de photons, mais en lançant des photons à partir des échantillons de la caméra, on obtiendra les endroits dans la scène qui sont susceptibles d'être les plus importants pour les échantillons de la caméra, permettant ainsi d'améliorer possiblement la qualité des résultats.

Les échantillons de la caméra sont triviaux à trouver, il s'agit de faire un lancer de rayons simple de la caméra vers la scène, et de garder la première intersection.

Pour obtenir $\mathbf{g}_d$, qui est le vecteur d'illumination directe des échantillons de collecte, on doit organiser ces échantillons dans un vecteur. Une technique de regroupement hiérarchique ordonne les échantillons en utilisant l'algorithme des k-moyennes.

Pour calculer $\mathbf{T}$, la méthode utilisée se divise en deux étapes, qui sépare la matrice de transfert en deux :

1. Une matrice $\mathbf{M}$ transfère la lumière due aux diverses interréflexions dans la scène ; $\mathbf{M}$ peut être de moindre précision.
2. Une matrice $\mathbf{F}$ fait le transfert final entre les échantillons de collecte et les échantillons de la caméra ; $\mathbf{F}$ doit avoir une meilleure précision.

Ces deux matrices sont compressées en éliminant les coefficients d'ondelettes les moins significatifs. Donc la formulation du problème global devient

$$\mathbf{v}_i = [\mathbf{F} \cdot (\mathbf{M} + \mathbf{I})] \cdot \mathbf{g}_d, \tag{2.9}$$

où $\mathbf{I}$ est la matrice identité.

La matrice $\mathbf{M}$ est calculée en lançant, à partir des n_g échantillons, des photons dans la scène, ce qui va permettre de calculer la contribution de la lumière des échantillons de collecte entre eux. La matrice $\mathbf{M}$ est donc carrée, où la i-ème ligne de la matrice donne la contribution de chaque échantillon vers le i-ème échantillon.

De façon similaire, la matrice $\mathbf{F}$ est calculée en lançant des photons à partir des échantillons de la caméra, ce qui va permettre de calculer la contribution de lumière des échantillons de collecte vers les échantillons de la caméra. Donc, la i-ème ligne de la matrice $\mathbf{F}$ donne la contribution de chaque échantillon de collecte vers le i-ème échantillon de vue.

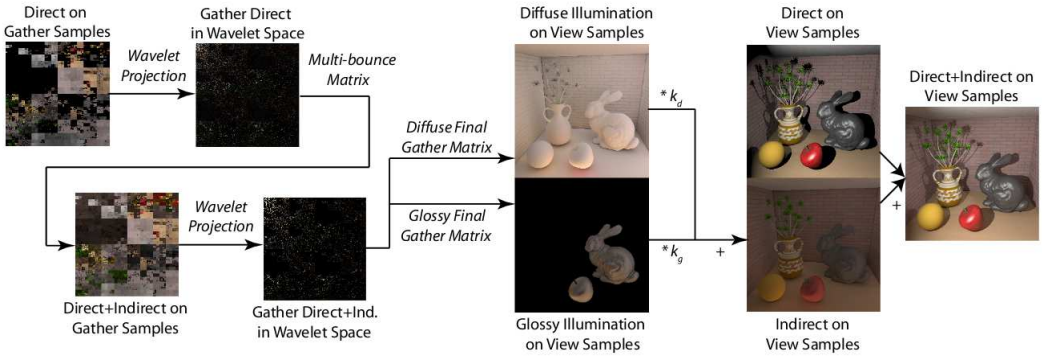


FIGURE 2.2 – Aperçu de la technique de Hašan et al. [HPB06]. Source [HPB06].

2.3.2 Rendu

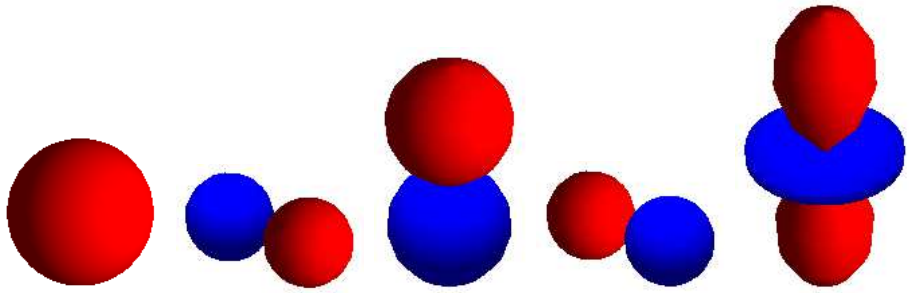
Voici les étapes nécessaires pour obtenir l'image finale. La figure 2.2 donne un aperçu schématisé de l'approche.

1. Jusqu'ici, la technique ne fait aucune supposition sur la position, l'orientation et l'émission des sources de lumière. Ceci permet donc de modifier les paramètres de ces sources au rendu et d'avoir le résultat en temps réel. Quand les paramètres des sources changent, on doit calculer la lumière reçue par les échantillons de collecte à partir des sources de lumière (le vecteur $\mathbf{g}_d$). Pour ceci, on peut utiliser n'importe quelle technique d'illumination directe connue et efficace. On projette ensuite $\mathbf{g}_d$ dans les bases d'ondelettes de Haar.
2. On calcule l'équation 2.9, où toutes les matrices et les vecteurs sont projetés dans les bases d'ondelettes, ce qui donne le vecteur $\mathbf{v}_i^w$, qui est la couleur de chaque pixel de l'image, exprimée seulement en coefficients d'ondelettes.
3. On applique la transformation inverse en ondelettes sur $\mathbf{v}_i^w$, ce qui donne $\mathbf{v}_i$, c'est-à-dire les couleurs des pixels de l'image.

L'article explique comment stocker efficacement les coefficients d'ondelettes des matrices et des vecteurs et faire les produits sur la carte vidéo afin d'accélérer grandement le traitement.

2.4 Représentations efficaces

Ici on introduit des outils qui permettent de représenter certains signaux (p. ex. une BRDF) afin de pouvoir les manipuler, combiner et compresser efficacement et facile-



(a) Y_0^0 (b) Y_1^{-1} (c) Y_1^0 (d) Y_1^1 (e) Y_2^0

FIGURE 2.3 – Visualisation des fonctions de la base des harmoniques sphériques pour les basses fréquences. Les zones bleues représentent des coefficients négatifs et les zones rouges des coefficients positifs.

ment. En infographie, il est souvent nécessaire d'effectuer l'intégrale de la convolution de plusieurs signaux (p. ex. entre l'environnement et la BRDF pour obtenir la couleur réfléchie pour un certain point sur la surface d'un objet) et ces représentations offrent une manière de calculer ce genre d'opérations plus facilement.

2.4.1 Harmoniques sphériques

La transformée de Fourier représente une fonction linéaire comme une somme pondérée de fréquences élémentaires (fonctions sinusoïdales). Les harmoniques sphériques sont l'équivalent de la transformée de Fourier, mais pour les fonctions sphériques. Elles sont donc idéales pour représenter des données naturellement définies sur la sphère, par exemple une carte d'environnement. Comme pour la transformée de Fourier, les harmoniques sphériques possèdent également des propriétés mathématiques utiles pour simplifier le traitement de fonctions.

La figure 2.3 montre la visualisation de quelques fonctions de base des harmoniques sphériques. Une fonction de base est définie par deux indices, $L \in \mathbb{N}$ et $M \in \mathbb{Z}$. L est appelé l'indice de bande. Pour chaque L, M est compris entre $-L$ et L. Les petites valeurs de L représentent les fonctions de base de basses fréquences sur la sphère.

Puisque les fonctions de la base des harmoniques sphériques sont orthonormales, il s'ensuit qu'une fonction scalaire f, définie sur la sphère, peut être représentée par des coefficients de fonctions d'harmoniques sphériques, qui pondèrent chaque fonction de base Y_l^m.

Les coefficients c_l^m peuvent être calculés en projetant f sur chacune des fonctions de base Y_l^m avec une intégrale :

$$c_l^m = \int f(s) Y_l^m(s) ds. \tag{2.10}$$

La fonction originale f peut alors être reconstruite à partir de ces coefficients par la sommation des fonctions de base pondérée par les coefficients correspondants :

$$\tilde{f}(s) = \sum c_l^m Y_l^m(s). \tag{2.11}$$

Comme pour la transformée de Fourier, la fonction reconstruite sera toujours une approximation s'il y a une quantité finie de coefficients. Par contre, pour des fonctions sphériques de basses fréquences, une quantité relativement petite de coefficients est suffisante pour reconstruire une bonne approximation.

Étant donné que les fonctions de base des harmoniques sphériques sont orthonormales, si on a deux fonctions a et b définies sur la sphère, leur projection dans les bases satisfait l'équation suivante :

$$\int \tilde{a}(s)\tilde{b}(s) ds = \sum a_l^m b_l^m. \tag{2.12}$$

Les harmoniques sphériques sont invariantes sous la rotation. La fonction reconstruite $\tilde{f}(s)$ qui a subi une rotation peut être projetée dans les harmoniques sphériques en utilisant une transformation linéaire des coefficients de $f(s)$.

2.4.2 Ondelettes

Les ondelettes sont un outil mathématique pour décomposer hiérarchiquement un signal. Elles permettent à une fonction d'être représentée en termes d'une forme grossière, en ajoutant progressivement des détails de plus en plus fins. Les ondelettes offrent donc une technique élégante pour représenter des niveaux de détails.

Ici on abordera seulement les ondelettes pour des fonctions discrètes et linéaires. Ce sont les plus simples et les plus utilisées en infographie. Elles ont la limitation qu'elles ne fonctionnent que sur des fonctions linéaires. Donc, pour représenter une fonction sphérique, il faudra trouver une transformation de cette fonction vers, par exemple, un carré. Les ondelettes ont par contre un inconvénient majeur : elles ne sont pas invariantes sous la rotation.

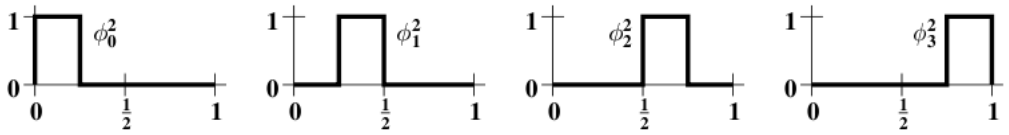


FIGURE 2.4 – Les quatre fonctions boîte de la base de V^2. Source [SDS95].

Ondelettes en une dimension

On peut penser à un signal discret (p. ex. une image) comme étant une fonction divisée en parties égales sur l'intervalle $[0, 1)$ et où chaque partie a une valeur constante. Pour ce faire, on utilisera la notion d'espace vectoriel de l'algèbre linéaire. Dans cette section, on appellera une fonction divisée en 2^n parties égales f^n. Pour f^0, ceci correspond à une fonction constante sur l'intervalle $[0, 1)$. On définit V^0 comme l'espace vectoriel de toutes les f^0 possibles. Pour f^1, ceci correspond à une fonction qui contient deux parties constantes sur les intervalles $[0, 0.5)$ et $[0.5, 1)$. On définit V^1 l'espace vectoriel de toutes les f^1 possibles. Si on continue de cette façon, l'espace vectoriel V^j contiendra toutes les f^j, qui ont des valeurs constantes pour chacun des 2^j sous-intervalles.

Parce que toutes les fonctions ont été définies sur l'intervalle $[0, 1)$, chaque fonction dans V^j fait également partie de V^{j+1}, en divisant chaque sous-partie de la fonction en deux.

Maintenant, on va trouver une base pour chaque espace vectoriel V^j. C'est ce qu'on appelle les *scaling functions* notées par le symbole ϕ. La base la plus simple pour V^j est donnée par l'ensemble des fonctions boîte suivantes :

$$\phi_i^j(x) = \phi(2^j x - i), \qquad i = 0, \ldots, 2^j - 1 \tag{2.13}$$

où

$$\phi(x) = \begin{cases} 1 & \text{pour } 0 \leq x < 1 \\ 0 & \text{autrement.} \end{cases} \tag{2.14}$$

La figure 2.4 montre les quatre fonctions boîte qui forment la base de V^2.

La prochaine étape est de définir le produit scalaire entre deux éléments de l'espace vectoriel V^j. Le produit scalaire standard,

$$\langle f|g\rangle = \int_0^1 f(x)g(x)dx, \tag{2.15}$$

où $f, g \in V^j$ sera utilisé pour l'instant. On peut maintenant définir un nouvel espace vectoriel W^j des vecteurs qui sont orthogonaux à V^j et V^{j+1}. Donc, de façon informelle,

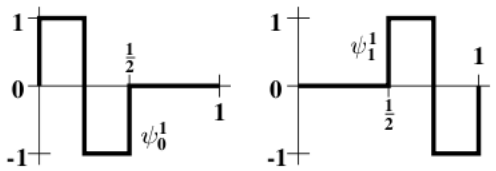


FIGURE 2.5 – Les ondelettes de Haar pour W^1. Source [SDS95]

on peut dire que les ondelettes dans W^j sont une façon de représenter les parties d'une fonction dans V^{j+1} qu'on ne peut pas représenter dans V^j.

L'ensemble des vecteurs $\psi_i^j(x)$ linéairement indépendants dans W^j sont appelés ondelettes. Ces vecteurs de base ont deux propriétés importantes :

1. Les vecteurs $\psi_i^j \in W^j$ avec les fonctions de base $\phi_i^j \in V^j$ forment une base de V^{j+1}.
2. Chaque vecteur $\psi_i^j \in W^j$ est orthogonal à chaque fonction de base $\phi_i^j \in V^j$ sous le produit scalaire choisi.

Les ondelettes correspondant aux fonctions boîte choisies plus haut sont connues sous le nom d'ondelettes de Haar, et sont définies par

$$\psi_i^j(x) = \psi(2^j x - i) \qquad\qquad i = 0, \ldots, 2^j - 1 \tag{2.16}$$

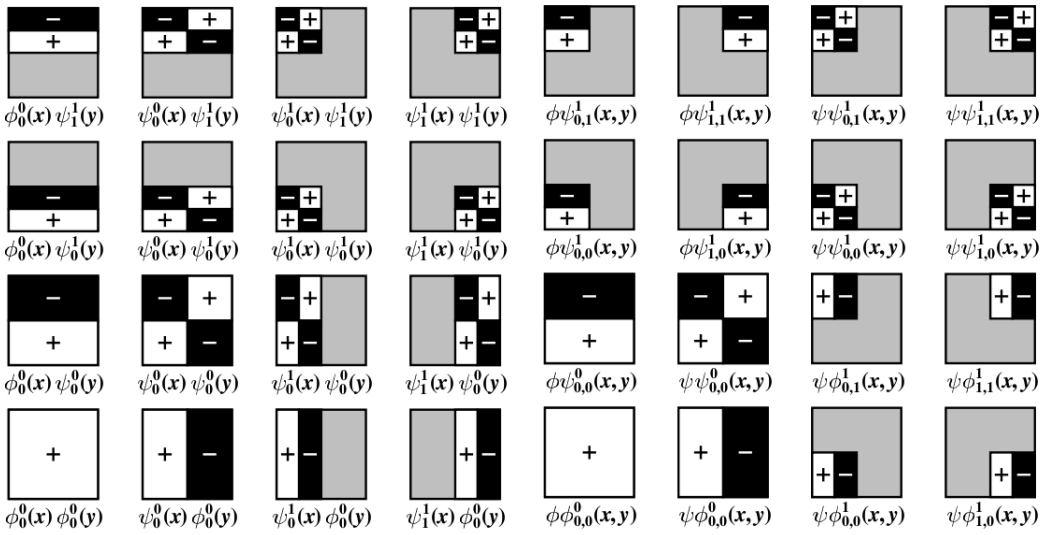
où

$$\psi(x) = \begin{cases} 1 & \text{pour } 0 \leq x < 0.5 \\ -1 & \text{pour } 0.5 \leq x < 1 \\ 0 & \text{autrement.} \end{cases} \tag{2.17}$$

La figure 2.5 montre les deux ondelettes de Haar pour W^1.

Donc, on peut décomposer un signal discret en ondelettes en commençant par représenter notre signal S^j comme la combinaison linéaire des vecteurs de base de V^j, où j est la résolution du signal. Ensuite, récursivement, on trouve la combinaison linéaire des vecteurs de base de V^{j-1} qui représente le mieux S^j, qui donne S^{j-1}, et on trouve la combinaison linéaire des vecteurs de W^j qui permet de récupérer S^j. On continue jusqu'à tant que $j = 1$.

En infographie, on utilise les ondelettes de Haar parce qu'elles possèdent une propriété additionnelle importante : les vecteurs dans W^j sont orthonormaux entre eux. Donc si on a deux signaux, a et b, projetés dans une base d'ondelettes, l'équation suivante est satisfaite



(a) (b)

FIGURE 2.6 – (a) L'ensemble des bases standards pour un signal de résolution 4×4. (b) L'ensemble des bases non-standards pour un signal de la même résolution.

$$\int a(s)b(s)ds = \sum \hat{a}_i \hat{b}_i \tag{2.18}$$

où $\hat{a}_i$ et $\hat{b}_i$ sont les coefficients des ondelettes.

Une autre utilité intéressante des ondelettes est la compression. Soit $\tilde{a}$, un signal reconstruit à partir d'une compression des coefficients d'ondelettes de a, c'est-à-dire en mettant à zéro les n plus petits coefficients (en valeur absolue). Alors on peut prouver que le taux d'erreur $\Delta = \sum(\tilde{a}_i - a_i)^2$ est minimal pour le taux de compression par ondelettes choisi. Donc la méthode de compression est très simple et efficace.

Ondelettes en plusieurs dimensions

On peut étendre la technique à un signal 2D en transformant de façon indépendante les lignes et les colonnes. Il existe deux façons de décomposer un signal 2D (voir figure 2.6 pour un signal 2D de résolution 4×4) :

1. Méthode standard : on effectue toutes les itérations de la transformation sur les lignes du signal. On effectue ensuite toutes les itérations sur les colonnes. Elle a l'avantage d'être très simple à calculer, en récupérant une implémentation de la transformation 1D et en l'appliquant sur les lignes et ensuite sur les colonnes. Par

contre, le support de chaque base (région où la fonction est non-nulle) n'est pas nécessairement carré.

2. Méthode non-standard : on effectue, en alternance, une itération de la transformation sur les lignes et une itération sur les colonnes. Cette méthode est un peu plus efficace à calculer et le support de chaque base est carré.

Il est possible de travailler en plus de deux dimensions. Il existe aussi des types d'ondelettes pour des fonctions sphériques [SS95]. Elles sont généralement beaucoup plus complexes et leur avantage n'est pas énorme.

2.4.3 SRBF

Tsai et Shih [TS06] expliquent comment les SRBFs peuvent représenter une carte d'environnement et une BRDF arbitraire. Les SRBFs sont des fonctions circulairement symétriques par rapport à un axe défini sur S^m, qui est la sphère unité dans $\mathbb{R}^{m+1}$. Soient η et ξ deux points sur S^m, et θ la distance géodésique entre η et ξ, c'est-à-dire $\theta = \arccos(\eta \cdot \xi)$. Une SRBF est une fonction qui dépend de θ, et qui peut être exprimée en terme d'une expansion de polynômes de Legendre comme

$$G(\cos\theta) = G(\eta \cdot \xi) = \sum_{l=0}^{\infty} G_l P_l(\eta \cdot \xi) \tag{2.19}$$

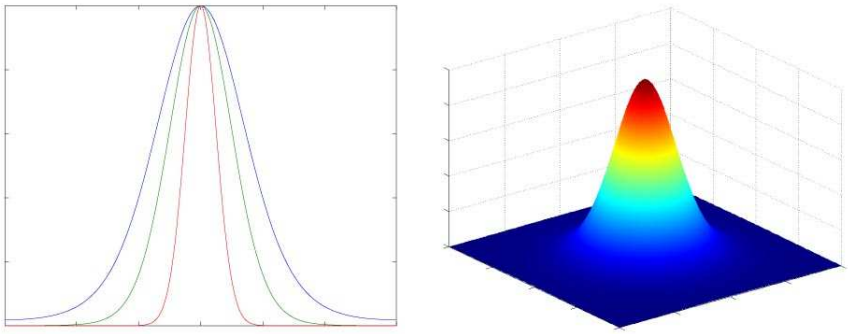
où $P_l(\eta \cdot \xi)$ est le polynôme de Legendre normalisé de degré l, et G_l sont les coefficients qui satisfont $G_l \leq 0$ et $\sum_{l=0}^{\infty} G_l < \infty$. ξ est le centre de la SRBF. Par la propriété que les polynômes de Legendre sont orthogonaux dans l'intervalle $[-1, +1]$, les expansions facilitent la convolution de deux SRBFs, nommée aussi intégrale sphérique singulière, par

$$(G *_m H)(\xi_g \cdot \xi_h) = \int_{S^m} G(\eta \cdot \xi_g) H(\eta \cdot \xi_h) d\omega(\eta) \tag{2.20}$$

$$= \sum_{l=0}^{\infty} G_l H_l \frac{\omega_m}{d_{m,l}} P_l(\xi_g \cdot \xi_h) \tag{2.21}$$

où ω_m est la surface totale de S^m, $d_{m,l}$ est la dimension de l'espace des harmoniques sphériques d'ordre l sur S^m et $d\omega$ est la dérivée d'un élément de surface sur S^m.

Deux exemples de SRBFs sont la fonction Abel-Poisson et la fonction gaussienne. Dans le cas de la gaussienne, l'évaluation de l'intégrale sphérique singulière est efficace



(a) (b)

FIGURE 2.7 – (a) Le tracé de SRBFs gaussiennes avec des paramètres de bande passante différents. (b) Un rendu 3D d'une SRBF gaussienne. Source [TS06].

$$(G_{*m}^{Gau} H^{Gau})(\xi_g \cdot \xi_h; \lambda_g, \lambda_h) = e^{-(\lambda_g+\lambda_h)} \omega_m \Gamma \left(\frac{m+1}{2}\right) I_{\frac{m-1}{2}}(||r||) \left(\frac{2}{||r||}\right)^{\frac{m-1}{2}} \quad (2.22)$$

où λ est le paramètre de bande passante, qui contrôle la couverture de la SRBF et $r = \lambda_g \xi_g + \lambda_h + \xi_h$. La figure 2.7 montre un exemple de SRBFs gaussiennes.

Soit un ensemble de points distincts $\Xi = \{\xi_1, \ldots, \xi_n\}$ sur S^m, qui sont les centres des SRBFs, et un autre ensemble de nombres réels $\Lambda = \{\lambda_1, \ldots, \lambda_n\} \in \mathbb{R}$, qui sont les paramètres de bande passante correspondants, alors une fonction sphérique $F(\eta)$ peut être représentée comme une expansion en SRBFs comme

$$F(\eta) \approx \sum_{k=1}^{n} F_k G(\eta \cdot \xi_k; \lambda_k). \quad (2.23)$$

Le net avantage des SRBFs est qu'elles définissent une fonction sphérique directement dans l'espace S^m, et non pas dans un espace fréquentiel. Ceci implique qu'il n'est pas nécessaire d'effectuer la rotation de la fonction pour calculer l'intégrale entre deux frames orientés différemment.

Chapitre 3

Rendu inverse

3.1 Introduction

Les problèmes inverses sont habituellement beaucoup plus complexes à résoudre, mais forment de plus en plus un sujet important dans la recherche en infographie, car leurs applications sont multiples. Quelques exemples d'utilisations de problèmes inverses abordés en infographie sont : le design de l'éclairage (p. ex. positionnement automatique des sources de lumière ou modification de leur intensité, sujet du présent mémoire), le design d'animations (p. ex. la cinématique inverse) [WW92], le positionnement automatique de la caméra et d'objets [PP03]. On peut définir les problèmes inverses comme étant des problèmes où l'on doit inférer les valeurs de certains paramètres d'un système à partir de données observables ou de ce que l'on souhaite comme résultat final. Ceci est donc bien différent des problèmes dits directs, où les valeurs de tous les paramètres du système sont connues et utilisées pour simuler ou calculer le résultat final.

Par exemple, en infographie, un problème direct classique est de calculer la lumière réfléchie des objets dans une scène tridimensionnelle à partir des informations comme les sources de lumière, la géométrie et les matériaux (propriétés de réflectance) des objets, la position de la caméra, etc. Les problèmes directs peuvent être prouvés comme étant bien posés [HMH95]. Ce n'est pas le cas pour les problèmes inverses, qui manquent au moins un des critères de Hadamard [Had02] pour être considérés comme bien posés : la solution n'est pas linéaire sur les données, ce qui implique qu'une petite erreur de mesure peut engendrer des erreurs considérables dans la solution [HMH95].

Dans ce chapitre, nous discutons principalement des problèmes inverses reliés à

l'éclairage dans le domaine de l'infographie. Quoique la problématique peut être associée à la vision par ordinateur, nous nous limitons à l'infographie où les paramètres de la caméra et de plusieurs aspects de la scène tridimensionnelle sont connus. Contrairement aux problèmes directs, ce sont des problèmes où un ou plusieurs aspects de la scène sont inconnus. Une caractéristique commune de ce type de problèmes est que l'on connaît d'avance l'apparence finale de certaines surfaces de la scène. C'est donc à l'algorithme inverse de modifier ou trouver les valeurs des paramètres des surfaces ou de la scène pour obtenir le résultat désiré. Comme mentionné dans l'introduction du mémoire, ce genre d'outils est d'une très grande importance entre autres pour les architectes, l'industrie du jeu vidéo et des effets spéciaux de films, surtout en réalité augmentée où le réel et le synthétique doivent coexister de façon naturelle.

3.2 Théorie

Dans cette section, on suppose qu'on utilise l'illumination globale. En réalité, on pourrait simplifier davantage le problème en considérant seulement l'illumination locale, en utilisant seulement des sources de lumière pontuelles et en omettant les interréflexions. C'est ce que plusieurs des travaux antérieurs ont fait. Par contre, l'illumination globale est souvent importante dans une scène, car elle englobe tous les phénomènes reliés aux interréflexions de la lumière (p. ex. les caustiques) qui ajoutent du réalisme au rendu.

L'illumination globale est reliée à la théorie du transport de la lumière et peut être vue comme un cas particulier de celui-ci [CW93, SP94]. Le comportement de la lumière qui se déplace est caractérisé par les propriétés des photons alors qu'ils traversent l'environnement. La radiance $L(\mathbf{r}, \omega)$ est définie comme la puissance lumineuse qui est irradiée à un certain point $\mathbf{r}$ dans une direction ω pour une unité d'angle solide pour une fréquence donnée :

$$L(\mathbf{r}, \omega) = L_e(\mathbf{r}, \omega) + \int_{\Omega_i} f_r(\mathbf{r}, \omega, \omega_i) L(\mathbf{r}, \omega_i) \cos\theta d\omega_i \tag{3.1}$$

où $L_e(\mathbf{r}, \omega)$ est l'émission initiale, f_r est la fonction bidirectionnelle de réflexion et/ou transmission (BRDF, BTDF), θ est l'angle entre la normale de la surface au point $\mathbf{r}$ et la direction ω, Ω_i est l'hémisphère des directions incidentes au point $\mathbf{r}$, et ω_i est la direction incidente de la lumière.

Cette équation peut être exprimée comme un opérateur linéaire. Premièrement, on définit l'opérateur de réflexion locale $\hat{K}$ par

$$(\hat{K}h)(\mathbf{r},\omega) \equiv \int_{\Omega_i} k(\mathbf{r};\omega_i \rightarrow \omega)h(\mathbf{r},\omega_i)\cos\theta d\omega_i \tag{3.2}$$

qui calcule la répartition de la lumière incidente. h correspond à la fonction qui décrit toute la lumière qui arrive au point $\mathbf{r}$. L'opérateur $\hat{K}$ transforme la distribution de la lumière incidente vers la distribution de la lumière réfléchie résultant d'une réflexion locale.

Soit $\mathcal{M}$ l'ensemble des surfaces dans la scène. On définit la fonction de visibilité de surface par

$$\nu(\mathbf{r},\omega) \equiv \inf\{x > 0 : \mathbf{r} + x\omega \in \mathcal{M}\} \tag{3.3}$$

où "inf" désigne la plus grande borne inférieure. Cette fonction donne la distance entre $\mathbf{r}$ et le point le plus près sur $\mathcal{M}$ dans la direction ω. Si un tel point n'existe pas, alors $\nu(\mathbf{r},\omega) = \infty$. On définit également la fonction du lancer de rayon par

$$\mathbf{p}(\mathbf{r},\omega) \equiv \mathbf{r} + \nu(\mathbf{r},\omega)\omega \tag{3.4}$$

qui est le point d'intersection entre le rayon $R(t) = \mathbf{r} + t\omega$ et $\mathcal{M}$. S'il n'existe pas d'intersection, alors la fonction n'est pas définie.

