Université de Montréal

Représentation hiérarchique et efficace des sources lumineuses dans le cadre du rendu d'images

par

Eric Paquette

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Mémoire présenté à la faculté des études supérieures

en vue de l'obtention du grade de

Maître ès sciences (M.Sc.)

en informatique

Décembre 1997

Université de Montréal

Faculté des études supérieures

Ce mémoire de maîtrise intitulé

Représentation hiérarchique et efficace des sources lumineuses dans le cadre du rendu d'images

présenté par

Eric Paquette

a été évalué par un jury composé des personnes suivantes :

Président :	Jean Meunier
Directeur de recherche :	Pierre Poulin
Membre :	François Major

Sommaire

Le réalisme dans les images de synthèse est très important. Pour l'atteindre, une bonne approximation de l'éclairage est nécessaire. L'éclairage d'une scène réelle est souvent composé de plusieurs sources lumineuses. Le calcul de l'illumination résultant d'un ensemble de sources représente souvent la partie la plus coûteuse du rendu car les sources sont traitées individuellement.

Notre recherche se place dans le contexte de l'éclairage direct provenant de sources ponctuelles et calculé par tracé de rayon sur des surfaces à modèle de réflexion de Phong. Nous réduisons les temps de calcul en présence de centaines de sources lumineuses grâce à l'utilisation d'une représentation hiérarchique des sources. Une structure d'*octree* nous procure différents niveaux de précision de l'illumination en tout point de la scène.

Pour les scènes sans ombre visible, nous avons développé une borne efficace sur l'erreur maximale encourue par l'utilisation de la hiérarchie pour les surfaces diffuses. Nous présentons un critère d'erreur pour la réflexion spéculaire et suggérons comment l'adapter pour qu'il fournisse aussi une borne sur l'erreur maximale.

Les accélérations réalisées par notre méthode sont très bonnes, étant jusqu'à 5 fois plus rapide en utilisant la borne sur l'erreur et jusqu'à 38 fois plus rapide sans borne sur l'erreur. L'erreur maximale par pixel que nous observons est très petite (3.5%) et toujours inférieure à la borne fixée par l'utilisateur. De plus, l'erreur moyenne par pixel est négligeable (0.004%).

Nous exposons également une méthode qui permet d'utiliser la hiérarchie pour les scènes comportant des ombres. Cette méthode se base sur l'utilisation d'une visibilité approximative dérivée de la visibilité volumétrique.

Mots-clés :

Infographie 3D, image de synthèse, rendu, tracé de rayon, hiérarchie, illumination, agrégation, *octree*, réflexion, visibilité, ombre.

Table des matières

Liste des tableaux

Table des figures

À Jacqueline et Rolland

Remerciements

Je remercie mes parents pour les encouragements qu'ils m'ont en tout temps donnés à poursuivre mes études toujours plus loin. Je remercie ma fiancée Karyne pour la patience dont elle a su faire preuve pendant ces longues semaines de rédaction. Je remercie tous mes amis qui ont su me rester fidèles malgré les longs moments sans nouvelles de moi lors des fins de trimestre et de la rédaction. Je remercie Monique avec qui j'ai eu des discussions qui m'ont fait poursuivre aux études supérieures.

J'ai particulièrement apprécié travailler avec mes deux grands conseillers, mon directeur Pierre Poulin et Georges Drettakis qui m'a dirigé en France et qui poursuit son implication appréciée dans mes recherches. Je les remercie d'avoir su guider mes travaux vers ce résultat, un mémoire de maîtrise dont je suis fier. Je remercie les gens du laboratoire d'infographie de l'Université de Montréal ainsi que ceux du laboratoire iMAGIS de Grenoble. J'ai passé avec ces amis de très beaux moments, autant au travail que dans la vie de tous les jours. Je remercie également le directeur du laboratoire iMAGIS, M. Claude Puech, d'avoir rendu possible et agréable la collaboration entre son équipe et celle du laboratoire d'infographie de l'Université de Montréal.

Ma maîtrise a été rendue possible grâce à une bourse ÉS/A du CRSNG. Je suis très reconnaissant au gouvernement canadien de l'appui qu'il donne à la recherche.

J'aimerais finalement mentionner plusieurs organismes qui ont affecté de près ou de loin mes recherches : le CRSNG, l'Université de Montréal et Taarna Studios pour les ressources qu'ils ont mis à la disposition des membres du laboratoire d'infographie, les services informatiques pour le matériel informatique qui est mis à la disposition des étudiants en infographie, l'AUPELF pour la bourse CIME qui m'a permis de me rendre en France pour travailler avec l'équipe iMAGIS et l'OFQJ pour le support qui m'a été fourni lors de la planification de mon séjour en France.

Remarques préliminaires

Certains mots qui peuvent porter à confusion ou qui sont tirés de l'anglais se retrouvent dans le glossaire en page 87. La première occurrence de ces mots sera mise en évidence par une *police de caractère oblique*. La *police de caractère italique* a été choisie pour marquer les mots anglais.

À moins d'indication contraire, les vecteurs présents dans les formules sont des vecteurs en trois dimensions et sont normalisés. Le produit scalaire de deux vecteurs $\vec{V_1}$ et $\vec{V_2}$ est noté $\vec{V_1} \cdot \vec{V_2}$.

Nous avons implanté notre méthode en C++ dans le logiciel de tracé de rayon *OORT* de Nicholas Wilt [Wil93]. Les tests ont été réalisés sur différents ordinateurs Silicon Graphics, allant de Indy R46000 à Indigo2 R10000, touts utilisant le système d'exploitation IRIX6.2. La plupart des images sont calculées avec 24 bits de couleur (8 R, 8 V, 8 B) sans *correction d'aliassage* (un rayon par pixel). Pour la présentation dans le mémoire, les images sont converties en 256 tons de gris.

Chapitre 1

Introduction

Nous décrivons ici les problèmes qui motivent cette recherche puis nous parlons des buts de notre travail et situons brièvement notre contribution. Nous terminons par un aperçu du contenu des différents chapitres.

1.1 Problématique

Le *rendu* d'images occupe une place dominante en *infographie* et fait l'objet d'un grand nombre de publications depuis les années 1970. La recherche en infographie se dirige toujours vers deux buts souvent opposés : l'augmentation du réalisme des images générées et la diminution des temps de calcul.

1.1.1 Réalisme des images

Le réalisme des scènes provient, en première part, d'une description géométrique suffisamment correcte des *objets* qui y sont présents. Des objets simples, tels les cubes et les sphères, ne peuvent donner toute la richesse présente dans notre environnement quotidien. Plusieurs primitives puissantes, telles les quadriques, les surfaces splines, ou les surfaces implicites, peuvent être ajoutées. Elles peuvent aussi être combinées à l'aide de la *construction de solides* pour créer des formes encore plus complexes. Tout ceci permet d'approcher assez bien les formes présentes dans un environnement réel. Les détails fins des surfaces, telle la rugosité, ne sont pas représentés par des primitives. Ils font plutôt partie des propriétés des objets. Ces propriétés sont traitées par différents modèles comme nous l'expliquons plus loin. Évidemment, une scène réaliste,

en plus d'être composée de primitives relativement complexes, contiendra énormément de ces dernières. Une grande quantité de primitives est très souvent nécessaire pour bien montrer tous les détails des objets nous entourant.

Que la forme des objets de notre modèle soit fidèle à ce que nous voulons représenter n'est évidemment pas suffisant. L'*apparence de la surface* des objets doit aussi correspondre à la réalité que nous recherchons. L'apparence des objets nous est tout d'abord visible grâce à l'éclairage qu'ils reçoivent des sources lumineuses. Un éclairage réaliste nous permet de bien percevoir la forme et la position des objets. Encore une fois, les types de lumières et leur nombre sont importants pour nous permettre d'avoir des scènes riches.

La détermination de la contribution d'une lumière à l'illumination d'un objet est importante et complexe. L'irradiance qui atteint l'objet doit être déterminée en tenant compte de la distance et de l'orientation de la lumière ainsi que des possibilités d'occlusion par les objets (ombrages). Nous devons ensuite traiter la façon selon laquelle la surface affectera (réflexion, transmission, réfraction) la lumière reçue. Encore une fois, si nous désirons des résultats intéressants, il faut s'assurer que le modèle d'interaction de la lumière avec la surface est suffisamment complet pour traduire les effets d'*illumination* recherchés.

À la contribution de toutes les lumières, il faut ajouter la contribution lumineuse qui, venant des sources, est reflétée (transmise, réfractée) sur les objets (*éclairage indirect*). Tenir compte de ces effets est très important pour avoir des ombres et une illumination correcte de tous les objets.

1.1.2 Diminution des temps de calcul

Le réalisme des images se traduit souvent par des temps de calcul très grands. Les phénomènes physiques qui régissent l'apparence des objets sont très complexes et certains d'entre eux restent encore mal compris.

L'utilisation de primitives complexes donne un réalisme accru, mais engendre du même coup des calculs souvent plus longs lors du rendu. Les algorithmes visant à accélérer le rendu assument souvent un type particulier de primitive. Ceci complique les choses quand une scène comporte plusieurs types de primitives. Par exemple, plusieurs techniques, comme les volumes d'ombre [Cro77], s'appliquent seulement à des

objets polygonaux. Plutôt que d'utiliser les objets complexes directement, une approximation de ces objets est souvent utilisée. La *tessellation* est l'exemple le plus évident de ces approximations. Un objet complexe est remplacé par une plus grande quantité de polygones. Ces derniers sont plus simples à traiter par les algorithmes et peuvent souvent être envoyés directement au *matériel graphique*.

Oubliant la complexité au niveau des primitives, la présence de plusieurs milliers d'objets dans une scène ralentit dramatiquement le rendu. Déterminer quelle partie de chacun des objets est visible d'un certain *point de vue* est très coûteux. De plus, quand on remplace un objet par son approximation composée d'une multitude d'objets plus simples, les problèmes ne font que s'aggraver.

Plusieurs méthodes s'attaquent à ce problème, les plus répandues étant celles utilisant la subdivision de l'espace, la cohérence et le parallélisme. Les hiérarchies de volumes englobants [RW80], les grilles régulières [FTI86, AW87], les *octrees* [Gla84] et les *arbres BSP* [Kap87] sont de bons exemples des méthodes de subdivision de l'espace. L'utilisation de ces algorithmes, sauf pour le *BSP*, permet de calculer la visibilité à partir d'une liste réduite d'objets. Les arbres *BSP* subdivisent plutôt l'espace de façon à pouvoir afficher les objets du plus éloigné au plus rapproché. De cette manière, aucun calcul de visibilité n'a à être fait par rapport au point de vue lors de l'affichage. Les méthodes de cohérence [HG86, Woo93], quant à elles, tirent avantage des cohérences habituellement présentes dans les scènes. Par exemple, un objet qui bloque la lumière à un certain endroit dans l'image finale va probablement bloquer la même lumière dans une zone autour de cet endroit. Il est alors possible de vérifier si la lumière est cachée par le dernier *bloqueur* avant de regarder tous les autres objets.

Le parallélisme est une forme bien particulière d'accélération du rendu. Plutôt que de fournir un algorithme plus rapide, le parallélisme scinde un calcul en plusieurs parties indépendantes, qui seront exécutées en même temps plutôt qu'une après l'autre. L'accélération qu'il est possible de tirer du parallélisme dépend des ressources disponibles et de la possibilité de diviser le problème en parties indépendantes. Le rendu se prête bien à une division et beaucoup d'architectures graphiques basées sur ce principe existent depuis longtemps. La limite vient du coût du matériel qui comporte beaucoup de mémoire spécialisée pour emmagasiner les calculs intermédiaires, les textures et l'image finale. L'analyse des accélérations venant du parallélisme s'arrêtera ici, notre

but étant limité au développement d'un algorithme plus rapide.

Une autre façon très efficace de réduire les temps de calcul est de remplacer une géométrie complexe d'objets par une texture à deux [MS95] ou trois dimensions [Ney96]. Il est alors plus rapide de calculer l'apparence de la texture, que de calculer la géométrie qu'elle remplace. Pour remplacer la géométrie par une texture, il faut tout d'abord calculer la texture à partir de cette géométrie. La texture doit être utilisée à plusieurs reprises pour que la méthode amène une réduction du temps de calcul. Ces méthodes sont donc principalement utilisées dans les animations.

Pour faire le rendu d'objets il faut tout d'abord déterminer les parties d'objets qui doivent être affichées. Une fois cela fait, il faut trouver quelles sources éclairent les objets, et de quelle façon. La cohérence nous donne une façon rapide de trouver une occlusion, mais si cette dernière n'a pas lieu ou n'est pas complète, nous devons revenir aux méthodes habituelles. Celles-ci sont très lentes lorsqu'il y a beaucoup d'objets et de sources lumineuses. Une façon efficace de réduire le temps de calcul de la visibilité des sources est de les échantillonner de façon sélective [War91, SWZ96]. En déterminant la visibilité d'un sous-ensemble des sources lumineuses plutôt que de toutes, nous gagnons beaucoup de temps. Tant que l'échantillon est bien choisi, les gains en vitesse surpassent de beaucoup les pertes en qualité d'image.

Après avoir trouvé quels objets sont visibles et comment ils sont éclairés, il reste à calculer leur apparence. Calculer l'apparence de la surface peut être particulièrement coûteux si le modèle de réflexion est complexe. Le modèle d'illumination de Phong [Pho75] remplace donc souvent des modèles plus généraux tels le modèle de He *et al.* [HTSG91] ou les *BRDF* [CMS87]. L'image générée perd en réalisme, mais son calcul est beaucoup plus rapide et ses coûts mémoire sont diminués.

1.2 Motivation

Nous avons vu dans les sections précédentes que la vitesse de rendu est surtout déterminée par le réalisme désiré. Les méthodes hiérarchiques sont de puissants outils pour réduire les problèmes liés à la présence de beaucoup d'objets. Par contre, une scène comportant beaucoup de sources lumineuses peut poser de sérieux problèmes. La présence de centaines ou même de milliers de sources lumineuses implique beaucoup de calculs de visibilité. Contrairement à la majorité des autres méthodes, les méthodes

d'échantillonnage sélectif permettent de réduire la quantité d'information à traiter. Plutôt que de traiter toutes les sources lumineuses, seulement un sous-ensemble est pris en considération. Ces méthodes donnent en général de bons résultats, mais dans le cas où les sources lumineuses ont des contributions similaires, l'échantillonnage devra considérer presque toutes les sources ou en négliger une grande quantité. Ceci peut se traduire par des erreurs trop grandes dans l'image finale.

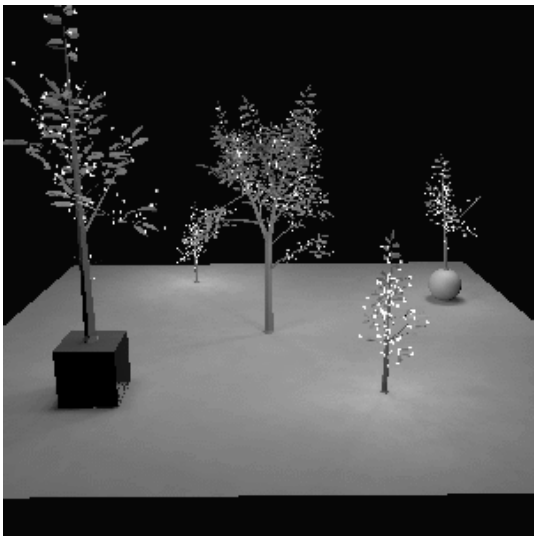
Les scènes composées de beaucoup de sources lumineuses ne sont pas utopiques. De bons exemples seraient la simulation de flammes ou de feux d'artifice par un système de particules, un sapin de Noël ou une rue éclairée par des guirlandes de lumières. La figure 1.1 montre des exemples de scènes comportant beaucoup de sources lumineuses.

Des *sources lumineuses étendues* peuvent être remplacées par une approximation composée de beaucoup de sources ponctuelles [BB84]. Remplacer un éclairage complexe par des sources lumineuses ponctuelles fournit aussi énormément de sources. Un *photon map* [JC95] pour lequel chaque photon serait remplacé par une source ponctuelle utilisée pour l'éclairage indirect contiendrait énormément de sources lumineuses (plusieurs centaines de milliers). Une idée similaire est présentée par Keller [Kel97] où plusieurs sources lumineuses ponctuelles sont utilisées pour faire l'approximation des résultats de méthodes de radiosité.

L'utilisation de beaucoup de sources lumineuses est souvent nécessaire pour obtenir un réalisme adéquat. Remplacer des flammes par une seule source ne donne pas de bons résultats. L'exemple de la rue se prête encore moins bien à un remplacement de sources par quelques sources.

Présentement, les artistes qui créent des scènes sont contraints de jouer avec différents effets pour essayer de générer des images pour lesquelles l'éclairage semble complexe mais se résume en réalité à très peu de sources. Le rendu de scènes complexes comportant plus de dix sources est pour le moment prohibitif. Par exemple, la scène des arbres dans la figure 1.1 a demandé plus de treize heures de calcul contre moins d'une minute s'il n'y avait qu'une seule source éclairant la scène de l'extérieur.

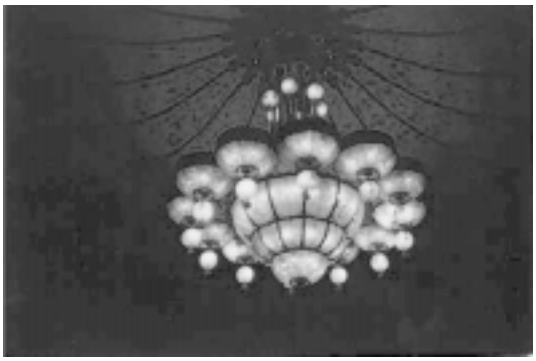
Nous avons besoin d'un algorithme adaptatif qui, pour certaines parties de la scène, remplace beaucoup de sources par quelques-unes selon la précision désirée. L'algorithme doit aussi pouvoir détecter les endroits où il est impossible de remplacer les sources. Pour ces endroits, l'échantillonnage complet doit être fait.