Ensuite, on peut définir l'opérateur $\hat{G}$ qui transforme la distribution de la lumière sortante vers la distribution de lumière incidente qui résulte de l'interréflexion entre les surfaces dans une scène :

$$(\hat{G}h)(\mathbf{r},\omega) \equiv \begin{cases} h(\mathbf{p}(\mathbf{r},-\omega),\omega) & \text{si } \nu(\mathbf{r},\omega) < \infty \\ 0 & \text{autrement.} \end{cases} \tag{3.5}$$

Avec ces opérateurs, on peut réexprimer l'intégrale de l'équation 3.1 comme suit :

$$L = L_e + \hat{K}\hat{G}L. \tag{3.6}$$

Cette équation permet de classifier les différents travaux précédents en fonction des éléments inconnus de celle-ci [Mar98].

Les problèmes directs sont ceux où l'on connaît les valeurs pour L_e, $\hat{K}$ et $\hat{G}$ afin de résoudre L. Si par contre on connaît (partiellement) L, on peut caractériser les problèmes d'éclairage inverse en fonction des inconnues de l'équation :

1. Si L_e est inconnue, mais qu'on connaît (partiellement) $\hat{K}$, $\hat{G}$ et L, alors c'est un problème de design de lumière. C'est-à-dire qu'on doit trouver les sources de lumière (L_e) qui, en affectant les objets ($\hat{G}$) constitués de certains matériaux ($\hat{K}$), permettront de retrouver L.
2. Si $\hat{K}$ est inconnue et qu'on connaît (partiellement) $\hat{G}$, L_e et L, alors c'est un problème de design de matériau. C'est-à-dire qu'on doit trouver la réflectance des objets qui, en fonction de la géométrie des objets ($\hat{G}$) permettra de réfléchir la lumière incidente L_e afin de donner L.
3. Si $\hat{G}$ est inconnue et qu'on connaît (partiellement) $\hat{K}$, L_e et L, alors c'est un problème de design de géométrie. C'est-à-dire qu'on doit déformer les objets ($\hat{G}$) afin d'obtenir une illumination désirée.

Dans ce mémoire, on s'intéresse à la première classe de problèmes. Il est à noter qu'un problème particulier pourrait combiner plus d'une classe de problèmes (p. ex. si on ne connaît pas les sources de lumière, ni la réflectance). Dans ce cas-ci, le problème serait encore plus mal posé, car une explosion (infinité) de possibilités d'illuminations incidentes et de réflectances des objets pourraient donner le même résultat.

Si l'illumination incidente est distante, qu'aucune interréflexion n'est prise en compte, que les BRDFs sont isotropes et que la géométrie des objets ainsi que les paramètres de la caméra sont connus, alors la lumière réfléchie d'un objet peut être considérée comme étant la convolution de deux signaux, la BRDF et la lumière incidente [RH01]. Ceci revient à dire qu'on filtre la lumière incidente par la BRDF. Le problème dans ce cas revient à faire la déconvolution des deux signaux. On peut alors en conclure que si l'illumination incidente et/ou la BRDF contiennent des hautes fréquences, alors le problème de la déconvolution des signaux est bien posé. Même si les restrictions de ce modèle sont fortes, c'est ce que nous utilisons dans notre premier système implémenté, car elles présentent des avantages certains (voir chapitre 5) et correspondent néanmoins à des besoins réels. Notre deuxième approche ne suppose pas que l'illumination incidente est distante, et donc les résultats sont plus précis mais à un coût supplémentaire en précalculs et utilisation mémoire.

3.3 Travaux antérieurs

Parmi les travaux antérieurs, on s'intéresse en particulier à deux classes de problèmes inverses : les problèmes de design de lumière et les problèmes de design de BRDF.

Les problèmes de design de lumière s'intéressent à modifier les paramètres des sources de lumière pour obtenir une illumination réfléchie désirée. L'utilisateur manipule directement l'illumination réfléchie en peignant de la lumière directement sur les surfaces des objets, soit en manipulant certains effets lumineux (p. ex. une ombre) directement comme des objets, ou en changeant certains paramètres globaux de la scène (p. ex. la luminosité).

Les problèmes de design de BRDF s'intéressent à modifier les paramètres de réflectance des surfaces des objets pour obtenir une illumination réfléchie désirée. Comme pour le design de lumière, l'utilisateur manipule soit directement l'illumination réfléchie ou change manuellement les paramètres de la surface.

Dans les deux cas, le but est le même : obtenir une certaine illumination. L'idéal pour ce genre de système de design est la possibilité d'obtenir des résultats dans des temps raisonnables, tout en ayant un rendu de scène d'assez bonne qualité. Pour y parvenir, les travaux antérieurs ont imposé certaines contraintes à leurs systèmes : par exemple de fixer la caméra ou de restreindre le problème à l'illumination directe.

Comme mentionné plus haut, il existe également une troisième classe de problèmes inverses pour l'illumination : la modification de la géométrie. On ne s'intéresse pas à cette classe parce que dans notre contexte, nous croyons que la géométrie de la scène est habituellement bien définie par l'artiste et que la forme d'un objet permet de l'identifier. La modification des objets n'est donc pas nécessairement une bonne idée dans ce contexte.

3.3.1 Problèmes de design de lumière

Dans cette section, nous présentons quelques travaux qui tentent de résoudre le problème inverse où l'on veut retrouver la lumière incidente L_e.

Dans le contexte où l'on souhaite positionner et/ou créer des sources de lumière, en considérant seulement l'illumination directe, un des premiers travaux sur le problème inverse d'illumination a été introduit par Poulin et Fournier [PF92]. Ils proposent d'utiliser les *highlights* et les ombres sur les objets dans le but de modéliser les sources de

lumière.

Dans le cas des *highlights*, ils considèrent le terme spéculaire du modèle de Phong tel qu'exprimé par Blinn [Bli77]. En laissant l'utilisateur indiquer le point d'intensité maximale du *highlight*, ils trouvent analytiquement la direction de la lumière. En choisissant un autre point sur la surface de l'objet, l'utilisateur indique où le terme spéculaire atteint un certain seuil ; le système peut alors déterminer la valeur du coefficient de rugosité. En imposant d'autres restrictions, comme par exemple de contraindre la lumière à rester dans un certain plan, on peut gérer certains cas particuliers de sources de lumière ponctuelles. Cette méthode ne donne donc que des sources de lumière ponctuelles et directionnelles.

Pour les ombres générées par une source directionnelle, la direction de la source est spécifiée en laissant l'utilisateur choisir deux points arbitraires dans la scène, le dernier étant dans l'ombre du premier. Pour une source ponctuelle, le volume d'ombre [WPF90] généré par celle-ci doit être utilisé. Pour une source surfacique, des nouvelles sources ponctuelles qui définissent les sommets de celle-ci sont nécessaires.

Cette méthode est limitée à l'illumination locale et à une BRDF spécifique. Aussi, l'interface n'est pas nécessairement très intuitive.

Le système de Poulin et al. [PRJ97] étend la technique précédente en plaçant des sources de lumière ponctuelles en dessinant des ombres ou des *highlights*, ou des sources de lumière surfaciques en dessinant des ombres et pénombres. Lorsque l'utilisateur dessine dans le système, une série continue de points sont créés et immédiatement déposés dans la scène en 3D. Ces points sont alors considérés comme étant tous dans l'ombre ou dans le *highlight* selon le cas. La technique commence en définissant pour chacun des points tracés un cône de positions possibles pour la source de lumière. Le volume où peut être placée la lumière correspond alors à l'intersection de tous ces cônes. Un volume infini produira une source directionnelle, sinon une source ponctuelle. Le problème est posé comme une optimisation contrainte avec comme fonction objectif de maximiser la distance entre les points tracés et la source de lumière en satisfaisant les contraintes que la source de lumière doit être à l'intérieur de tous les cônes et pour les ombres, que l'obstacle soit entre les points tracés et la source de lumière.

Pour les sources de lumière surfaciques, l'utilisateur dessine l'ombre et la pénombre (voir figure 3.1). Pour l'ombre, chaque point de la source doit être dans tous les cônes.

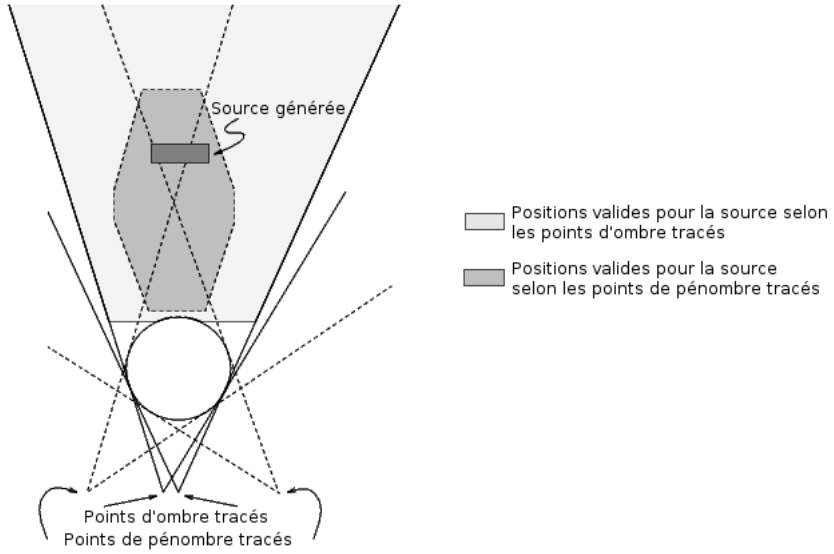


FIGURE 3.1 – Exemple de contraintes imposées à une source de lumière surfacique selon le dessin d'une ombre et d'une pénombre dans le système de Poulin et al. [PRJ97].

Pour la pénombre, au moins un point de la lumière doit être dans tous les cônes. Quand l'utilisateur dessine des *highlights*, un point sur la surface est considéré comme étant dans le *highlight* si l'évaluation de la fonction spéculaire de Phong à ce point donne une valeur plus grande qu'un certain seuil. Ensuite, le coefficient de rugosité n est diminué (ce qui élargit le cône) jusqu'à ce qu'une intersection soit possible.

L'avantage de ce système par rapport au précédent est que son interface est beaucoup plus intuitive que de manipuler des points. Par contre, les mêmes restrictions s'appliquent au niveau des capacités du système : exclusivement de l'illumination directe avec une BRDF selon le modèle de Phong.

Dans la même veine, Pellacini et al. [PTG02] présentent un système pour modifier en temps réel l'apparance d'une ombre déjà existante dans une scène, en considérant celle-ci comme un objet à part entière que l'on peut déplacer, agrandir, etc. Après une opération, la position, l'orientation et l'angle de défilement (*cutoff angle*) de l'unique spot de lumière sont corrigés pour obtenir le résultat. Le système permet aussi d'ajouter des fausses ombres ou des zones de lumière, qui sont appelées dans le contexte de l'article "biscuits" (*cookies*). La figure 3.2 présente les opérations possibles.

Le système semble être limité à une seule source de lumière, et utilise seulement l'illumination directe. De plus, l'ajout de ces "biscuits" fait que le résultat n'est pas

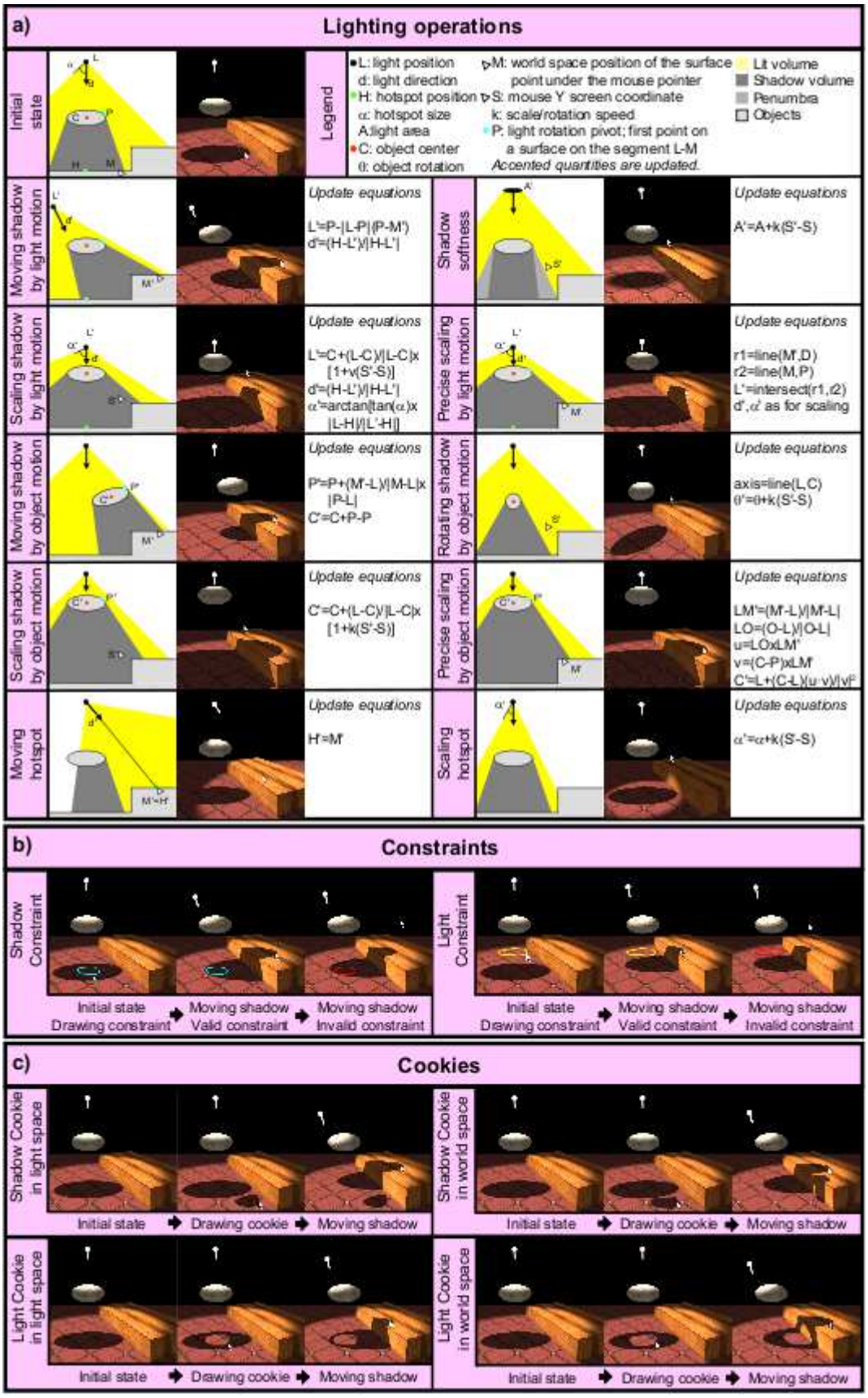
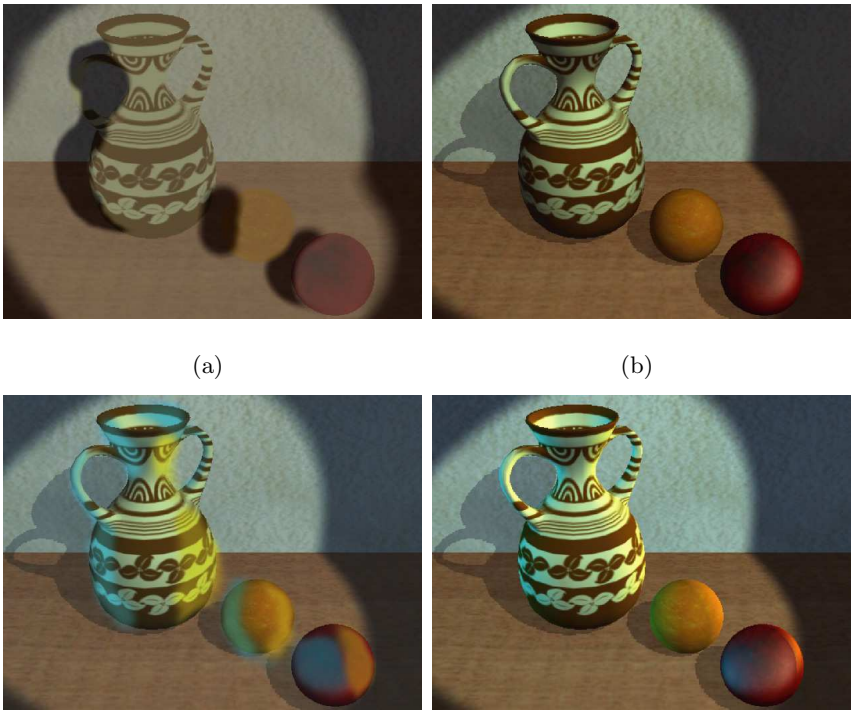


FIGURE 3.2 – Un tableau des opérations possibles dans le système de Pellacini et al. [PTG02]. Source [PTG02].



(a) (b)

(c) (d)

FIGURE 3.3 – Un exemple d'une séance de travail dans le système décrit de Pellacini et al. [PBMF07]. (a) L'utilisateur peinture de la lumière sur la scène pour donner l'atmosphère générale de la scène, et (b) le résultat de l'optimisation est affiché. (c) L'artiste peinture d'autres détails, et (d) le système optimise d'autres sources de lumière pour s'approcher du résultat désiré. Source [PBMF07].

nécessairement physiquement réaliste.

Le dernier système de positionnement de sources présenté ici (dans le contexte d'illumination directe) est celui de Pellacini et al. [PBMF07]. Ils introduisent un système fait sur mesure pour l'industrie du film. Pour une image d'une scène tridimensionnelle donnée, l'utilisateur peinture de la couleur, de la lumière et des ombres. Ils présentent une méthode de travail pour ajouter et modifier itérativement des sources de lumière. Un exemple de séance de design de sources de lumière est présenté à la figure 3.3.

Le système utilise une technique similaire à celle de Schoeneman et al. [SDS+93], c'est-à-dire que le système tente d'optimiser la fonction objectif suivante

$$\epsilon = \left(\sum_i I_i \cdot ||T_i - R_i||_2 \right) / \left(\sum_i I_i \right) \tag{3.7}$$

où I est une carte d'importance (où l'utilisateur peinture dans l'image), T_i est l'intensité

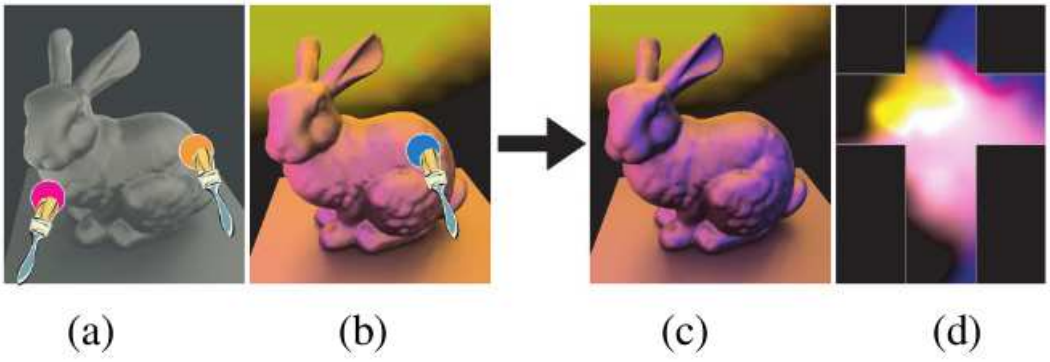


FIGURE 3.4 – Un exemple de manipulation possible dans le système de Okabe et al. [OMSI07]. (a) L'utilisateur peinture la couleur diffuse désirée directement sur l'objet (un modèle de lapin avec une BRDF blanche spéculaire). (b) La scène est rendue en utilisant la carte d'environnement estimée. L'utilisateur ajoute un *highlight* bleu sur l'objet. (c) Tous les effets lumineux désirés sont satisfaits en faisant le rendu du lapin avec la carte d'environnement estimée, montrée en (d). Source [OMSI07].

du pixel i dans l'image désirée et R_i est l'intensité du pixel i après le rendu. Le système essaie ainsi d'optimiser les paramètres des sources de lumière (position, orientation, angle de défilement, etc.) afin de minimiser ϵ. Il s'agit d'un problème non-linéaire avec beaucoup de variables, qui peut être instable, ou très difficile et coûteux à optimiser.

Le système a l'avantage de considérer le moteur de rendu comme une boîte noire (et donc de permettre n'importe quel moteur), mais utiliser un moteur avec illumination globale complexifirait énormément l'optimisation qui n'est déjà pas évidente. De plus, la caméra est fixe, donc la solution trouvée peut ne pas convenir pour les images subséquentes dans un film, ou pour l'utilisation dans un logiciel d'architecture.

Okabe et al. [OMSI07] présentent un système où l'on peinture l'apparence désirée sur un objet illuminé par une carte d'environnement (voir figure 3.4). Le pinceau dans ce cas-ci assigne aux sommets de la géométrie des couleurs, qui représenteront par la suite des contraintes. Après un coup de pinceau, le système effectue une optimisation pour retrouver la carte d'environnement qui satisfera le mieux possible ces contraintes. Pour résoudre le problème dans un temps raisonnable, ils utilisent une forme de PRT (section 2.2) avec SRBFs (section 2.4.3) pour représenter la carte d'environnement et des BRDFs arbitraires. Plusieurs autres fonctionnalités sont implémentées, comme par ex-

emple la possibilité de glisser sur la surface de l'objet les coups de pinceau précédemment appliqués, de pivoter la carte d'environnement pour déplacer les ombres, et le support pour la gamme dynamique étendue (*high dynamic range* ou HDR).

Ce système comporte aussi des limitations. D'abord, il s'agit d'un système où un seul objet est affecté par une carte d'environnement. Donc, c'est un bon système dans un contexte où, par exemple, l'on veut insérer un objet artificiel dans une photo existante. Mais, si on veut modifier une scène 3D en général, alors ce système est peu adéquat.

Leur système par optimisation est une façon élégante d'obtenir des résultats exacts, si c'est vraiment ce que l'on recherche. On préfèrera dans le contexte de ce mémoire l'approche additive pour ajouter de la lumière, ce qui s'apparente plus à l'approche d'un peintre (chapitre 4). Outre la divergence "philosophique" de l'approche, l'optimisation de la carte d'environnement peut causer des problèmes : comme elle essaye à tout prix d'obtenir la couleur à un certain sommet, elle pourra, pour y arriver, affecter des couleurs totalement contre-intuitives à l'environnement. Ainsi pour le point de vue au moment du coup de pinceau, le résultat sera correct, mais que dès que l'utilisateur visualisera l'objet sous un autre angle, il pourrait être éclairé d'une manière tout à fait autre à ce que l'utilisateur s'attendrait.

Dans le contexte de l'illumination globale, une des premières approches pour résoudre le problème inverse d'illumination incidente a été décrite par Shoeneman et al. [SDS+93]. Leur système permet de peinturer des couleurs directement sur les surfaces pour trouver les couleurs et les intensités des sources de lumière déjà existantes. Pour résoudre le problème, ils introduisent la formulation mathématique suivante. Supposons $\{\Phi^1, \ldots, \Phi^n\}$ l'ensemble des fonctions des contributions distinctes pour n sources de lumière dans l'environnement. Ces fonctions peuvent être calculées par n'importe quel algorithme d'illumination globale qui approxime l'équation 3.1. Alors, on peut décrire l'illumination de la scène comme une simple combinaison linéaire de la forme $\Psi = \sum_i^n u_i \Phi^i$, où u_i est le poids non-négatif de la source i de lumière sur l'environnement. Si on suppose une relation linéaire Ξ entre les valeurs d'intensités α_j dans la scène ou sur l'écran, alors on peut formuler le calcul d'illumination comme suit

$$\alpha_j = \Xi(\Psi) = \sum_{i=1}^{n} u_i \Xi(\Phi^i). \tag{3.8}$$

Avec une telle formulation, le problème inverse peut se traduire en une minimisation des moindres carrés $\min(\beta_j - \alpha_j)^2$ entre les valeurs désirées (peinturées) β_j et les valeurs obtenues α_j. Dans le contexte de cet article, la solution est trouvée en utilisant une version modifiée de Gauss-Seidel. Aussi, pour assurer un temps interactif en 1993, ils utilisent la méthode de la radiosité, dont le résultat est indépendant du point de vue, ce qui permet d'obtenir des fonctions Φ^i constantes. Ceci implique que toute réflexion spéculaire est absente.

Costa et al. [CSF99] ont implémenté une technique automatique qui cherche le meilleur emplacement et les intensités pour des sources de lumière. Dans une étape de prétraitement, les exigences de l'utilisateur (appelées les *inverse luminaries* ou *IL*s) sont considérées comme des sources d'intensité unitaire, qui se propagent dans l'environnement. Ceci permet d'utiliser n'importe quel système de simulation d'illumination globale pour calculer la distribution d'importance des *IL*s sur un ensemble de surfaces. Ensuite, une fonction fournie par l'utilisateur au moyen de scripts détermine les buts fixés et les contraintes de l'illumination, ainsi que les contraintes géométriques qui pourraient être souhaitables en termes de restrictions sur l'emplacement des sources. Cette fonction est en réalité une fonction de coût que le système d'optimisation doit minimiser.

À partir de cette fonction, une étape de validation essaie de trouver les incompatibilités entre les buts fixés et les sources de lumière déjà placées, suivie d'une étape de calcul qui essaie de trouver l'emplacement et l'orientation des sources de lumière. La minimisation de la fonction est effectuée par l'algorithme du recuit simulé.

Quoique l'approche soit prometteuse, l'utilisation de scripts pour définir les contraintes n'est pas nécessairement très intuitive, contrairement à un système qui permet de peinturer directement de la lumière sur les surfaces. Le choix du recuit simulé comme algorithme de minimisation est aussi discutable, car il ne s'agit pas de l'algorithme le plus rapide qui soit.

Dans un contexte un peu différent où l'illumination incidente et la réflectance sont inconnues, Kawai et al. [KPC93] proposent un système pour résoudre ce problème combiné. Ils utilisent la radiosité pour minimiser l'énergie globale de la scène au lieu de minimiser les moindres carrés entre les valeurs désirées et courantes. L'énergie de la scène correspond à la somme pondérée de l'émission des sources et de la réflexion des éléments de surface en ajoutant des termes physiques et des termes basés sur la per-

ception humaine. Les termes physiques comprennent les émissions, la distribution et la directionalité des sources de lumière et les réflectances des éléments de surface, tandis que les termes de perception humaine comprennent des quantités subjectives comme la clarté, la chaleur, etc. Tous ces termes sont considérés dans la fonction de minimisation comme des contraintes. Le système utilise la technique de Broyden-Fletcher-Goldfarb-Shanno (BFGS) d'optimisation non-linéaire pour résoudre le problème.

Leur système ne permet donc pas de "peinturer" de la lumière dans la scène, ce qui est ce qu'on essaie d'accomplir dans ce mémoire. Il s'agit vraiment ici de changer globalement l'atmosphère générale de la scène en utilisant potentiellement un grand nombre de contraintes qui permettent d'obtenir des buts, comme par exemple "je veux que la pièce soit plus invitante". Leur système utilise aussi la radiosité comme technique de rendu, ce qui élimine toute possibilité de réflexions spéculaires.

3.3.2 Design de BRDF

Quoique le problème de design de BRDFs n'est pas le problème visé par notre système, il reste que ce problème permet de modifier par rendu inverse l'apparence des objets dans une scène, ce qui rejoint en partie ce que nous voulons accomplir. Nous présentons ici quelques travaux importants sur le sujet.

Ben-Artzi et al. [BAOR06] proposent un système pour modifier les paramètres des BRDFs (p. ex. la rugosité pour la BRDF de Phong, l'indice de réfraction, le coefficient d'extinction et la distribution moyenne des pentes pour la BRDF de Cook-Torrance) et de voir le résultat en temps réel.

Ils déterminent premièrement tout ce qui est statique dans les calculs de la scène pour factoriser ce qui sera modifiable par l'utilisateur. En partant de l'équation de la réflexion de la lumière

$$R(x, \omega_o) = \int_\Omega L(x, \omega_i) V(x, \omega_i) \rho(\omega_i, \omega_o) S(\omega_i, \omega_o) d\omega_i \tag{3.9}$$

où L est la lumière incidente, V la visibilité binaire, ρ la BRDF et S le terme du cosinus tel que $\max(\omega_i \cdot \omega_o, 0)$. En fixant les paramètres de la caméra et en faisant l'hypothèse que n'importe quelle BRDF peut être exprimée sous forme d'une combinaison linéaire de bases

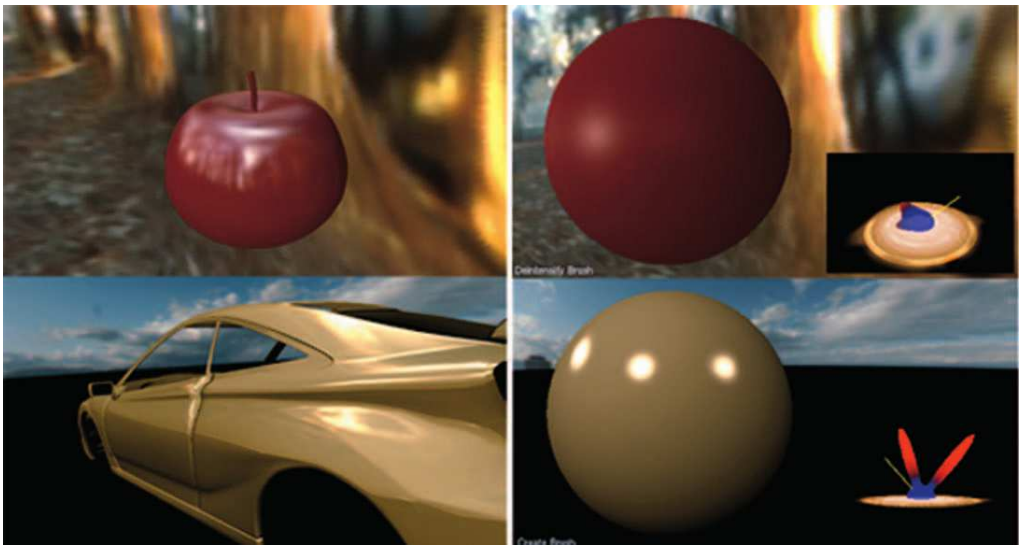


FIGURE 3.5 – L'interface du système *BRDF-shop* [CPK06] où l'on voit le canvas sphérique pour apporter les modifications sur la BRDF, et à gauche la fenêtre avec l'objet complexe pour apprécier les modifications. Source [CPK06].

$$\rho(\omega_i, \omega_o) = \sum_{j=1}^{J} c_j b_j(\omega_i, \omega_o) \tag{3.10}$$

on peut, en précalculant la fonction de transport, arriver à une formulation de la réflexion qui est un simple produit scalaire (comme dans le cas de la technique de PRT)

$$R(x, \omega_o(x)) = \sum_{j} c_j T_j(x) = \mathbf{c} \cdot \mathbf{T}(x) \tag{3.11}$$

où $T_j(x)$ sont les coefficients de la lumière réfléchie en chaque point, en ne considérant pas la BRDF. Cette quantité est précalculée et projetée dans la même base que la BRDF. La différence avec le PRT, c'est la BRDF qui peut être modifiée, et non pas l'illumination incidente. Dans l'article, les ondelettes sont utilisées comme bases.

Les limitations majeures de la technique sont que la caméra est fixe, et que les paramètres doivent être modifiés manuellement pour obtenir le résultat voulu.

Un système un peu différent des autres dans le contexte du rendu inverse pour le design de BRDF est celui de Colbert et al. [CPK06]. Un canvas sphérique est présenté afin de modifier la BRDF directement, et il permet de visualiser en même temps un objet complexe pour en apprécier les modifications. Ici, on ne peinture pas directement sur l'objet pour accomplir des résultats spécifiques (figure 3.5).

Le modèle de BRDF utilisé est une extension du modèle de Ward [Lar92]. Le modèle de Ward est

$$F(\omega_i, \omega_o) = \frac{1}{4\pi\alpha_x\alpha_y\sqrt{\cos\theta_i \cos\theta_o}} \exp\left[-2\frac{\left(\frac{\hat{h}\cdot\hat{x}}{\alpha_x}\right)^2 + \left(\frac{\hat{h}\cdot\hat{y}}{\alpha_y}\right)^2}{1+\hat{h}\cdot\hat{n}}\right], \tag{3.12}$$

où :

- ω_i et ω_o sont respectivement les directions incidente et sortante normalisées ;
- $\hat{x}$ et $\hat{y}$ sont les directions principales de l'anisotropie ;
- θ_i et θ_o sont les angles que font ω_i et ω_o avec $\hat{n}$;
- α_x et α_y sont les variances de la pente de la surface en directions de $\hat{x}$ et $\hat{y}$;
- $\hat{n}$ est la normale à la surface ;
- $\hat{h}$ est le vecteur bissecteur tel que $\hat{h} = (\omega_i + \omega_o)/||\omega_i + \omega_o||$.

Pour être capable de dessiner des *highlights* n'importe où sur la sphère et de placer plusieurs lobes spéculaires, une extension est faite, donnant le modèle suivant

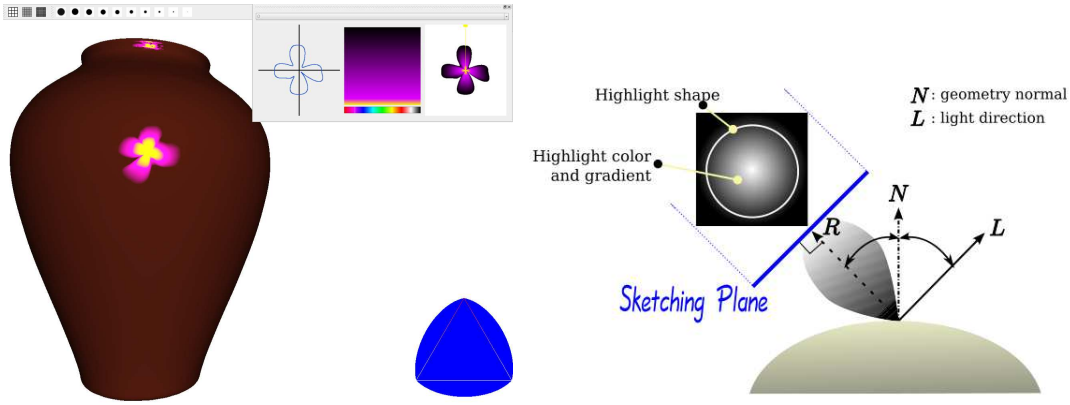
$$f(\omega_i, \omega_o) = \frac{\rho_d}{\pi} + \sum_{k=1}^{\#\text{ lobes}} \rho_{s_k} \cdot F_k(\omega_i, R_k\omega_o), \tag{3.13}$$

où le paramètre ρ_d représente l'albédo diffus pour le matériel et ρ_{s_k} est l'albédo spéculaire pour le k-ème lobe. F_k est le modèle de Ward pour le k-ème lobe. La matrice de transformation $\mathbf{R}_k$ permet de placer arbitrairement le k-ème lobe sur la sphère.

Ils ont implémenté plusieurs types de pinceaux : un pour créer un nouvel *highlight*, un pour modifier la rugosité d'un *highlight*, un pour étirer un *highlight* (changer l'anisotropie) et un dernier pour ajuster la distribution de l'énergie entre les différents lobes. Ces pinceaux modifient les différents paramètres du modèle étendu.

La technique de Pacanowski et al. [PGSP08] présente une interface où l'utilisateur peut dessiner des *highlights* arbitraires. Dans ce contexte, le *highlight* est considéré comme un objet artificiel projeté sur l'objet. Lors du déplacement de la caméra, le *highlight* est distortionné pour qu'il se comporte similairement à un véritable *highlight*. Ici, la BRDF et l'illumination de la scène n'influencent aucunement le *highlight*. L'interface du système est présentée à la figure 3.6(a).

Comme illustré à la figure 3.6(b), l'approche est basée sur la modification directe du *highlight*, qui est projeté sur un plan perpendiculaire à la direction réfléchie $\mathbf{R}$ de la lumière. Sur ce plan, l'artiste change la forme et le gradient de couleur avec des



(a) (b)

FIGURE 3.6 – (a) L'interface du système de Pacanowski et al. [PGSP08] pour dessiner des *highlights* très précis sur un objet. (b) La représentation d'un *highlight* sous le même système. Le *highlight* est défini sur un plan perpendiculaire de la direction réfléchie **R** de la lumière. Source [PGSP08].

outils 2D. Par conséquent, le *highlight* est défini pour une certaine direction de L, ici appelée frame de référence. Une séance de travail habituelle modifie donc la forme du *highlight* et ses couleurs, pour ensuite visualiser le résultat pour différentes directions de L. Le système réplique par défaut le *highlight* pour chaque frame de référence, mais un utilisateur peut modifier le *highlight* pour certains frames de référence. Alors le système interpolera le *highlight* entre les différents frames.

Pour interpoler le *highlight*, une représentation par champ de distance définie par une courbe est utilisée pour la forme, et une texture pour les couleurs et les intensités.

La technique ne peut pas tout à fait être considérée comme une solution du rendu inverse, car ici le résultat n'est pas nécessairement physiquement réaliste ; ainsi on ne retrouve pas les éléments manquants de l'équation de l'illumination 3.1. Par contre, elle permet d'ajouter des détails très précis, ce qui dans un contexte physiquement réaliste, serait à peu près impossible de faire.

Chapitre 4

Paradigme de peinture

Le paradigme de peinture est un peu le côté “philosophique” derrière un trait de pinceau appliqué par l'utilisateur. C'est-à-dire, qu'est-ce que signifie un trait de pinceau sur un objet ? Étant donné qu'il y a une multitude d'interprétations pour une multitude de contextes différents, nous avons décidé de diviser le problème en plusieurs stratégies, qui sont chacune orthogonales entre elles. Donc, l'utilisateur peut choisir, à sa guise, comment les modifications apportées à la scène doivent être menées. Au risque d'ajouter une certaine lourdeur dans l'utilisation de notre prototype, ce choix permet d'investiguer des tâches mieux ciblées, espérant avec l'expérience de pouvoir dégager des grandes lignes d'interfaces intuitives.

Mais tout d'abord, définissons ce que nous essayons d'accomplir. Nous supposons premièrement les faits suivants :

1. La scène est entièrement modélisée : la disposition, la géométrie et la réflectance (p. ex. la BRDF, les textures, etc.) des objets sont connues.
2. La position, la forme, la couleur et l'intensité d'émission des sources de lumière sont connues.
3. N'importe quels phénomènes lumineux (p. ex. réflexion, réfraction, illumination globale, etc.) peuvent se produire dans la scène. Cependant, la diffusion de lumière à l'intérieur des matériaux (*subsurface scattering*) n'est pas gérée dans notre implémentation.
4. La navigation dans la scène doit se faire librement.
5. L'utilisation du système doit se faire au moins en temps interactif.

Étant données ces contraintes fortes imposées au système, nous avons décidé de réduire le problème inverse général, où tout peut être modifié, à un problème de sélection et de manipulation de l'intensité des sources de lumière déjà existantes. Malgré cette forte réduction, le problème reste intéressant tant au point de vue théorique que pratique, où un tel outil devrait grandement aider les "designers" d'éclairage de scènes synthétiques.

Avant de présenter la division du problème, nous décrirons les deux modèles de base qui ont été considérés dans l'implémentation de nos systèmes.

4.1 Modèles

Nous distinguons deux modèles de modifications des sources de lumière dans notre paradigme de peinture.

Modèle exact. Le modèle exact, comme son nom l'indique, cherche à retrouver exactement la couleur peinturée sur la surface de l'objet. Les coups de pinceau sur l'objet se traduisent alors comme des contraintes sur la couleur peinturée sur ces zones. Habituellement, ce modèle est accompagné d'un système d'optimisation qui cherche les valeurs des paramètres des sources de lumière qui, après application de la formule de la réflexion, permettent d'obtenir une image finale où les zones contraintes seront aux couleurs peinturées. Un bon exemple d'un tel système est celui de Okabe et al. [OMSI07] où l'optimisation des coefficients de SRBFs d'une carte d'environnement éclairant l'objet repose sur une minimisation des moindres carrés entre le résultat de la réflexion des sommets et la couleur désirée. Le modèle est tout indiqué si l'on désire obtenir des couleurs exactes (ou à un delta près) à certains endroits.

Modèle additif. Le modèle additif ajoute ou enlève progressivement de la couleur à chaque coup de pinceau. On peut interpréter ce modèle comme faire de la peinture avec de l'encre semi-transparente ou de l'aquarelle : la superposition de plusieurs couches nous approche de la couleur désirée. Contrairement au modèle exact, la peinture des sommets d'un objet n'est pas vue comme une contrainte forte sur la couleur de celui-ci, et donc l'optimisation des paramètres des sources de lumière n'est pas nécessaire.

Le modèle exact est préférable si l'on veut un résultat précis. Mais est-ce que l'on

désire vraiment qu'un sommet particulier soit d'une couleur précise quand vient le temps de retoucher la distribution de la lumière de la scène ? Surtout que la réflectance autre que diffuse fera changer cette couleur si la caméra change de position. Le plus important est, à notre avis, le contraste entre les zones ombragées et éclairées, la luminance générale de la scène, etc. Ce sont ces facteurs qui donnent une atmosphère particulière à une scène. Donc si l'on désire, par exemple, éclairer davantage un endroit sur un objet, le modèle exact n'est pas aussi approprié dans ce cas.

C'est le modèle additif qui a été choisi dans nos deux systèmes. Donc quand l'usager peinture des sommets pour augmenter ou diminuer l'éclairage, nous sélectionnons les sources de lumière selon une des stratégies décrites ci-dessous à la section 4.3, sans vraiment se soucier des autres coups de pinceau. Bien sûr, cette façon de penser peut être contre-productive, car on peut "défaire" le travail réalisé sur une partie de l'objet par un coup de pinceau fait ailleurs sur celui-ci. On contrevient à ce problème en introduisant un pinceau de contrainte décrit à la section 4.2.

4.2 Pinceaux

Nous avons implémenté trois types de pinceaux dans nos deux systèmes :

Pinceau positif. Ce pinceau augmente l'éclairage sur les sommets où il est appliqué. L'augmentation de l'éclairage est faite en choisissant la/les sources selon la stratégie de sélection (section 4.3) et en les modifiant selon la stratégie de modification des sources de lumière choisie (section 4.4).

Pinceau négatif. Ce pinceau diminue l'éclairage sur les sommets où il est appliqué. Les sources choisies et la façon de les modifier sont les mêmes que pour le pinceau positif.

Pinceau de contrainte. Ce pinceau permet de marquer des sommets comme étant non modifiables. C'est-à-dire que l'on restreint la modification des sources de lumière afin de minimiser les changements sur les sommets marqués de ce pinceau.

4.3 Stratégie de sélection

Les stratégies de sélection déterminent, pour un coup de pinceau, quelles sont les sources de lumière sujettes à modification. Les stratégies utilisent l'importance absolue

d'un coup de pinceau (section 5.2.2), qui est en fait la quantité de lumière reçue de chaque source, filtrée par la BRDF pour le point de vue actuel. Si des contraintes sur des sommets ont été imposées par l'utilisation du pinceau de contrainte, alors c'est la stratégie de sélection qui doit gérer ces contraintes en pénalisant les sources qui influencent ces sommets. Les détails sur l'implémentation du pinceau de contrainte sont décrits pour chaque système aux sections 5.2.3 et 6.2.

Les stratégies de sélection sont les suivantes :

La plus importante. Cette stratégie ne considère que la source qui influence le plus les sommets peinturés.

Histogramme. Cette stratégie ordonne d'abord les sources en ordre décroissant selon leur importance absolue et sélectionne progressivement, dans l'ordre, des sources jusqu'à ce qu'un certain pourcentage de la quantité totale de lumière reçue par les sommets peinturés soit atteint.

Roulette russe. Cette stratégie commence de la même façon que l'histogramme : elle construit la liste des sources en ordre décroissant d'importance relative. Ensuite, elle choisit aléatoirement une source, selon la même probabilité que son importance relative.

4.4 Modification de l'intensité

Les stratégies de modification des sources de lumière déterminent le facteur d'augmentation (ou de diminution) de l'intensité lumineuse. Pour simplifier l'écriture, nous parlons seulement d'augmentation dans les descriptions : on applique une diminution dans le cas du pinceau négatif.

Les types de modifications sont les suivantes :

Constant. On ajoute une certaine valeur constante à l'intensité des sources.

Double. On double l'intensité des sources.

Direct. On calcule l'intensité des sources pour que la moyenne de la couleur des sommets soit la même que la couleur du pinceau. Ceci correspond à une implémentation approximative du modèle exact. Cette stratégie est utilisée pour converger plus rapidement à un certain résultat.

Fraction de direct. On calcule l'intensité comme pour la stratégie précédente, mais on applique seulement une certaine fraction de l'intensité aux sources de

lumière. Cette stratégie est utilisée pour s'approcher progressivement et rapidement à un certain résultat.

Chapitre 5

Système avec cartes d'environnement

Nous rappellons que le but du système est de fournir des outils qui permettent de peinturer sur des objets des intentions d'éclairage et d'identifier les sources de lumière qui contribuent le plus aux changements demandés. Nous modifions ensuite l'intensité de ces sources pour essayer d'obtenir l'effet désiré. Donc ici nous supposons que l'artiste ayant modélisé la scène est satisfait par sa géométrie, ses réflectances et l'emplacement des sources de lumière, et qu'il désire maintenant modifier les intensités des sources pour obtenir une atmosphère désirée. On verra à la section 8.5.3 qu'on peut aussi utiliser le système pour positionner des sources. Les autres fonctionnalités désirées dans le système sont la possibilité de déplacer librement la caméra, d'obtenir des résultats au moins en temps interactif et d'inclure les interréflexions dans la scène (l'illumination globale). Il serait bien aussi que le précalcul nécessaire pour une scène ne soit pas trop coûteux ni en temps ni en mémoire. Donc, en combinant toutes ces contraintes, il y a bien sûr des compromis à faire, et il faut utiliser des représentations qui permettent de compresser les données et d'accélérer les calculs.

Étant donné que le PRT (voir section 2.2) permet de visualiser un objet complexe sous illumination distante en temps réel, incluant le traitement des ombres, c'est ce dont notre premier système s'inspire. Par contre, nous sommes dans un contexte d'une scène arbitrairement complexe où l'hypothèse de l'illumination distante peut être souvent invalide. Donc, le système utilise la carte d'environnement comme représentation intermédiaire pour capturer la lumière incidente provenant des sources et de l'illumina-

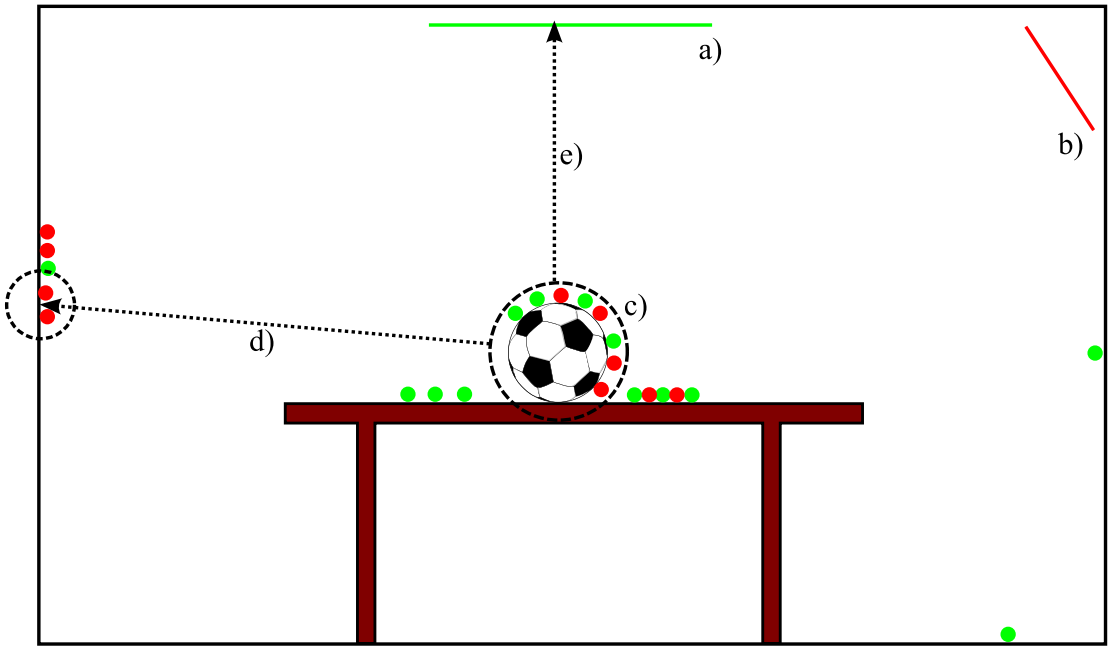


FIGURE 5.1 – Aperçu de la représentation utilisée pour encoder l'illumination incidente d'une scène arbitrairement complexe. Les sources de lumière surfaciques **a)** et **b)** émettent des photons (petits cercles de couleur). La carte d'environnement **c)** autour de l'objet (le ballon) stocke, pour chaque élément discret de la carte, la lumière incidente de la scène pour cette direction. Pour les interréflexions sur les surfaces de la scène, par exemple pour **d)**, on utilise le lancer de photons pour obtenir la quantité de lumière réfléchie pour chaque source. Lorsque le rayon partant de la carte d'environnement arrive directement sur une source de lumière, comme en **e)**, alors nous prenons la contribution directe.

tion globale vers un objet donné, en supposant que cette lumière provienne de l'infini. Bien sûr, ceci reste une approximation, mais il a été démontré (*far-field illumination*) que si la distance entre l'objet et les sources de lumière est environ cinq fois plus grande que la taille de l'objet, alors cette approximation est suffisamment satisfaisante. Un aperçu de cette représentation est illustré à la figure 5.1. Cette représentation pose cependant quelques contraintes sur le type d'objets qui fonctionnent bien avec ce système : l'objet doit être relativement compact comparativement à la scène. Par exemple, le système fonctionne bien pour un personnage, une chaise ou un bibelot, mais fonctionne moins bien pour un mur ou un plancher. C'est pourquoi nous limitons la modification par la peinture sur certains de ces objets de la scène. Aussi, dans ce système, on ne considère que l'auto-occultation, et non pas l'auto-réflexion. Donc plus l'objet est concave et spéculaire, plus le résultat pourra diverger de l'intention peinturée.

5.1 Aperçu de la technique

La technique comprend deux aspects importants : la visualisation et la peinture sur les objets. La clé de la technique est l'utilisation d'une carte d'environnement et d'une carte d'importance des sources de lumière. La carte d'environnement encode la lumière incidente totale pour chaque direction échantillonnée autour de l'objet. Elle sera utilisée pour calculer les couleurs aux sommets de l'objet peinturé. La carte d'importance des sources de lumière donne, également pour chaque direction autour de l'objet, la fraction de l'intensité lumineuse de chaque source de lumière qui arrive de cette direction. Il est possible de construire cette carte, car on suppose que les sources de lumière sont déjà à leurs positions finales.

Pour l'affichage, la carte d'environnement est mise à jour, si besoin est, à partir de la carte d'importance des sources de lumière, exprimée en ondelettes pour calculer la couleur de chaque sommet de l'objet. Pour pouvoir utiliser les ondelettes comme représentation, il faut être capable d'effectuer une rotation des ondelettes pour que les référentiels de la carte d'environnement, de la BRDF et de la carte de visibilité pour chaque sommet soient alignés (voir section 5.2.1).

Lors d'un coup de pinceau sur des sommets de l'objet afin de modifier l'intensité d'une ou plusieurs sources de lumière, on doit identifier les sources à modifier selon la stratégie choisie (voir section 4.3). La carte d'importance des sources de lumière est

alors utilisée pour extraire cette information. La représentation exacte de cette carte est définie à la section 5.2.2.

5.2 Représentations

Pour représenter la BRDF, la carte d'environnement et la carte de visibilité de façon compacte, et permettre d'utiliser la carte vidéo pour faire certains calculs rapidement, nous utilisons la représentation en ondelettes (voir section 2.4.2). Il y a cependant quelques problèmes avec la représentation en ondelettes, le plus important étant qu'il n'y a aucune façon analytique efficace d'effectuer la rotation des ondelettes pour aligner la BRDF, la carte d'environnement et la carte de visibilité, pour ensuite pouvoir calculer le triple produit scalaire pour obtenir la couleur réfléchie en un sommet [NRH04]. C'est pourquoi on utilise la méthode présentée par Wang et al. [WNLH06], qui permet de faire la rotation des ondelettes, et qu'on présente ici.

5.2.1 Rotation des ondelettes

Dans cette section, nous dérivons la formule pour effectuer la rotation d'une base générale pour une fonction définie sur la sphère. Ceci aboutit à une matrice de transformation pour chaque référentiel local. Dans le cas des ondelettes, la matrice résultante est très creuse. Ici, on suppose que la lumière est distante et qu'on ignore les interréflexions.

On commence par l'équation de la réflexion à un point de normale $\mathbf{n}$ dans la direction ω_o dans le cas de l'illumination distante :

$$L_r(\mathbf{n}, \omega_o) = \int_{\Omega_{(\mathbf{n})}} \tilde{L}(\mathbf{n}, \omega) f_r(\omega, \omega_o) V(\omega)(\omega \cdot \mathbf{n}_y) d\omega. \tag{5.1}$$

Ceci est un rappel de l'équation de la réflexion de la lumière L_r, pour un référentiel local comme l'intégrale sur l'hémisphère Ω de toute la lumière incidente. Ici la normale $\mathbf{n}$ est exprimée dans le système de coordonnées de l'objet, et la direction incidente ω et la direction de vue ω_o sont toutes deux exprimées dans le référentiel local. $\tilde{L}$ est la lumière incidente après la rotation dans le référentiel local ; f_r est la BRDF de la surface ; $V(\omega)$ est la fonction binaire de visibilité pour la direction incidente ω ; et $\omega \cdot \mathbf{n}_y$ est le terme du cosinus (où $\mathbf{n}_y$ correspond à la normale $\mathbf{n}$ dans le référentiel local).

Habituellement, le terme du cosinus est incorporé dans la définition de la BRDF, ce qui rend l'équation 5.1 essentiellement l'intégrale triple entre la lumière incidente,

la BRDF et la fonction de visibilité. Si on connaît la lumière incidente locale $\tilde{L}$ et la direction de vue ω_o, on peut utiliser des approximations avec des bases comme les harmoniques sphériques ou les ondelettes pour représenter $\tilde{L}$, f_r et V comme des vecteurs de coefficients $\tilde{\mathbf{L}}$, $\mathbf{F}_r$ et $\mathbf{V}$ (voir chapitre 2). L'intégrale devient alors un simple produit scalaire de trois vecteurs : $L_r = \mathbf{F}_r \cdot \tilde{\mathbf{L}} \cdot \mathbf{V}$.

La lumière incidente $\tilde{L}(\mathbf{n}, \omega)$ dans le référentiel local est reliée à la lumière incidente globale L par une rotation :

$$\tilde{L}(\mathbf{n}, \omega) = L(R_{(\mathbf{n})} \cdot \omega). \tag{5.2}$$

Ici, $R_{(\mathbf{n})} = [\mathbf{s}, \mathbf{n}, \mathbf{t}]$ est simplement une matrice 3×3 qui transforme des coordonnées locales vers des coordonnées globales, où $\mathbf{n}$ est la normale et $\mathbf{s}$ et $\mathbf{t}$ sont des vecteurs tangents du référentiel local et perpendiculaires entre eux. Puisque habituellement les objets tridimensionnels n'ont pas de vecteurs tangents définis, nous utilisons une recette simple (voir appendice A.1) pour les générer à partir de $\mathbf{n}$, ce qui veut dire que chaque référentiel local est identifié et indexé uniquement par $\mathbf{n}$. Pour la simplification de la description de la technique, nous fixons à partir d'ici un référentiel local, défini par $\mathbf{n}$ et la direction de vue ω_o. Dans ce cas, on peut écrire la rotation $R_{(\mathbf{n})}$ comme R et la BRDF $f_r(\omega, \omega_o)$ comme $f_r(\omega)$. Autrement, les vecteurs $\mathbf{s}$ et $\mathbf{t}$ peuvent être fournis au préalable avec la définition de l'objet lui-même.