(a) Arbres

(b) Rue

(c) Hall



(d) Lustre

FIG. 1.1: Exemples de scènes comportant beaucoup de lumières. Les arbres et la rue sont des scènes synthétiques tandis que le hall et le lustre sont des scènes réelles. Dans la scène des arbres, chacun des points blancs dans les arbres est une source lumineuse. Cette scène comporte 590 sources lumineuses. La scène de la rue comporte 57 sources lumineuses. La scène de hall montre que les éclairages sont souvent complexes. L'éclairage du lustre serait extrêmement complexe à modéliser. Il pourrait être approximé par une grande quantité de sources lumineuses ponctuelles.

Nous avons donc choisi d'explorer la construction d'une hiérarchie de sources lumineuses. Dans notre hiérarchie, les sources sont regroupées en *agrégats*. Nous associons à chaque agrégat une nouvelle source qui représente une approximation de l'illumination du groupe. Cette nouvelle source lumineuse, que nous appelons une source virtuelle, doit être une bonne approximation de celles qu'elle remplace.

Nous avons développé des critères d'erreur qui permettent de choisir le niveau le plus élevé de la hiérarchie qui fournit une approximation respectant une borne d'erreur désirée. Notre méthode fournit des façons de quantifier l'erreur commise lors de son utilisation. Les bornes d'erreur que nous utilisons sont suffisamment efficaces pour permettre de calculer des images au niveau de qualité voulu. De plus, le temps de calcul de ces bornes est suffisamment petit pour que les gains en performance restent considérables.

1.3 Contexte

Nous avons décidé de traiter seulement l'*éclairage direct* dans le cadre du *tracé de rayon* [Whi80]. Cette décision a été motivée par la simplicité et la puissance de la méthode du tracé de rayon. Le tracé de rayon permet de bien traiter les phénomènes d'illumination directe, d'ombre, de réflexion, de transparence et de réfraction[1]. Il ne considère cependant pas les réflexions (transparences, réfractions) de lumière entre les objets. Ce phénomène de *simulation d'éclairage*, bien que nécessaire pour atteindre un réalisme adéquat, n'est pas considéré dans notre recherche et fait partie de travaux futurs. Nous avons aussi choisi de ne considérer que des sources lumineuses ponctuelles et le modèle de réflexion de Phong, laissant encore à des travaux futurs l'extension à des modèles plus complexes.

Ce contexte restreint n'est pas limitatif. Tel que nous le montrons dans les travaux futurs, notre méthode peut être étendue. De plus, le tracé de rayon est une méthode de rendu très populaire et la majorité des images faites avec le tracé de rayon sont calculées avec un modèle de Phong.

[1] En tracé de rayon, la réfraction est bien traitée pour les rayons de l'oeil. La réfraction des rayons d'ombre est beaucoup plus complexe et souvent n'est pas traitée par les programmes de tracé de rayon. Notre méthode ne tient pas compte de la réfraction des rayons d'ombre.

1.4 Aperçu des différents chapitres

Dans le chapitre 2, nous exposons certaines notions utiles à une bonne compréhension de notre méthode. Ces explications, bien que parfois sommaires, fournissent à la personne qui connaît l'infographie toutes les pistes nécessaires. Au chapitre 3 nous décrivons les autres travaux qui s'attaquent à l'accélération du rendu par rapport aux sources lumineuses. Différentes stratégies y sont exposées avec leurs avantages et inconvénients. Le chapitre 4 décrit notre hiérarchisation. Les raisons de nos choix de représentation ainsi que la méthode de construction s'y trouvent. Au chapitre 5 nous présentons les critères d'erreur que nous utilisons avec la hiérarchie de lumières présentée au chapitre précédent. L'élaboration et l'utilisation de ces critères y sont décrites. Ce chapitre ne traite que les scènes où il n'y a pas d'ombres. L'extension aux scènes générales par la détermination de la visibilité des sources est faite au chapitre 6. Les résultats sont présentés tout au long du développement des différentes méthodes d'utilisation de la hiérarchie. Nous terminons au chapitre 7 avec l'analyse de nos résultats et les avenues de recherche ouvertes par ce travail.

Chapitre 2

Préalables

Dans ce chapitre, nous survolons certains concepts qui sont utiles pour mieux comprendre les sections qui suivent. Les descriptions que nous faisons sont assez succinctes, mais fournissent l'essentiel. Nous référons aux articles où une description plus détaillée peut être trouvée lorsque c'est nécessaire.

2.1 Tracé de rayon

Le tracé de rayon [Whi80] est une méthode puissante de rendu permettant la simulation de plusieurs effets. Il permet également l'utilisation d'objets non polygonaux. Avec le tracé de rayon, une image est calculée en lançant dans la scène des rayons à partir de l'oeil. Ces rayons traversent les pixels qui se trouvent sur le plan de l'image, tel qu'on peut le voir à la figure 2.1. Une fois le point d'intersection avec un objet déterminé, il reste à trouver la contribution des sources lumineuses à l'apparence de la surface. Dans le cas d'une source ponctuelle, un *rayon d'ombre* est lancé en direction de la source. Si aucun objet ne bloque le rayon, la source est visible. Nous appellerons "bloqueur" un objet qui cache, partiellement ou entièrement, une source lumineuse. Pour des sources lumineuses étendues [VG84, CPC84], un échantillonnage ou une méthode analytique peuvent être utilisés. Il est à noter que la visibilité doit être calculée pour chaque source lumineuse. Le coût du calcul de la visibilité peut devenir beaucoup plus important que celui de la détermination des surfaces visibles faite par les *rayons de l'oeil*. Un rayon doit être lancé vers chaque source lumineuse pour chacun des rayons de l'oeil, ce qui peut faire un beaucoup plus grand nombre de rayons.

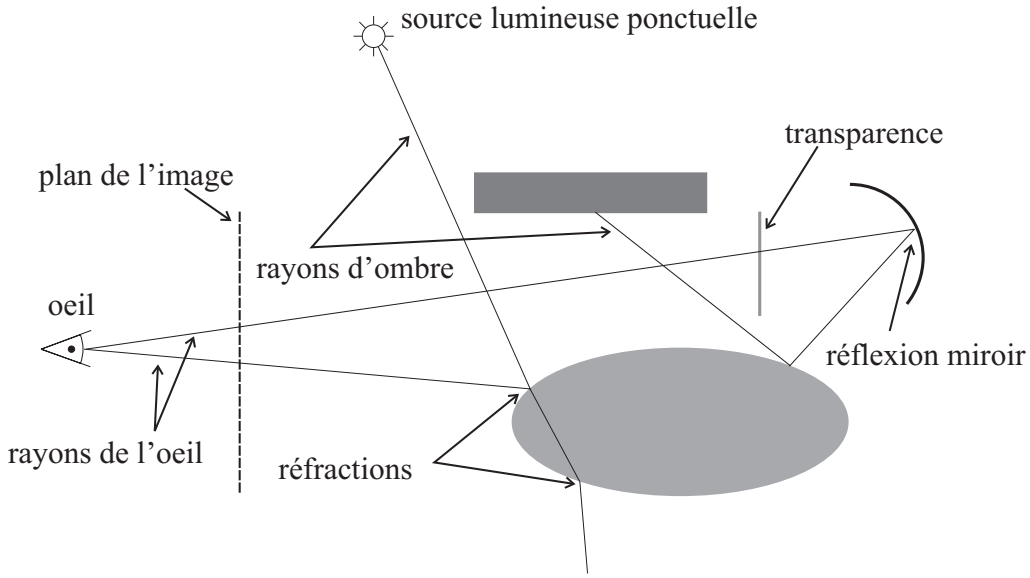


FIG. 2.1: Le tracé de rayon.

Des effets de réflexion miroir et de transparence peuvent également être simulés avec cette méthode. En présence de ces phénomènes, en plus de lancer les rayons vers les sources au point d'intersection, il faudra encore lancer les rayons aux sources à partir du point d'intersection du rayon réfléchi ou transmis. Il y aura donc un nombre grandissant de calculs d'illumination à faire.

2.2 Subdivision de l'espace

Les méthodes de subdivision de l'espace sont au coeur de beaucoup d'algorithmes en infographie. Typiquement, une subdivision de l'espace permet d'identifier rapidement un sous-ensemble réduit de tous les objets de la scène, nécessaire pour le traitement d'un certain phénomène.

Une méthode de subdivision de l'espace divise la scène en sections relativement restreintes. Pour chacun des éléments de la subdivision, que nous appellerons maintenant *voxels*, la liste des objets qui s'y trouvent est construite. Quand nous devons trouver les objets qui respectent un certain critère, nous n'avons qu'à regarder quels *voxels* peuvent contenir des objets respectant le critère. Ceci peut être un gain considérable car nous n'avons pas à vérifier tous les objets, mais seulement les *voxels* et les objets des *voxels* retenus. Par exemple, pour le tracé de rayon, plutôt que de vérifier l'intersection

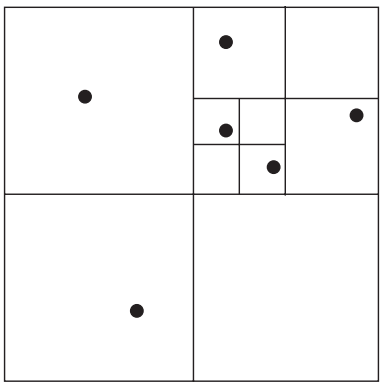


FIG. 2.2: Subdivision en *quadtree*.

du rayon avec tous les objets, il est possible de regarder seulement les objets qui se trouvent dans les *voxels* traversés par le rayon.

2.2.1 *Octree*

L'*octree* [Gla84] est une méthode intéressante de subdivision de l'espace. Plutôt que de subdiviser régulièrement, l'*octree* subdivise seulement là où c'est nécessaire. Il représente l'espace de façon hiérarchique. À sa racine, un seul *voxel* contient tout l'espace. Les *voxels* sont construits selon trois axes orthogonaux qui sont généralement les axes x, y et z de la scène, bien qu'il puisse en être autrement. Si un *voxel* ne représente pas bien l'espace qu'il contient, il est subdivisé en huit *voxels* enfants. Ces *voxels* enfants sont eux aussi alignés selon les axes, leur dimension est la moitié de celle de leur parent et ils couvrent entièrement l'espace de leur parent. La figure 2.2 montre un exemple d'une telle subdivision en deux dimensions. En deux dimensions, l'équivalent de l'*octree* s'appelle un *quadtree*. Dans cet exemple, si un *voxel* contient plus de deux points, il est subdivisé. Le nombre d'objets est souvent le critère utilisé pour faire la subdivision des *voxels*.

La construction de l'*octree* se fait généralement en ajoutant tous les éléments à la racine. Si un *voxel* ne représente pas bien ce qu'il contient, il est subdivisé. Pour éviter qu'une subdivision infinie se produise, la profondeur de l'arbre des *voxels* est limitée.

2.3 Équation fondamentale du rendu

Une définition simple et complète des phénomènes d'illumination est donnée par l'*équation fondamentale du rendu*, introduite par Kajiya [Kaj86] :

$$I(x, x') = g(x, x') \left[\epsilon(x, x') + \int_S \rho(x, x', x'') I(x', x'') dx'' \right] . \tag{2.1}$$

Cette équation décrit l'intensité I de la lumière arrivant à un point x provenant d'un autre point x'. Cette intensité est fonction d'un facteur géométrique g. Ce facteur décrit quelle proportion de l'énergie quittant x' se rend à x. La lumière quittant x' pour se rendre à x est issue de deux phénomènes différents. Premièrement, il y a la lumière qui est émise ϵ par x' dans la direction de x. À cette lumière s'ajoute celle qui provient d'un troisième point x'' et est réfléchie en x' dans la direction de x. L'intensité $I(x', x'')$ de x'' vers x' doit être intégrée sur tous les points de toutes les surfaces S que contient l'environnement. La fonction ρ décrit comment la lumière est réfléchie en x' lorsqu'elle provient de x'' pour se rendre en x. Avec l'émission ϵ et l'intégration sur toutes les surfaces S, nous avons couvert tout l'environnement et connaissons exactement l'énergie qui arrive en x par x'.

L'équation fondamentale traite tous les échanges de lumière de l'environnement. Elle traite l'éclairage direct, c'est-à-dire la lumière qui va directement des sources vers les objets. Elle traite aussi l'éclairage indirect qui lui représente la lumière qui est réfléchie une ou plusieurs fois sur des objets avant d'éclairer celui qui nous intéresse.

Cette équation, bien que très intéressante au niveau théorique, l'est beaucoup moins en pratique. Pour résoudre l'équation, il faut tenir compte de tous les points de toutes les surfaces. Ceci donne un système d'équations intégrales simultanées qui n'a pas de solution analytique dans le cas général. Il est possible de calculer une approximation de la solution du système, mais les temps de calcul sont très souvent trop grands en pratique.

2.3.1 Modèle de réflexion de Phong

D'autres modèles simulent plus ou moins fidèlement l'équation fondamentale. Le modèle de Phong [Pho75] en est un qui simule exclusivement l'éclairage direct et qui est très populaire. Il est illustré à la figure 2.3. Ce modèle est souvent présenté comme

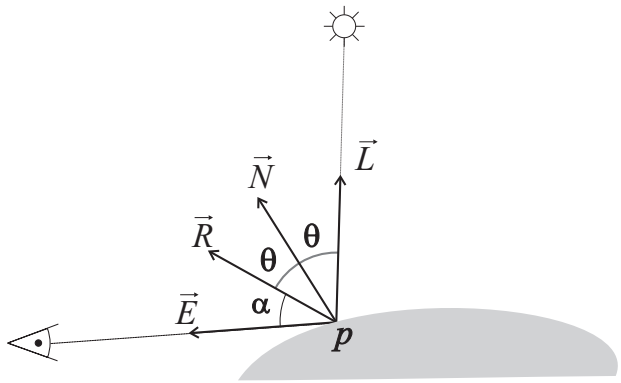


FIG. 2.3: Illustration du modèle de réflexion de Phong.

suit :

$$I_\lambda = \sum_{i=1}^{m} S_i f_{att_i} I_{\lambda_i} \left(k_d O_{d\lambda} (\vec{N} \cdot \vec{L_i}) + k_s O_{s\lambda} (\vec{R_i} \cdot \vec{E})^n \right) \tag{2.2}$$

où

$I =$ l'intensité lumineuse allant vers l'oeil à partir d'un point p d'une surface,
$\lambda =$ une certaine longueur d'onde (couleur monochromatique),
$i =$ la source lumineuse i,
$m =$ le nombre total de sources lumineuses,
$S =$ le facteur de visibilité de la source lumineuse,
$f_{att} =$ l'atténuation de la lumière provenant de la source lumineuse,
$I_i =$ l'intensité de la source lumineuse i,
$k_d =$ le coefficient de réflexion diffuse de la surface,
$O_d =$ la couleur de la réflexion diffuse de la surface,
$\vec{N} =$ la normale à la surface au point p,
$\vec{L} =$ le vecteur de p vers la lumière,
$k_s =$ le coefficient de réflexion spéculaire de la surface,
$O_s =$ la couleur de la réflexion spéculaire de la surface,
$\vec{R} =$ le vecteur $\vec{L}$ réfléchi (de façon miroir) en p par rapport à la normale $\vec{N}$,
$\vec{E} =$ le vecteur de p vers l'oeil,
$n =$ le coefficient de rugosité spéculaire de la surface.

Cette équation fait une approximation de l'équation fondamentale pour des surfaces qui ont des propriétés de réflexion diffuse et une réflexion spéculaire particulière. Regardons les différences de cette équation 2.2 par rapport à l'équation fondamentale 2.1. Nous calculons maintenant la lumière qui arrive à l'oeil en passant par le point p. Comme l'oeil est un point et que par définition p est visible, nous n'avons plus besoin du facteur géométrique. L'émission aussi est négligée car nous ne traitons maintenant que des surfaces qui n'émettent pas de lumière. L'intégrale est remplacée par une somme par rapport à toutes les sources lumineuses. La fonction de réflexion de surface ρ est remplacée par deux composantes de réflexion de surface. Une première compo-

sante traite la réflexion diffuse par approximation de surfaces mates (lambertiennes). La deuxième composante traite les réflexions spéculaires, imitant les surfaces plastifiées ou métalliques qui reflètent la lumière un peu comme un miroir. Comme nous pouvons le voir, le modèle de Phong ne traite que l'illumination venant directement des sources lumineuses. La lumière qui est réfléchie par les objets n'est pas prise en compte.

En général, toutes les longueurs d'onde λ ne sont pas traitées. Les intensités de lumières et les couleurs de surfaces sont décrites selon trois couleurs de base : le rouge, le vert et le bleu. Évidemment, ceci est encore une approximation car pour bien traduire les phénomènes de propagation de la lumière, il faudrait tenir compte de tout le spectre des couleurs.

Une autre simplification commune est l'utilisation de sources ponctuelles. L'éclairage produit par ces sources n'est évidemment pas réaliste, mais son traitement est simple. Comme nous l'avons vu plus haut, déterminer la visibilité d'une source ponctuelle par rapport à un point revient à tout simplement regarder si un objet bloque le rayon qui les unit. Pour des sources lumineuses étendues, la tâche est plus difficile car nous devons déterminer la proportion de la source qui est visible ou du moins en faire une approximation.

Chapitre 3

Travaux antérieurs

Dans ce chapitre, nous exposons les travaux qui visent l'accélération du rendu par des économies au niveau du calcul de l'apparence des surfaces. Nous commençons par les méthodes qui visent une réduction du temps passé aux calculs de visibilité. Nous voyons ensuite les techniques qui réduisent le temps de rendu en faisant des approximations, et nous terminons avec les méthodes hiérarchiques.

Le calcul de l'apparence d'une surface en un point peut assez facilement être découplé en deux tâches indépendantes. Une première partie du problème consiste à déterminer la visibilité des sources lumineuses. Dans le cas de sources ponctuelles, la décision est binaire. Pour des sources étendues par contre, il faut déterminer la proportion de la source qui est visible. La deuxième partie du travail est de calculer, selon la visibilité de la source et les propriétés de la surface, l'apparence de l'objet.