Nous projetons d'abord la lumière incidente L sur un ensemble de bases orthonormales ξ_i (appelées la base source) qui est défini dans le système de coordonnées de l'objet :

$$L(R \cdot \omega) = \sum_i L_i \xi_i (R \cdot \omega). \tag{5.3}$$

Puisque la base $\xi_i(R \cdot \omega)$ est elle-même une fonction sphérique, elle peut être projetée à nouveau sur un ensemble (possiblement différent) de bases orthonormales $\zeta_j(\omega)$ (appelées la base cible) qui est défini dans les coordonnées du référentiel local :

$$\xi_i(R \cdot \omega) = \sum_j R_{ij} \zeta_j(\omega). \tag{5.4}$$

Les coefficients R_{ij} forment une matrice appelée la matrice de transformation de base. Chaque R_{ij} stocke la projection de la base source ξ_i vers la base cible ζ_j sous

la rotation R. En d'autres mots, cette matrice représente la transformation linéaire des bases sources vers les bases cibles. En substituant l'équation 5.4 dans l'équation 5.3 et en réarrangeant les termes, on exprime la lumière incidente sur le référentiel local (exprimée comme un ensemble de bases cibles) comme :

$$L(R \cdot \omega) = \sum_j \left(\sum_i R_{ij} L_i \right) \zeta_j(\omega). \tag{5.5}$$

Nous projetons également la BRDF et la carte de visibilité dans l'ensemble de bases cibles :

$$f_r(\omega) = \sum_k \rho_k \zeta_k(\omega) \tag{5.6}$$

$$V(\omega) = \sum_k v_k \zeta_k(\omega). \tag{5.7}$$

Maintenant que la lumière incidente, la BRDF et la carte de visibilité sont exprimées dans la même base orthonormale, on peut écrire leur intégrale dans une forme matricielle compacte comme

$$L_r = \int L(R \cdot \omega) f_r(\omega) V(\omega) d\omega = \sum_k \rho_k v_k \left(\sum_i R_{ik} L_i \right) = \mathbf{F}_r \cdot \mathbf{V} \cdot (\mathbf{R} \times \mathbf{L}) \tag{5.8}$$

où $\mathbf{F}_r$ est le vecteur colonne (ρ_k) représentant la BRDF dans l'ensemble des bases cibles ; $\mathbf{V}$ est le vecteur colonne (v_k) représentant la carte de visibilité dans l'ensemble des bases cibles ; $\mathbf{L}$ est le vecteur colonne (L_i) représentant la lumière incidente globale dans l'ensemble des bases sources ; et $\mathbf{R}$ est la matrice de transformation de base (R_{ik}) entre les deux bases sous la rotation R. La figure 5.2 illustre la rotation dans l'espace des ondelettes ainsi que la rotation dans l'espace spatial.

Il est à noter que la matrice $\mathbf{R}$ n'est pas obligatoirement carrée et qu'il est possible de transformer un signal entre deux bases totalement arbitraires.

Jusqu'à maintenant, nous avons présenté la théorie derrière la rotation de bases. On doit maintenant l'appliquer aux ondelettes. On utilise les ondelettes de Haar à cause de leur simplicité. Les ondelettes de Haar ne peuvent représenter que des signaux linéaires (1D) ou rectangulaires (2D). Il faut donc utiliser une paramétrisation qui permette de prendre un signal défini sur la sphère (pour la lumière globale incidente) et sur l'hémisphère (pour la lumière locale incidente, la BRDF et la carte de visibilité pour

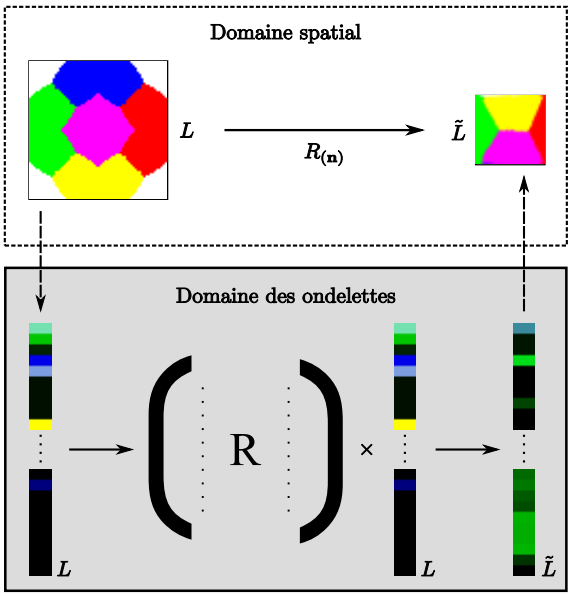


FIGURE 5.2 – Schéma de la transformation d'un environnement global vers un environnement local (qui implique la rotation d'une sous-section) dans le domaine spatial et le domaine des ondelettes.

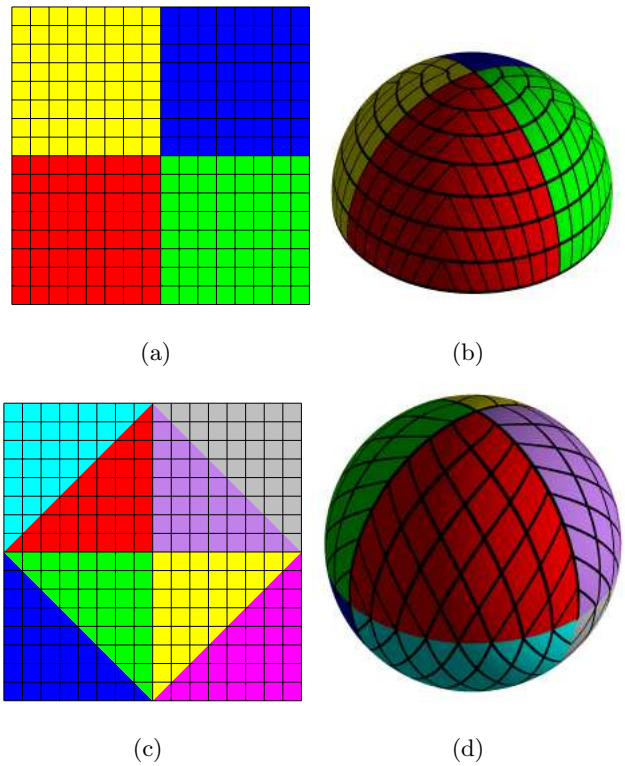


FIGURE 5.3 – (haut) Paramétrisation entre le carré et l'hémisphère telle que décrite dans [SC97]. (bas) Paramétrisation entre le carré et la sphère selon [CAM08].

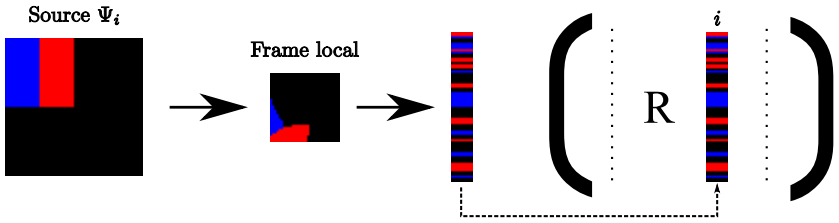


FIGURE 5.4 – Ce diagramme montre comment une colonne de la matrice de rotation **R** est calculée. On commence par construire l'ondelette de la source Ψ_i (Haar dans cet exemple) dans le domaine spatial. La couleur bleue dénote une valeur positive et la couleur rouge une valeur négative. L'ondelette est alors échantillonnée dans le référentiel local à la résolution de la cible et transformée en ondelettes dans un vecteur creux. Ce vecteur creux devient alors la i-ème colonne de la matrice **R**.

chaque sommet). Pour la sphère, on utilise la paramétrisation telle que décrite par Clarberg et al. [CAM08] qui combine la carte octaédrique [PH03] avec la paramétrisation de Shirley et Chiu [SC97], ce qui permet d'obtenir un carré où chaque pixel représente exactement la même aire une fois projeté sur la sphère et où la distorsion est faible. Pour l'hémisphère, on utilise directement la représentation décrite par Shirley et Chiu [SC97], qui possède les mêmes propriétés que pour la sphère. La figure 5.3 illustre les deux représentations.

Pour les prochaines explications, la résolution de la carte de lumière incidente source est $N = n^2$ et la résolution de la carte de lumière incidente cible, de la BRDF et des cartes de visibilité est $M = m^2$. Habituellement, la résolution choisie pour M est $N/4$.

Précalcul des matrices de rotation. Comme mentionné plus haut, il n'existe pas de façon analytique et efficace d'effectuer la rotation des ondelettes. L'astuce est donc de précalculer numériquement des matrices de rotation pour un sous-ensemble de toutes les normales possibles définies sur la sphère. La discrétisation de la sphère utilise la paramétrisation mentionnée ci-dessus. Étant donné qu'un référentiel local est identifié et indexé selon sa normale **n**, il sera possible par la suite de choisir la matrice de rotation qui effectuera le plus près possible l'opération nécessaire pour obtenir la carte d'illumination incidente locale à partir de la carte globale.

Soit **R** la matrice de rotation précalculée pour une certaine orientation **n**. **R** est une matrice $M \times N$ où N et M sont le nombre d'ondelettes pour la source et la cible respec-

(a) (b) (c)

FIGURE 5.5 – (a) Une tranche 2D de la BRDF *shiny metal* (les couleurs sont inversées). (b) La carte de visibilité du sommet encerclé en jaune pour (c) le modèle Suzanne. La couleur blanche correspond aux endroits où l'on voit l'environnement et la couleur noire aux occultations.

tivement. La i-ème colonne de $\mathbf{R}$ correspond à la projection de la i-ème ondelette de la source vers toutes les ondelettes de la cible pour le référentiel local. Nous précalculons une colonne à la fois. Pour ce faire, il faut construire la i-ème ondelette dans le domaine spatial pour le référentiel global (source). Ensuite, nous échantillonnons l'image de l'ondelette source à partir du référentiel local, ce qui donne l'image de l'ondelette source ξ_i vue à partir du référentiel local. Nous transformons ensuite l'image obtenue en ondelettes, et les coefficients donnent directement la i-ème colonne de la matrice $\mathbf{R}$. La figure 5.4 illustre le processus.

Précalcul des BRDFs. Les BRDFs doivent également être exprimées sous forme d'ondelettes pour que le système fonctionne. Pour ce faire, nous échantillonnons la BRDF dans la direction de vue ω_o, qui est un élément de la discrétisation de l'espace de toutes les directions de vue possibles (l'hémisphère), en utilisant la paramétrisation mentionnée ci-dessus. Pour chaque direction de vue, nous échantillonnons la direction de lumière incidente ω en utilisant la même discrétisation de l'hémisphère, avec une résolution M. Ceci donne une tranche 2D de la BRDF pour la direction de vue ω_o. On applique ensuite une transformation en ondelettes de Haar sur cette tranche, puis la compresse en mettant à zéro les coefficients les moins significatifs. La figure 5.5(a) montre une tranche 2D pour la BRDF *shiny metal*.

Précalcul de la carte de visibilité. De façon similaire à la BRDF, la carte de visibilité pour chaque sommet du modèle géométrique doit être précalculée et transformée en ondelettes. Pour chaque sommet, on échantillonne l'hémisphère autour du sommet en utilisant toujours la même paramétrisation, et en effectuant un lancer de rayons pour déterminer si la direction incidente ω est obstruée. Dans le cas d'une occultation, la valeur de la carte est zéro, autrement la valeur est un. Ensuite, on applique une transformation en ondelettes de Haar sur la carte. On empile ainsi les cartes de visibilité, en ordre croissant de l'identifiant des sommets. La figure 5.5(b) montre une carte de visibilité pour un sommet du modèle Suzanne (figure 5.5(c)).

Le rendu. Le rendu d'un objet sous l'influence d'une carte d'environnement consiste en deux étapes. La première étape est effectuée quand la carte d'environnement change. On itère au travers de toutes les matrices de rotation et effectue le calcul $\mathbf{R} \times \mathbf{L}$ de l'équation 5.8. Cette opération précalcule et stocke la lumière incidente locale $\tilde{L}$, en ondelettes, pour chaque normale échantillonnée. La deuxième étape, effectuée chaque fois que le point de vue change, itère sur tous les sommets de l'objet et calcule la couleur du sommet comme étant le produit scalaire entre $\tilde{L}$ pour la normale la plus près de celle du sommet, de la tranche de la BRDF $\mathbf{F}_r$ pour le point de vue et de la carte de visibilité $\mathbf{V}$ du sommet. On trouve la bonne tranche de la BRDF en calculant le vecteur de la direction de vue dans l'espace objet $\omega_{ob} = \mathbf{p_c} - \mathbf{p_s}$ où $\mathbf{p_c}$ est la position de la caméra et $\mathbf{p_s}$ est la position du sommet, toutes les deux dans l'espace objet. Ensuite on transforme ω_{ob} vers ω_o en effectuant une rotation selon la normale du sommet $\mathbf{n}$. On utilise ensuite ω_o comme index pour retrouver la bonne tranche de la BRDF. Finalement, la couleur au sommet est $L_r = \mathbf{F}_r \cdot \mathbf{V} \cdot \tilde{\mathbf{L}}$. Une implémentation du produit scalaire creux en GPU a été faite pour accélérer le traitement et est expliquée à la section 7.3.

5.2.2 Carte d'importance des sources de lumière

La carte d'importance des sources de lumière donne, pour chaque direction ω_i échantillonnée autour de l'objet, la fraction de lumière émise de chaque source qui arrive vers l'objet dans la direction ω_i. Dans le contexte d'une scène où l'on considère l'illumination globale, cette quantité de lumière est l'intégrale de tous les chemins de lumière possibles partant d'une source et arrivant sur l'objet dans la direction ω_i, en pondérant ces chemins par la fraction de lumière réfléchie sur les surfaces. Pour être

capable de calculer cette quantité, on peut utiliser une technique d'illumination globale, par exemple le lancer de photons [Jen01] ou le tracer de chemins bidirectionnel [DBB02], qui permet d'échantillonner l'espace des chemins de lumière de façon efficace. Une fois l'espace des chemins de lumière échantillonné, la fraction de l'intensité que l'on cherche devient alors une somme pondérée. Soit s_i la i-ème source de lumière dans la scène, on a

$$\hat{L}_{\omega_i,s_i} = P_{s_i}a_{i,1} + P_{s_i}a_{i,2} + \ldots + P_{s_i}a_{i,n}, \tag{5.9}$$

où $\hat{L}_{\omega_i,s_i}$ est la quantité de lumière totale arrivant sur la carte d'environnement de l'objet dans la direction ω_i à partir de la source s_i, P_{s_i} est l'intensité lumineuse totale de la source s_i et $a_{i,c}$ est l'atténuation totale de P_{s_i} pour un chemin arbitrairement complexe c arrivant dans la direction ω_i sur la carte d'environnement de l'objet. $a_{i,c}$ peut comprendre, sans s'y limiter, des réflexions diffuses, spéculaires et miroirs, des réfractions, de l'atténuation dans un volume participant, etc.

En factorisant la somme, on obtient

$$\hat{L}_{\omega_i,s_i} = P_{s_i}\left(a_{i,1} + a_{i,2} + \ldots + a_{i,n}\right). \tag{5.10}$$

Maintenant, il n'est pas nécessaire de conserver le détail des facteurs d'atténuation $a_{i,c}$ étant donné que les seules opérations qui sont faites dans le système modifient l'intensité des sources de lumière. Donc on définit $A_i = \sum_{c=1}^{n} a_{i,c}$, qui est l'approximation du facteur de transfert de la lumière provenant de s_i arrivant sur la carte d'environnement dans la direction incidente ω_i. On calcule et conserve donc les A_i pour chaque source de lumière, et ce pour chaque direction.

5.2.3 Contraintes sur les sommets

Le problème avec la résolution du problème inverse pour modifier l'illumination sur des objets est qu'une modification faite peut affecter un autre objet ou une autre partie du même objet. Nous mettons donc à la disposition de l'utilisateur un outil qui permet de pénaliser les modifications faites à l'environnement autour de certains sommets. On applique ces contraintes en peignant sur les sommets dont on veut préserver l'illumination. Pour que le sommet conserve sa couleur pour tous les points de vue (surtout dans le cas où la BRDF est spéculaire), nous contraignons toutes les modifications faites à

l'environnement dans l'hémisphère autour du sommet. Nous implémentons cette fonctionnalité en pénalisant les sources de lumière lors de la sélection quand un coup de pinceau est appliqué.

Les contraintes sont implémentées comme une pénalisation de la modification des sources de lumière qui affectent les environnements autour des sommets contraints. Nous avons observé que si peu de sommets sont contraints, plus ces contraintes sont importantes.

Étant donné que l'on connaît d'avance les sources, on peut calculer globalement un coefficient de pénalité pour chaque source. Ce coefficient est calculé selon la formule suivante :

$$C_{s_i} = \frac{1}{T} \sum_{j=1}^{T} \left(1 - \frac{S^c_{O_j}}{S_{O_j}} \right) \sum_{k=1}^{S^c_{O_j}} A_i^k, \tag{5.11}$$

où T est le nombre d'objets modifiables dans la scène, S_{O_j} est le nombre de sommets pour l'objet O_j, $S^c_{O_j}$ est le nombre de sommets contraints pour l'objet et A_i^k est la fraction totale de l'intensité lumineuse de la source i arrivant sur le sommet contraint k. Donc, si peu de sommets sur un objet sont contraints, alors leurs importances sont plus grandes, ce qui se traduit par une pénalité plus substantielle.

De toutes les informations qu'on conserve dans le système, A_i^k n'en est pas une. Donc pour obtenir cette information, on échantillonne A_i par rapport à l'hémisphère autour du sommet k, en sommant toutes les valeurs récupérées.

5.3 Calculs

5.3.1 Précalculs

L'avantage de cette technique est le temps relativement faible des précalculs qu'elle nécessite. Voici les différents précalculs qui doivent être faits :

Carte de visibilité. Pour les objets modifiables par coups de pinceau, les cartes de visibilité doivent être précalculées. Une explication sommaire est donnée à la section 5.2.1. Pour un objet, la carte ne doit être calculée qu'une fois, et l'objet pourra être utilisé dans plusieurs scènes par la suite.

BRDF. Pour chaque BRDF utilisée sur des objets modifiables, la conversion dans la représentation par ondelettes (voir section 5.2.1) doit être faite. Encore une fois,

ce précalcul ne doit être fait qu'une seule fois pour chaque BRDF, qui pourra être utilisée sur n'importe quel objet, dans n'importe quelle scène.

Matrices de rotation. C'est probablement le plus gros précalcul pour cette technique. Par contre, ce calcul est fait une seule fois, et les matrices seront ensuite utilisables dans n'importe quel contexte, pour une résolution de source et de cible fixe (voir section 5.2.1).

Lancer de photons. Pour être capable de calculer la carte d'importance des sources de lumière (section 5.2.2), nous utilisons la technique du lancer de photons [Jen01] pour retrouver les coefficients d'atténuation A_i de chaque direction de chaque carte d'importance de chaque source de lumière. Cette technique a été choisie car elle est relativement simple et les photons sont facilement stockables, contrairement par exemple au tracer de chemins bidirectionnel où la structure de données est plus complexe. Étant donné que l'information des photons qui sont envoyés dans la scène est indépendante des objets et du point de vue, le lancer des photons se fait une seule fois quand on lance le programme, pour une scène choisie. Certaines petites modifications sont faites à l'algorithme original du lancer de photons :

1. Un nombre identique de photons sont lancés pour chaque source de lumière.
2. L'identifiant de la source de lumière d'origine est stocké pour chaque photon.
3. L'intensité initiale de chaque photon est calculée en fonction que la source de lumière ait une intensité unitaire.

5.3.2 Sélection de l'objet

Lors de la sélection de l'objet que l'utilisateur veut peinturer, le calcul des coefficients d'atténuation est fait pour chaque source de lumière pour chaque direction de la carte d'environnement. Nous utilisons la technique du lancer de photons pour obtenir ces coefficients. Voici la procédure :

1. Nous trouvons le centre de l'objet. Il correspond au centre de sa boîte englobante orientée sur les axes.
2. Pour chaque direction échantillonnée, nous lançons un rayon à partir de ce centre dans cette direction.
3. Tant que le premier objet intersecté est l'objet sélectionné, on ignore cette intersection et on continue le rayon.

4. Pour une intersection valide, on prend les n photons les plus proches (en définissant une distance maximale) du point d'intersection. Les photons ont déjà été précédemment lancés lors du précalcul au démarrage du programme. Les photons considérés pourraient avoir des sources de lumière d'origines différentes. Étant donné que nous gardons l'information sur la source pour chaque photon, nous pouvons tout simplement faire la somme des valeurs des photons pour chaque source, pour ainsi calculer les A_i.

Maintenant qu'on a la carte d'importance des sources de lumière, on peut utiliser l'intensité actuelle des sources pour calculer la carte d'environnement pour l'affichage. Nous projetons ensuite la carte d'environnement en ondelettes, pour appliquer ensuite les matrices de rotation pour obtenir tous les frames locaux d'illumination incidente (section 5.2.1).

5.3.3 Mouvement de la caméra

Lorsque l'utilisateur se promène dans la scène, nous devons recalculer les couleurs des sommets de l'objet (section 5.2.1), puisque n'importe quelle BRDF peut être appliquée sur un objet.

5.3.4 Peinture

Comme mentionné au chapitre 4, la peinture d'un objet se fait toujours aux sommets de celui-ci. Donc, lorsque l'utilisateur peinture un objet avec un des pinceaux (voir section 4.2), on vérifie d'abord quels sont les sommets affectés par le coup de pinceau, et la couleur de la peinture sur chacun d'eux. Plus de détails sont fournis au chapitre 7.

Une fois ces informations trouvées, nous cherchons l'importance des sources de lumière pour le coup de pinceau donné. Pour ce faire, nous avons implémenté deux techniques : une technique qui utilise l'échantillonnage et une autre qui utilise un produit scalaire (voir ci-dessous). Une fois les importances trouvées, nous appliquons une des stratégies de sélection de sources (section 4.3), pour ensuite les modifier selon une des stratégies de modification (section 4.4).

Échantillonnage. Nous échantillonnons une certaine quantité de directions à partir des sommets affectés sur leur hémisphère respectif. L'échantillonnage des directions

se fait à partir de la BRDF. Une fois que nous connaissons les directions choisies dans l'hémisphère pour chacun des sommets, nous consultons la carte d'importance des sources de lumière, et nous sommons les importances pour chaque source dans les directions choisies.

Voici en détail les étapes effectuées :

1. Nous trouvons les sommets affectés par le coup de pinceau. Voir section 7.2.
2. Pour chaque sommet, nous cherchons la tranche de la BRDF et la carte de visibilité qui affecte le sommet pour le point de vue donné (section 5.2.1).
3. Nous créons une version filtrée de la BRDF par la carte de visibilité, $\mathbf{F}_r\mathbf{V}$, en multipliant, élément par élément, les coefficients d'ondelettes de la BRDF avec ceux de la carte de visibilité. Ceci donne les coefficients d'ondelettes de $\mathbf{F}_r\mathbf{V}$.
4. En utilisant $\mathbf{F}_r\mathbf{V}$, nous transformons un ensemble prédéterminé d'échantillons (directions) pour qu'il représente les zones importantes (section 5.3.4). Nous rappelons que $\mathbf{F}_r\mathbf{V}$, qui est une fonction hémisphérique, est paramétrisée sur un carré et est représentée par des ondelettes. Donc les échantillons prédéterminés sont en fait définis sur le carré unitaire. Dans notre implémentation, on utilise l'ensemble de Hammersley, dont les points sont bien distribués dans le plan et sont d'apparance aléatoire [WLH97]. Plusieurs autres distributions auraient pu être utilisées.
5. Pour chaque $\mathbf{F}_r\mathbf{V}$, les échantillons, qui sont à présent définis dans le frame local du sommet en traitement, sont transformés dans la carte d'importance des sources de lumière. Pour ce faire, nous transformons les échantillons sur le plan vers des points sur l'hémisphère en utilisant la paramétrisation inverse entre l'hémisphère et le plan. Nous transformons ensuite les échantillons vers le frame global en utilisant la matrice de rotation pour la normale $\mathbf{n}$ au sommet (voir appendice A.1). Finalement, nous transformons ces points (maintenant sur la sphère) vers le même plan que la carte d'importance des sources en utilisant la paramétrisation sphère-plan.
6. Chaque échantillon se trouve maintenant sur un élément discret de la carte d'importance des sources. Nous additionnons les importances pour chacune des sources de lumière, pour chaque élément de la carte qui contient un échantillon. Si un élément contient plus d'un échantillon, nous multiplions les importances par le nombre de ces échantillons.

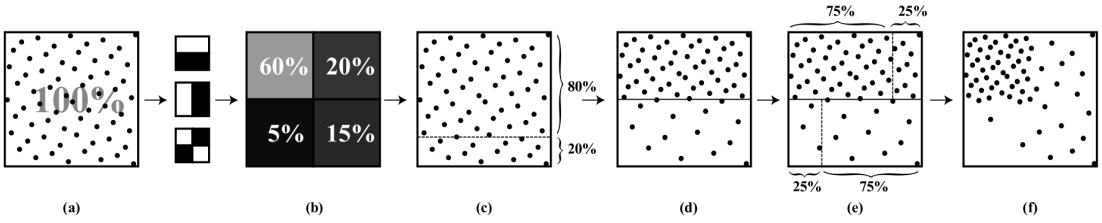


FIGURE 5.6 – Illustration de la transformation de points **(a)** selon un niveau d'un signal bidimensionnel compressé par des ondelettes où les pourcentages des quadrants **(b)** sont dérivés à partir des coefficients d'ondelettes pour la région courante en utilisant l'équation 5.13. L'ensemble initial de points est d'abord séparé en deux rangées **(c)** où leurs hauteurs sont déterminées par les probabilités totales et sont ensuite mises à l'échelle **(d)** pour qu'elles soient de la même hauteur. Finalement, chaque rangée est séparée horizontalement **(e)** selon les probabilités de ses sous-régions et les points sont encore une fois mis à l'échelle **(f)** pour que les sous-régions soient de la même taille. Le processus recommence à l'étape **(a)** pour chaque sous-région. Source [CJAMJ05].

Transformation d'échantillons avec les ondelettes. Pour bien échantillonner une BRDF (filtrée par une carte de visibilité) paramétrée sur un carré sous forme d'ondelettes, nous avons utilisé la technique décrite dans Clarberg et al. [CJAMJ05], que nous expliquons ici.

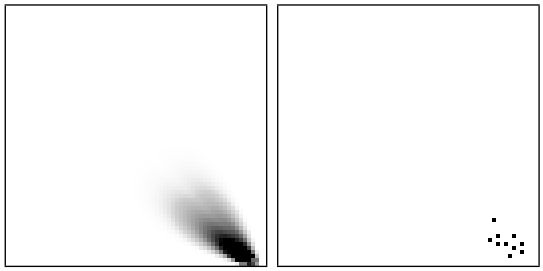
Soit une fonction en n dimensions, compressée avec des ondelettes :

$$H = \sum_i H_i \Psi_i. \tag{5.12}$$

Échantillonner la fonction requiert de calculer les probabilités des différentes régions de l'arbre des ondelettes. Nous définissons ces régions récursivement pour que la somme des probabilités des sous-régions pour chaque noeud de l'arbre soit égale à un. En utilisant les coefficients des fonctions de mise à l'échelle (*scaling functions*) pour une région donnée, les probabilités des sous-régions sont alors définies comme :

$$P_i^l = \frac{H_{i,0}^l}{\sum_t H_{t,0}^l}. \tag{5.13}$$

L'algorithme est hiérarchique : il commence par le niveau le plus bas et procède récursivement pour chaque niveau de la hiérarchie des ondelettes. La figure 5.6 illustre l'algorithme pour une tranche de BRDF.



(a) (b)

FIGURE 5.7 – (a) Une tranche de la BRDF *shiny metal*. (b) L'échantillonnage résultant avec la transformation de dix points selon l'ensemble de Hammersley. Les couleurs ont été inversées pour une meilleure visualisation : blanc équivaut à zéro et noir à un.