3.1 Zone d'influence des sources

Dans cette section, nous présentons des méthodes qui réduisent le temps de calcul en déterminant pour chaque source une zone d'influence. Si un point donné est à l'intérieur de la zone d'influence, la lumière doit être prise en compte pour le calcul de l'apparence en ce point. Par contre, pour toutes les lumières pour lesquelles le point est à l'extérieur de leur zone d'influence, aucun calcul n'est fait.

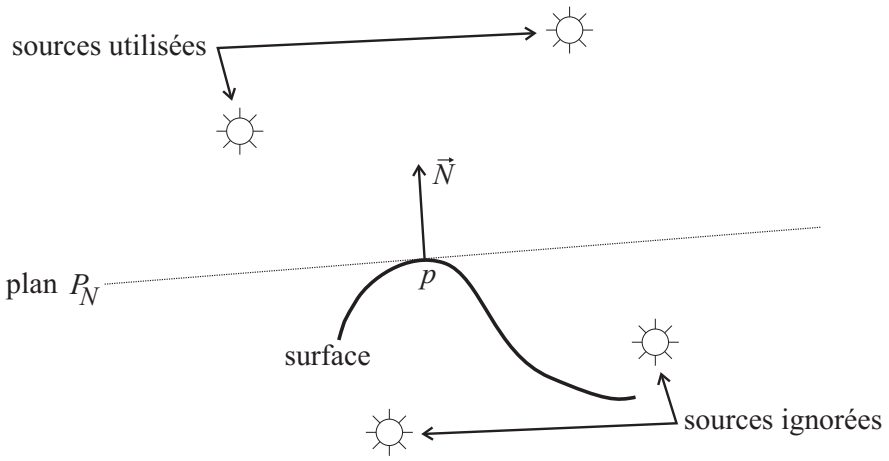


FIG. 3.1: Élimination du traitement de sources lumineuses en considérant la normale à la surface.

3.1.1 Normale à la surface

Plusieurs sources lumineuses peuvent ne pas être visibles d'un certain endroit. Pour un point d'une surface, toutes les lumières qui sont derrière le plan déterminé par la normale à la surface et le point lui-même sont invisibles. La figure 3.1 montre cette sélection des sources lumineuses. Cette méthode peut s'appliquer à tous les types d'objets et de sources lumineuses.

Dans le cas où l'objet est un polygone, la détermination de la visibilité des sources se fait pour toute la surface puisque tous les points du polygone déterminent le même plan de division entre lumières visibles et invisibles. Éliminer des sources lumineuses tôt dans le processus de calcul de l'apparence de la surface est très intéressant. Pour les sources éliminées, nous n'avons pas besoin de calculer la visibilité. La détermination de cette dernière étant très coûteuse, les économies faites sont grandes. Nous ne devons pas non plus évaluer l'apparence de la surface, processus qui est cependant généralement peu coûteux par rapport au calcul de visibilité. Le coût du calcul de la visibilité dépend de la complexité de la scène, contrairement au coût du calcul de l'apparence de la surface qui dépend du modèle de réflexion. Pour une distribution uniforme de sources lumineuses et d'objets, la moitié des sources seront éliminées par cette technique puisque le plan divise l'espace en deux régions.

Cette accélération n'a évidemment aucun désavantage car elle élimine des calculs qui de toute façon resteront vains.

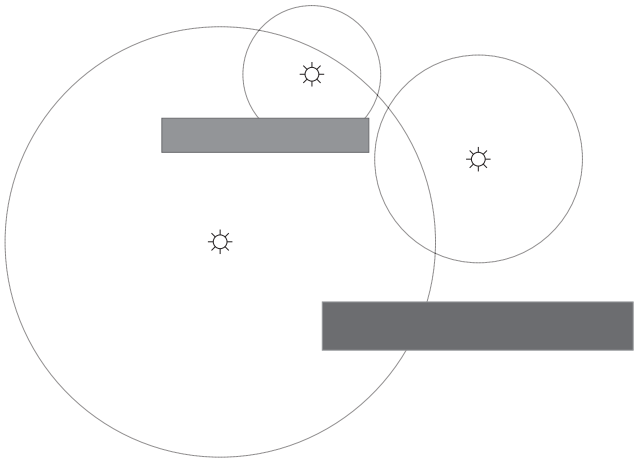


FIG. 3.2: Sphère d'influence déterminant si une source doit être prise en considération ou non.

Cette méthode est similaire au *back-face culling*. Ce dernier élimine les polygones qui font dos à l'oeil. Si nous avons affaire à des objets fermés, le *back-face culling* permet d'enlever des polygones qui sont invisibles parce que cachés par les autres polygones qui ferment l'objet.

3.1.2 Sphère d'influence

Une autre façon simple de réduire le temps de rendu est encore une fois d'éliminer les calculs pour des sources, mais cette fois en se basant sur leur proximité. Cette méthode peut s'appliquer à tous les types d'objets et de sources lumineuses. Elle est cependant moins bien adaptée à d'autres types de sources que ponctuelles ou sphériques.

Nous pouvons définir autour de chaque source, une sphère d'influence qui détermine la zone à l'intérieur de laquelle la source va éclairer. Nous assumons qu'à l'extérieur de la sphère, la contribution de la source peut être négligée [Ber86]. L'intensité de la lumière arrivant à un point varie selon le carré de la distance à la source lumineuse (pour une source ponctuelle). À une grande distance, l'intensité qui vient d'une source est donc très faible, d'où l'idée de limiter sa zone d'influence. Nous montrons à la figure 3.2 un exemple de sphères d'influence. Plus une lumière est intense, plus le rayon de sa sphère d'influence est grand. Encore une fois, nous éliminons tôt dans le calcul les contributions de certaines sources. Nous pourrions aussi déterminer la zone d'influence à l'aide d'un polyèdre ou de tout autre volume adéquat. Cependant, l'utilisation de la

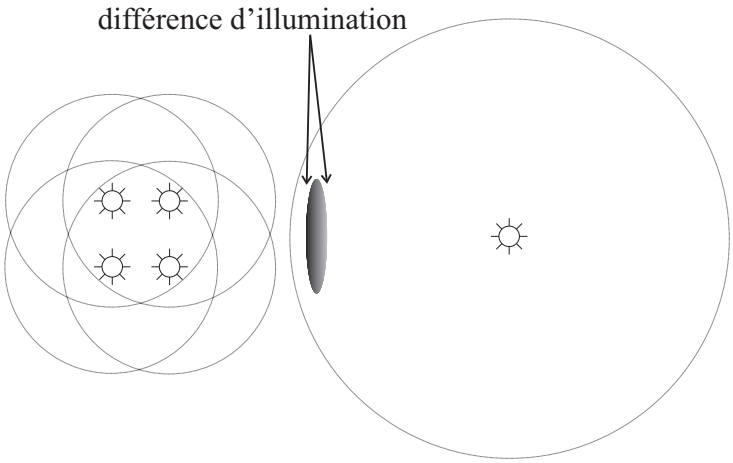


FIG. 3.3: Problème de la sphère d'influence. À droite nous avons une source d'intensité I et à gauche quatre sources d'intensités $I/4$. Nous voyons que les deux côtés de l'ellipse devraient avoir une illumination semblable. À cause de la sphère d'influence, aucune contribution n'est prise en compte sur le côté gauche.

sphère est particulièrement appropriée car elle permet de déterminer facilement si un point en fait partie ou non. Si la distance entre un point et la source est plus grande que le rayon de la sphère, le point est à l'extérieur de la sphère d'influence. Inversement, si la distance est plus petite ou égale au rayon, le point est à l'intérieur de la sphère d'influence et la contribution de la source doit être évaluée.

Bien que cette méthode soit intéressante pour éliminer les sources qui pourraient être peu importantes, la détermination des zones d'influence est difficile lorsqu'il y a plusieurs lumières qui éclairent les mêmes objets. Par exemple, une façon simple de déterminer la dimension de la sphère d'influence est de se baser sur la racine carrée de l'intensité de la source. Cependant, plusieurs sources d'une intensité totale équivalente à une autre source auront une illumination différente. L'intensité de chacune des sources du groupe étant plus petite, leurs sphères d'influence sont elles aussi plus petites. Si les sources sont dans un amas compact, l'illumination résultante sera différente de celle de la source plus intense, même si l'intensité totale est égale. La figure 3.3 illustre ce problème.

La méthode est donc surtout intéressante quand on peut facilement déterminer une zone en dehors de laquelle la contribution de la source est partout négligeable. En présence de beaucoup de sources, la distribution spatiale de ces dernières est très impor-

tante. Ne considérer que l'intensité d'une source pour déterminer sa sphère d'influence n'est pas suffisant.

3.2 Restriction du nombre de bloqueurs potentiels

Cette section présente des techniques qui réduisent la quantité de calcul à faire lors du tracé des rayons d'ombre. Ces techniques créent des structures qui permettent de trouver rapidement un sous-ensemble petit de bloqueurs potentiels parmi tous les objets de la scène.

3.2.1 Subdivision de l'espace

Les hiérarchies d'objets et les grilles, utilisées pour accélérer la détermination des objets qui sont touchés par les rayons de l'oeil, vont tout aussi bien accélérer la détermination de la visibilité des sources lumineuses. Elles s'appliquent encore à tous les types d'objets et de sources lumineuses.

Ces accélérations pour le tracé de rayon sont décrites dans le livre de Glassner [Gla89]. Pour les rayons d'ombre, nous n'avons pas besoin du bloqueur qui est le plus près de l'origine du rayon, mais d'un bloqueur quelconque. Dès qu'un objet est trouvé, la procédure peut le retourner immédiatement. Ceci n'est pas le cas pour les rayons provenant de l'oeil où on doit poursuivre les tests. Ces derniers sont continués jusqu'à ce que le rayon entre dans un *voxel* qui est plus éloigné que l'intersection la plus près que nous avons trouvée.

Le plus grand problème de cette méthode est la construction de la hiérarchie. Déterminer la taille des *voxels* et la subdivision hiérarchique est assez difficile, bien que des heuristiques assez fiables existent.

Cette technique est utilisée dans nos travaux avec un *octree* comme méthode de subdivision de l'espace.

3.2.2 *Shadow cache*

Une façon efficace de déterminer s'il y a occlusion ou non par rapport à une certaine source lumineuse est de tester en premier les objets qui ont de plus grandes chances de cacher la source. De cette façon, on réduit le temps passé à tester chacun des bloqueurs potentiels. Le *shadow cache*, introduit par Haines et Greenberg [HG86] dans leur article

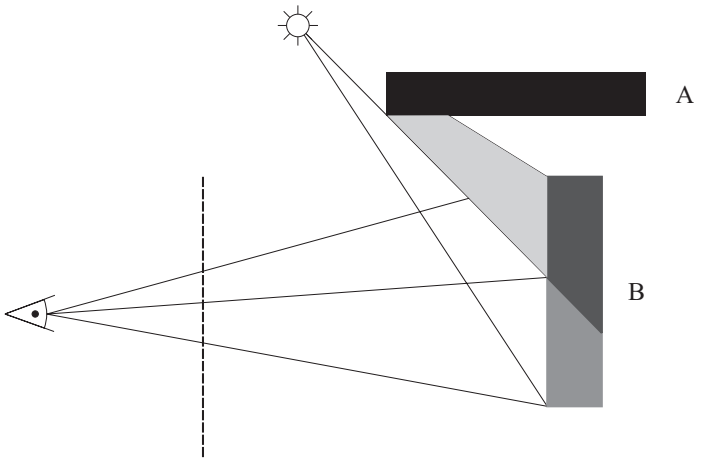


FIG. 3.4: Exemple de cohérence dont tire profit le *shadow cache*. Tous les rayons de l'oeil qui touchent la partie ombragée de l'objet B sont cachés par le même objet A.

sur le *light buffer*, est un excellent exemple de cette stratégie. Cet algorithme s'applique à tous les types d'objets et aux sources ponctuelles ou directionnelles.

Dans une scène typique, les occlusions des sources d'un pixel au suivant sont très fortement corrélées. La figure 3.4 montre le genre de cohérence dont tire profit le *shadow cache*. Pour chacune des sources lumineuses, nous gardons un pointeur vers l'objet qui nous a donné la dernière occlusion. Au pixel suivant, nous regardons tout d'abord si cet objet cache encore la source lumineuse. Si c'est le cas, nous arrêtons immédiatement le traitement pour cette source lumineuse.

Un problème de cette méthode est que la cohérence peut se retrouver au niveau des objets composés plutôt que des primitives. Pour un bloqueur issu d'une forte tessellation (beaucoup de petits bloqueurs), la méthode sera peu efficace. Une tentative pour régler ce problème est présentée par Pearce et Jevans [PJ91]. Tout d'abord, une subdivision de l'espace est appliquée à la scène. Plutôt que de garder seulement l'objet qui a produit la dernière occlusion, ils gardent de la même façon le *voxel* qui contenait cet objet. L'objet du *shadow cache* est testé en premier suivi des objets contenus dans le *voxel* conservé. Les accélérations de cette méthode sont cependant décevantes, elle n'est qu'un à six pour cent plus rapide que le *shadow cache* standard.

3.2.3 *Light buffer*

Le *light buffer* est une technique intéressante introduite par Haines et Greenberg [HG86]. Elle traite les sources ponctuelles et les objets quelconques.

Cette méthode construit autour de chaque source un cube où chaque face est divisée en une grille régulière. Une pyramide est déterminée par la source et chaque carré de la grille d'une face. Tous les objets qui intersectent cette pyramide sont associés à l'élément de grille correspondant. Quand vient le temps de déterminer la visibilité de la source pour un point d'une surface, nous déterminons l'élément de grille qui est intersecté par le rayon du point vers la source. Il ne reste plus qu'à tester les objets associés à l'élément de grille pour trouver la visibilité car eux seuls peuvent faire occlusion. Une bonne économie de temps est ainsi faite car les objets intéressants sont rapidement trouvés.

Certaines optimisations sont également faites. Les objets sont triés selon leur distance à la lumière, ce qui permet d'arrêter la recherche d'une occlusion quand les objets pouvant faire ombre sont en arrière du point par rapport à la lumière. Les listes d'objets peuvent aussi être tronquées dès qu'un objet fait ombre sur tout l'élément de la grille car les objets suivants seront nécessairement dans son ombre.

Malgré ses aspects positifs, le *light buffer* possède beaucoup de défauts. L'efficacité de l'algorithme dépend beaucoup de la résolution de la grille utilisée. Une résolution plus fine réduit les temps de calcul lors du rendu, mais demande plus de mémoire et un pré-calcul plus important. Une résolution trop faible entraînera une liste de candidats trop volumineuse. La pyramide formée de la source et de l'élément de grille est aussi un problème car plus on s'éloigne des objets, plus sa coupe transversale est grande, pouvant ainsi contenir beaucoup d'objets. La consommation mémoire est multipliée si la scène contient plusieurs sources car chacune d'elles possède sa structure de grille qui est déjà énorme.

3.2.4 Pyramide d'occlusion

La pyramide d'occlusion [CK92] est une autre méthode visant à trouver les objets plus propices à faire occlusion à une source lumineuse. Elle s'applique à des sources ponctuelles ou directionnelles et à des objets polygonaux ou ayant un contour simple (les sphères par exemple).

Pour chaque objet, une *pyramide* est formée. Cette pyramide a pour base la surface de l'objet et pour sommet une source lumineuse. Elle contient tous les objets qui peuvent faire une ombre sur la surface. De cette façon, ayant une intersection sur une surface, nous n'avons qu'à interroger sa pyramide pour trouver les bloqueurs potentiels. Cette stratégie est très similaire au *shaft culling* [HW91] utilisé en radiosité.

Pour accélérer la recherche d'un bloqueur dans la pyramide, la liste des objets est triée par rapport à un critère de *facteur de forme pour ombres*. Ce critère, apparenté au *facteur de forme* utilisé en radiosité, favorise les objets qui ont une aire plus grande, qui sont plus près de la source et qui sont perpendiculaires au vecteur allant du centre de masse de la surface vers la source lumineuse. En général, ce critère va placer les objets qui ont une plus grande chance de faire occlusion au début de la liste, et les autres à la fin. De cette manière, l'occlusion de la source est trouvée plus rapidement. Évidemment, pour un point qui n'est pas dans l'ombre, comme il n'y a pas occlusion, la liste doit encore être traversée complètement.

Certaines optimisations ont été faites à l'algorithme de base. Pour ne pas construire une pyramide qui ne sera jamais utilisée, les pyramides sont construites seulement pour les objets qui sont visibles. Cette méthode s'applique bien aux objets de taille moyenne, mais mal aux autres. Pour les objets trop petits, aucune pyramide ne sera construite car trop peu de pixels utiliseront cette information. Des objets trop gros risquent fort de mettre trop d'objets dans la pyramide. Dès qu'une pyramide contient trop d'objets, son information est jetée et une méthode classique est utilisée.

Une idée similaire est présentée par Woo [Woo93]. La pyramide est cependant inversée et utilisée pour accélérer le calcul de visibilité d'une source étendue. Pour un point où nous voulons connaître la visibilité d'une source polygonale, la pyramide formée du point et de la source est construite et utilisée pour échantillonner la visibilité.

Le problème majeur de la pyramide d'occlusion est sa consommation de mémoire. Pour chaque objet visible, une pyramide est construite pour chaque source lumineuse. Chaque pyramide contient un grand nombre de pointeurs vers les objets qui risquent de se répéter dans d'autres pyramides.

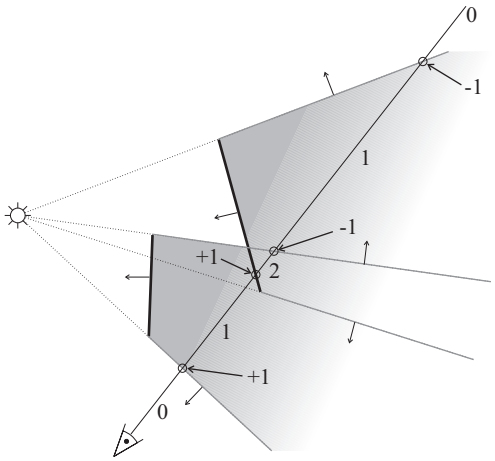


FIG. 3.5: Volume d'ombre déterminé par une source ponctuelle et un polygone. Il est à noter que les normales des polygones d'ombre pointent vers l'extérieur de la zone ombragée.