Si on transforme des points multidimensionnels, l'algorithme alterne entre les dimensions. En commençant par la première dimension, l'algorithme sépare l'ensemble de points en deux. Ensuite, pour la prochaine dimension, on sépare les deux nouveaux ensembles également en deux, donnant quatre ensembles. Nous continuons ainsi pour les autres dimensions. Ce processus peut être vu comme la construction d'un arbre k-dimensions avec les points de l'ensemble.

Pour obtenir la bonne distribution lors de la séparation des points dans une des dimensions, on doit calculer la probabilité totale de chaque moitié de cette dimension. Elles correspondent simplement à la somme des probabilités des sous-régions à l'intérieur de chaque moitié. Soient ces probabilités $P^l_{i,x-}$ et $P^l_{i,x+}$, où $P^l_{i,x+} = 1 - P^l_{i,x-}$. Pour effectuer la séparation, on divise d'abord les points de l'ensemble par rapport à un plan de séparation positionné de telle sorte que la première moitié contienne la fraction $P^l_{i,x-}$ du nombre d'échantillons et que la deuxième moitié contienne la fraction $P^l_{i,x+}$. Ensuite, chacun de ces deux nouveaux ensembles est agrandi pour que son domaine redevienne $[0, 1]$. Cette procédure est alors répétée sur chacun de ces sous-ensembles sur les dimensions subséquentes, jusqu'à ce que tous les points aient été transformés sur chaque dimension.

Lorsqu'une passe de transformation (c.-à-d. sur les n dimensions) pour un niveau a été complétée, l'algorithme réexécute récursivement cette étape sur les quatre sous-ensembles. Les deux conditions d'arrêt de l'algorithme sont : 1) si un ensemble est vide, alors on n'applique pas la transformation sur celui-ci et 2) si on a atteint le dernier niveau de l'arbre des ondelettes. La figure 5.7 montre un exemple de transformation

pour une tranche de BRDF.

Produit scalaire. Une méthode plus simple et plus exacte que d'échantillonner la BRDF filtrée ($\mathbf{F}_r\mathbf{V}$) afin d'accumuler les contributions de la carte d'importance des sources est tout simplement de faire le produit scalaire entre $\mathbf{F}_r\mathbf{V}$ et une version locale de la carte d'importance des sources. Le désavantage est qu'on doit, comme pour la carte d'environnement lorsqu'on fait le rendu, calculer des référentiels locaux pour chaque matrice de rotation d'ondelettes, pour chaque source de lumière. L'avantage est qu'on a déjà accès aux matrices de rotation pour les ondelettes, et que le calcul des frames locaux ne doit être fait qu'une seule fois pour chaque objet, et non pas chaque fois que l'environnement change.

Voici les étapes de cette version de l'algorithme :

1. Tout d'abord, quand un objet est sélectionné, nous calculons normalement la carte globale d'importance des sources (section 5.3.1). Nous projetons ensuite chacune des cartes d'importance (une pour chaque source) en ondelettes. Finalement, en utilisant les matrices de rotation d'ondelettes, nous créons les frames locaux de chacune des cartes. Nous compressons chaque frame local en gardant les coefficients d'ondelettes les plus significatifs.
2. Lorsque l'utilisateur peinture sur l'objet, nous trouvons les sommets affectés par le coup de pinceau (section 7.2).
3. Pour chaque sommet, nous cherchons la tranche de la BRDF, la carte de visibilité et le frame local des cartes d'importance correspondant le plus près à la normale du sommet.
4. Pour chaque carte d'importance et pour chaque sommet, nous faisons le produit scalaire triple entre la tranche de la BRDF, la carte de visibilité et la carte d'importance, ce qui donne l'importance de chaque source pour chaque sommet.
5. Nous faisons la somme des importances pour chaque source, ce qui donne l'importance totale par source pour le coup de pinceau.

On voit ici que la méthode en elle-même est beaucoup plus simple et n'approxime pas la BRDF filtrée par un échantillonnage. Elle requiert cependant plus de mémoire pour stocker les frames locaux des cartes d'importance. Les temps de calculs sont simi-

laires : on remplace un échantillonnage de fonctions hémisphériques par plusieurs produits scalaires. Ces produits sont calculés sur la carte vidéo (section 7.3).

5.4 Conclusion

Le système avec carte d'environnement a des avantages non négligeables : la vitesse de rendu d'un objet est rapide, le temps de précalcul est amorti si on utilise le système sur plusieurs scènes, surtout si un même objet est utilisé. Effectivement, le plus gros des précalculs sont pour des données qui sont partagées pour toutes les scènes. Une fois qu'ils sont calculés, le temps de calcul pour charger une scène est court (quelques secondes).

Par contre, la limitation majeure du système est qu'il suppose que la lumière provient de l'infini (lumière distante) et ne gère pas certains phénomènes comme la réflexion de la lumière par un objet sur lui-même. Donc le système fonctionne bien si la distance entre les sources de lumière et l'objet est environ cinq fois plus grande que la taille de l'objet (*far field*) et que l'objet n'est pas trop concave. Malheureusement, ceci est loin d'être toujours vrai.

C'est pourquoi on présente dans le prochain chapitre un autre système qui règle quelques-uns de ces problèmes, en échange d'un précalcul plus coûteux et d'une utilisation mémoire par objet plus importante.

Chapitre 6

Système avec points

Le système avec points est similaire à celui avec les cartes d'environnement (chapitre 5). Encore une fois, ce système fournit des outils pour peinturer des intentions d'illumination sur un objet afin d'identifier et de modifier l'intensité des sources de lumière selon des stratégies (voir section 4.3). On suppose aussi que la géométrie de la scène (positionnement des sources de lumière, des objets, le choix des textures, propriétés de réflexion, etc.) satisfait l'artiste et que la tâche à accomplir est de trouver l'intensité des sources.

Comme mentionné à la conclusion du chapitre précédent, le fait que le système avec carte d'environnement suppose que la lumière est distante représente une approximation de la réalité qui peut diverger rapidement si la distance réelle entre l'objet et les sources de lumière est plus petite que cinq fois la taille de l'objet. Cette approximation ne permet pas non plus de bien gérer les auto-réflexions de la lumière sur l'objet. Ceci limite grandement la généralité du type de scènes et d'objets qui fonctionnent bien avec ce système. Par exemple, peinturer sur un mur n'est pas très recommandé, à moins qu'on le subdivise en plusieurs petites zones du mur.

En utilisant un peu plus de mémoire et de précalcul, on peut cependant corriger la plupart de ces problèmes. Le rendu de l'objet sera également un peu plus coûteux, mais comme on le verra, restera dans des temps interactifs.

L'idée est tout simplement, pour chaque sommet des objets sélectionnables, de précalculer les cartes d'importance des sources de lumière (de la même façon que pour l'autre système). Cette fois-ci par contre, les cartes d'importance sont paramétrées sur un hémisphère au lieu d'une sphère. En précalculant ces informations, nous éliminons le

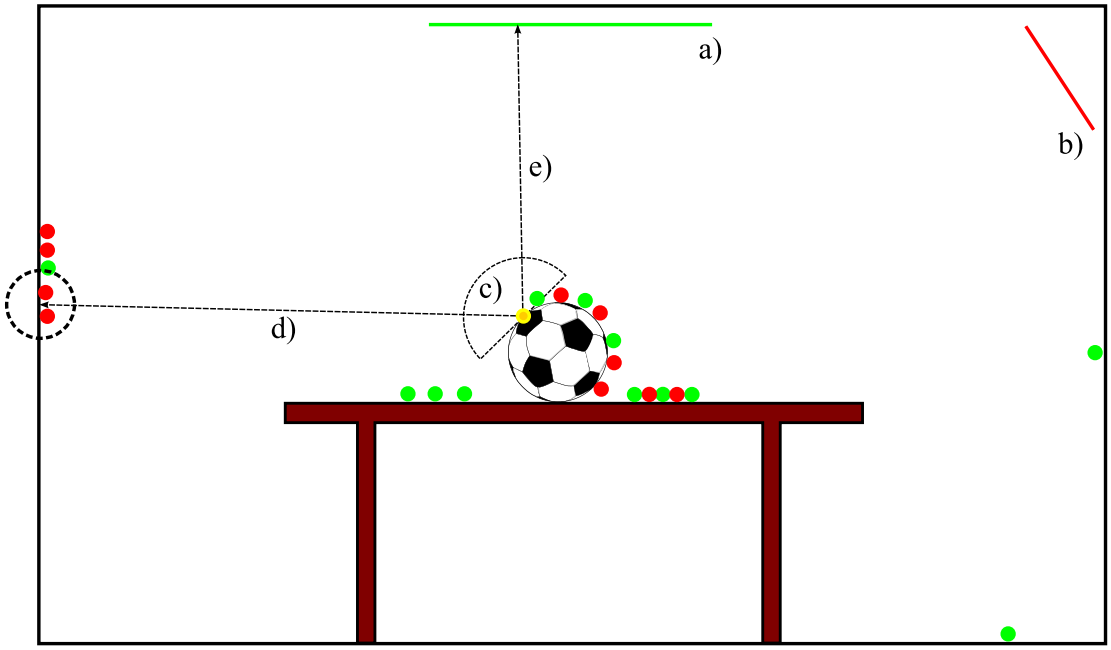


FIGURE 6.1 – Aperçu de la représentation utilisée pour encoder l'illumination incidente d'une scène arbitrairement complexe. Les sources de lumière surfaciques **a)** et **b)** émettent des photons (petits cercles de couleur). La carte d'environnement locale **c)** d'un sommet (cercle jaune) de l'objet (ballon) contient, pour chaque élément discret de la carte, la lumière incidente de la scène pour une certaine direction. Pour les interréflexions sur les surfaces de la scène, par exemple pour **d)**, on utilise le lancer de photons pour obtenir la quantité de lumière réfléchie pour chaque source. Lorsque le rayon partant de la carte d'environnement arrive directement sur une source de lumière, comme en **e)**, alors on prend la contribution directe.

besoin d'effectuer la rotation et la transformation de fonctions sphériques exprimées en ondelettes vers une fonction hémisphérique, exprimée également en ondelettes. Ceci est vrai, car toute l'information précalculée est déjà en fonction de l'environnement local de chaque sommet, et non pas en fonction de l'environnement global.

6.1 Aperçu de la technique

Comme pour le système précédent, la technique comprend deux aspects importants : la visualisation et la peinture des objets. Contrairement à l'autre système, ici on doit précalculer, pour chaque scène et pour chaque objet modifiable, la carte d'importance de

chaque lumière sur l'hémisphère de chaque sommet. On le fait en précalcul, car il s'agit d'un calcul relativement long. Dans l'autre système, le calcul des cartes d'importance globales était fait au démarrage du programme, car ce calcul est relativement court (moins de 10 secondes).

Pour l'affichage, nous utilisons les cartes d'importance pour calculer les cartes d'environnement de chaque sommet. Nous devons refaire ce calcul à chaque fois que l'intensité d'une ou plusieurs sources de lumière est modifiée. Une fois les cartes d'environnement mises à jour, pour obtenir la couleur d'un sommet, nous faisons exactement comme avant : nous faisons le produit scalaire entre la bonne tranche de la BRDF et la carte d'environnement. Ici, il n'y a pas de carte de visibilité : les auto-occultations et les interréflexions sont déjà prises en compte dans les cartes d'importance des sources lors du précalcul.

Lors d'un coup de pinceau sur des sommets d'un objet afin de modifier l'intensité d'une ou plusieurs sources de lumière, on doit savoir quelles sources modifier selon la stratégie choisie (voir section 4.3). Nous utilisons les cartes d'importance des sources pour obtenir cette information. La représentation exacte de cette carte est définie à la section 6.2.

6.2 Représentations

BRDF. La BRDF est encodée de façon identique au système avec la carte d'environnement. C'est-à-dire que la discrétisation de l'ensemble des directions de vue possibles sur l'hémisphère dans le frame local de la BRDF est faite. Ensuite nous échantillonnons la BRDF sous chaque direction de vue pour produire des tranches, qui forment des images carrées en utilisant la paramétrisation entre le carré et l'hémisphère.

Cartes d'importance. On peut voir les cartes d'importance par sommet comme une carte à plusieurs couches où chaque couche donne la fraction de l'intensité lumineuse d'une source de lumière atteignant le sommet (directement ou après un nombre arbitraire de rebonds). Pour un sommet, la lumière peut provenir de n'importe où autour, ceci est donc une fonction hémisphérique, et nous utilisons la même paramétrisation que pour la BRDF.

Cartes d'environnement. Les cartes d'environnement par sommet capturent la lumière totale arrivant au sommet. Nous utilisons, comme pour les autres informations, une représentation en ondelettes paramétrisée sur un carré.

Contraintes sur les sommets. La formule pour les contraintes globales sur les sources de lumière est la même que pour le système précédent (section 5.2.3). Cependant, le précalcul des cartes d'importance pour chaque sommet séparément facilite la tâche concernant le facteur R_i^k. En effet, on n'a plus besoin d'échantillonner une carte d'importance globale : on peut simplement faire la somme des coefficients d'ondelettes de la carte d'importance de la source i pour le sommet k afin d'obtenir R_i^k.

6.3 Calculs

6.3.1 Précalculs

Le précalcul nécessaire différencie le plus cette technique du système précédent. Ici un précalcul doit être fait pour chaque nouvelle scène et pour chaque objet qui pourra être peinturé.

Le précalcul consiste en une passe de lancer de photons modifiée :

1. En premier, nous lançons des photons dans la scène en supposant que les sources de lumière ont une intensité lumineuse unitaire. Les photons donnent donc la fraction relative de l'intensité de chaque source qui est dispersée dans la scène. Pour chaque photon, nous gardons l'identifiant de la source d'origine.
2. Pour chaque sommet de chaque objet modifiable, nous échantillonnons l'hémisphère autour du sommet en lançant des rayons dans la scène et en regardant la première intersection. Pour chaque intersection, nous cherchons les photons dans un rayon prédéterminé autour de l'intersection. Étant donné que nous connaissons la source pour chaque photon, nous pouvons déterminer la fraction d'intensité pour chaque source qui atteint le sommet dans la direction incidente actuelle. Nous répétons cette opération pour chaque direction dans l'hémisphère. La paramétrisation de l'hémisphère en carré donne une image pour chaque sommet, pour chaque source.
3. Nous compressons ces images en utilisant les ondelettes de Haar et en ne gardant que les coefficients les plus significatifs.

6.3.2 Sélection de l'objet

Lors de la sélection de l'objet que l'on veut peinturer, nous calculons et stockons les cartes d'environnement par sommet. Étant donné que nous avons déjà précalculé les cartes d'importance, les cartes d'environnement deviennent seulement une combinaison linéaire de ceux-ci. Soit P_i l'intensité lumineuse de la i-ème source et $\mathbf{I}_{i,s}$ le vecteur de coefficients d'ondelettes de la carte d'importance de la i-ème source pour le sommet k. Alors la carte d'environnement pour le sommet k est seulement

$$\mathbf{E}_k = \sum_{i=0}^{N} P_i \mathbf{I}_{i,k}, \tag{6.1}$$

où $\mathbf{E}_k$ est le vecteur de coefficients d'ondelettes pour la carte d'environnement autour du sommet k.

Les cartes d'environnement seront également recalculées pour tout changement à l'intensité d'une des sources.

6.3.3 Rendu

La couleur au sommet est simplement le produit scalaire entre sa carte d'environnement $\mathbf{E}_s$ et la tranche de la BRDF $\mathbf{F}_r$. On connaît déjà $\mathbf{E}_k$, donc il reste à trouver la bonne tranche de la BRDF en obtenant le vecteur de la direction de vue dans l'espace objet $\omega_{ob} = \mathbf{p_c} - \mathbf{p_s}$ où $\mathbf{p_c}$ est la position de la caméra et $\mathbf{p_s}$ est la position du sommet, tous les deux dans l'espace objet. Ensuite nous transformons ω_{ob} vers ω_o en effectuant une rotation selon la normale du sommet $\mathbf{n}$. Nous utilisons ω_o comme index pour retrouver la bonne tranche de la BRDF. Finalement, la couleur au sommet est $L_r = \mathbf{F}_r \cdot \mathbf{E}_s$. Une implémentation en GPU a été faite pour accélérer le traitement et est expliquée à la section 7.3.

6.3.4 Peinture

Comme mentionné au chapitre 4, la peinture d'un objet se fait toujours aux sommets de celui-ci. Donc, lorsque l'utilisateur peinture un objet avec un des pinceaux (voir section 4.2), on vérifie d'abord quels sont les sommets affectés par le coup de pinceau, et la couleur de la peinture sur chacun d'eux. Plus de détails sont fournis à la section 7.2.

Une fois ces informations trouvées, nous devons déterminer l'importance des sources de lumière pour le coup de pinceau donné. Nous utilisons la méthode du produit scalaire,

très semblable à celle du système précédent :

1. Nous trouvons les sommets qui ont été affectés par le coup de pinceau.
2. Pour chaque sommet, on cherche la tranche de la BRDF qui affecte le sommet pour le point de vue donné (section 6.3.3).
3. Nous calculons l'importance de chaque source, pour un sommet, en effectuant le produit scalaire entre chaque carte d'importance de celui-ci avec la tranche de la BRDF.
4. Pour chaque sommet, nous sommons l'importance pour chaque source, ce qui donne l'importance totale par source pour le coup de pinceau.

6.4 Conclusion

On peut remarquer que ce système partage plusieurs éléments avec le précédent : les représentations pour les BRDFs et les cartes d'environnements locales sont les mêmes et les intégrales se calculent de la même façon (par produit scalaire). Cependant, ce système est beaucoup plus simple car on élimine le besoin d'effectuer la transformation d'une carte d'environnement globale vers un référentiel local. De plus, ce système prend en compte que les sources de lumière ne sont pas distantes et prend en compte les auto-réflexions de la lumière sur un objet. Le rendu de l'objet reste en temps interactif.

Cependant, nous ajoutons un coût de précalcul pour chaque scène qui n'était pas nécessaire avec le système précédent et chaque objet occupe un peu plus d'espace mémoire.

Chapitre 7

Implémentation

Dans ce chapitre nous détaillons certaines parties de notre implémentation des systèmes présentés aux chapitres 5 et 6.

7.1 Outils

Les systèmes ont été implémentés en C++, en utilisant OpenGL comme API graphique. *Cg* [NVIa] a également été utilisé pour les *shaders* et *CUDA* [NVIb] pour les calculs parallèles sur la carte vidéo.

Le système d'exploitation utilisé est Linux 2.6.29.4 avec le compilateur GCC 4.2.

Nous avons utilisé le code de *PBRT* [PH04] (avec plusieurs modifications) pour l'implémentation des techniques d'illumination globale comme le tracer de chemins bidirectionnel et le lancer de photons.

7.2 Peinture

Nos systèmes reposent sur le fait qu'on peinture sur les sommets des objets. Nous détaillons ici brièvement comment on détecte si un coup de pinceau touche à des sommets ou pas :

1. D'abord, lorsque l'utilisateur peinture sur le canvas (pixels), nous gardons les pixels du trait et sa boîte englobante 2D.
2. Lorsque l'utilisateur termine le trait, on fait le rendu de l'objet en mode "point", sans le trait : c'est-à-dire qu'on dessine seulement les sommets. Le rendu est fait

dans un *framebuffer object*. Il faut aussi faire attention d'activer le *backface culling* ainsi que le test avec le tampon de profondeur. Chaque sommet aura une couleur différente qui détermine l'indice de celui-ci dans le modèle géométrique. Pour ce faire, on transcode un indice sur 32 bits vers une couleur RGBA, également de 32 bits (4×8 bits). Les endroits où il n'y a pas de sommet ont une couleur blanche opaque parfaite (0xFFFFFFFF).

3. On dessine le trait en utilisant les points enregistrés dans un autre *framebuffer object*.
4. Nous vérifions si le trait touche un des sommets en balayant la boîte englobante 2D du trait. Si un pixel contient un sommet et que le pixel correspondant a un pixel du trait, alors on a trouvé un sommet qui a été atteint. Étant donné que la couleur du pixel contenant un sommet est l'identifiant de celui-ci, alors on peut retrouver les informations concernant ce sommet.

7.3 Utilisation du GPU

Dans les techniques présentées, nous utilisons les ondelettes pour représenter certaines fonctions. Afin de compresser ces fonctions, nous gardons seulement les coefficients les plus significatifs, ce qui donne des vecteurs creux. Pour le premier système, on précalcule également des matrices pour transformer des vecteurs de coefficients (section 5.2.1). Ces matrices sont très creuses (moins de 1% des éléments sont non nuls).

Nous utilisons donc très souvent le produit entre une matrice creuse et un vecteur plein ainsi que le produit scalaire entre deux vecteurs creux. Afin d'augmenter l'interactivité de notre application, il est important que ces produits soient faits très rapidement.

Il existe plusieurs librairies qui utilisent une forme d'accélération pour effectuer ces opérations, par exemple *uBLAS*, la librairie C++ d'algèbre linéaire de *Boost* [Boo]. La méthode d'accélération dans ce cas repose sur l'encodage des matrices et des vecteurs : l'encodage permet de chercher directement les informations sur les éléments non nuls de la matrice ou du vecteur, et d'effectuer les multiplications où les opérandes sont toutes les deux non nulles. L'autre avantage des encodages pour les matrices et vecteurs creux est qu'ils permettent de réduire grandement la quantité de mémoire nécessaire pour les stocker.

Un exemple d'encodage pour les vecteurs creux est de créer deux vecteurs : un qui

$$
\begin{aligned}
\mathbf{v} &= \begin{bmatrix} 0 & 0 & 1 & -10 & 0 & 0 & 3 & 0 & 0 & 9 \end{bmatrix} \\
\mathbf{v}_v &= \begin{bmatrix} 1 & -10 & 3 & 9 \end{bmatrix} \\
\mathbf{v}_p &= \begin{bmatrix} 2 & 3 & 6 & 9 \end{bmatrix}
\end{aligned}
$$

FIGURE 7.1 – Un exemple d'encodage d'un vecteur $\mathbf{v}$ en gardant seulement les valeurs non nulles ainsi que leurs positions respectivement dans $\mathbf{v}_v$ et $\mathbf{v}_p$.

$$
\begin{aligned}
\mathbf{M} &= \begin{pmatrix} 0 & 0 & 8 & 0 & 1 \\ 7 & 0 & 0 & 9 & 0 \\ 0 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 5 & 0 \end{pmatrix} \\
\mathbf{M}_v &= \begin{bmatrix} 8 & 1 & 7 & 9 & 2 & 1 & 4 & 5 \end{bmatrix} \\
\mathbf{M}_p &= \begin{bmatrix} 2 & 4 & 0 & 3 & 1 & 2 & 0 & 3 \end{bmatrix} \\
\mathbf{M}_l &= \begin{bmatrix} 0 & 2 & 4 & \underbrace{6}_{\text{4ème l. vide}} & 6 & 8 \end{bmatrix}
\end{aligned}
$$

FIGURE 7.2 – Un exemple d'encodage d'une matrice $\mathbf{M}$ en gardant, pour chaque ligne, les valeurs non nulles, leurs positions (dans la ligne) et les indices, dans les vecteurs compressés, de début (et de fin) de chaque ligne respectivement dans $\mathbf{M}_v$, $\mathbf{M}_p$ et $\mathbf{M}_l$.

contient les positions des éléments non nuls et l'autre, les valeurs non nulles. Un exemple est donné à la figure 7.1.

Pour les matrices creuses, un encodage semblable peut être utilisé. On crée alors trois vecteurs : un pour les indices de début et de fin des éléments non nuls de chaque ligne, un pour les positions des éléments non nuls dans chaque ligne, et un pour les valeurs des éléments non nuls. Un exemple est donné à la figure 7.2.

Data : les vecteurs de positions $\mathbf{P}_1$ et $\mathbf{P}_2$;
les vecteurs de valeurs $\mathbf{V}_1$ et $\mathbf{V}_2$;
$i, j = 0$; $R = 0$
Result : le produit scalaire des deux vecteurs encodés avec les vecteurs de positions et de valeurs

foreach $p \in \mathbf{P}_1$ **do**
  while $p > \mathbf{P}_{2,i}$ **do** //On saute tous les éléments non nuls du deuxième
  //vecteur qui sont avant l'élément courant du premier
    $i = i + 1$;
  end
  $v = \mathbf{V}_{1,j}$;
  if $p = \mathbf{P}_{2,i}$ **then** //On fait la multiplication des éléments des deux
  //vecteurs qui sont à la même position
    $R = R + (v \cdot \mathbf{V}_{2,i})$;
  end
  $j = j + 1$;
end
return R;

Algorithme 1 : Algorithme pour calculer le produit scalaire de deux vecteurs creux, où un vecteur est encodé par l'ensemble des positions des valeurs non nulles et l'ensemble des valeurs non nulles.

En utilisant ces encodages et en supposant que les matrices et les vecteurs sont très creux, on accélère déjà grandement les produits. L'algorithme 1 fait le produit scalaire entre deux vecteurs creux et l'algorithme 2 permet de calculer le produit entre une matrice creuse et un vecteur plein.

Quoique les algorithmes 1 et 2 sont déjà une amélioration considérable par rapport à faire le produit standard (avec toutes les valeurs nulles), il reste qu'on calcule une quantité gigantesque de produits dans nos systèmes et qu'il est encore trop coûteux de

Data : La matrice $\mathbf{M}$ représentée par les vecteurs $\mathbf{M}_r$, $\mathbf{M}_p$ et $\mathbf{M}_v$ qui sont respectivement les vecteurs d'indices des lignes, des positions et des valeurs non nulles ;
Le vecteur $\mathbf{V}$; Le vecteur résultant $\mathbf{R}$ initialisé à zéro ;
$l = 0$

Result : Le vecteur issu du produit de $\mathbf{M} \cdot \mathbf{V}$

for $i = 0; i < ||\mathbf{M}_r|| - 1; i = i + 1$ **do**
  $p_1 = \mathbf{M}_{r,i}$;
  $p_2 = \mathbf{M}_{r,i+1}$;
  for $j = p_1; j < p_2; j = j + 1$ **do**
    $\mathbf{R}_l = \mathbf{R}_l + (\mathbf{M}_v \cdot \mathbf{V}_{\mathbf{M}_{p,j}})$;
  end
end
return R;

Algorithme 2 : Algorithme pour calculer le produit entre une matrice creuse et un vecteur plein. La matrice est encodée par : un vecteur indiquant les indices de début et de fin de chaque ligne dans le vecteur de positions et de valeurs ; le vecteur de positions des valeurs non nulles dans chaque ligne ; le vecteur des valeurs.

les effectuer.

On remarque immédiatement, dans le cas de la multiplication entre une matrice et un vecteur, que le calcul de chaque valeur du vecteur résultant est indépendant par rapport aux lignes de la matrice. La clé afin d'obtenir une accélération beaucoup plus substantielle est de calculer en parallèle chacune de ces valeurs. Pour ce faire, on aurait pu utiliser tout simplement le fait que les processeurs actuels ont généralement plusieurs coeurs. Mais pour l'instant, ces processeurs ont au maximum quatre coeurs. Au lieu d'utiliser le processeur, nous utilisons la carte graphique, qui, de nos jours, peut être considérée comme une machine très puissante permettant de traiter en parallèle un flux de données, ce qui est exactement ce qu'on essaie d'accomplir. De plus, l'évolution de la puissance de calcul des cartes graphiques est beaucoup plus rapide que celle des processeurs grâce à la pression issue de l'industrie du jeu vidéo (figure 7.3).

On utilise *CUDA*, qui est un langage de programmation, un pilote et un ensemble de librairies développés par *NVIDIA*, et qui permet d'exécuter du code arbitraire sur la carte vidéo pour traiter n'importe quelles données.