3.3 Détermination des ombres en pré-calcul

Dans cette section nous verrons des méthodes qui pré-calculent les zones d'ombre avant de faire le rendu. Au moment de calculer l'image, la détermination de la visibilité des sources se limite à interpréter l'information pré-calculée. Les calculs de visibilité ne dépendent plus du nombre d'objets, mais de la facilité avec laquelle il est possible de retrouver l'information à partir du pré-calcul.

Woo *et al.* [WPF90] présentent un résumé des méthodes de calcul des ombres dans lequel on retrouve la majorité des méthodes que nous venons de décrire. Cet article contient une description un peu plus détaillée et une liste assez exhaustive de références.

3.3.1 Volume d'ombre

La technique du volume d'ombre introduite par Crow [Cro77], utilise un pré-calcul pour que la détermination de la visibilité des sources soit plus rapide lors du rendu. Dans cette technique, tous les objets sont des polygones et les sources sont ponctuelles ou directionnelles.

Une source et un polygone déterminent une ombre qui est délimitée par le volume défini par le polygone lui-même ainsi que par la source et les arêtes du polygone. Un exemple en deux dimensions est présenté à la figure 3.5. Tous les polygones des différents volumes d'ombre sont ajoutés aux polygones de la scène. Ils sont cependant marqués

comme étant des polygones d'ombre qui ne font donc pas partie de l'affichage. Lors du rendu, quand nous traversons le polygone d'un volume d'ombre, le compteur de la visibilité de la source associée est modifié. Si le polygone nous fait face, le compteur est incrémenté et s'il nous fait dos, le compteur est décrémenté. Quand vient le temps de calculer la couleur d'un pixel, si le compteur est à zéro, la lumière est visible et s'il est plus grand que zéro, nous sommes dans l'ombre.

Déterminer la visibilité d'une source est très simple, mais coûteux car lors du rendu il faut intersecter tous les volumes d'ombre rencontrés. Le pré-calcul des volumes d'ombre est aussi relativement coûteux et multiplié par le nombre de sources présentes dans la scène. Là où la méthode devient encore moins intéressante, c'est dans sa consommation mémoire. Pour chaque arête de chaque polygone, un nouveau polygone d'ombre est ajouté. Pour que ce polygone ne s'étende pas jusqu'à l'infini, il est généralement limité à la boîte englobante de la scène. Le polygone d'ombre a donc au moins 4 sommets. Ce processus génère un minimum de trois polygones d'ombre si nous n'avons affaire qu'à des triangles. Nous venons donc d'au moins quadrupler le nombre de polygones qui doivent être en mémoire pour le rendu. Il faut également multiplier le tout par le nombre de sources. Cela fait beaucoup de polygones allongés qui doivent être intersectés lors du rendu.

Une méthode hybride [EK89] visant à calculer plus efficacement la visibilité a été développée. Cette méthode se base sur des critères pour utiliser les rayons d'ombre ou les volumes d'ombre quand une des deux méthodes est plus efficace. Bien qu'elle réduise les problèmes des intersections avec trop de volumes d'ombres, cette méthode garde la complexité du pré-calcul et les coûts mémoire de la méthode des volumes d'ombre.

Le test d'occlusion des *voxels* [WA90], basé sur l'idée des volumes d'ombre, vient réduire certains coûts associés à cette méthode. Les objets sont tout d'abord insérés dans une grille régulière. Plutôt que de conserver tous les volumes d'ombre, les *voxels* sont marqués par rapport à chaque source. Pour chaque source, le *voxel* possède un état : dans l'ombre, illuminé ou visibilité partielle. Lors du calcul de l'apparence en un point, si ce point est dans un *voxel* avec l'attribut ombre ou illuminé, il n'y a aucun calcul de visibilité à faire. Un rayon d'ombre est lancé pour les *voxels* qui ont une visibilité partielle. Ceci élimine les coûts mémoire des volumes d'ombre et élimine les calculs d'intersection de ces derniers lors du rendu. L'initialisation de la méthode reste

tout de même coûteuse avec des tests, pour chaque lumière, de tous les segments de tous les polygones.

Une autre méthode utilisant les arbres *BSP* se base sur le principe des volumes d'ombre [CF89, CF92]. Cette méthode a l'avantage de faire l'union des volumes d'ombre, ce qui accélère la détermination de la visibilité lorsque les volumes d'ombre de plusieurs objets se superposent. Le pré-traitement reste énorme, mais il y a un gain au niveau de la consommation mémoire. Cette consommation reste tout de même proportionnelle au nombre de sources présentes.

3.3.2 Subdivision des surfaces

Une accélération assez similaire aux volumes d'ombre est présentée par Atherton *et al.* [AWG78]. Elle utilise une procédure de *clipping* pour déterminer quelle partie d'un polygone est visible de la source. Encore une fois, cette méthode est conçue pour les objets polygonaux et les sources ponctuelles ou directionnelles.

La méthode est basée sur le *clipping* comme procédure de détermination des surfaces visibles lors de l'affichage. Le même *clipping* effectué lors de l'affichage est fait par rapport à la source lumineuse pour déterminer les polygones qui sont dans l'ombre. Les polygones invisibles sont marqués comme étant dans l'ombre et ceux qui sont visibles sont marqués comme étant illuminés. Pour les polygones qui ne sont pas complètement visibles, le contour que l'ombre projette sur eux est déterminé. Cette information est ajoutée comme propriété de la surface. Cette subtilité est importante, car si nous avions plutôt coupé le polygone en sa partie éclairée et sa partie dans l'ombre, nous aurions eu beaucoup plus de polygones à traiter lors du rendu. La visibilité des ombres n'aurait pas demandé de calculs, mais la détermination des surfaces visibles se serait faite avec un plus grand nombre de polygones. Nous aurions aussi pu avoir des problèmes de fentes dans les objets à cause des erreurs de notation en virgule flottante. De plus, il aurait pu y avoir des polygones concaves, ce qui est un problème pour beaucoup de procédures de rendu.

Cet algorithme est intéressant, mais demande la projection d'énormément de polygones. Dans le pire des cas, chaque polygone doit être projeté sur chaque autre polygone. La consommation mémoire augmente également car chaque polygone peut contenir un nombre arbitraire de segments déterminant le contour de l'ombre qui l'affecte. Comme

chaque source fera des ombres différentes, l'information supplémentaire sera multipliée par le nombre de sources.

3.3.3 *Shadow map*

Une autre technique pour réduire le temps du calcul d'ombre est l'utilisation du *shadow map* [Wil78]. L'idée se base sur l'utilisation du *tampon-z* [Cat74] comme méthode de détermination des surfaces visibles. Cette méthode s'applique à toutes sortes d'objets qui peuvent être traités par le tampon-z, mais ne supporte correctement que les sources ponctuelles ou directionnelles.

Une première image de la scène est faite à partir de la source. De cette image, nous ne calculons que les valeurs du tampon-z. C'est ce tableau de profondeurs [1] que nous appelons *shadow map*. Une deuxième image est calculée, mais cette fois-ci à partir de l'oeil. Pour chaque point où nous voulons calculer l'apparence de la surface, nous transformons les coordonnées du point dans le système de coordonnées de la source lumineuse et comparons la profondeur avec celle qui se trouve dans le *shadow map* pour le pixel correspondant. Si la profondeur du point est plus grande que la valeur du *shadow map*, un objet se trouve devant et cache la lumière. Inversement, si la profondeur est égale à celle du *shadow map*, l'objet est celui qui était visible de la lumière et il se trouve illuminé.

Un intérêt de cette méthode est que la même technique est utilisée pour le rendu de l'image et pour déterminer la visibilité des sources. Un avantage de ceci est que le tampon-z est souvent implanté en matériel, ce qui accélère largement la méthode.

Par contre, il y a beaucoup de problèmes associés à l'utilisation du *shadow map*. La correspondance d'un point en trois dimensions avec le *shadow map* n'est pas exacte. Presque toujours, le point ne projettera pas au centre d'un pixel. La détermination de la profondeur qui correspond à un point non centré entraînera une première forme d'imprécision.

Comme les profondeurs sont gardées sur un nombre fini de bits, le problème de quantification se pose. La représentation discrète des profondeurs dans le tampon-z entraîne de forts problèmes d'aliassage. Nous devons ajouter un certain ε aux profondeurs

[1] Les méthodes de projections standard utilisant un système droitier placent les objets dans une pyramide définie par l'axe négatif des z. Pour éviter toute confusion, nous parlons plutôt de profondeur. Une profondeur plus grande implique un objet plus éloigné de l'oeil.

du *shadow map* pour être certains qu'un point de la surface ne se retrouve pas caché par lui-même. Pour qu'un point soit visible il faut alors que sa profondeur soit plus petite ou égale à celle du *shadow map*.

Pour réduire les problèmes d'aliassage, nous devons augmenter la résolution du *shadow map* et peut-être augmenter le nombre de bits utilisés pour quantifier les profondeurs. Ceci ne résout pas tous les problèmes et ne garantit en rien que l'image résultante sera exempte de problèmes d'aliassage dus à la dimension finie des pixels du *shadow map*. De plus, les augmentations de la résolution et du nombre de bits demandent plus de mémoire ainsi qu'un temps de calcul accru. Un nouveau *shadow map* doit être construit pour chaque source.

Un dernier inconvénient de la méthode est que si la source lumineuse est dans le volume de vue, un seul *shadow map* ne peut couvrir tout ce qui est visible. Il faut donc calculer plusieurs *shadow maps* dans des directions différentes pour couvrir toute la scène. Typiquement, un cube d'images est calculé, multipliant par six les coûts autant en mémoire qu'en temps de construction.

Quelque part entre le *light buffer* et le *shadow map*, il y a le *ZZ-Buffer* [SS89, SS90]. Avec cette méthode, le problème de quantification des profondeurs est éliminé. Plutôt que de garder une profondeur, c'est un intervalle de valeur de profondeurs qui est conservé avec les objets qu'il contient. Le nombre d'objets conservés étant généralement petit, le calcul d'une visibilité correcte avec ces derniers est relativement peu coûteux. La demande en mémoire est supérieure à celle du *shadow map*, et encore multipliée par le nombre de sources.

3.4 Méthodes par approximation

Nous venons de voir plusieurs méthodes visant à réduire les temps de calcul de l'apparence des surfaces. La plupart d'entre elles sont exactes, sauf pour celles qui souffrent d'aliassage ou de stabilité par rapport à la notation en virgule flottante. La méthode des sphères d'influence est aussi inexacte car on a nécessairement une erreur présente. Elle peut cependant être moins visible à cause de la discrétisation des couleurs dans l'image finale.

Nous présentons maintenant des méthodes qui visent à faire une approximation de l'apparence de la surface en échantillonnant un minimum de sources. La grande

difficulté à laquelle s'attaquent ces méthodes est de trouver rapidement quelles sources sont importantes pour avoir une approximation de qualité.

3.4.1 Calcul adaptatif des rayons d'ombre

Ward [War91] présente une méthode simple, élégante et générale pour réduire les coûts dus aux tests de visibilité des sources. Dans le calcul de l'apparence de la surface, l'évaluation de la visibilité de la source n'est faite que lorsque le reste du calcul montre que la source peut apporter une contribution appréciable. La technique s'applique à tous les types de sources et d'objets.

La méthode de Ward repose sur l'idée de tester en premier les sources qui peuvent créer une différence importante d'illumination et d'arrêter les calculs quand la contribution potentielle des sources restantes est négligeable. Donc, pour tout point où nous désirons calculer l'apparence de la surface, nous devons calculer la contribution potentielle de chaque source. Pour des surfaces éclairées par des sources ponctuelles, cette contribution potentielle pourrait s'exprimer comme suit :

$$I_{approx} = I_i f_{att_i} \left(k_d O_d (\vec{N} \cdot \vec{L}_i) + k_s O_s (\vec{R}_i \cdot \vec{E})^n \right) .$$

Dans cette équation, I_i est l'intensité de la source pour laquelle nous voulons calculer la contribution potentielle et les autres paramètres sont ceux du modèle standard de Phong.

Nous voyons tout de suite un problème à cette approximation, par l'absence du terme géométrique (S de l'équation 2.2 du modèle de Phong en page 13) qui détermine la visibilité de la source. Évidemment, comme c'est justement cette visibilité que nous voulons éviter de calculer, nous devons nous contenter de cette approximation. Une fois toutes les contributions potentielles calculées, nous trions les sources par ordre décroissant de contribution. Une fois le tri terminé, nous testons la visibilité des sources une à une et l'incorporons à la contribution potentielle pour obtenir la contribution exacte de la source. La contribution exacte des sources est calculée au fur et à mesure que nous traversons la liste et elle est accumulée dans une somme Som. Avant de tester une nouvelle source, nous regardons si la contribution potentielle r des n prochaines sources est plus petite qu'une certaine proportion p de Som. Les paramètres n et p sont définis par l'usager et lui permettent de contrôler la qualité des images générées. Dès que $Som\ p > r(n)$, la visibilité des sources restantes n'est pas calculée.

Pour ne pas biaiser le résultat en incluant ou ignorant systématiquement les sources restantes, une approximation de leur contribution est ajoutée. Chaque source conserve son nombre de tests de visibilité ainsi que le nombre de tests de visibilité qui ont conclu que la source était visible. À partir de ces informations, le pourcentage de visibilité de la source peut être calculé. De même, à chaque pixel, le nombre de tests de visibilité de source ainsi que le nombre de sources visibles sont conservés. Le pourcentage de visibilité des sources pour ce pixel peut ainsi être estimé. Pour chaque nouveau pixel, le nombre de tests de visibilité et le nombre de sources visibles sont remis à zéro avant de commencer à parcourir la liste. Pour chacune des sources qui ne sont pas échantillonnées, leur contribution potentielle est multipliée par leur pourcentage de visibilité respectif. La somme de ces contributions potentielles pondérées est multipliée par le pourcentage de visibilité du pixel courant. De cette façon, une approximation raisonnable de la contribution des sources dont la visibilité n'est pas échantillonnée est calculée.

Dans les scènes où quelques sources sont beaucoup plus importantes que les autres pour la plupart des endroits visibles, cette méthode s'avérera très puissante. Elle permettra de réduire considérablement les tests de visibilité en ne testant les sources que lorsque c'est nécessaire. Cette méthode s'apparente un peu aux sphères d'influence, mais elle s'adapte aux surfaces et à l'éclairage différent dans chaque portion de la scène.

Cependant, pour les scènes qui ont un grand nombre de sources, le temps passé à trier les contributions des sources peut devenir considérable. La méthode fonctionne donc mieux pour les scènes où le nombre d'objets est beaucoup plus grand que le nombre de sources. Les calculs de visibilité sont alors bien plus coûteux que le coût du tri. De plus, si toutes les sources ont des contributions semblables, la liste devra être traversée presque en entier avant que les sources restantes représentent une proportion assez petite pour ne pas tester leur visibilité.

Cette méthode est celle qui résout le mieux le problème que nous attaquons. Nous ferons une étude plus approfondie de ses résultats face aux nôtres au chapitre 5.

3.4.2 Monte Carlo pour sources multiples

Dans leur article sur les techniques *Monte Carlo*, Shirley *et al.* [SWZ96] exposent une méthode pour faire un échantillonnage plus efficace dans les scènes avec plusieurs

centaines de sources lumineuses. Cette méthode s'applique à tous les types d'objets et de sources.

Commençons tout d'abord par une brève explication des méthodes Monte Carlo. Pour chaque pixel de l'image, une grande quantité de rayons sont envoyés dans la scène pour calculer une approximation de l'éclairage direct et indirect. Typiquement, quelques centaines de rayons sont tracés pour chaque pixel. Un rayon voyage donc de surface en surface jusqu'à ce qu'il atteigne une source lumineuse à partir de laquelle sera déterminée la contribution du rayon au pixel. Cette contribution est calculée en tenant compte des différentes surfaces que le rayon a rencontrées avant d'arriver à la source lumineuse. Plusieurs méthodes différentes guident les rayons dans la scène pour essayer d'aller chercher le plus efficacement la lumière qui parvient au pixel.

Un grave problème des méthodes Monte Carlo est la grande quantité de bruit présent dans l'image finale. C'est pour réduire ce bruit qu'un si grand nombre d'échantillons sont requis. Comme les rayons vont un peu n'importe où dans la scène, il doit y en avoir un grand nombre pour que leur moyenne donne une illumination correcte. Le grand problème du Monte Carlo est de trouver quelle surface ou source échantillonner.

Pour les sources, une probabilité d'être échantillonnée doit être associée à chacune d'entre elles. Déterminer cette probabilité est difficile et Shirley *et al.* simplifient cette tâche en divisant les sources lumineuses en deux groupes : les sources importantes et les sources tamisées. Ils déterminent alors des probabilités d'échantillonner chaque source importante et une probabilité d'échantillonner une des sources tamisées. Comme pour la méthode de Ward [War91], on fait ici l'hypothèse que quelques sources seront importantes et que les autres seront négligeables pour tout endroit de la scène. La détermination des sources importantes et tamisées se fait ici une seule fois en pré-calcul. Une sphère d'influence est associée à chaque source. La scène est divisée spatialement et pour chaque élément de subdivision qui intersecte la sphère d'influence d'une source, cette dernière sera ajoutée à la liste des sources importantes. Les problèmes de la sphère d'influence seront encore présents ici. De plus, nous ne tenons pas compte des propriétés de surface comme c'était le cas pour Ward, ni de la visibilité des sources.

Bien qu'assez simpliste, cette façon de séparer les sources importantes des sources tamisées peut donner d'assez bons résultats quand le nombre d'échantillons par pixel est grand car nous échantillonnons alors un nombre raisonnable de sources tamisées.

Les avantages de cette méthode sont sa simplicité et sa généralité. Elle traite la simulation complète de l'éclairage plutôt que simplement l'éclairage direct. Ceci donne des images d'un grand réalisme.

Deux problèmes de la méthode sont inhérents à la philosophie Monte Carlo : le bruit dans les images et le temps de calcul énorme. Un autre problème est la division en sources importantes et sources tamisées. Comme il n'y a qu'une seule probabilité associée aux sources tamisées, aucune d'elles n'est favorisée dans le choix de celles qui seront échantillonnées. Un groupe de sources plus importantes, bien que faisant partie des sources tamisées, pourrait donc être échantillonné de manière aléatoire, ajoutant beaucoup de bruit dans l'image finale.

Nous avons vu des méthodes qui calculent approximativement l'apparence de la surface. Notre approche est similaire car nous utilisons des approximations pour éviter d'échantillonner la totalité des sources lumineuses.

3.5 Méthodes hiérarchiques

Les méthodes que nous présentons dans les sections précédentes utilisent parfois une hiérarchie d'objets pour le tracé de rayon. Cette hiérarchisation se limite aux objets, laissant les sources à part. Plusieurs font des pré-calculs pour accélérer la détermination de la visibilité des sources, mais ces informations sont gardées à un même niveau. Nous voyons maintenant la hiérarchisation utilisée en radiosité.

3.5.1 Radiosité

Le monde de la radiosité est très vaste. Deux livres expliquent bien les méthodes de radiosité, l'un par Sillion et Puech [SP94] et l'autre par Cohen et Wallace [CW93]. Les techniques de représentation hiérarchique que nous exposons ici sont en partie décrites dans ces livres et dans quelques articles qui traitent plus particulièrement de l'agrégation [Sil94, SAG94]. La radiosité s'applique surtout à des objets et sources polygonaux. La détermination du facteur de forme rend plus complexe l'extension de la méthode à des sources ou objets plus généraux.

La radiosité traite les échanges de lumière qui ont lieu entre tous les objets et lumières. En général, les transferts traités sont ceux résultant de réflexions diffuses de la lumière. Comme calculer analytiquement les échanges entre surfaces est extrêmement

difficile, les calculs se font entre éléments de surface. Quand l'échange n'est pas bien représenté en utilisant les surfaces entières, celles-ci sont subdivisées et les échanges sont calculés au niveau des sous-éléments. En radiosité, comme les surfaces et les sources lumineuses sont traitées de la même façon, il est plus facile de construire une hiérarchie de la scène. La subdivision des surfaces en sous-éléments, donne déjà une hiérarchisation [HSA91]. Il est aussi possible d'aller plus loin en regroupant les surfaces en agrégats [Sil94, SAG94]. Les échanges peuvent maintenant être faits entre des agrégats, des surfaces ou des sous-éléments de surface selon le résultat du critère d'erreur utilisé.

Les images de radiosité sont calculées en deux phases. Premièrement, les échanges de lumière entre les surfaces sont calculés au niveau de précision requis. Une fois le calcul des échanges terminé, l'apparence de chaque surface est connue. Il ne reste plus qu'à afficher les objets sans se préoccuper de la visibilité des sources puisque l'effet de la lumière venant de celles-ci est déjà connu. Une fois le pré-calcul de l'échange de lumière terminé, l'affichage des scènes de radiosité est donc très rapide et indépendant du point de vue.

Les méthodes de radiosité génèrent des images de très bonne qualité car elles font la simulation de l'éclairage pour des surfaces diffuses. Les ombres correctes et l'éclairage indirect se retrouvent dans ces images.

Malgré ses qualités, la radiosité souffre de gros problèmes de consommation mémoire à cause de la subdivision des surfaces. Une seule surface peut être subdivisée maintes et maintes fois pour bien représenter les différentes discontinuités d'ombres qui l'affectent. Le calcul des échanges de lumière entre les surfaces est également coûteux. Les méthodes qui adaptent la radiosité à d'autres types de surface augmentent de beaucoup les demandes en mémoire et en temps de calcul.

Les critères d'erreur qui déterminent le niveau de hiérarchie à utiliser dans les échanges d'énergie ne sont pas bien adaptés à des scènes qui contiendraient beaucoup de sources lumineuses. Le critère BF, par exemple, demandera d'aller trop bas dans la hiérarchie. Ce critère, comme son nom l'indique, utilise la radiosité B et le facteur de forme F entre les deux surfaces pour déterminer si l'échange doit être fait sur des éléments de surface plus petits. À cause de leur forte radiosité B, les sources lumineuses entraîneront une grande subdivision des objets auxquels elles envoient leur radiosité. L'utilisation de beaucoup de sources résultera en beaucoup d'échanges à des niveaux

fins de la hiérarchie.

L'agrégation de sources en radiosité n'a pas encore été traitée. Une représentation capturant l'illumination de plusieurs sources permettrait de traiter plus simplement l'effet cumulé de cet ensemble de sources.

Chapitre 4

Hiérarchie de lumières

Comme nous venons de le voir, les méthodes de radiosité tirent un grand profit de la hiérarchisation des échanges de lumière. Dans les méthodes de tracé de rayon, très peu d'efforts dans ce sens ont été faits. En se basant sur le succès de la hiérarchisation en radiosité, on peut être optimiste quant au succès d'une telle stratégie pour le tracé de rayon.

Il faut cependant prendre en considération que le tracé de rayon est différent de la radiosité. Le tracé de rayon est une méthode très efficace pour représenter l'éclairage direct. Comme nous l'avons vu, la radiosité traite les échanges de lumière au niveau des éléments de surface. Représenter les discontinuités dans ce cadre demande une grande subdivision des surfaces. Les discontinuités dans l'illumination sont principalement dues aux frontières d'ombres. La subdivision pour les discontinuités fortes entraîne beaucoup plus de calculs à des niveaux fins de la hiérarchie. La radiosité représente bien l'éclairage indirect car celui-ci est souvent très continu. Dès qu'il y a beaucoup de discontinuités, la radiosité demande beaucoup plus de temps pour obtenir des résultats d'aussi bonne qualité.

Le tracé de rayon se distingue de la radiosité par le fait qu'il fasse une distinction entre les objets et les sources lumineuses. De plus, le tracé de rayon calcule l'illumination pour un point de vue fixe, contrairement aux méthodes de radiosité. Ces dernières étant basées sur un pré-calcul indépendant du point de vue, elles permettent de naviguer en temps réel dans un environnement.

Il serait intéressant, en tracé de rayon, de pouvoir regrouper les sources lumineuses en agrégats. Lorsque cette représentation serait adéquate, nous pourrions utili-

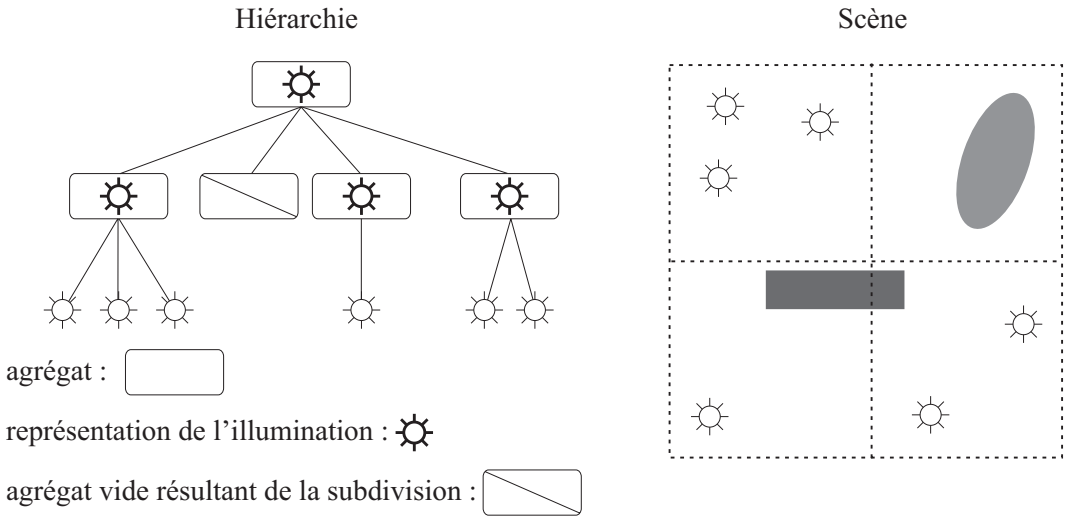


FIG. 4.1: Exemple de représentation hiérarchique de lumières.

ser l'agrégat pour calculer l'illumination plutôt que d'effectuer le calcul avec chacune des sources. Intuitivement, lorsque l'agrégat est suffisamment petit par rapport à la distance le séparant du point où nous désirons calculer l'apparence de la surface, remplacer toutes les sources par une seule devrait introduire très peu d'erreur et réduire le temps de calcul. Nous développerons à partir de notre hiérarchisation des critères basés sur cette idée et qui permettront de quantifier l'erreur commise par l'utilisation de la hiérarchie.

Ce chapitre décrit la hiérarchie de lumières que nous utilisons et le prochain introduit les critères d'erreur qui déterminent comment utiliser la hiérarchie.

4.1 Description de la hiérarchie

Nous voulons une hiérarchie d'agrégats dans laquelle les sources lumineuses sont contenues aux feuilles. Chaque noeud interne de la hiérarchie représente une agrégation d'autres agrégats. Il possède une représentation compacte de l'illumination des sources qu'il regroupe. Un noeud peut donc être un agrégat de sources lumineuses ou un agrégat contenant d'autres agrégats. La figure 4.1 montre un exemple d'une telle hiérarchie. Lors du rendu, nous devons choisir quels noeuds sont utilisés pour le calcul de l'illumination. Nous choisissons les noeuds qui sont les plus près de la racine et qui n'introduisent pas une trop grande erreur dans le calcul de l'apparence.

Notre hiérarchie de sources est donc différente des hiérarchies normalement utilisées

en tracé de rayon. Habituellement, les hiérarchies servent à accélérer la recherche d'une intersection entre un objet et un rayon. Ces hiérarchies ne possèdent pas vraiment de représentation intermédiaire des objets qu'elles contiennent. Notre hiérarchie sert plutôt à représenter à différents niveaux de précision l'illumination directe contenue dans la scène.

Pour construire une hiérarchie représentant efficacement des agrégations d'objets, une méthode de type *bottom-up* est souvent le meilleur choix. Par contre, les critères de regroupement des objets en agrégats sont souvent difficiles à définir et la forme des agrégats résultants peut être arbitraire. Une construction de type *top-down* est souvent plus simple à réaliser, mais peut donner une moins bonne représentation. Sa construction est aussi plus simple avec une complexité de $O(n \log n)$ contre au moins $O(n^2)$ pour une construction *bottom-up*.

Nous utilisons une méthode de type *top-down* parce qu'elle est simple et efficace. Cette méthode sert à démontrer l'utilité de l'approche et il est toujours possible de remplacer cette méthode par une autre donnant une meilleure agrégation.

Nous utilisons l'*octree* car c'est une structure de données simple, très connue et éprouvée. L'*octree* permet de très simplement regrouper les sources lumineuses en agrégats. Évidemment, un groupe naturel de sources pourrait être scindé entre deux ou plusieurs *voxels* puisque la division se fait uniquement selon des axes et des dimensions prédéfinies.

4.2 Deux hiérarchies

Comme nous en avons déjà parlé, la hiérarchie que nous construisons pour les sources lumineuses est assez différente des hiérarchies utilisées pour le tracé de rayon. Nous avons donc décidé de construire deux hiérarchies distinctes pour les objets et pour les sources.

Nous avons choisi de faire ainsi pour plusieurs raisons. Tout d'abord, l'utilisation faite des deux hiérarchies n'est pas la même. Dans un cas nous y lançons des rayons pour trouver rapidement les intersections. Dans l'autre, nous utilisons les représentations approximatives des *voxels* pour limiter les calculs d'illumination.

Les critères de construction sont par conséquent différents. Dans les deux cas, nous subdivisons le *voxel* s'il contient plus d'un certain nombre d'items. Le nombre maxi-

mal d'items pour un *voxel* de sources et pour un *voxel* d'objets n'est cependant pas nécessairement le même. Une valeur efficace pour les objets ne le sera peut-être pas pour les sources. De plus, un *voxel* qui contient un nombre raisonnable de sources et d'objets peut se retrouver subdivisé parce que le total des items ne respecte pas le critère.

La répartition des objets et des sources dans la scène pourrait également poser problème. Beaucoup de sources dans une région donnée pourraient amener une trop grande subdivision par rapport aux objets. L'utilisation de cette hiérarchie pour faire le tracé de rayon ne serait plus efficace dans cette région. La flamme d'une bougie simulée par plusieurs dizaines de sources est un bon exemple de cette situation. Le contraire est aussi vrai : quelques sources pourraient être subdivisées sur beaucoup de niveaux à cause d'une abondance d'objets. Il y aurait donc beaucoup de niveaux qui résumeraient pratiquement la même information d'illumination. La répartition spatiale des objets et des sources est un autre problème. La présence d'un ciel étoilé au-dessus d'une maison amènera une subdivision beaucoup trop grande du ciel pour faire un tracé de rayon efficace.

D'un autre côté, mettre les objets et les sources dans deux structures différentes n'a pas que des avantages. L'information de l'occlusion entre les sources, les *voxels* de sources et les objets, est dispersée dans les deux hiérarchies. Nous devrons donc trouver comment faire interagir les deux hiérarchies pour obtenir toute l'information nécessaire pour faire le rendu.

4.3 Construction

Maintenant que nous connaissons suffisamment les raisons et les motivations derrière l'utilisation d'un *octree* de lumières, regardons comment nous le construisons et comment nous représentons l'illumination de chaque *voxel*.

La construction de la hiérarchie débute par la détermination de la boîte englobante minimale alignée sur les axes qui contient toutes les sources. Cette boîte sera la racine de notre *octree* et nous y ajoutons toutes les sources. Quand un *voxel* possède plus d'un certain nombre de sources lumineuses, il est subdivisé. Diminuer le nombre maximal d'items par *voxel* augmente le nombre de *voxels* vides au dernier niveau de subdivision. La hiérarchie possède alors plus de niveaux. Inversement, augmenter le nombre maximal

d'items diminue le nombre de *voxels* vides et compacte l'information sur moins de niveaux dans la hiérarchie.

Il faut trouver un juste équilibre entre une hiérarchie avec beaucoup de niveaux et une très compacte. Dans une hiérarchie avec beaucoup de niveaux, nous retrouvons une large palette de représentations de l'illumination, mais elle nécessite plus de mémoire et son parcours est plus long. Une hiérarchie compacte ne donne que très peu de représentations de l'illumination. Nous sommes alors trop souvent obligés d'utiliser les sources elles-mêmes car la représentation de l'agrégat est trop grossière.

D'après les tests que nous avons faits, un maximum de sept sources par *voxel* donne de bons résultats. Si la répartition des sources est uniforme, un *voxel* subdivisé engendre des enfants ayant en moyenne une source. Nous exposerons aussi plus loin des raisons de ne pas avoir un nombre maximal d'items trop petit. Le nombre maximal de sources n'est cependant pas très important car le nombre de niveaux dans la hiérarchie est proportionel à $\log_8(\textit{nombre de sources})$ plutôt qu'au nombre maximal de sources. C'est pour cette raison que nous ne nous attardons pas plus longtemps sur le choix judicieux du nombre maximal de sources.

Une fois que nous avons construit notre *octree* de lumières, nous devons avoir à chaque *voxel* une représentation de l'information d'illumination des sources qu'il contient. Nous avons choisi d'utiliser une source virtuelle pour cette représentation. Cette source appartient au *voxel* seulement et ne se retrouve pas dans la scène. La source virtuelle est elle aussi une source ponctuelle et possède une intensité ainsi qu'une position. L'intensité de la source virtuelle est tout simplement la somme des intensités des sources que le *voxel* contient. Cette intensité est gardée sous forme d'un triplet *RVB*. Pour ce qui est de la position de la source virtuelle, elle pourrait se trouver à la moyenne des positions des sources, mais ceci ne tiendrait pas compte de l'intensité relative des sources. La source virtuelle doit se rapprocher des sources les plus intenses. Nous positionnons la source virtuelle à la moyenne des positions des sources, pondérée par leur intensité relative.

Nous avons aussi besoin d'une information sur la distribution des sources dans un *voxel*. Nous savons déjà que toutes les sources sont à l'intérieur du *voxel*. La boîte du *voxel* ne donne cependant pas d'information sur la répartition des sources puisque sa taille est fixe. Nous gardons donc la boîte englobante minimale alignée sur les axes qui

Attributs d'une source	
Intensité	I_i
Position	P_i

Information d'un *voxel*		
Intensité	$I_v = \sum_{i=1}^{m} I_i$	
Position	$P_v = \frac{\sum_{i=1}^{m} P_i I_i}{\sum I_i}$	
Dispersion	$B_{min} =$ BoîteAlignée(	$(\min(Px_i), \min(Py_i), \min(Pz_i))$, $(\max(Px_i), \max(Py_i), \max(Pz_i))$)

TAB. 4.1: Description des attributs d'une source ainsi que les informations gardées dans un *voxel* de sources. L'intensité est conservée comme un triplet RVB. La pondération de la position se fait en considérant la dimension des vecteurs RVB des intensités I_i.

contient toutes les sources. Si les sources sont dans un petit amas compact, notre boîte minimale reflétera assez bien cette distribution. La boîte minimale alignée sur les axes n'est pas une solution idéale, mais elle donne d'assez bons résultats et est extrêmement simple à calculer. Nous discutons d'améliorations à cette représentation au chapitre 7.

La table 4.1 résume les informations gardées dans un *voxel* ainsi que les attributs d'une source lumineuse.

Chapitre 5

Illumination sans occlusion

Nous venons de décrire la hiérarchie de lumières, sa construction et l'information qu'elle renferme. Nous allons maintenant regarder comment nous pouvons utiliser cette hiérarchie pour accélérer le rendu. Dans ce chapitre, nous nous restreignons aux scènes où il n'y a pas d'ombres. Ceci simplifie les explications en ignorant le calcul de la visibilité d'un *voxel* de lumière. Nous traitons tout d'abord l'utilisation de la hiérarchie pour des surfaces diffuses, puis pour des surfaces spéculaires. Dans le prochain chapitre nous présentons une façon d'intégrer le calcul de la visibilité des *voxels* de sources et les difficultés que cela représente.