Ici, on n'entre pas dans les détails du langage : la documentation de *NVIDIA* est très

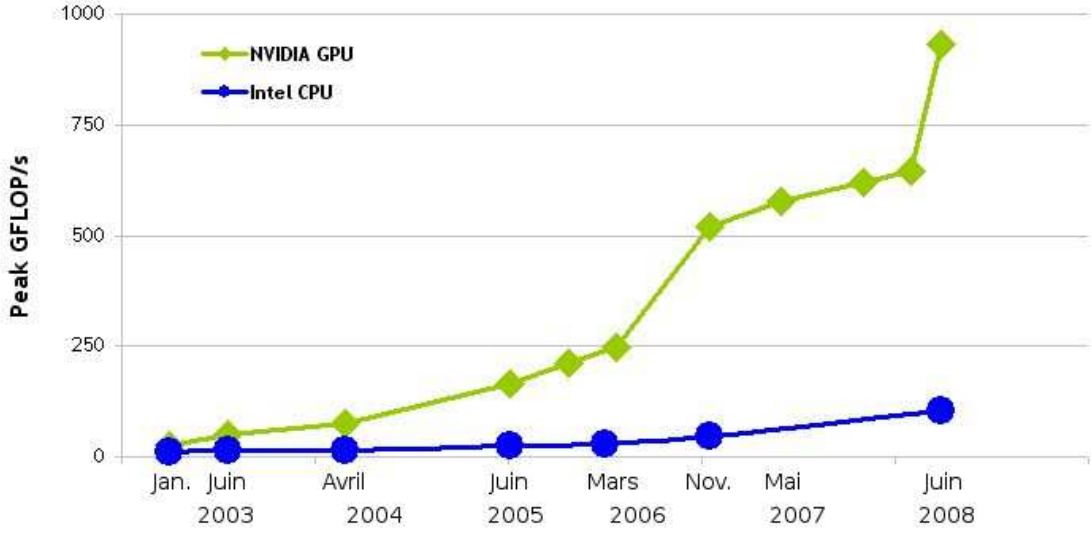


FIGURE 7.3 – L'évolution de la puissance de calcul (en terme de giga-flops) entre les GPUs de *NVIDIA* et les CPUs de *Intel*. Source [NVIc].

complète.[0] On décrit seulement le modèle d'exécution de *CUDA* pour ensuite expliquer comment on l'applique pour notre problème. Le code est présenté à l'annexe A.3.

Premièrement, comme mentionné plus haut, les processeurs graphiques sont hautement parallèles, ce qui veut dire que plusieurs fils d'exécution (*threads*) sont exécutés en même temps. Afin d'être capable d'assigner les différentes parties du problème à résoudre aux fils d'exécution, *CUDA* utilise un modèle hiérarchique d'assignations. D'abord une grille (de une à trois dimensions, la grandeur est spécifiée par l'utilisateur) est créée et chaque élément contient un bloc. Les éléments de chaque bloc (qui est également de une à trois dimensions, de grandeur variable) contient un fil d'exécution. Ici on parle de fils d'exécution virtuels : en réalité, le processeur graphique contient un nombre limité de coeurs (p. ex. la famille des processeurs graphiques G80 ont 240 coeurs). Ces différents coeurs seront assignés aux fils d'exécution. La figure 7.4 illustre le système de grille et de blocs en deux dimensions. Chaque fil d'exécution a donc un identifiant unique, qui peut être utilisé pour déterminer quelle partie du problème parallèle il doit résoudre.

Un exemple simple pour illustrer est l'addition de deux vecteurs de longueur N ($\mathbf{a} + \mathbf{b} = \mathbf{c}$). Si on utilise une grille unitaire (un seul élément) qui contient un bloc unidimensionnel de grandeur N, alors le fil d'exécution i a la charge d'effectuer l'addition $\mathbf{a}_i + \mathbf{b}_i = \mathbf{c}_i$ (voir figure 7.5).

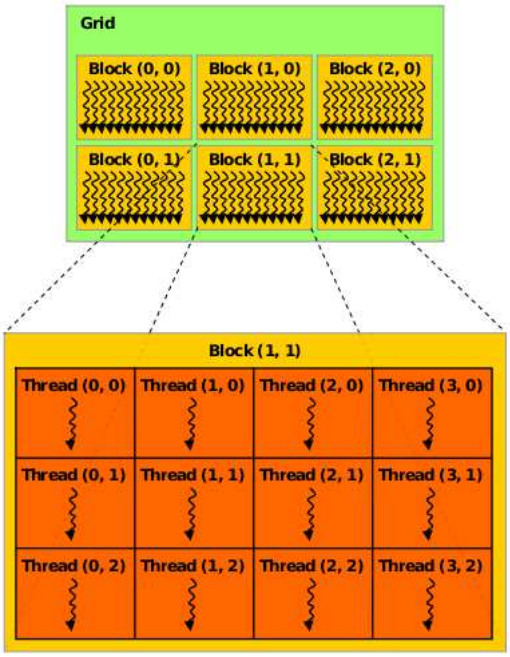


FIGURE 7.4 – Illustration de la méthode hiérarchique d'assignation des fils d'exécution afin de paralléliser un calcul. Source [NVIc].

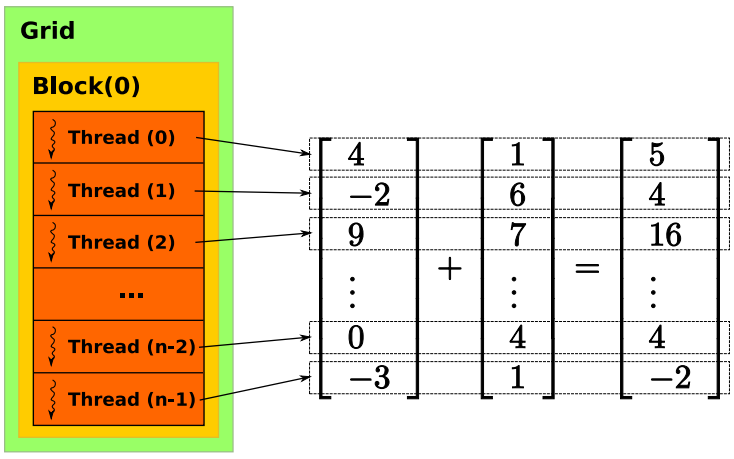


FIGURE 7.5 – Illustration d'une façon d'effectuer l'addition de deux vecteurs en parallèle, où chaque fil d'exécution additionne les éléments correspondants d'une ligne.

$$
\begin{aligned}
\Upsilon &= \left\{ \begin{bmatrix} 3 \\ 0 \\ 0 \\ 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ -1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 3 \\ 0 \\ 0 \\ 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 8 \\ 0 \\ 2 \\ 4 \\ 0 \end{bmatrix} \right\} \\
M &= \begin{bmatrix} 0 & 1 & 2 & 0 & 3 \end{bmatrix} \\
L &= \begin{bmatrix} 0 & 2 & 4 & 4 & 7 \end{bmatrix} \\
P &= \begin{bmatrix} 0 & 3 & 2 & 4 & 0 & 2 & 3 \end{bmatrix} \\
V &= \begin{bmatrix} 3 & 2 & -1 & 1 & 8 & 2 & 4 \end{bmatrix}
\end{aligned}
$$

FIGURE 7.6 – Représentation compacte d'un ensemble Υ de vecteurs creux, où M est le vecteur d'indirection, qui permet d'éliminer la duplication des vecteurs identiques dans l'ensemble, L est l'indice de début (et de fin) des données compressées de chaque vecteur, P est la position des valeurs non nulles dans chaque vecteur et V contient les valeurs des éléments non nuls.

On rappelle que dans nos systèmes, les calculs substentiels sont la multiplication des matrices de rotation des coefficients avec des vecteurs pleins lorsque la carte d'environnement change (section 5.2.1) et le produit scalaire entre des vecteurs creux pour effectuer l'intégrale entre la BRDF et la carte d'environnement locale (et la carte de visibilité pour le système avec carte d'environnement) à chaque sommet.

7.3.1 Produit scalaire

Le problème avec les produits scalaires est qu'ils ne sont pas aussi facilement parallélisables que l'exemple de l'addition simple de vecteurs : la somme finale des éléments, si elle est faite en parallèle, pourrait causer des erreurs dues au manque de synchronisation entre les différents fils d'exécution. Nous exploiterons donc le parallélisme d'une autre façon : nous regroupons et envoyons sur la carte vidéo tous les calculs de produits scalaires à faire. Après, on peut calculer en parallèle chaque produit scalaire. Pour pousser encore plus le parallélisme, nous effectuons la multiplication des éléments de chaque vecteur par un groupe de fils d'exécution. Nous présentons les grandes lignes de l'algorithme.

Données. Les vecteurs sont exprimés selon la représentation illustrée à la figure 7.1. Pour compacter plusieurs vecteurs, nous utilisons un tableau qui décrit la position du premier et dernier (qui est le premier du vecteur suivant) élément dans les données compressées. Un exemple est donné à la figure 7.6.

Les données nécessaires pour faire le produit scalaire entre les vecteurs correspondants de deux ensembles sont :

1. V_1, V_2 : Les valeurs compactées non nulles des vecteurs pour chaque ensemble.
2. P_1, P_2 : Les indices des valeurs non nulles correspondantes dans chaque vecteur, pour chaque ensemble.
3. L_1, L_2 : Les positions où commence et termine chaque vecteur dans les données V_i et I_i, pour chaque ensemble.
4. M_1, M_2 : Étant donné qu'un vecteur peut revenir plusieurs fois dans un ensemble parmi les calculs de produits scalaires, on compresse davantage les données en ne conservant qu'une seule copie de celui-ci. M_i donne la liste d'indices des vecteurs de chaque ensemble.

Parallélisation. D'abord nous associons un élément de la grille de blocs de fils d'exécution pour chaque produit scalaire : elle est donc unidimensionnelle et de la même longueur que le nombre de produits scalaires à faire. L'indice de l'élément de la liste donne l'indice du couple de vecteurs à multiplier que le bloc de fils d'exécution doit accomplir. Cette assignation constitue la première exploitation du parallélisme. Chaque élément de la liste contient un bloc avec une quantité fixe de fils d'exécution, qui vont effectuer un produit scalaire. Dans notre implémentation, on utilise 32 fils.

Calcul d'un produit scalaire. Soient **a** et **b** les deux vecteurs dont l'on souhaite le produit scalaire. **a** est situé à l'indice $\hat{M_1} \in M_1$ dans L_1. $L_1[\hat{M_1}]$ et $L_1[\hat{M_1}+1]$ donnent les indices de début et de fin des données compressées dans P_1 et V_1. De façon similaire, on peut retrouver les données de **b** dans les tableaux M_2, L_2, P_2 et V_2. Soit $\hat{L}_1^0 = L_1[\hat{M_1}], \hat{L}_1^1 = L_1[\hat{M_1}+1], \hat{L}_2^0 = L_2[\hat{M_2}]$ et $\hat{L}_2^1 = L_2[\hat{M_2}+1]$. Soit F_b le fils d'exécution à l'indice b dans le bloc.

Le rôle de F_b est d'effectuer la multiplication des éléments $V_1[\hat{L}_1^0 + b + Q_i]$ aux positions $P_1[\hat{L}_1^0 + b + Q_i]$ dans **a** avec les éléments correspondants dans **b** ; où Q est le

nombre de fils d'exécution dans un bloc, $\forall i \in \mathbb{N}$ tel que $\hat{L}_1^0 + b + Q_i < \hat{L}_1^1 - \hat{L}_1^0$. En d'autres mots, F_b fait la multiplication des données en commençant par la position b dans les données compressées de **a** en incrémentant par le nombre de fils d'exécution dans le bloc.

Il faut faire la multiplication de la donnée $V_1[\hat{L}_1^0 + b + Q_i]$ de **a** avec la donnée correspondante à la position $P_1[\hat{L}_1^0 + b + Q_i]$ dans **b**. On doit alors chercher dans P_2 si **b** a une valeur non nulle à cette position : on doit obligatoirement faire une recherche linéaire entre $P_2[\hat{L}_2^0]$ et $P_2[\hat{L}_2^1]$.

Une fois la multiplication des données correspondantes entre les deux vecteurs faites, on utilise le fil d'exécution F_0 pour faire la somme de ces multiplications pour obtenir le produit scalaire. On utilise un seul fil pour ne pas avoir de problèmes de synchronisation.

Le code complet pour la multiplication entre deux ensembles de vecteurs creux est disponible à l'annexe A.3.1.

7.3.2 Produit matrice-vecteur

Pour les multiplications matrice-vecteur creux, il y a deux façons d'exploiter le parallélisme tel que défini dans *CUDA* : premièrement en effectuant en parallèle plusieurs multiplications entre une matrice et un vecteur, mais aussi en calculant en parallèle le produit scalaire entre une ligne de chaque matrice et le vecteur.

Données. Les matrices sont exprimées selon la représentation illustrée à la figure 7.2. Pour compacter plusieurs matrices, on utilise un tableau qui décrit la position du premier et dernier (qui est le premier de la matrice suivante) éléments dans les données compressées. Un exemple est donné à la figure 7.7.

Les données nécessaires pour faire le produit scalaire entre les vecteurs correspondants de deux ensembles sont :

1. V : Les valeurs compactées non nulles des matrices.
2. P : Les positions (dans les lignes) des valeurs non nulles de chaque matrice.
3. L_l : Les positions où commence et termine chaque ligne de chaque matrice dans les données compressées (relatives au début des données de la matrice).
4. M : Les positions du début (et la fin) de chaque matrice dans les données compressées.

$$
\begin{aligned}
\Upsilon &= \left\{ \begin{bmatrix} 0 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 3 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 2 \\ 6 & 2 & 3 \end{bmatrix}, \begin{bmatrix} 0 & 2 & 0 \\ 0 & 9 & 0 \\ 4 & 0 & 0 \end{bmatrix} \right\} \\
M &= \begin{bmatrix} 0 & 4 & 8 & 11 \end{bmatrix} \\
L_l &= \begin{bmatrix} 0 & 1 & 2 & 4 & 0 & 0 & 1 & 4 & 0 & 1 & 2 & 3 \end{bmatrix} \\
P &= \begin{bmatrix} 2 & 1 & 0 & 2 & 2 & 0 & 1 & 2 & 1 & 1 & 0 \end{bmatrix} \\
V &= \begin{bmatrix} 1 & 2 & 1 & 3 & 2 & 6 & 2 & 3 & 2 & 9 & 4 \end{bmatrix}
\end{aligned}
$$

FIGURE 7.7 – Représentation compacte d'un ensemble Υ de matrices creuses, où M est le vecteur des positions du début (et de la fin) de chaque matrice dans les données compressées, L_l est le vecteur des positions où commence et termine chaque ligne de chaque matrice, P est le vecteur des positions des valeurs non nulles dans chaque ligne et V contient les valeurs des éléments non nuls.

5. X : Les valeurs des vecteurs.

Parallélisation. D'abord nous créons une grille de blocs de fils d'exécution bidimensionnelle : la première composante (x) de la grille représente la multiplication entre une matrice et un vecteur. Pour chacune de ces multiplications, la deuxième composante (y) est la multiplication entre une ligne et le vecteur. Donc, si on a n_m matrices ayant chacune n_l lignes, alors la grille sera de dimension $n_m \times n_l$. On utilise, comme pour le produit scalaire de deux vecteurs, plusieurs fils d'exécution pour effectuer le produit scalaire entre une ligne d'une matrice et un vecteur. Donc chaque bloc de la grille aura une grandeur fixe. Comme pour le produit scalaire de vecteurs, on utilise 32 fils.

Calcul d'une multiplication matrice-vecteur. Étant donné qu'on divise le problème de la multiplication de plusieurs matrices et vecteurs en plusieurs problèmes de multiplication entre deux vecteurs (une ligne de la matrice et le vecteur), l'algorithme est le même que pour le produit scalaire (section 7.3.1). Les seules différences sont :

1. La façon de trouver les valeurs de la ligne à multiplier dans les données compactées. D'abord la composante x du fils d'exécution donne l'indice de la matrice et du vecteur qu'il doit multiplier ensemble. La composante y donne la ligne de cette matrice qu'il doit multiplier avec le même vecteur. Avec ces informations, on a

que $M[x]$ et $M[x+1]$ donnent les indices de début et fin des informations sur les débuts et fins des lignes de la x-ième matrice dans L_l. Soit $L_l^M = L_l[M[x]]$. On peut alors trouver les indices de début et fin sur les informations de la ligne y de la matrice x dans P et V, qui sont $P[L_l^M[y]], P[L_l^M[y+1]], V[L_l^M[y]]$ et $V[L_l^M[y+1]]$.

2. Les vecteurs multipliés sont pleins, donc pas besoin de faire la recherche linéaire pour trouver les valeurs correspondantes dans les vecteurs lors d'une multiplication avec une ligne.

Le code complet pour la multiplication entre deux ensembles de vecteurs creux est disponible à l'annexe A.3.2.

Chapitre 8

Résultats

Nous présentons ici quelques résultats et exemples d'applications pour les systèmes présentés aux chapitres 5 et 6. Nous discutons aussi des statistiques ainsi obtenues pour mieux dégager les forces et faiblesses de chacun de nos deux systèmes.

L'équipement utilisé pour les résultats est un processeur Core 2 Duo 6600 (2.4 GHz) avec 2 Go de mémoire vive et une carte vidéo GeForce 8800 GTX avec 768 Mo de mémoire. Le système d'exploitation utilisé est Linux 2.6.29.4 en 64 bits, avec les pilotes *NVIDIA* 180.29. On utilise également *CUDA* 2.2.

Toutes les scènes ont été modélisées dans *Blender* [Ble] et exportées dans un format propre au projet. Le pinceau utilisé dans tous les exemples est un cercle de 11 pixels de diamètre. Pour le lancer de photons, on utilise un noyau de 50 photons, ce qui est suffisant pour toutes les scènes. Il faudrait ajuster ce paramètre dans le cas où une scène contiendrait beaucoup plus de sources de lumière.

On rappelle qu'une version couleur électronique est disponible au `http://www.iro.umontreal.ca/labs/infographie/theses/rozonfre/`.

8.1 Système avec carte d'environnement

Dans cette section, nous présentons les temps de calculs, la quantité de mémoire requise ainsi que quelques résultats pour le système avec carte d'environnement.

Résolution (N)	Temps calculs	Taille	% non nuls
16×16	10 s.	11.8 Mo	18%
32×32	1.2 m. (72 s.)	58.4 Mo	5.6%
64×64	30 m. (160 s.)	266 Mo	1.5%
128×128	3 h. (10800 s.)	1.1 Go (1126.4 Mo)	0.38%

FIGURE 8.1 – Les statistiques sur différentes résolutions pour la carte d'environnement globale (la résolution pour la carte d'environnement locale est toujours $N/4$).

8.1.1 Précalculs

Les plus gros calculs pour ce système est la création des matrices de transformation des ondelettes et la conversion des BRDFs.

Pour les matrices de rotation, nous avons choisi une résolution de 64×64 pour la carte d'environnement globale et une résolution de 32×32 pour les cartes d'environnement locales. Ce choix est un compromis entre la qualité et la vitesse de rendu, et la quantité de mémoire nécessaire. Nous avons échantillonné également 32×32 normales sur la sphère. Donc, au total, on a 1024 matrices de transformation de taille 4096×1024. Puisqu'il s'agit d'un très grand nombre de données, on quantifie les valeurs (*float*) des matrices, qui sont dans le domaine $[-1, 1]$, vers des entiers courts, afin de réduire la quantité mémoire de moitié. On présente à la figure 8.1 les temps de calculs, la quantité de mémoire et le pourcentage des éléments non nuls pour différentes résolutions de matrices (le nombre de normales reste le même, soit 32×32). On rappelle qu'on ne doit calculer les matrices qu'une seule fois ; par la suite, ces matrices peuvent être utilisées pour toutes les scènes.

Pour les BRDFs, la résolution des tranches doit être la même que la résolution des cartes d'environnement locales (dans notre cas 32×32), pour être capable de faire le produit scalaire entre les deux. Le temps nécessaire pour transformer une BRDF est d'environ une heure. Chaque BRDF prend environ 13 Mo de mémoire. Chaque BRDF ne doit être convertie qu'une seule fois, après quoi elle peut être appliquée sur plusieurs objets.

Le dernier précalcul est la carte de visibilité de chaque objet. Le temps de calcul est évidemment dépendant du nombre de sommets de l'objet. Par exemple, la tête de singe *Suzanne* utilisée dans la première scène est constituée de 10640 sommets, requiert environ 2 minutes et occupe environ 10 Mo de mémoire.

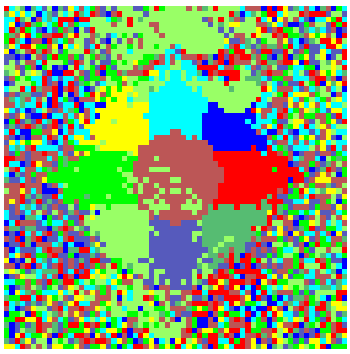


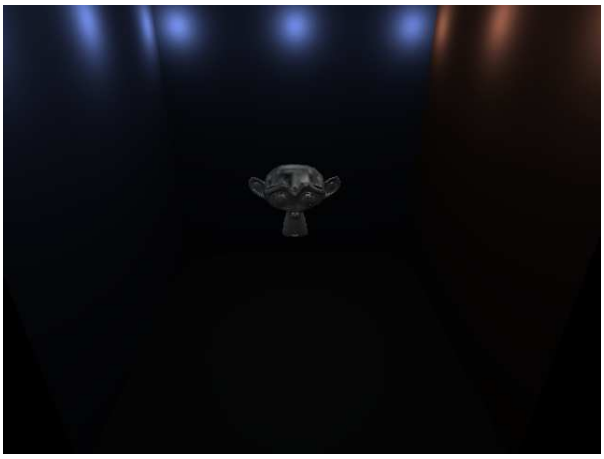
FIGURE 8.2 – La carte d'importance des sources de lumière : chaque pixel correspond à la source (identifiée par la couleur) la plus importante dans une certaine direction autour de l'objet. La projection utilisée est illustrée à la figure 5.3 (en bas). Cette image sert seulement pour la visualisation : en réalité, le système conserve l'importance de chaque source pour chaque direction.

8.1.2 Scène simple

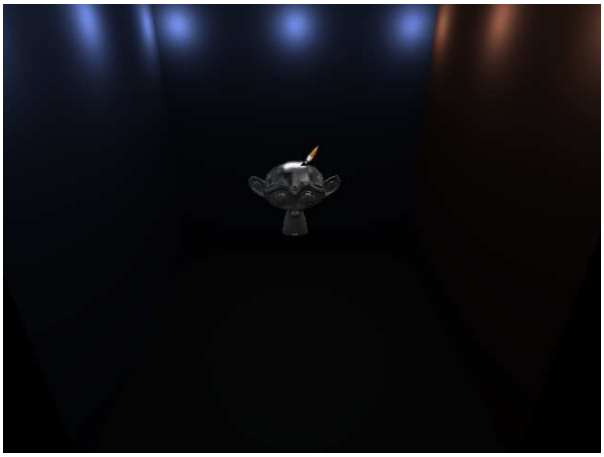
La première scène que nous avons utilisée est très simple mais démontre quelques-unes des fonctionnalités importantes des systèmes. Il s'agit d'une pièce cubique avec neuf points de lumière répartis également au plafond, avec un diffuseur placé sous celles-ci (couvrant tout le plafond). Les murs ont une seule BRDF de peinture mais de différentes couleurs, et le plancher est diffus et gris. Le modèle utilisé pour ces BRDFs est celui de Lafortune et al. [LFTG97], qui est une extension du modèle de Blinn-Phong permettant d'utiliser plusieurs lobes spéculaires. N'importe quel autre modèle aurait pu être utilisé. L'objet d'intérêt dans la scène est la tête de singe appelée Suzanne provenant de *Blender*. Le modèle comporte 10640 sommets. Le matériel utilisé pour la tête est un métal brossé, qui est également modélisé par le modèle de Lafortune et al.

Cette scène comporte quelques phénomènes lumineux intéressants comme l'interréflexion diffuse de la lumière (pour les murs et le plancher), et la transmission de lumière au travers d'un matériau (pour le diffuseur).

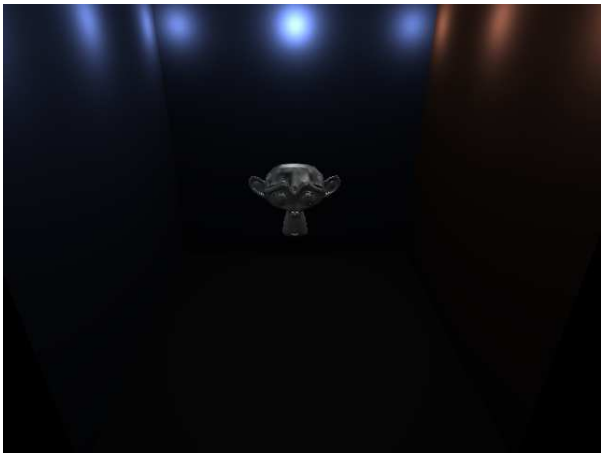
La figure 8.3 montre quelques captures d'écran de la visualisation et de la manipulation de la scène avec le système utilisant les cartes d'environnement. Le temps nécessaire pour trouver et modifier les sources de lumière est très court : habituellement moins d'une seconde. On remarque que les éléments non modifiables de la scène sont rendus avec de l'illumination directe seulement en utilisant le modèle *distribution-based BRDF*



(a)



(b)



(c)

FIGURE 8.3 – Exemple d'utilisation du système avec carte d'environnement : (a) La scène sans modification. (b) On applique un coup de pinceau afin d'augmenter l'éclairage sur le dessus de la tête. (c) Le résultat de la modification : le système a augmenté l'intensité de la source de lumière au fond de la salle.

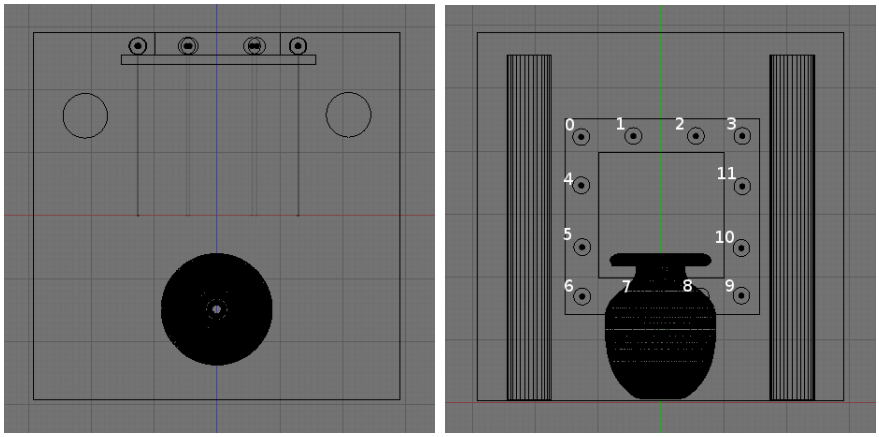
introduit par Ashikhmin et Premoze [AP].

Le lancer de photons nécessaire lors du départ du système est de 23 secondes pour 900,000 photons (100,000 par lumière). La sélection de l'objet prend environ 5.43 secondes, ce qui comprend le calcul de la carte d'importance des sources pour chaque direction (4.63 s.), la création de l'environnement pour la visualisation de l'objet (0.01 s.) et le calcul des environnements locaux (0.79 s.). Une image de la visualisation d'une carte d'importance est montrée à la figure 8.2. On y voit l'importance directe des neuf sources de lumière, mais aussi celle plus indirecte des murs. On remarque que la carte est très bruitée sur les côtés. Ceci correspond au dessous de l'objet, où le plancher diffus rend la sélection d'une source de lumière très difficile. La vitesse de rendu de la scène (avec l'objet modifiable sélectionné) est entre 30 et 45 images par seconde.

8.1.3 Scène plus complexe

La deuxième scène que nous avons utilisée est une autre pièce, où 12 sources de lumière ponctuelles sont cachées derrière un bloqueur. Deux colonnes de verres créent des caustiques dans la scène. Les murs sont d'une peinture bleue diffuse. L'objet modifiable est un vase ancien en argile un peu luisante. Toutes les BRDFs sont exprimées dans le modèle de Lafortune et al. Le vase comporte 105,738 sommets. Le schéma de la scène est présenté à la figure 8.4.

Nous présentons à la figure 8.6 quelques modifications faites à l'éclairage de l'objet. Le premier coup de pinceau augmente l'intensité due à une des caustiques réfléchies par le vase. Le système trouve que la source 6 est responsable de la caustique, et augmente son intensité. Ce genre d'opération serait à peu près impossible dans les autres systèmes de rendu inverse, car ils ne prennent pas en compte les multiples rebonds de la lumière. Le deuxième coup de pinceau assombrit légèrement la partie droite du vase. Encore une fois, le système trouve correctement que la source 9 est la plus importante. On présente à la figure 8.5 les histogrammes des importances de chaque source pour chaque coup de pinceau. Ici on a utilisé la stratégie de sélection "la plus importante" pour les deux coups de pinceau. On remarque que dans le cas du deuxième coup de pinceau, plusieurs sources ont une importance non négligible. Dans ce genre de situation, l'utilisation de la stratégie de l'histogramme aurait effectué le même effet en diminuant un peu plusieurs sources de lumière. Par contre, le risque de conflit avec la première modification aurait



(a) (b)

FIGURE 8.4 – Schéma de la scène plus complexe pour le système avec carte d'environnement. (a) Vue du dessus. (b) Vue de face, avec la numérotation des sources de lumière.