Nous devons développer une stratégie pour trouver rapidement le niveau de la hiérarchie des lumières qui représente bien l'illumination par rapport à un point d'une surface et aux propriétés de réflexion de cette dernière. Il faut en même temps essayer d'utiliser les niveaux qui sont les plus élevés dans la hiérarchie car ce sont eux qui regroupent le plus de lumières et donc qui offrent une plus importante réduction des calculs.

Pour nous assurer que l'erreur introduite par l'utilisation de la hiérarchie n'est pas trop grande, nous baserons notre choix du niveau sur des critères d'erreur. Ces critères garantissent que l'erreur sur l'illumination ne dépasse pas une certaine borne fixée par l'usager. Notre approche est basée sur ces critères d'erreur.

Dans le but d'alléger un peu la notation, les coefficients de réflexion k_d et k_s ont été fusionnés avec les couleurs O_d et O_s pour ne donner que deux coefficients. Ces nouveaux coefficients k_d et k_s sont des triplets RVB équivalents à la multiplication des coefficients avec leur couleur respective. Plutôt que de dériver des équations pour

les trois couleurs de base, les intensités des sources et des *voxels* de sources (I_i, I_v) ainsi que les couleurs diffuses et spéculaires (k_d, k_s) sont utilisées directement dans les dérivations. Ces "vecteurs" de couleur ne sont pas normalisés. Ceci résulte en une certaine incohérence vu le produit de deux vecteurs RVB de même dimension. Ce produit est donc interprété comme le produit terme à terme de deux vecteurs V_1 et V_2, et est noté V_1V_2. Par exemple, pour deux vecteurs ligne de dimension trois, $V_1 = (v_{11}, v_{12}, v_{13})$ et $V_2 = (v_{21}, v_{22}, v_{23})$, $V_1V_2 = (v_{11}v_{21}, v_{12}v_{22}, v_{13}v_{23})$. De même, $V_1 > V_2$ signifie que $v_{11} > v_{21}$, $v_{12} > v_{22}$, et $v_{13} > v_{23}$.

Pour simplifier la compréhension du développement des critères d'erreur, nous commençons par définir un critère d'erreur pour des surfaces diffuses, puis pour des surfaces spéculaires.

5.1 Surfaces diffuses

Nous développons, dans cette section, une borne d'erreur sur l'effet de l'utilisation de la hiérarchie pour le calcul de l'apparence de surfaces diffuses. Dans un premier temps, nous introduisons quelques définitions et hypothèses que nous utilisons plus tard pour élaborer la borne d'erreur.

5.1.1 Hypothèses et définitions pour les surfaces diffuses

À la figure 5.1 nous voyons les différentes variables auxquelles nous faisons référence dans les sections qui suivent. Nous assumons que, pour tout point visible de l'oeil, aucune source lumineuse n'est obstruée. Les indices i font référence à une source en particulier tandis qu'en l'absence d'un indice, nous assumons qu'il s'agit d'une valeur associée à la source virtuelle. [1] L'angle θ est celui utilisé pour le calcul de la réflexion diffuse (lambertienne) selon le modèle de Phong.

Nous définissons les relations suivantes :

$$\cos(\theta) = \vec{N} \cdot \vec{L}$$

$$\cos(\theta_i) = \vec{N} \cdot \vec{L}_i$$

$$\Delta d_i = d_i - d$$

[1] La notation I_v est une exception à cette règle. Le v est utilisé ici pour signifier que l'intensité est celle de la source virtuelle.

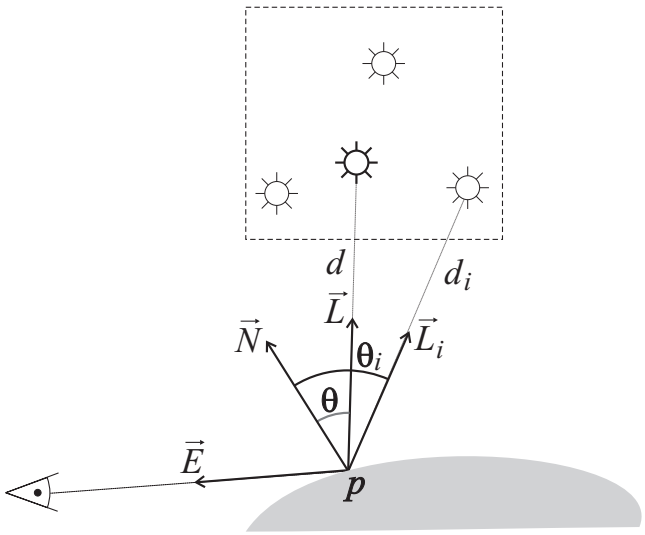


FIG. 5.1: Illustration des différentes valeurs utilisées pour définir la borne d'erreur diffuse.

$$\Delta\theta_i = \theta_i - \theta$$

$$\Delta d_{max} \geq \max(|\Delta d_i|)$$

$$\Delta\theta_{max} \geq \max(|\Delta\theta_i|)\,.$$

$$0 \leq \Delta d_{max} < d$$

Nous savons que :

$$(0,0,0) \leq k_d$$

$$(0,0,0) < I_i$$

$$0 \leq \theta \leq \pi/2$$

$$0 \leq \theta_i \leq \pi/2$$

$$0 \leq \cos(\theta) \leq 1$$

$$0 \leq \cos(\theta_i) \leq 1\,.$$

Nous devons trouver des valeurs pour Δd_{max} et $\Delta\theta_{max}$ qui respectent les définitions que nous venons de donner. La figure 5.2 montre en deux dimensions les bornes maximales de $|\Delta d_i|$ et $|\Delta\theta_i|$ que nous utilisons. Nous avons donc :

$$\Delta d_{max} = \text{diag}$$

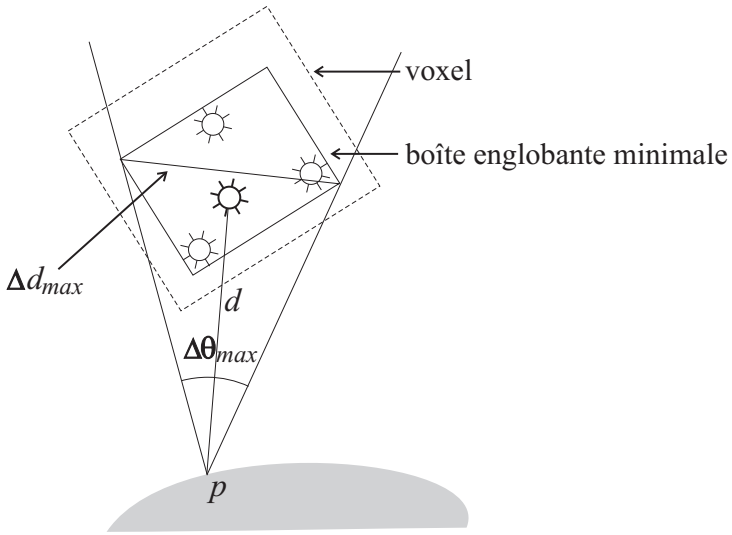


FIG. 5.2: Erreurs maximales des valeurs d et θ dues à l'utilisation de la hiérarchie.

$$\Delta\theta_{max} = \arctan\left(\frac{\Delta d_{max}}{d - \Delta d_{max}}\right)$$

où diag est la plus grande diagonale de la boîte englobante minimale des sources lumineuses.

5.1.2 Description de la borne diffuse

Maintenant nous voulons trouver une borne sur l'erreur causée par l'utilisation de la hiérarchie. Nous adaptons l'équation du modèle de Phong (équation 2.2 de la page 13) à notre problème particulier. Nos scènes sont constituées de m sources ponctuelles dont l'atténuation de la lumière diminue selon le carré de la distance. Les surfaces de ces scènes sont parfaitement diffuses et aucune d'elles ne fait d'ombres visibles. Selon ces conditions, l'équation devient :

$$I = \sum_{i=1}^{m} I_i \frac{1}{d_i^2} k_d \cos(\theta_i) \,. \tag{5.1}$$

Faire l'approximation du calcul de l'apparence de la surface en utilisant une source virtuelle nous donne l'équation suivante :

$$I_{approx} = \left(\sum_{i=1}^{m} I_i\right) \frac{1}{d^2} k_d \cos(\theta) \,. \tag{5.2}$$

Comme la somme des intensités est déjà calculée dans l'information du *voxel* (I_v), il ne nous reste qu'à calculer d et θ, puis évaluer l'approximation. Les économies faites proviennent de ne pas avoir à évaluer la somme ni calculer tous les d_i et tous les

$\cos(\theta_i)$. Le *voxel* doit contenir au moins deux sources pour qu'il vaille la peine d'utiliser la méthode hiérarchique.

L'erreur de l'utilisation de l'intensité approximative plutôt que la vraie intensité s'exprime simplement comme :

$$\Delta I = I - I_{approx} \, .$$

Nous cherchons une fonction ΔI_{sup} qui nous donne une borne supérieure sur l'erreur ΔI :

$$|\Delta I| \leq \Delta I_{sup} = f(I_v, k_d, d, \Delta d_{max}, \theta, \Delta\theta_{max}) \, .$$

Les paramètres de cette fonction sont calculés à partir de l'information de la source virtuelle et de celle de la surface au point p.

Notons que la fonction que nous cherchons doit être rapide à évaluer. Si le résultat de cette fonction est plus petit que l'erreur que nous tolérons, nous n'avons pas besoin de calculer l'apparence avec toutes les sources, mais seulement avec la source virtuelle. Nous devons donc calculer ΔI_{sup}, puis peut-être I_{approx}. Si l'évaluation de ΔI_{sup} prend plus de temps que d'évaluer l'apparence avec toutes les sources, utiliser la méthode hiérarchique sera plus lent. Il faut donc que l'évaluation de la fonction ne prenne pas beaucoup plus de temps que l'évaluation de l'apparence pour une seule source. En nous basant sur le temps d'évaluation de la fonction, nous pourrons déterminer combien de sources il doit y avoir au minimum dans un *voxel* pour que l'utilisation de la méthode hiérarchique soit efficace.

Comme ΔI est composé d'une soustraction, nous devons décomposer le problème en deux parties pour avoir une borne sur la valeur absolue. Dans un premier temps, si I est plus grand que l'approximation, la borne s'exprime comme $\Delta I = I - I_{approx}$. Il est aussi possible que l'approximation soit plus grande que la valeur exacte de l'illumination et alors la borne s'exprime comme $\Delta I = I_{approx} - I$.

Évaluer chacune de ces deux parties équivaut à utiliser l'illumination minimale I_{min} et maximale I_{max} du *voxel*. La valeur la plus grande entre $I_{approx} - I_{min}$ et $I_{max} - I_{approx}$ est la borne sur l'erreur ΔI_{sup} que nous cherchons. Nous utilisons cette stratégie plutôt que de calculer les deux parties de la valeur absolue.

5.1.3 Illumination minimale et maximale

Nous expliquons tout d'abord quelques petites astuces que nous utilisons pour minimiser ou maximiser certains termes de nos équations.

Tout d'abord, si $0 < d$ et $0 < \varepsilon$, alors

$$\frac{|x|}{d} > \frac{|x|}{d+\varepsilon}\,. \tag{5.3}$$

Sachant que $\frac{d}{d\theta}\cos(\theta) = -\sin(\theta)$, nous avons $-1 \leq \frac{d}{d\theta}\cos(\theta) \leq 1$. Dans notre cas particulier, comme $0 \leq \theta \leq \pi/2$ nous avons plutôt $-1 \leq \frac{d}{d\theta}\cos(\theta) \leq 0$. Partant d'un angle ϕ pour lequel nous connaissons la valeur de $\cos(\phi)$, nous nous déplaçons de $\Delta\phi$ le long de la courbe. En utilisant simplement les bornes sur les dérivés, nous pouvons borner comme suit la valeur du cosinus à cette nouvelle position :

$$\begin{aligned}
\cos(\phi) - 1\Delta\phi \leq \cos(\phi+\Delta\phi) \leq \cos(\phi) + 0\Delta\phi \quad \text{si} \quad \Delta\phi \geq 0\,,\\
\cos(\phi) + 0\Delta\phi \leq \cos(\phi+\Delta\phi) \leq \cos(\phi) - 1\Delta\phi \quad \text{si} \quad \Delta\phi < 0\,.
\end{aligned}$$

Pour un $\Delta\phi$ quelconque nous avons donc :

$$\cos(\phi) - |\Delta\phi| \leq \cos(\phi+\Delta\phi) \leq \cos(\phi) + |\Delta\phi| = \cos(\phi)\,. \tag{5.4}$$

Une propriété du cosinus nous permettra d'avoir des illuminations minimale et maximale encore plus serrées autour de la valeur exacte. Nous savons que :

$$0 \leq \cos(\theta_i) \leq 1\,.$$

Donc,

$$\cos(\theta_i) = \min(1, \cos(\theta_i)) = \max(0, \cos(\theta_i))\,. \tag{5.5}$$

Trouvons maintenant l'illumination minimale en partant de l'équation 5.1.

$$\begin{aligned}
I &= \sum_{i=1}^{m} I_i \frac{1}{d_i^2} k_d \cos(\theta_i) \\
&= k_d \sum_{i=1}^{m} I_i \frac{1}{(d+\Delta d_i)^2} \cos(\theta + \Delta\theta_i) \\
&\geq k_d \sum_{i=1}^{m} I_i \frac{1}{(d+\Delta d_{max})^2} \cos(\theta + \Delta\theta_i) && \text{par 5.3} \\
&\geq k_d \sum_{i=1}^{m} I_i \frac{1}{(d+\Delta d_{max})^2} \max(0, \cos(\theta + \Delta\theta_i)) && \text{par 5.5} \\
&\geq k_d \sum_{i=1}^{m} I_i \frac{1}{(d+\Delta d_{max})^2} \max(0, \cos(\theta) - |\Delta\theta_i|) && \text{par 5.4} \\
&\geq k_d \sum_{i=1}^{m} I_i \frac{1}{(d+\Delta d_{max})^2} \max(0, \cos(\theta) - \Delta\theta_{max}) \\
&\geq k_d I_v \frac{\max(0, \cos(\theta) - \Delta\theta_{max})}{(d+\Delta d_{max})^2}\,.
\end{aligned}$$

L'illumination maximale est dérivée de la même façon.

$$\begin{aligned}
I &= \sum_{i=1}^{m} I_i \frac{1}{d_i^2} k_d \cos(\theta_i) \\
&= k_d \sum_{i=1}^{m} I_i \frac{1}{(d+\Delta d_i)^2} \cos(\theta + \Delta\theta_i) \\
&\leq k_d \sum_{i=1}^{m} I_i \frac{1}{(d-\Delta d_{max})^2} \cos(\theta + \Delta\theta_i) && \text{par 5.3} \\
&\leq k_d \sum_{i=1}^{m} I_i \frac{1}{(d-\Delta d_{max})^2} \min(1, \cos(\theta + \Delta\theta_i)) && \text{par 5.5} \\
&\leq k_d \sum_{i=1}^{m} I_i \frac{1}{(d-\Delta d_{max})^2} \min(1, \cos(\theta) + |\Delta\theta_i|) && \text{par 5.4} \\
&\leq k_d \sum_{i=1}^{m} I_i \frac{1}{(d-\Delta d_{max})^2} \min(1, \cos(\theta) + \Delta\theta_{max}) \\
&\leq k_d I_v \frac{\min(1, \cos(\theta) + \Delta\theta_{max})}{(d-\Delta d_{max})^2} .
\end{aligned}$$

Nous avons donc deux bornes pour la réflexion diffuse en remplaçant m sources par une source virtuelle :

$$I_{min} = k_d I_v \frac{\max(0, \cos(\theta) - \Delta\theta_{max})}{(d+\Delta d_{max})^2} \tag{5.6}$$

et

$$I_{max} = k_d I_v \frac{\min(1, \cos(\theta) + \Delta\theta_{max})}{(d-\Delta d_{max})^2} . \tag{5.7}$$

5.1.4 Borne d'erreur pour les surfaces diffuses

À partir des équations approximative 5.2, minimale 5.6 et maximale 5.7, nous trouvons la borne sur l'erreur de l'utilisation de la hiérarchie pour des surfaces diffuses :

$$\begin{aligned}
\Delta I_{sup} = k_d I_v \max(\quad & \frac{\cos(\theta)}{d^2} - \frac{\max(0, \cos(\theta) - \Delta\theta_{max})}{(d+\Delta d_{max})^2}, \\
& \frac{\min(1, \cos(\theta) + \Delta\theta_{max})}{(d-\Delta d_{max})^2} - \frac{\cos(\theta)}{d^2} \quad) .
\end{aligned} \tag{5.8}$$

La borne est environ deux à trois fois plus longue à évaluer que l'illumination par rapport à une seule source. Comme nous l'avons déjà dit, une fois que le test de la borne est positif, il faut évaluer l'illumination de la source virtuelle. Nous devons donc avoir un minimum de trois à quatre sources dans un *voxel* pour que l'utilisation de la hiérarchie réduise le temps de calcul.

L'utilisation de la borne se fait après l'étape de détermination du point d'intersection du rayon de l'oeil avec un objet. Une fois cette intersection trouvée, nous appelons la procédure d'illumination hiérarchique montrée à la figure 5.3. Nous remarquons que

l'accélération basée sur la normale à la surface est utilisée pour éviter des calculs sur les *voxels* qui sont entièrement du mauvais côté de la normale. Il est à noter que lorsque le seuil diffus S_{diffus} est respecté, le *voxel* n'est pas directement utilisé, mais plutôt ajouté à une liste de *voxels*. L'utilisation directe du *voxel* n'est pas possible car notre borne donne l'erreur commise par le remplacement des sources contenues dans un *voxel* par la source virtuelle de ce dernier. Cette erreur est différente de l'erreur commise lors du remplacement des sources contenues dans plusieurs *voxels* par les sources virtuelles correspondantes. Quand un *voxel* ne respecte pas la borne, nous considérons tous ses enfants ce qui nous donne éventuellement l'utilisation de plusieurs sources virtuelles. Il faut donc s'assurer que l'erreur totale du remplacement des sources de tous les *voxels* par leurs sources virtuelles respectives ne depasse pas le seuil S_{diffus}. Une fois la traversée de la hiérarchie terminée, la liste des *voxels* respectant le seuil est examinée et si la somme des erreurs de chacun des *voxels* dépasse le seuil, les *voxels* entraînant les plus grandes valeurs d'erreur sont traités à un niveau plus bas de la hiérarchie. Ceci est répété jusqu'à ce que la somme des erreurs soit plus petite que le seuil. Les sources virtuelles des *voxels* restant dans la liste sont alors utilisées pour calculer la contribution de ces *voxels* qui est ajoutée à l'illumination déjà calculée.

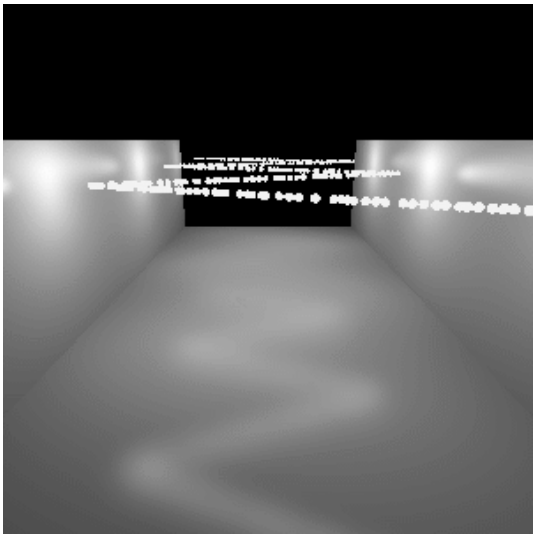
5.1.5 Description des scènes de test

Avant de discuter les résultats de la borne diffuse, nous décrivons nos scènes de tests. Les figures 5.4, 5.5, 5.6 et 5.7 montrent les quatre configurations de base que nous utilisons. Pour chaque configuration, nous faisons varier le nombre de sources lumineuses. Pour que l'intensité globale des images n'augmente pas, nous diminuons l'intensité de chaque source à mesure que nous augmentons leur nombre. Tous les calculs concernant une configuration ont été réalisés sur un même ordinateur. Les comparaisons de temps de calcul pour cette configuration sont donc correctes. Des comparaisons de temps de calcul d'une configuration par rapport à une autre ne seraient cependant pas valables car ils sont relatifs à des ordinateurs de vitesse différente. Les images de nos tests ont une résolution de 100 par 100 pixels.

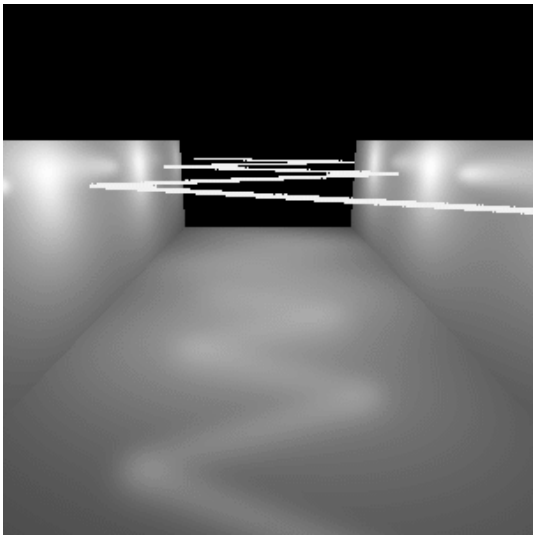
```
IlluminationHiérarchique( Point p, Voxel v )
{
    if v.Vide()
        return( 0 )

    if EstComplètementDuCôtéOpposé( v, p, NormaleÀLaSurface( p ) )
        return( 0 )

    if v.NombreLumières() ≤ 3
        return( IlluminationNormale( p, v.sources ) )

    Évaluer I_v, k_d, d, Δd_max, θ, Δθ_max à partir de p et v.

    if d_max > d
        // Nous ne pouvons évaluer notre borne si d_max > d.
        return( IlluminationNiveauInférieur( p, v ) )

    erreur = BorneDiffuse( I_v, k_d, d, Δd_max, θ, Δθ_max)

    // Comparaison avec le seuil d'erreur diffus
    if erreur < S_diffus
        AjoutListeDiffus( v, erreur )
        return( 0 )
    else
        return( IlluminationNiveauInférieur( p, v ) )
}

IlluminationNiveauInférieur( Point p, Voxel v )
{
    if v.Subdivisé()
        ∀ v.voxelEnfant
            IlluminationHiérarchique( p, v.voxelEnfant )
    else
        IlluminationNormale( p, v.sources )
}
```

FIG. 5.3: Algorithme de calcul de l'illumination à l'aide de la hiérarchie de lumières pour des surfaces diffuses.

Guirlandes 1
nombre de sources = 61



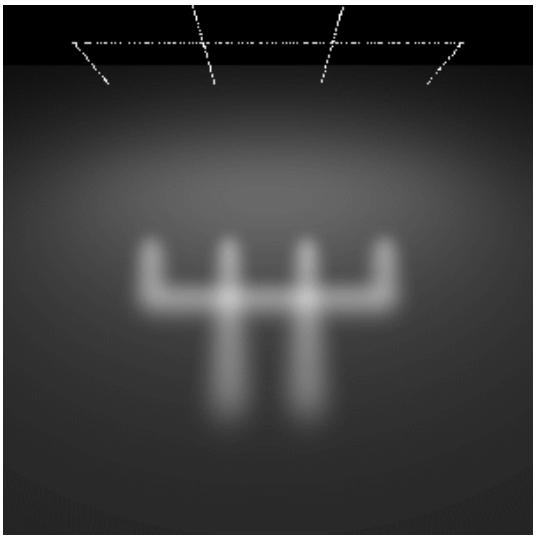
Guirlandes 2
nombre de sources = 261



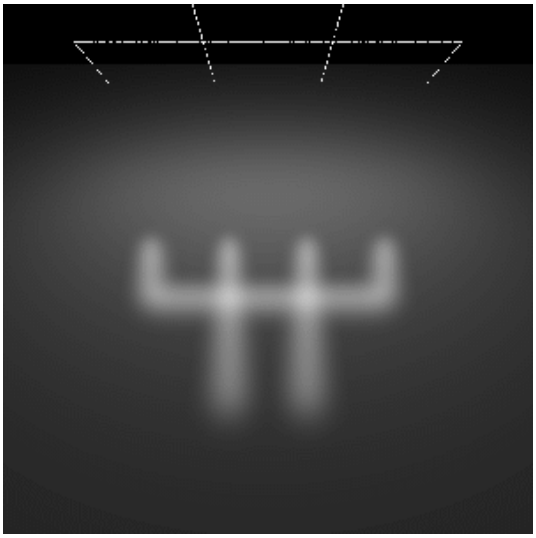
Guirlandes 3
nombre de sources = 1056

FIG. 5.4: Description des scènes de guirlandes de lumières. (nombre d'objets = 3, $k_d = (0.6, 0.6, 0.6)$, $k_s = (4, 4, 4)$, $n = 200$) Ces scènes sont composées de trois polygones formant un "canal". Des sources sont ajoutées le long de cinq segments reliant les deux côtés du "canal".

Spéculaire 1
nombre de sources = 62

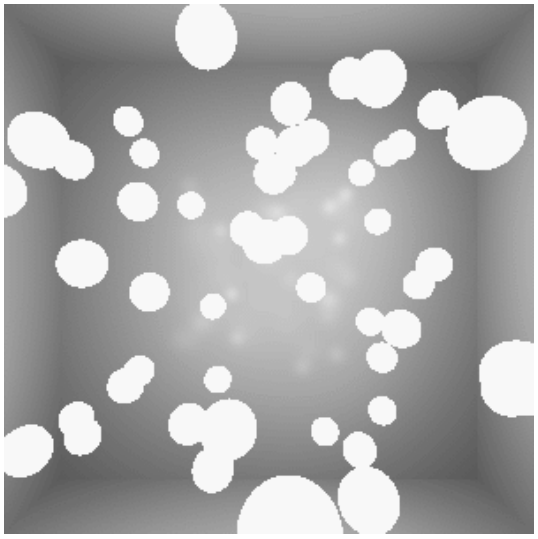


Spéculaire 2
nombre de sources = 257

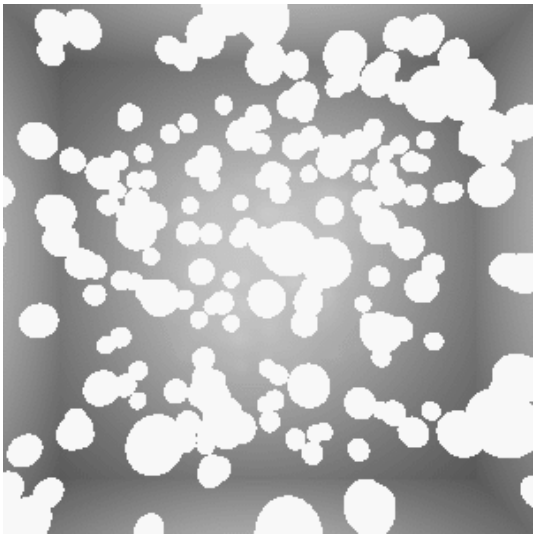


Spéculaire 3
nombre de sources = 1029

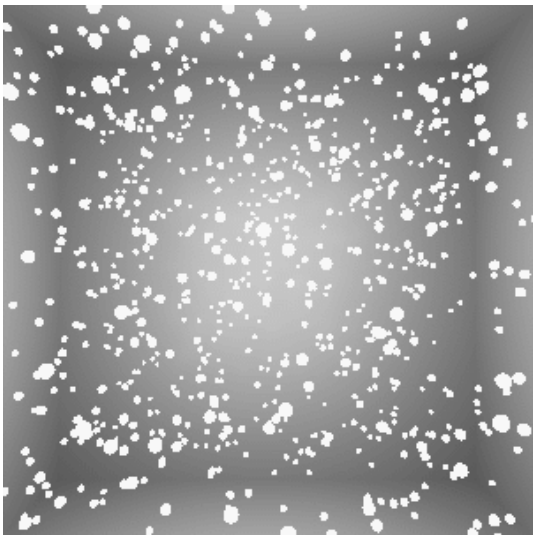
FIG. 5.5: Description des scènes spéculaires. (nombre d'objets = 1, k_d = (0.1, 0.1, 0.1), k_s = (4, 4, 4), n = 200) Ces scènes sont composées d'un seul polygone illuminé par des sources placées de façon à bien voir leur réflexion spéculaire sur le polygone.



Distribution 1
nombre de sources = 64
k_s = (1, 1, 1)

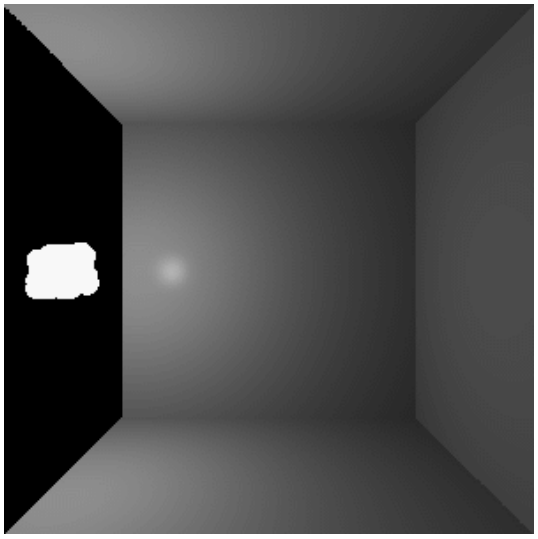


Distribution 2
nombre de sources = 256
k_s = (1.2, 1.2, 1.2)

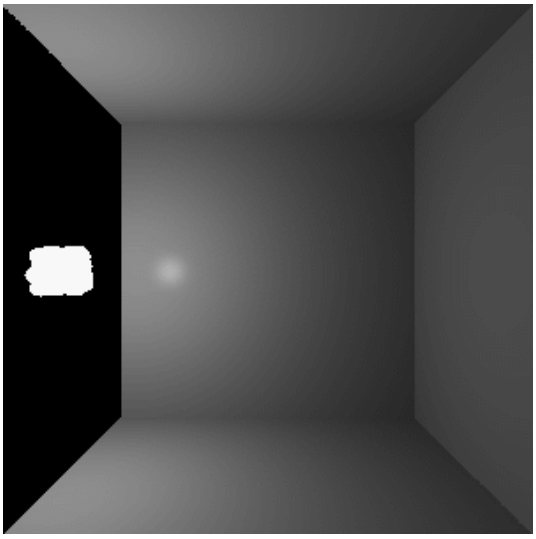


Distribution 3
nombre de sources = 1024
k_s = (1.4, 1.4, 1.4)

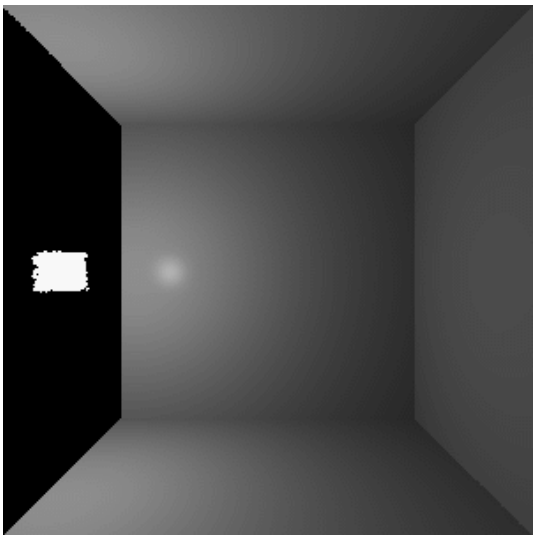
FIG. 5.6: Description des scènes de sources distribuées. (nombre d'objets = 6, k_d = (0.65, 0.65, 0.65), n = 200) Dans ces scènes, des sources sont distribuées aléatoirement dans un cube. L'observateur fait dos à une des faces. Les sources lumineuses sont affichées en sphères pour donner une bonne idée de leur distribution dans la scène. Lors des tests, le rendu des sources lumineuses n'est pas fait pour ne pas cacher les surfaces du cube. Le coefficient spéculaire est légèrement augmenté en même temps que le nombre de sources. Ceci était nécessaire pour que les réflexions spéculaires d'un plus grand nombre de sources de moindre intensité restent bien visibles.



Agrégat 1
nombre de sources = 64



Agrégat 2
nombre de sources = 256



Agrégat 3
nombre de sources = 1024

FIG. 5.7: Description des scènes d'agrégat de sources. (nombre d'objets = 5, $k_d = (1, 1, 1)$, $k_s = (1, 1, 1)$, $n = 200$) Ces scènes sont composées d'un amas relativement compact de sources (à gauche) qui éclairent un cube dont une des faces est enlevée (face de gauche). Nous présentons ici une vue de côté de la scène. Lors des tests, l'observateur est à la gauche, juste en arrière des sources dont le rendu n'est pas fait à ce moment pour ne pas obstruer la vue des surfaces du cube.

5.1.6 Résultats de la borne diffuse

Nous présentons dans cette section les résultats de l'utilisation de la borne que nous venons de développer. Nous comparons ces résultats avec le tracé de rayon standard et avec le calcul adaptatif des rayons d'ombre. Même si les scènes sont exemptes d'occlusions au niveau des rayons d'ombre, nous lançons tout de même ces rayons. Ceci donne une idée des gains qui pourraient résulter de l'utilisation de notre méthode pour des scènes générales. Pour la méthode hiérarchique un rayon d'ombre est lancé vers les sources virtuelles lorsqu'elles sont utilisées. Nous avons utilisé un seuil $S_{diffus} = 0.01$.

La table 5.1 résume les résultats.

Nous décrivons maintenant les différents champs de cette table :

TR seul Temps en secondes du calcul du tracé des rayons seulement, c'est-à-dire que tous rayons sont tracés, mais que l'illumination n'est pas calculée. Ceci permet de voir le coût de l'illumination par rapport au coût du tracé des rayons.

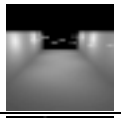
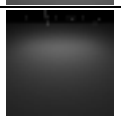
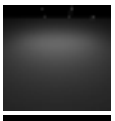
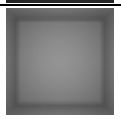
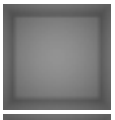
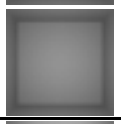
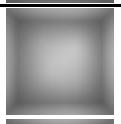
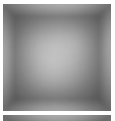
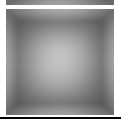
TR Temps en secondes du calcul du rendu par tracé de rayon standard.

Ward Pourcentage du temps de rendu avec calcul adaptatif des rayons d'ombre [War91] par rapport au tracé de rayon standard. Nous utilisons une implantation légèrement modifiée de la méthode du calcul adaptatif des rayons d'ombre. Avec cette modification, nous pouvons garantir une borne absolue sur l'erreur résultant de l'utilisation de la méthode. Nous arrêtons les tests quand la somme des contributions potentielles maximales est plus petite que $S_{diffus} = 0.01$. La comparaison avec notre méthode est ainsi équitable.

HL Pourcentage du temps de rendu avec la méthode hiérarchique par rapport au tracé de rayon standard. Les temps de rendus qui sont plus longs que le tracé de rayon standard sont soulignés en **caractères gras**. Lorsque la hiérarchie n'est pas utilisée parce que le critère d'erreur n'est pas respecté une seule fois pour toute l'image, le caractère † est ajouté devant le temps de rendu.

err. max Erreur maximale à un pixel. Cette erreur est la valeur maximale de la somme de l'erreur d'un pixel pour R, V et B. Les couleurs RVB sont entières et entre 0 et 255.

err. pixel Erreur moyenne par pixel. Cette erreur est la somme des erreurs pour tous les pixels (somme de R, V et B entre 0 et 255) divisée par le nombre de pixels.