été plus grand.

Le lancer de photons nécessaire lors du départ du système est de 231 secondes pour 12M photons (1M par source de lumière). La sélection de l'objet prend 8.18 secondes, ce qui comprend le calcul de la carte d'importance des sources pour chaque direction (7.38 s.), la création de l'environnement pour la visualisation de l'objet (0.01 s.) et le calcul des environnements locaux (0.79 s.). La vitesse de rendu de la scène (avec l'objet modifiable sélectionné) est entre 15 et 25 images par seconde.

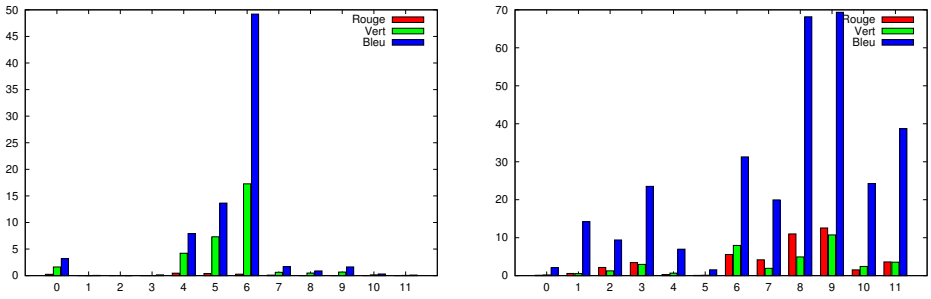
8.2 Système avec points

Dans cette section, nous présentons les temps de calculs, la quantité de mémoire requise ainsi que quelques résultats pour le système avec points.

8.2.1 Précalculs

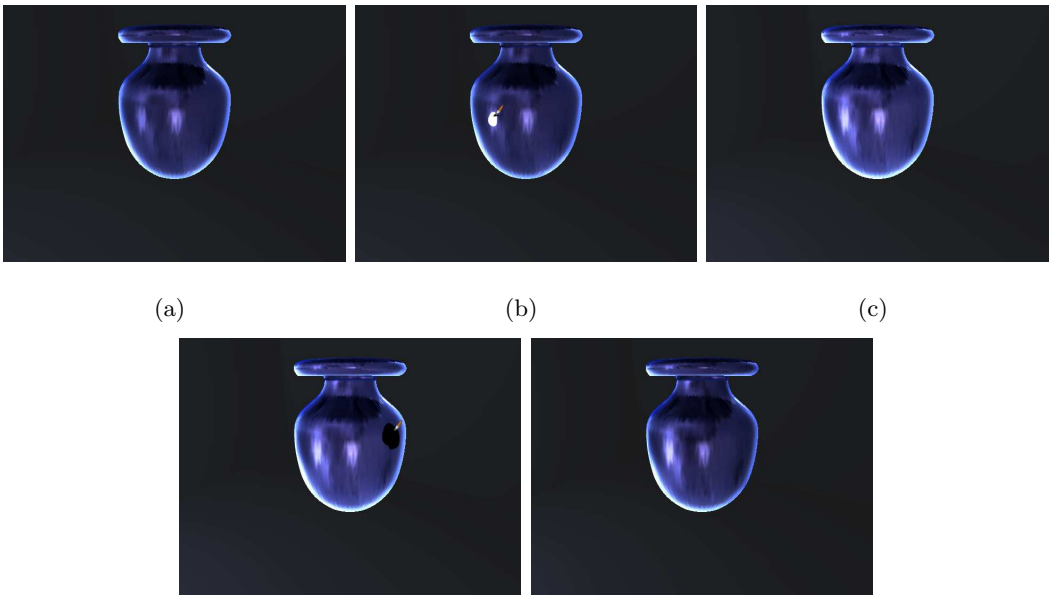
Le seul précalcul qui est réutilisable sur plusieurs scènes est la conversion des BRDFs. Il s'agit de la même conversion que pour le système avec carte d'environnement. Comme mentionné à la section 8.1.1, chaque BRDF prend environ une heure à convertir et occupe 13 Mo d'espace mémoire.

Pour chaque scène, on doit précalculer les importances des sources de lumière pour chaque sommet de chaque objet modifiable.



(a) (b)

FIGURE 8.5 – Distribution des importances de chaque source pour chaque couleur pour (a) l'augmentation de la caustique et (b) l'assombrissement du côté droit du vase pour le système avec carte d'environnement.



(a) (b) (c)

(d) (e)

FIGURE 8.6 – Exemple d'utilisation du système avec carte d'environnement. (a) Visualisation de l'objet dans l'état initial. (b) Un coup de pinceau est donné à la caustique de gauche pour l'accentuer et (c) le résultat est affiché. (d) Un coup de pinceau est donné pour diminuer l'éclairage dans la partie droite du vase et (e) le résultat est affiché.

8.2.2 Scène simple

Nous avons utilisé la même scène simple que pour le système avec carte d'environnement pour pouvoir faire une comparaison. La figure 8.7 montre quelques captures d'écran de la visualisation et de la manipulation de la scène. De la même façon qu'avec le système avec carte d'environnement, moins d'une seconde est nécessaire pour trouver et modifier les sources de lumière. Le modèle de Ashikhmin et Premoze [AP] est utilisé également pour rendre les objets non modifiables de la scène.

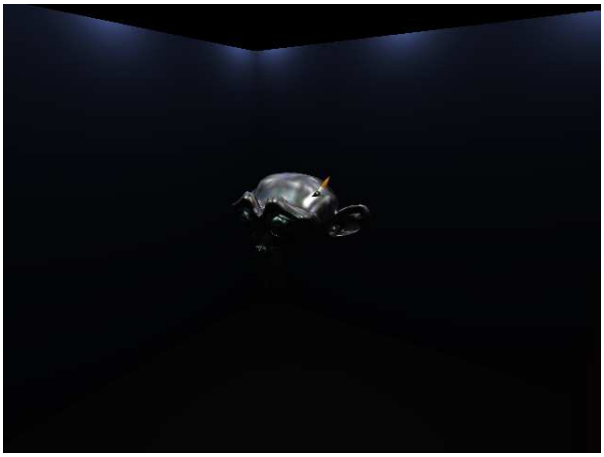
Le temps de précalcul de la scène est d'environ 20 minutes et prend 27 Mo d'espace mémoire. On rappelle que le précalcul est une forme de lancer de photons où la contribution de chaque source de lumière est calculée pour chaque direction de l'hémisphère autour de chaque sommet de chaque objet modifiable. La sélection d'un objet est ici très rapide (environ une seconde) étant donné que les cartes d'importance des sources sont précalculées et qu'on doit seulement les charger. La vitesse de rendu de la scène (avec l'objet modifiable sélectionné) est entre 30 et 45 images par seconde.

8.2.3 Caustiques

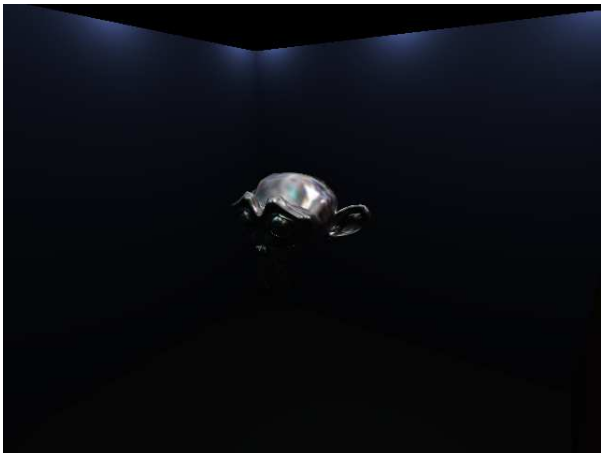
La scène avec les caustiques montre que le système avec points peut prendre en compte des effets lumineux complexes sur n'importe quel type de surface. Il s'agit d'une pièce (piscine) recouverte de feutre noir (pour empêcher que la lumière se reflète sur les murs), le fond de la piscine est recouverte d'une peinture bleue et la surface des vagues est de l'eau. Nous avons enlevé l'atténuation de la lumière dans l'eau pour mieux voir les caustiques. Toutes les BRDFs sont exprimées dans le modèle de Lafortune et al. Le fond de la piscine est le seul objet modifiable et est très finement subdivisé (173902 sommets) pour obtenir plus de détails. Le schéma de la scène est montré à la figure 8.8.

La figure 8.9 montre quelques captures d'écran de la visualisation et de la manipulation de la scène. Moins d'une seconde est nécessaire pour trouver et modifier les sources de lumière. Ici, il y a plusieurs choses à remarquer. Premièrement, la source de lumière choisie pour effectuer la modification n'est pas celle qu'intuitivement une personne choisirait : elle est un peu en retrait par rapport à la direction de la caméra et la zone peinturée. La distribution des importances des sources est montrée à la figure 8.10. Ceci démontre l'utilité du système, qui rend intuitif ce genre de manipulation. La seconde chose à remarquer est que la modification de la source a engendré beaucoup

(a)

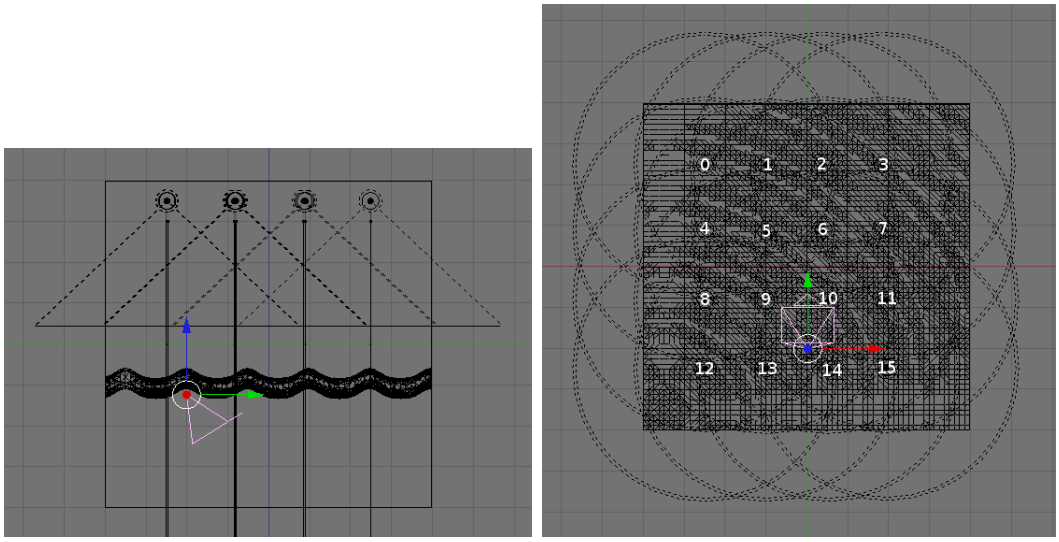


(b)



(c)

FIGURE 8.7 – Exemple d'utilisation du système avec points : (a) La scène sans modification. (b) On applique un coup de pinceau afin d'ajouter de la lumière sur le côté droit du crâne. (c) Le résultat de la modification : le système a augmenté l'intensité de la source de lumière à gauche de la salle.



(a) (b)

FIGURE 8.8 – Schéma de la scène de caustiques pour le système avec points. (a) Vue de côté. (b) Vue de dessus, avec la numérotation des sources de lumière.

(a) (b) (c)

FIGURE 8.9 – Exemple d'utilisation du système pour la scène de caustiques : (a) La scène sans modification. (b) On applique un coup de pinceau afin d'accentuer une caustique dans le fond de la piscine. (c) Le résultat de la modification : le système a augmenté l'intensité de la source de lumière numéro 7.

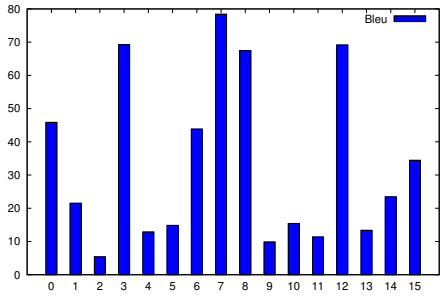


FIGURE 8.10 – Distribution des importances de chaque source dans le bleu pour l'augmentation de la caustique pour le systèmes avec points. On montre seulement les contributions pour le bleu puisque les autres canaux ne sont pas significatifs.

de changements dans la scène. Ceci est le désavantage de n'importe quel système de rendu inverse.

Le temps de précalcul de la scène est d'environ 2 heures et prend 114 Mo d'espace mémoire. Nous avons lancé 1M de photons par source (16M au total). La sélection d'un objet reste toujours très rapide (environ une seconde). La vitesse de rendu de la scène (avec l'objet modifiable sélectionné) est entre 5 et 15 images par seconde.

8.2.4 Scène diffuse

La scène diffuse est une pièce composée de colonnes à base carrée entourant le Buddha de Stanford. Les murs de la pièce et le Buddha sont mats et blancs, les colonnes sont en vert. Aucune des sources de lumière n'affecte directement le Buddha. Toutes les BRDFs sont exprimées dans le modèle de Lafortune et al. Le Buddha contient 543,652 sommets, donc très finement détaillé. Le schéma de la scène est montré à la figure 8.11.

La figure 8.12 montre quelques captures d'écran de la visualisation et de la manipulation de la scène. Encore une fois, moins d'une seconde est nécessaire pour trouver et modifier les sources de lumière. Dans cette scène, nous illustrons que le système peut gérer les chemins de lumière complexes et est capable de trouver les sources significatives même dans le cas où la lumière est réfléchie de manière diffuse. Dans le cas de la première modification, nous avons utilisé la stratégie "la plus importante" pour créer un *highlight* mieux défini. Dans le cas de la deuxième modification, nous avons utilisé la stratégie "histogramme" pour assombrir de façon plus générale le côté gauche du Buddha. La figure 8.13 montre la distribution de l'importance des sources de lumière pour

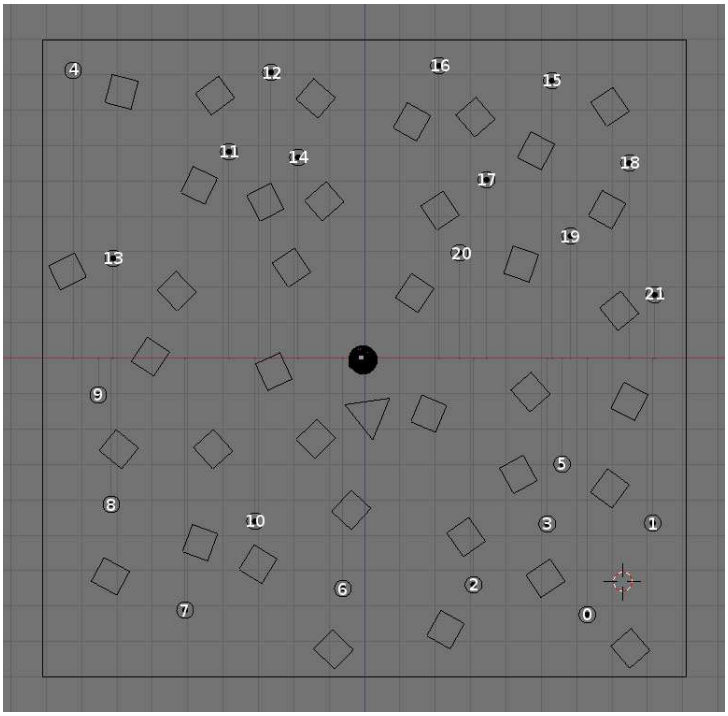


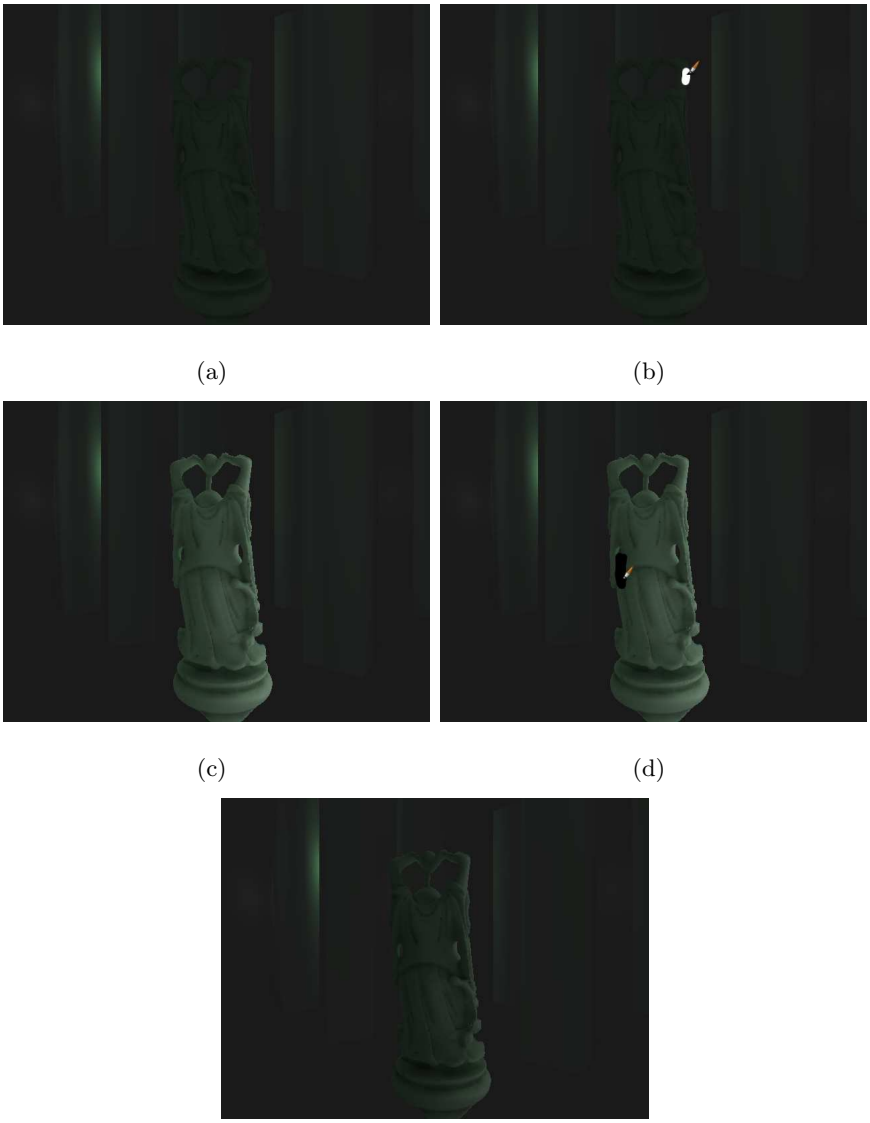
FIGURE 8.11 – Schéma de la scène diffuse pour le système avec points (vue de dessus) avec la numérotation des sources de lumière.

les deux opérations. On remarque que la distribution est quand même assez uniforme : il n'y a pas une source qui domine totalement les autres.

Le temps de précalcul de la scène est d'environ 3 heures et prend 266 Mo d'espace mémoire. Nous avons lancé 1M de photons par source (22M au total). La sélection d'un objet reste toujours très rapide (environ une seconde). La vitesse de rendu de la scène (avec l'objet modifiable sélectionné) est entre 5 et 13 images par seconde.

8.3 Outils

Dans cette section, nous présentons les différents résultats reliés à l'utilisation de différentes stratégies de sélection des sources de lumière ainsi que les différentes stratégies de manipulation de l'intensité des sources. Nous utilisons le système avec points pour produire les résultats. Nous aurions pu prendre n'importe lequel des deux systèmes et obtenir les même résultats (c'est-à-dire les mêmes différences entre les outils).

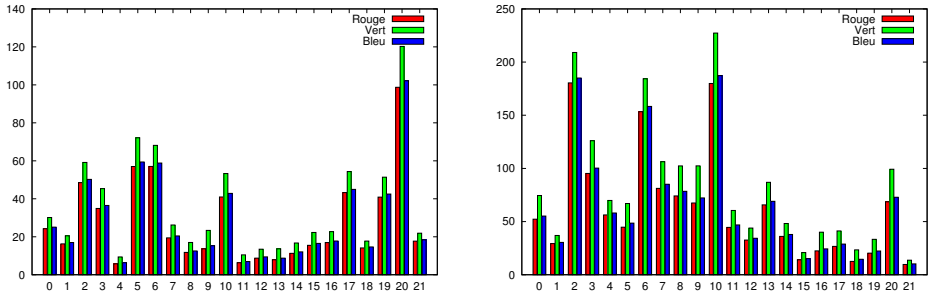


(a) (b)

(c) (d)

(e)

FIGURE 8.12 – Exemple d'utilisation du système avec points pour la scène diffuse. (a) La scène sans modification. (b) On applique un coup de pinceau sur le bras droit afin de créer un *highlight*. (c) Le résultat de l'opération : le système a augmenté l'intensité de la source numéro 20. (d) On applique un deuxième coup de pinceau pour assombrir le côté gauche du Buddha. (e) Le résultat de l'opération : ici nous avons choisi la stratégie de l'histogramme pour faire la sélection des sources, alors plusieurs sources ont été atténuées.



(a) (b)

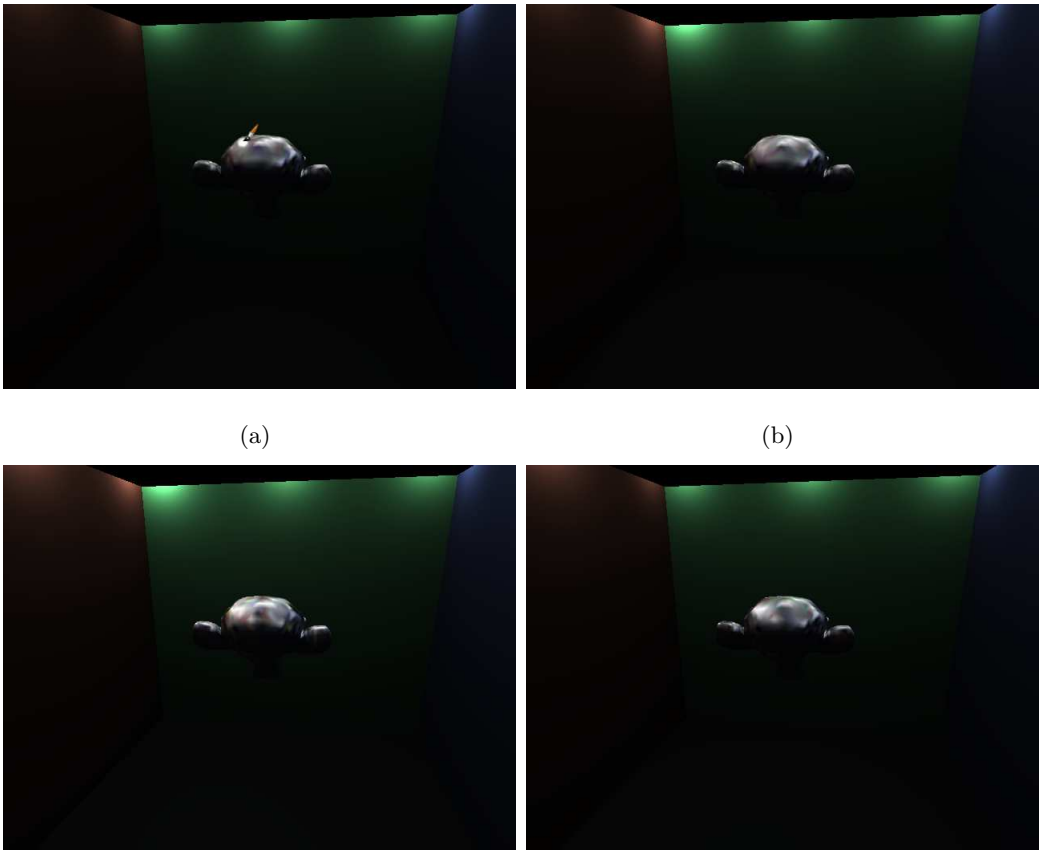
FIGURE 8.13 – Distribution des importances de chaque source pour les opérations effectuées à la figure 8.12 pour (a) l'augmentation de l'illumination sur le bras du Buddha et (b) l'assombrissement du côté gauche du Buddha.

8.3.1 Stratégies de sélection

Nous présentons à la figure 8.14 les différents comportements des stratégies de sélection. On remarque que pour une certaine zone peinturée, les trois stratégies donnent des résultats différents. Pour la stratégie "la plus importante", à la figure 8.14(b), seule la lumière qui contribue le plus à la zone est sélec/var/tmp/portage/gnome-base/gnome-keyring-2.22.3-r2/temp/build.logtionnée (source dans le coin en haut à gauche). Pour la stratégie de l'histogramme, à la figure 8.14(c), les sources les plus importantes qui contribuent à un certain pourcentage (50% pour l'exemple) de toute la lumière reçu par la zone sont sélectionnées. Dans l'exemple, les sources dans le coin en haut à gauche et à gauche ont été sélectionnées. Pour la stratégie de la roulette russe, à la figure 8.14(d), une source est choisie aléatoirement où la probabilité de chaque source est pondérée par l'importance pour la zone peinturée. Dans l'exemple, la deuxième source la plus importante a été choisie, c'est-à-dire la source à gauche.

8.3.2 Stratégies de modification

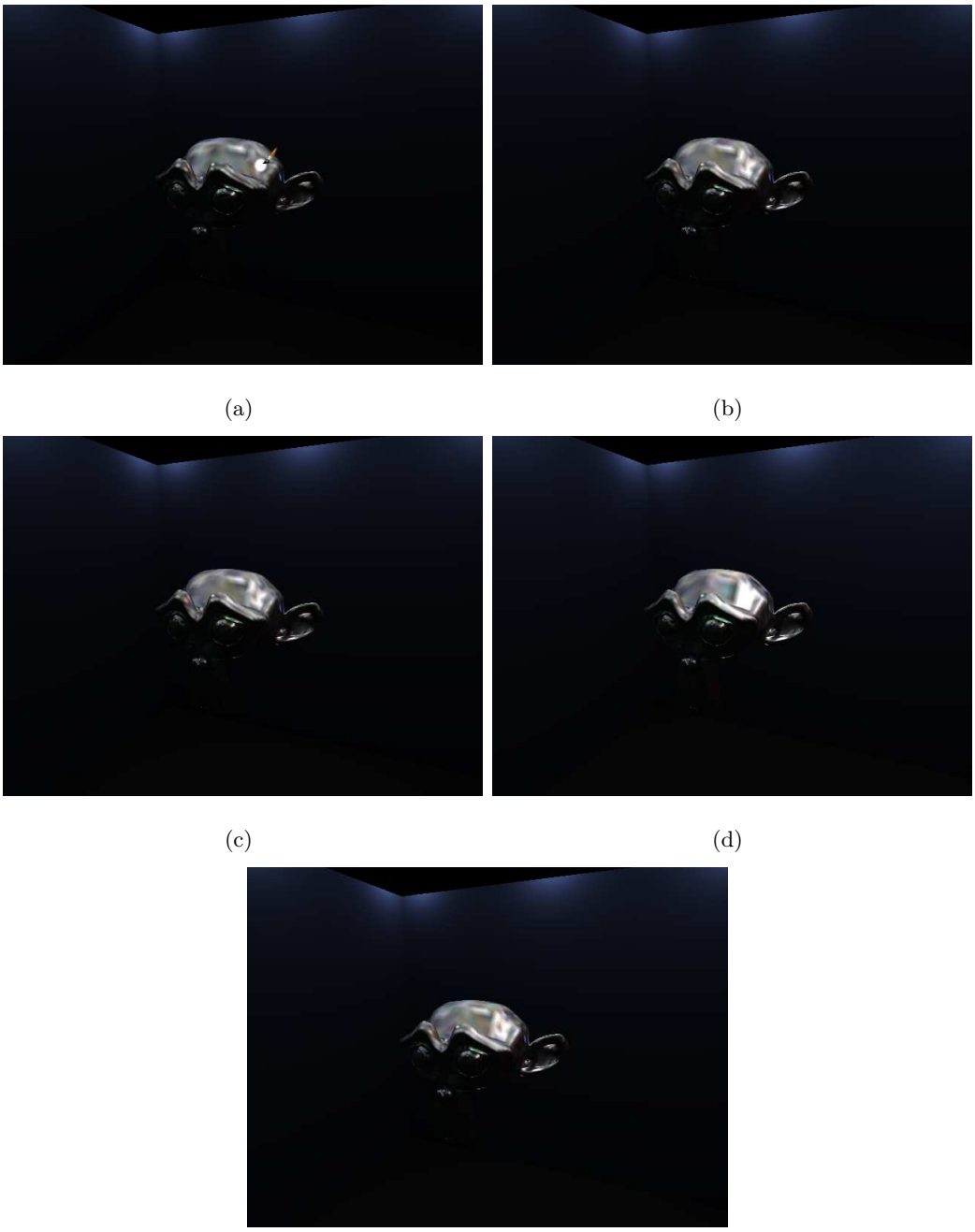
Nous présentons à la figure 8.15 les différents comportements des stratégies de modification de l'intensité des sources de lumière. Pour une certaine zone peinturée, les quatre stratégies donnent des résultats différents.



(a) (b)

(c) (d)

FIGURE 8.14 – Illustration des stratégies de sélection des sources de lumière après (a) la peinture d'une certaine zone sur le crâne du singe. (b) Sélection de la source la plus influente pour la zone. (c) Sélection des sources les plus influentes qui couvrent 50% de la lumière reçue par la zone. (d) Sélection aléatoire d'une source où la probabilité qu'une source soit choisie est directement proportionnelle de l'influence sur la zone.



(a) (b)

(c) (d)

(e)

FIGURE 8.15 – Illustration des stratégies de modification des sources de lumière après (a) la peinture d'une certaine zone sur le crâne du singe. (b) On augmente l'intensité de la source par une certaine quantité. (c) On double l'intensité de la source. (d) On résout le système pour que la zone soit de la même intensité que le pinceau. La couleur n'est pas prise en compte. (e) Stratégie identique à la précédente, mais on garde seulement une fraction de l'intensité du résultat.

(a) (b) (c)

FIGURE 8.16 – Comparaison du rendu de la même scène sous (a) le système avec cartes d'environnement, (b) le système avec points et (c) un rendu de référence par *PBRT*. On remarque que le système avec points est beaucoup plus près du rendu de référence que le système avec carte d'environnement. La principale raison dans ce cas-ci est le fait que le système avec carte d'environnement gère mal les concavités : ceci se voit particulièrement bien pour l'oreille.

8.4 Comparaison entre les deux systèmes

Étant donné que les deux systèmes ne font pas le rendu des objets modifiables de la même façon : ils produisent des résultats différents. La figure 8.16 montre le rendu de la tête de singe pour le système avec carte d'environnement, le système avec points et un rendu de référence par *PRBT*, en utilisant le tracer de chemins bidirectionel. Étant donné qu'on voulait comparer sommairement les images entre elles, le rendu de référence utilise relativement peu d'échantillons, ce qui explique pourquoi l'image est bruitée. On rappelle que le rendu des deux systèmes se fait par sommet, et donc le niveau de détails d'un objet est directement proportionnel à son nombre de sommets.

On remarque tout d'abord que l'image produite avec le système avec points est beaucoup plus près du rendu de référence que l'image produite par le système avec carte d'environnement. Ceci est causé par le fait que les concavités sont mal gérées avec le système par carte d'environnement, et donc on remarque que les différences sont plus marquées sur les endroits complexes (p. ex. l'oreille). D'autres différences sont dues à la discrétisation des normales. Pour un sommet donné, le système par carte d'environnement fait une interpolation bilinéaire entre les intensités capturées en direction des quatre normales les plus près de la normale du sommet, ce qui ne donne pas exactement la même intensité. Finalement, la distribution des photons étant aléatoire

(surtout dans une scène possédant des surfaces diffuses), il se peut qu'une zone n'ait pas été atteinte par des photons. Ceci explique le trou prononcé sur le crâne en haut à droite de l'image 8.16(a). Autrement, les deux systèmes convergent vers l'image de référence en apparence générale.

8.5 Discussion

Le système fonctionne comme attendu : pour les objets modifiables, un aperçu de son illumination directe et indirecte est disponible en temps interactif, les modifications de son illumination sont quasi instantanées et on peut se promener dans la scène librement. Lors d'un coup de pinceau le système trouve les sources de lumière qui affecteront cette zone, même les sources dans d'autres pièces ou derrière des obstacles.

Malgré que le système fonctionne, il y a deux limitations majeures inhérentes au problème de la sélection et modification de sources de lumière :

1. **Granularité de la solution.** Dans un contexte normal, une scène ne contient habituellement que quelques sources de lumière, par exemple un plafonnier, une lampe, etc. Un autre exemple est une scène extérieure, où la seule source est le soleil. Il est rare d'avoir une centaine, voire une dizaine de sources dans une scène. Donc, lorsque l'utilisateur peinture une zone d'un objet, il est fort probable qu'une grande partie de l'objet soit modifiée. Il est donc impossible, avec une petite quantité de sources, d'obtenir des résultats précis.
2. **Conflits.** Pour la même raison que le point précédent, le fait qu'une scène ne contienne qu'un petit nombre de sources, la modification d'une de ces sources va altérer considérablement le reste de la scène, pouvant causer des conflits avec d'autres zones déjà peinturées par l'utilisateur. Malgré que le système de contraintes fonctionne et peut aider à limiter ces conflits, il reste que si la zone peinte est affectée par une seule source, c'est celle-ci qu'on doit modifier pour obtenir les résultats, peu importe si la source est pénalisée ou pas.

Une façon de contourner le problème serait de subdiviser les sources de lumière surfaciques, ce qui augmente l'espace des solutions possibles. Cependant, ceci pourrait affecter le réalisme de la scène puisque des zones d'une lumière surfacique pourraient avoir des intensités très différentes. Pour les sources ponctuelles, la modification de leur

distribution angulaire d'émission de la lumière pourrait être une solution, mais ceci sort du contexte de ce mémoire et pourrait encore une fois affecter le réalisme et augmenter la complexité des structures et des algorithmes.

8.5.1 Consommation mémoire

Pour le système avec cartes d'environnement, une importante quantité mémoire est prise par les matrices de rotation des ondelettes. Nous utilisons une résolution source de 64×64 et une résolution cible de 32×32, ce qui donne 266Mo, comme indiqué à la figure 8.1. Ensuite, chaque objet modifiable doit avoir trois cartes d'environnement (une par canal de couleur) par source de lumière qui donne la fraction de l'intensité qui atteint l'objet dans chaque direction. Ces cartes sont représentées par des ondelettes, et ensuite compressées. Un triplet de carte d'environnement prend 1475 octets si on garde 3% des coefficients d'ondelettes, ce qui est négligeable, même pour plusieurs centaines de sources de lumière. Chaque BRDF occupe environ 12 Mo de mémoire. Finalement, nous gardons les environnements locaux (une pour chaque canal de couleur) pour chaque direction de normale échantillonnée pour faire le rendu de l'objet. Encore une fois, ils sont compressés avec des ondelettes (on garde 3% des coefficients), ce qui donne 94Ko par objet si on utilise un échantillonnage de 32×32 directions de normale.

Pour le système avec points, le plus gros de la quantité mémoire est le résultat du précalcul par objet. Nous conservons, pour chaque sommet de chaque objet, l'information sur la fraction de l'intensité qui atteint le sommet pour chaque direction. Nous utilisons une résolution de 32×32 pour ces cartes. Elles sont également compressées en ondelettes et nous gardons 3% des coefficients. Chaque carte prend environ 369 octets (pour les trois canaux). Par contre, il doit y avoir une carte par source de lumière, par sommet, par objet. Autrement, la technique n'a pas d'autres consommations de mémoire importantes.

8.5.2 Stratégies

Les différentes stratégies de sélection ont des utilités particulières. Par exemple, si on est dans une scène où il y a plusieurs sources dominantes et que l'utilisateur veut un résultat relativement précis, alors la stratégie "la plus importante" est conseillée. Sinon si un résultat précis n'est pas nécessaire, alors la stratégie "roulette russe" empêchera

le système de toujours choisir la même source, ce qui pourrait la rendre trop intense par rapport aux autres. Dans une scène où les sources ne sont pas dominantes ou sont cachées, la stratégie de l'histogramme pourra aller chercher plusieurs sources pour obtenir le résultat.

Les stratégies sont bien utiles et très rapidement l'utilisateur développe une intuition sur laquelle il doit choisir pour obtenir les meilleurs résultats.

Contrairement aux stratégies de sélection, le choix d'une stratégie de modification est totalement dépendant de ce que l'utilisateur veut accomplir. Pour des modifications incrémentales, ajouter une constante ou doubler l'intensité des sources de lumière sont les stratégies les plus appropriées. Pour obtenir un résultat plus rapidement, alors les stratégies "direct" et "fraction de direct" sont préférables.

8.5.3 Positionner des sources

Il est possible d'étendre le système pour qu'il puisse trouver l'emplacement de sources selon les coups de pinceaux. Il s'agit de placer une certaine quantité de sources d'intensité nulle aux endroits possibles où pourraient se trouver des sources. Quand l'utilisateur peinture un objet, le système sélectionnera quelques-unes des sources nulles et par conséquent, créera ces sources de lumière dans la scène finale.

L'avantage est qu'il n'y a aucune modification à faire aux deux systèmes pour ajouter cette fonctionnalité. Le désavantage est qu'on ajoute de la complexité étant donné qu'il y aura beaucoup de sources de lumière (voir section 8.5.1).

Chapitre 9

Conclusion

Dans ce mémoire, nous avons présenté deux systèmes qui permettent de choisir et de manipuler les intensités de sources de lumière en peignant de la lumière incidente sur des objets. Le premier système, avec les cartes d'environnement, sacrifie un peu l'exactitude des résultats pour moins de précalculs et une meilleure performance de rendu. Le deuxième système, avec les points, demande plus de précalculs et est légèrement moins rapide, mais offre des résultats plus exacts.

Dans les deux cas, les systèmes prennent en compte l'illumination globale de la scène avec tous les effets lumineux qui lui sont rattachés : les interréflexions, les caustiques, la diffusion de la lumière dans des milieux participatifs et les objets semi-transparents.

Nous avons donné des exemples qui montrent que le système fonctionne comme convenu : on peut modifier en temps interactif et de façon intuitive l'apparence d'objets due à l'illumination en peignant sur ceux-ci. Les résultats sont disponibles et affichés quasi instantanément.

Afin de pouvoir répondre à nos contraintes posées au début de ce mémoire, nous avons dû limiter le système rendre correctement certains objets cibles, et à rendre le reste de la scène en utilisant des techniques standards d'illumination directe. Ceci est évidemment une limitation majeure qu'il serait très important d'éliminer pour avoir un système complet utilisable dans un contexte professionnel. Des techniques comme le ré-éclairage direct vers indirect, présenté à la section 2.3, pourraient peut-être être adaptées pour obtenir rapidement une approximation de l'éclairage indirect de la scène complète.

Dans ce mémoire, nous avons aussi divisé le comportement des systèmes en différentes

stratégies pour la sélection et la manipulation des sources de lumière. Plus de recherche pourrait être faite pour automatiser le choix des stratégies selon le cas.

Plusieurs améliorations pourraient rendre les systèmes encore plus utiles. Présentement dans nos systèmes, seulement certains objets peuvent être modifiés. Il serait très utile que toutes les parties de la scène puissent être modifiées de façon transparente, où, par exemple, l'utilisateur peut peinturer un trait qui affecte plusieurs objets en même temps. Il faudrait aussi étudier la possibilité de modifier la distribution de l'émission des sources de lumière afin de pouvoir obtenir des résultats encore plus précis.

Plus d'optimisations pourraient être faites sur la gestion d'un grand nombre de sources de lumière : dans nos structures, on pourrait éliminer les sources de lumière peu importantes par objet, afin d'utiliser moins de mémoire. On pourrait aussi regrouper les sommets dans le système avec points afin de compresser leurs données. De façon similaire, on pourrait déterminer automatiquement le nombre de sommets optimal sur lesquels construire les hémisphères d'importance afin de réduire la mémoire en gardant une bonne qualité. Autrement, les systèmes peuvent gérer un grand nombre de sources de lumière sans problème.

Comme mentionné précédemment, nous avons dû limiter le contexte de l'application à la sélection et à la manipulation de l'intensité des sources de lumière. Il serait bien sûr intéressant d'étudier le problème du design des sources (positionnement, orientation et forme) tout en considérant l'illumination globale et la possibilité de bouger la caméra.

Malgré ces limitations, les systèmes présentés restent une solution intéressante au problème complexe de rendu inverse et démontrent bien l'utilité de la métaphore de la peinture pour modifier l'illumination sur un objet. Étant donné que le comportement de la lumière dans une scène est très complexe et que la manipulation des paramètres des sources de lumière est, de ce fait, également complexe, l'importance d'un bon outil de manipulation est cruciale. Nous croyons que les techniques et les résultats de ce mémoire sont un pas dans la bonne direction. La peinture d'illumination s'avère intuitive et simple et il s'agit donc d'un bon modèle.

Annexe A

Appendices

A.1 Matrice de rotation

Cet algorithme calcule une matrice de rotation qui transforme un frame local vers un frame global orienté autour d'un vecteur **n**.

Data : Le vecteur normal **n**

Result : La matrice de rotation **R**

$\mathbf{s} = [1.0, 0.0, 0.0]$;

$d = \text{dot}(\mathbf{n}, \mathbf{s})$;

if *abs(d)* > 0.6 **then**
    $\mathbf{s} = [0.0, 1.0, 0.0]$;

end

$\mathbf{t} = \text{cross}(\mathbf{s}, \mathbf{n})$;

$\mathbf{s} = \text{cross}(\mathbf{n}, \mathbf{t})$;

$$\mathbf{R} = \begin{bmatrix} \mathbf{s}_x & \mathbf{n}_x & \mathbf{t}_x \\ \mathbf{s}_y & \mathbf{n}_y & \mathbf{t}_y \\ \mathbf{s}_z & \mathbf{n}_z & \mathbf{t}_z \end{bmatrix};$$

return R;

A.2 Hammersley

Voici le code pour produire des points (x, y) de Hammersley sur le plan 2D, $x, y \in [0, 1]$. n est le nombre de points à générer.

```
void PlaneHammersley(float *result, int n)
{
  float p, u, v;
  int k, kk, pos;
  for (k=0, pos=0 ; k<n ; k++)
  {
    u = 0;
    for (p=0.5, kk=k ; kk ; p*=0.5, kk>>=1)
      if (kk & 1)                           // kk mod 2 == 1
        u += p;
    v = (k + 0.5) / n;
    result[pos++] = u;
    result[pos++] = v;
  }
}
```

A.3 Multiplication creuse

A.3.1 Produit scalaire de vecteurs creux

```
template<class T, class I>
__device__ float getVal(
  const I index,
  const unsigned int start,
  const unsigned int end,
  const I* indices,
  const T* vals)
{
  for(int i = start; i < end; ++i){
    if(index < indices[i]){
      return 0.;
    }
```

```
    else if(index == indices[i]){
      return vals[i];
    }
  }
  return 0.;
}

template<class T1, class T2, class I1, class I2>
__global__ void svsvm_kernel(
  unsigned int nbVecs,
  unsigned int offset,
  const int* map1,
  const int* map2,
  const unsigned int* ptr1,
  const unsigned int* ptr2,
  const I1* indices1,
  const I2* indices2,
  const T1* vals1,
  const T2* vals2,
  float* y)
{
  __shared__ float vals[WARP_SIZE];
  int vec_id = blockDim.x * blockIdx.x + threadIdx.x + offset;

  if(map1[vec_id] < 0 || map2[vec_id] < 0){
    y[vec_id] = 0;
    return;
  }

  int thread_id = blockDim.y * blockIdx.y + threadIdx.y;
  int lane = thread_id & (WARP_SIZE-1);
```

```
unsigned int vec1_start = ptr1[map1[vec_id]];
unsigned int vec1_end = ptr1[map1[vec_id] + 1];

unsigned int vec2_start = ptr2[map2[vec_id]];
unsigned int vec2_end = ptr2[map2[vec_id] + 1];

vals[lane] = 0;
for(int jj = vec1_start + lane; jj < vec1_end; jj += WARP_SIZE){
  vals[lane] += vals1[jj] * getVal<T2, I2>(indices1[jj], vec2_start,
                                           vec2_end, indices2, vals2);
}

__syncthreads();
if(lane < 16){
  vals[lane] += vals[lane + 16];
}
__syncthreads();
if(lane < 8){
  vals[lane] += vals[lane + 8];
}
__syncthreads();
if(lane < 4){
  vals[lane] += vals[lane + 4];
}
__syncthreads();
if(lane < 2){
  vals[lane] += vals[lane + 2];
}
__syncthreads();
if(lane < 1){
  vals[lane] += vals[lane + 1];
}
```

```
  __syncthreads();

  if(lane == 0){
    y[vec_id] = vals[lane];
  }
}
```

A.3.2 Produit scalaire entre matrices creuses et vecteurs pleins

```
__global__ void spmv_csr_vector_kernel(
  const unsigned int num_mat,
  const unsigned int num_rows,
  const unsigned int num_values,
  const unsigned int* ptrMat,
  const unsigned short* ptrRow,
  const unsigned short* indices,
  const short* values,
  const float* x,
  float* y)
{
  __shared__ float vals[WARP_SIZE];

  int matrix_id = blockDim.x * blockIdx.x + threadIdx.x;
  int thread_id = blockDim.y * blockIdx.y + threadIdx.y;
  int lane = thread_id & (WARP_SIZE-1);
  int row_id = thread_id / WARP_SIZE;

  unsigned int rowOffset = matrix_id * (num_rows + 1);
  unsigned int dataOffset = ptrMat[matrix_id];

  int row_start = ptrRow[rowOffset + row_id];
  int row_end = ptrRow[rowOffset + row_id + 1];
```

```
int index = lane;

vals[index] = 0;
for(int jj = row_start + lane; jj < row_end; jj += WARP_SIZE){
  vals[index] += values[dataOffset + jj] *
  fetch_x<false>(indices[dataOffset + jj], x);
}

__syncthreads();
if(lane < 16){
  vals[index] += vals[index + 16];
}
 __syncthreads();
if(lane < 8){
  vals[index] += vals[index + 8];
}
__syncthreads();
if(lane < 4){
  vals[index] += vals[index + 4];
}
__syncthreads();
if(lane < 2){
  vals[index] += vals[index + 2];
}
__syncthreads();
if(lane < 1){
  vals[index] += vals[index + 1];
}
__syncthreads();

if(lane == 0){
  int indexY = matrix_id * num_rows + row_id;
```

```
    y[indexY] = vals[index];
  }
}
```

Bibliographie

[AP] Michael Ashikhmin et Simon Premoze. « Distribution-based BRDFs ». `http://www.cs.utah.edu/~premoze/dbrdf/`.

[BAOR06] Aner Ben-Artzi, Ryan Overbeck et Ravi Ramamoorthi. « Real-time BRDF Editing in Complex Lighting ». *ACM Transactions on Graphics*, volume 25, numéro 3, pages 945–954, juillet 2006.

[Ble] « Blender ». `http://www.blender.org`.

[Bli77] James F. Blinn. « Models of Light Reflection for Computer Synthesized Pictures ». Dans *SIGGRAPH '77 : Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, pages 192–198, New York, NY, USA, 1977. ACM.

[Boo] Boost. « Boost C++ Libraries ». `http://www.boost.org/`.

[CAM08] Petrik Clarberg et Tomas Akenine-Möller. « Practical Product Importance Sampling for Direct Illumination ». *Computer Graphics Forum*, volume 27, numéro 2, pages 681–690, avril 2008.

[CJAMJ05] Petrik Clarberg, Wojciech Jarosz, Tomas Akenine-Möller et Henrik Wann Jensen. « Wavelet Importance Sampling : Efficiently Evaluating Products of Complex Functions ». *ACM Transactions on Graphics*, volume 24, numéro 3, pages 1166–1175, août 2005.

[CPK06] Mark Colbert, Sumanta Pattanaik et Jaroslav Krivanek. « BRDF-Shop : Creating Physically Correct Bidirectional Reflectance Distribution Functions ». *IEEE Comput. Graph. Appl.*, volume 26, numéro 1, pages 30–36, 2006.

[CSF99] António Cardoso Costa, António Augusto Sousa et Fernando Nunes Ferreira. « Lighting Design : A Goal Based Approach Using Optimisation ». Dans *Eurographics Rendering Workshop 1999*, pages 317–328, juin 1999.

[CW93] Michael F. Cohen et John Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, Inc., San Diego, CA, USA, 1993.

[DBB02] Philip Dutre, Kavita Bala et Philippe Bekaert. *Advanced Global Illumination*. A. K. Peters, Ltd., Natick, MA, USA, 2002.

[Had02] J. Hadamard. « Sur les problèmes aux dériveés partielles et leur signification physique ». Dans *Princeton University Bulletin*, numéro 23, pages 49–52, 1902.

[HLHS03] Jean-Marc Hasenfratz, Marc Lapierre, Nicolas Holzschuch et François Sillion. « A Survey of Real-Time Soft Shadows Algorithms ». *Computer Graphics Forum*, volume 22, numéro 4, pages 753–774, décembre 2003.

[HMH95] Vigain Harutunian, Juan Carlos Morales et John R. Howell. « Radiation Exchange Within an Enclosure of Diffuse-Gray Surfaces : The Inverse Problem ». *Proc. ASME/AIChE 1995 National Heat Trans. Conf.*, août 1995.

[HPB06] Miloš Hašan, Fabio Pellacini et Kavita Bala. « Direct-to-indirect Transfer for Cinematic Relighting ». *ACM Transactions on Graphics*, volume 25, numéro 3, pages 1089–1097, juillet 2006.

[Jen01] Henrik Wann Jensen. *Realistic Image Synthesis Using Photon Mapping*. A. K. Peters, Ltd., Natick, MA, USA, 2001.

[KPC93] John K. Kawai, James S. Painter et Michael F. Cohen. « Radioptimization - Goal Based Rendering ». Dans *Proceedings of SIGGRAPH 93*, Computer Graphics Proceedings, Annual Conference Series, pages 147–154, août 1993.

[Lar92] Gregory J. Ward Larson. « Measuring and Modeling Anisotropic Reflection ». Dans *Computer Graphics (Proceedings of SIGGRAPH 92)*, pages 265–272, juillet 1992.

[LFTG97] Eric P. F. Lafortune, Sing-Choong Foo, Kenneth E. Torrance et Donald P. Greenberg. « Non-Linear Approximation of Reflectance Functions ». Dans *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, pages 117–126, août 1997.

[Mar98] Stephen R. Marschner. *Inverse Rendering in Computer Graphics*. Thèse de doctorat, Cornell University, 1998.

[NRH04] Ren Ng, Ravi Ramamoorthi et Pat Hanrahan. « Triple Product Wavelet Integrals for All-Frequency Relighting ». *ACM Transactions on Graphics*, volume 23, numéro 3, pages 477–487, août 2004.

[NVIa] NVIDIA. « Cg ». `http://developer.nvidia.com/page/cg_main.html`.

[NVIb] NVIDIA. « CUDA ». `http://www.nvidia.com/object/cuda_home.html`.

[NVIc] NVIDIA. *CUDA Programming Guide*, 2.2.1 édition. `http://developer.download.nvidia.com/compute/cuda/2_21/toolkit/docs/NVIDIA_CUDA_Programming_Guide_2.2.1.pdf`.

[OMSI07] Makoto Okabe, Yasuyuki Matsushita, Li Shen et Takeo Igarashi. « Illumination Brush : Interactive Design of All-Frequency Lighting ». *Pacific Conference on Computer Graphics and Applications*, pages 171–180, 2007.

[PBMF07] Fabio Pellacini, Frank Battaglia, R. Keith Morley et Adam Finkelstein. « Lighting With Paint ». *ACM Transactions on Graphics*, volume 26, numéro 2, pages 9 :1–9 :14, juin 2007.

[PF92] Pierre Poulin et Alain Fournier. « Lights From Highlights and Shadows ». Dans *1992 Symposium on Interactive 3D Graphics*, pages 31–38, mars 1992.

[PGSP08] Romain Pacanowski, Xavier Granier, Christophe Schlick et Pierre Poulin. « Sketch and Paint-based Interface for Highlight Modeling ». Dans *Eurographics Workshop on Sketch-based Interfaces and Modeling 2008*, pages 17–23, juin 2008.

[PH03] Emil Praun et Hugues Hoppe. « Spherical Parameterization and Remeshing ». *ACM Transactions on Graphics*, volume 22, numéro 3, pages 340–349, juillet 2003.

[PH04] Matt Pharr et Greg Humphreys. *Physically Based Rendering : From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[Pho75] Bui T. Phong. « Illumination for Computer Generated Pictures ». *Commun. ACM*, volume 18, numéro 6, pages 311–317, jun 1975.

[PP03] Gustavo Patow et Xavier Pueyo. « A Survey of Inverse Rendering Problems ». *Computer Graphics Forum*, volume 22, numéro 4, pages 663–687, décembre 2003.

[PRJ97] P. Poulin, K. Ratib et M. Jacques. « Sketching Shadows and Highlights to Position Lights ». Dans *Computer Graphics International 1997*, pages 56–63, juin 1997.

[PTG02] Fabio Pellacini, Parag Tole et Donald P. Greenberg. « A User Interface for Interactive Cinematic Shadow Design ». *ACM Transactions on Graphics*, volume 21, numéro 3, pages 563–566, juillet 2002.

[RH01] Ravi Ramamoorthi et Pat Hanrahan. « A Signal-Processing Framework for Inverse Rendering ». Dans *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 117–128, août 2001.

[SC97] Peter S. Shirley et Kenneth Chiu. « A Low Distortion Map Between Disk and Square ». *Journal of Graphics Tools*, volume 2, numéro 3, pages 45–52, 1997.

[SDS+93] Chris Schoeneman, Julie Dorsey, Brian Smits, James Arvo et Donald Greenberg. « Painting With Light ». Dans *Proceedings of SIGGRAPH 93*, Computer Graphics Proceedings, Annual Conference Series, pages 143–146, août 1993.

[SDS95] Eric J. Stollnitz, Tony D. DeRose et David H. Salesin. « Wavelets for Computer Graphics : Part 1 ». *IEEE Computer Graphics & Applications*, volume 15, numéro 3, pages 76–84, mai 1995.

[SKS02a] Peter-Pike Sloan, Jan Kautz et John Snyder. « Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments ». SIGGRAPH Presentation slides : `http://online.cs.nps.navy.mil/DistanceEducation/online.siggraph.org/2002/Papers/08_LightingAndAppearance/sloan.ppt`, 2002.

[SKS02b] Peter-Pike Sloan, Jan Kautz et John Snyder. « Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting

Environments ≫. *ACM Transactions on Graphics*, volume 21, numéro 3, pages 527–536, juillet 2002.

[SP94] Francois X. Sillion et Claude Puech. *Radiosity and Global Illumination*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1994.

[SS95] Peter Schröder et Wim Sweldens. ≪ Spherical Wavelets : Efficiently Representing Functions on the Sphere ≫. Dans *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, pages 161–172, août 1995.

[TS06] Yu-Ting Tsai et Zen-Chung Shih. ≪ All-Frequency Precomputed Radiance Transfer Using Spherical Radial Basis functions and clustered tensor approximation ≫. *ACM Transactions on Graphics*, volume 25, numéro 3, pages 967–976, juillet 2006.

[WLH97] Tien-Tsin Wong, Wai-Shing Luk et Pheng-Ann Heng. ≪ Sampling with Hammersley and Halton points ≫. *Journal of Graphics Tools*, volume 2, numéro 2, pages 9–24, 1997.

[WNLH06] Rui Wang, Ren Ng, David Luebke et Greg Humphreys. ≪ Efficient Wavelet Rotation for EnvironmentMap Rendering ≫. Dans *Rendering Techniques 2006 : 17th Eurographics Workshop on Rendering*, pages 173–182, juin 2006.

[WPF90] Andrew Woo, Pierre Poulin et Alain Fournier. ≪ A Survey of Shadow Algorithms ≫. *IEEE Computer Graphics & Applications*, volume 10, numéro 6, pages 13–32, novembre 1990.

[WW92] Alan Watt et Mark Watt. *Advanced Rendering and Animation Techniques : Theory and Practice*. Addison-Wesley Professional, nov 1992.