Scène	TR seul.	TR	Ward	HL	err.max	err.pixel
Guirlande 1 61 sources 	10.7	11.1	73.0%	91.9%	3	0.001
Guirlande 2 261 sources 	85.0	87.4	71.1%	73.6%	3	0.000
Guirlande 3 1056 sources 	541.2	551.7	69.3%	42.7%	3	0.000
Spéculaire 1 62 sources 	30.1	34.0	67.6%	**120.0**%	3	0.002
Spéculaire 2 257 sources 	293.5	231.2	66.7%	45.6%	3	0.001
Spéculaire 3 1029 sources 	1335.9	1375.2	63.7%	17.0%	3	0.001
Agrégat 1 64 sources 	32.3	36.6	109.0%	99.2%	3	0.000
Agrégat 2 256 sources 	135.8	144.1	117.7%	95.4%	3	0.000
Agrégat 3 1024 sources 	611.6	577.0	129.9%	80.6%	3	0.000
Distribution 1 64 sources 	28.0	31.3	117.6%	**116.0**%	0	0.000
Distribution 2 256 sources 	109.4	122.2	125.0%	**117.3**%	3	0.000
Distribution 3 1024 sources 	435.6	487.9	133.9%	**118.2**%	3	0.000

TAB. 5.1: Résultats du critère diffus.

Ces mêmes valeurs sont utilisées pour les analyses de résultats que nous faisons plus loin.

Comme nous le voyons, notre méthode donne de bons résultats quand il y a suffisamment de sources pour former des agrégats d'assez petite taille. Les agrégats trop gros ne respectent jamais le critère d'erreur et résultent en une surcharge de travail inutile. Cette surcharge qui comporte l'évaluation de la borne et le parcours de la hiérarchie est d'environ 10% à 20% du temps de rendu total (Spéculaire 1 et Distribution). Comme nous utilisons des valeurs de pixel entre 0 et 1 qui sont ensuite converties en valeurs entières entre 0 et 255, un seuil de 0.01 donne une erreur potentielle totale de 7.5 par pixel. L'erreur maximale observée de 3 est deux fois et demi plus petite. L'erreur moyenne par pixel est négligeable. Comme nous le pensions, quand nous traitons des agrégats de sources qui sont petits par rapport à leur distance, la méthode performe bien avec des réductions de temps de calcul dépassant les 65% (Guirlande 3 et Spéculaire 2 et 3). La scène Agrégat ne donne pas d'aussi bons résultats que les scènes Guirlande et Spéculaire malgré que ses sources forment déjà un agrégat. Ceci est dû à une hiérarchie moins développée pour cette scène. Les sources étant uniformément distribuées dans un petit cube, peu de niveaux sont nécessaires pour que les feuilles contiennent suffisamment peu de sources pour arrêter la descente. À cause de la distribution uniforme, très peu de feuilles seront vides. Pour des scènes comme Guirlande et Spéculaire, les sources ne sont pas distribuées uniformément. Ceci donne une hiérarchie ayant beaucoup de feuilles vides. Plus de niveaux sont alors nécessaires pour que la subdivision des *voxels* cesse. Nous avons alors plus de niveaux et surtout des niveaux plus précis qui peuvent nous permettre de résumer avec la source virtuelle l'information du *voxel*. La scène Distribution comporte une forte distribution de sources lumineuses. Ces sources lumineuses uniformément distribuées donnent des agrégats de sources trop gros et souvent trop près des surfaces. Ils ne peuvent donc que rarement être utilisés et le coût de l'évaluation de la borne à chaque niveau explique le ralentissement observé.

La méthode du calcul adaptatif des rayons d'ombre donne de bons résultats en présence de peu de sources, contrairement à la nôtre. Cependant, en présence de beaucoup de sources ayant des contributions semblables (Distance), la méthode devient plus lente que le tracé de rayon standard. Même pour une scène où les sources ont des contributions très différentes (Distribution), la méthode est plus lente que le tracé de rayon

standard. Cette perte de performance est en partie attribuable au tri, qui pour nos scènes prend un temps considérable comparativement au temps requis pour tracer les rayons. Pour des scènes où la géométrie reste nettement plus complexe que le nombre de lumières, la méthode du calcul adaptatif des rayons d'ombre donnera de meilleurs résultats.

5.2 Surfaces spéculaires

Nous développons maintenant une méthode d'utilisation de la hiérarchie qui accélère le calcul de l'apparence de surfaces spéculaires. Nous utilisons ici une stratégie différente de celle pour les surfaces diffuses. Ce changement de façon de faire montre la versatilité de la hiérarchie. Comme nous l'exposons à la section 5.2.6, la stratégie utilisée pour les surfaces diffuses peut aussi servir pour les surfaces spéculaires.

Plutôt que de chercher une borne sur l'erreur de l'illumination causée par l'utilisation de la hiérarchie, nous cherchons les sources qui sont les plus importantes pour la réflexion spéculaire par rapport à une borne d'importance. Cette méthode est donc similaire à celle des sphères d'influence. Nous ne garantissons plus une erreur maximale sur le calcul d'illumination. Nous garantissons plutôt que l'illumination des sources que nous négligeons est multipliée par un facteur qui est plus petit ou égal au seuil S_{spec} que nous utilisons.

La méthode que nous présentons ici repose sur le fait que les sources ont une contribution spéculaire importante lorsqu'elles sont près de l'axe défini par la réflexion du vecteur de l'oeil par rapport à la normale de la surface.

5.2.1 Hypothèses et définitions pour les surfaces spéculaires

Les hypothèses et définitions introduites à la section 5.1.1 en page 41 sont encore applicables pour les surfaces spéculaires. À celles-là, nous en ajoutons quelques-unes ici.

Nous définissons :

$$\cos(\alpha_i) = \vec{R}_i \cdot \vec{E} = \vec{L}_i \cdot \vec{E}_r$$

$$0 \leq \alpha_i \leq \pi/2 \; .$$

5.2.2 Description de la stratégie spéculaire

Adaptant le modèle de Phong à la réflexion hautement spéculaire (*highlight*), nous avons :

$$I = \sum_{i=1}^{m} I_i \frac{1}{d_i^2} k_s \cos^n(\alpha_i) \, . \tag{5.9}$$

L'exposant n, pour des surfaces spéculaires peut être très élevé (de l'ordre de 100). C'est à ce type de réflexion que nous nous attaquons ici. Les réflexions spéculaires ayant un exposant faible (moins de 10) pourraient être traitées avec un algorithme similaire à celui pour les surfaces diffuses.

Un exposant n élevé diminue énormément les valeurs du terme cosinus pour les α_i qui ne sont pas très près de zéro. La figure 5.8 montre l'allure de $\cos^n(\alpha)$ pour différentes valeurs de n ainsi que des exemples de surfaces correspondantes. Pour les surfaces spéculaires, il y a donc une orientation bien précise que doivent respecter l'oeil et la lumière pour que se produise un *highlight* au point p. Pour une position d'oeil et un point d'une surface, la région où se trouvent les sources importantes est déterminée par un cône. La figure 5.9 montre ce cône en deux dimensions.

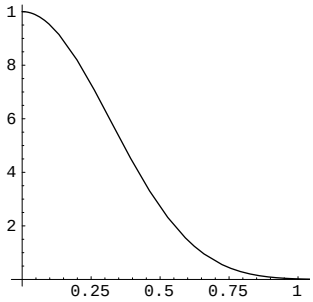
5.2.3 Cône de sources importantes

Nous voulons trouver l'ouverture du cône de telle sorte que les sources à l'extérieur aient une contribution très petite. Nous trouvons l'angle α_{max} pour lequel $\cos^n(\alpha)$ est égal au seuil désiré. De cette manière, les sources faisant un angle plus grand que α_{max} avec l'axe du cône $\vec{E}_r$, auront une contribution multipliée par un facteur plus petit que notre seuil. Pour une source sur l'axe du cône, $\cos^n(\alpha) = 1$. Nous choisissons donc un seuil représentant une fraction de la contribution d'une source sur l'axe, typiquement 0.0001. En choisissant un seuil suffisamment petit, les sources à l'extérieur contribueront pour très peu à l'illumination comparativement aux sources qui sont à l'intérieur du cône (prenant en considération des sources qui ont des intensités et des distances au point p similaires).

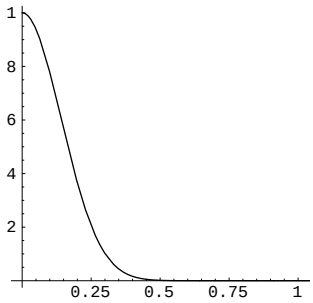
L'angle définissant le cône est calculé comme suit à partir du seuil S_{spec} :

$$\begin{aligned} \cos^n(\alpha_{max}) &= S_{spec} \\ \cos(\alpha_{max}) &= \sqrt[n]{S_{spec}} \\ \alpha_{max} &= \arccos\left(\sqrt[n]{S_{spec}}\right) \, . \end{aligned}$$

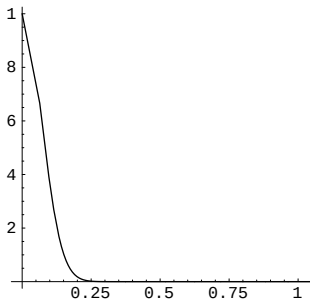
Nous utilisons la hiérarchie pour trouver rapidement les sources qui seront considérées dans le calcul. Comme on peut le voir à la figure 5.9, nous cherchons les



(a) $\cos^{10}(\theta)$



(b) $\cos^{50}(\theta)$



(c) $\cos^{200}(\theta)$

FIG. 5.8: Fonction cosinus élevée à différents exposants n et apparence des surfaces spéculaires correspondantes.

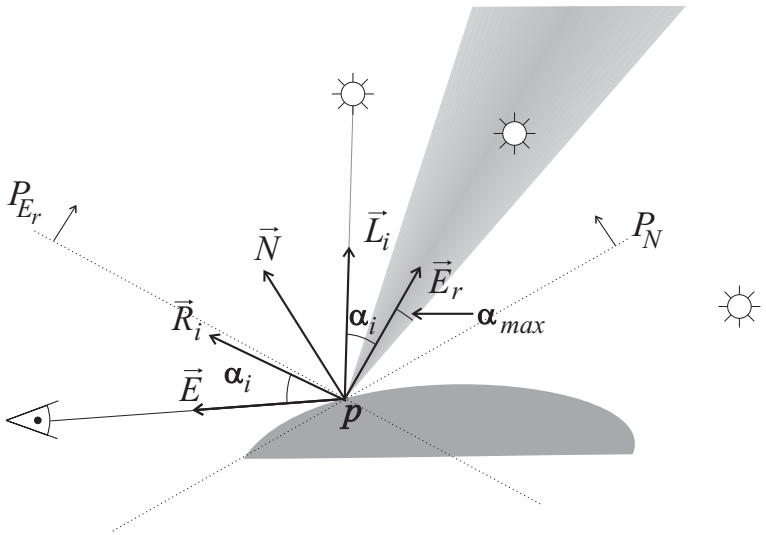


FIG. 5.9: Zone où se trouvent les sources plus importantes pour la réflexion spéculaire.

sources qui sont bien positionnées par rapport à la normale à la surface ($\vec{N}$), au rayon de l'oeil réfléchi ($\vec{E}_r$) et au cône (α_{max}). Pour la normale et le rayon de l'oeil réfléchi, nous n'avons qu'à éliminer les *voxels* et les sources qui sont complètement du mauvais côté du plan déterminé par le vecteur et le point où nous calculons l'illumination (les plans P_{Er} et P_N).

Déterminer si un *voxel* intersecte le cône spéculaire est relativement complexe par rapport au calcul de l'illumination avec une seule source. Nous avons décidé d'opter pour un rejet trivial qui est beaucoup plus rapide à calculer. Nous évaluons donc de façon conservatrice la position du *voxel* par rapport au cône.

Nous ne voulons pas traiter l'orientation du *voxel*, nous calculons donc notre possibilité d'intersection avec le centre du *voxel* et le Δd_{max} de la borne diffuse. La figure 5.10 résume l'information que nous utilisons pour évaluer la possibilité d'intersection. Définissons quelques valeurs :

$$\vec{C}_{cone} = c - p$$

$$d_{axe} = \vec{C}_{cone} \cdot \vec{E}_r$$

$$\vec{V}_{axe,c} = \vec{C}_{cone} - d_{axe}\vec{E}_r$$

$$d_{axe,c} = ||\vec{V}_{axe,c}||$$

$$dim_{min} = \tan(\alpha_{max})(d_{axe} - \Delta d_{max}/2)$$

$$dim_{max} = \tan(\alpha_{max})(d_{axe} + \Delta d_{max}/2)\ .$$

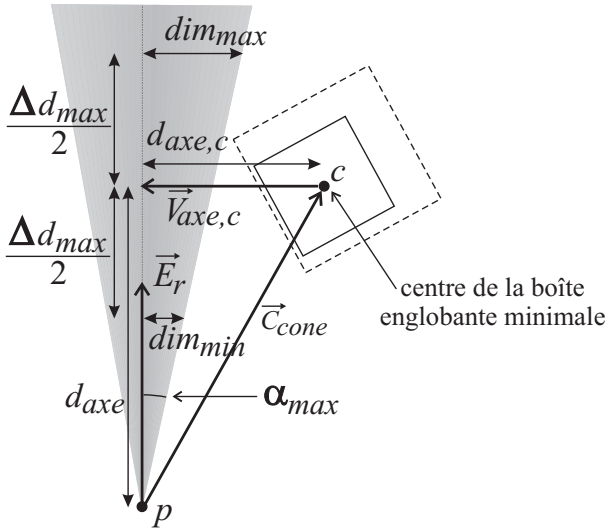


FIG. 5.10: Calcul conservateur de la position d'un *voxel* par rapport au cône spéculaire.

Dans ces dernières équations, les vecteurs $\vec{C}_{cone}$ et $\vec{V}_{axe,c}$ ne sont pas normalisés. Un *voxel* est hors du cône si

$$dim_{max} < d_{axe,c} - \Delta d_{max}/2 \,.$$

De même, un *voxel* est à l'intérieur du cône si

$$dim_{min} > d_{axe,c} + \Delta d_{max}/2 \,.$$

Les *voxels* qui ne sont pas à l'intérieur ou à l'extérieur du cône, selon nos critères conservateurs, sont considérés comme intersectant le cône.

De façon similaire, nous testons de manière conservatrice la position du *voxel* par rapport aux plans P_N et P_{E_r}.

Nous commençons à tester la position des *voxels* à la racine de la hiérarchie. Si un *voxel* est complètement à l'intérieur, il ne reste qu'à tester les sources qu'il contient. Un *voxel* à l'extérieur est rejeté ainsi que ses enfants et sources. C'est là qu'apparaît l'utilité de la hiérarchie. Plutôt que de tester les sources une à une, nous pouvons éliminer des groupes entiers avec un seul test. Pour les *voxels* qui intersectent le cône, s'ils sont subdivisés, nous recommençons les tests avec leurs enfants. Quant aux *voxels* intersectés mais non subdivisés, nous testons leurs sources par rapport aux plans et au cône, et évaluons leur illumination le cas échéant.

Nous constatons que c'est le long du cône que la méthode de calcul de la position des *voxels* passe le plus de temps. Ceci est dû à la subdivision pour déterminer l'appar-

tenance ou non au cône des *voxels* qui l'intersectent. Il serait intéressant de modifier la méthode pour qu'elle passe moins de temps à déterminer la position de *voxels* qui ont peu d'importance.

Encore une fois, il est intéressant de connaître le temps que prend le test d'un *voxel* par rapport au calcul de l'illumination d'une seule source. Déterminer la position d'un *voxel* par rapport aux plans et au cône est trois à quatre fois plus complexe que le calcul de l'illumination d'une seule source. Il faut donc que le *voxel* contienne au moins quatre sources pour que l'utilisation de la hiérarchie soit intéressante. Avec quatre sources ou moins, il vaut mieux calculer directement l'illumination avec celles-ci.

5.2.4 Critère spéculaire

Comme pour le critère diffus, le critère spéculaire s'applique quand nous avons trouvé le point où nous voulons calculer l'apparence de la surface. L'algorithme de calcul hiérarchique de l'apparence spéculaire est présenté à la figure 5.11.

Notons que le critère que nous avons développé ne se limite pas aux scènes où les sources sont toujours visibles. Nous économisons en ne calculant pas la contribution de sources qui sont à l'extérieur du cône spéculaire. Nous calculons cependant complètement la contribution des sources qui sont dans le cône, sans utiliser la source virtuelle. Ce critère pourrait donc facilement être adapté à un programme de tracé de rayon pour réduire le temps de calcul des surfaces spéculaires.

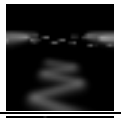
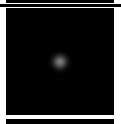
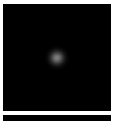
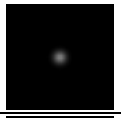
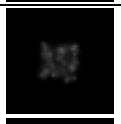
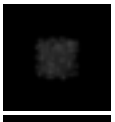
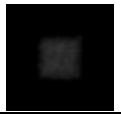
5.2.5 Résultats du critère spéculaire

Dans cette section, nous présentons les résultats de l'utilisation du critère spéculaire. Nous comparons ces résultats avec le tracé de rayon standard et avec le calcul adaptatif des rayons d'ombre [War91]. Nous faisons ici aussi le calcul des rayons d'ombre. Comme cette technique traite déjà la visibilité des sources, les résultats donnent la vraie performance du critère. Les résultats de l'utilisation du critère spéculaire avec $S_{spec} = 0.0001$ sont présentés à la table 5.2.

Dans cette série de tests, la méthode d'échantillonnage adaptatif des rayons d'ombre performe bien. Comme les sources qui sont hors du cône spéculaire ont une contribution infime, la méthode peut s'arrêter très tôt dans la traversée de la liste, amenant de fortes réductions de temps de calcul (plus de 80% avec Guirlande 2 et34 ainsi que

```
IlluminationHiérarchique( Point p, Voxel v )
{
    // Le calcul des plans P_N, P_Er et de α_max
    // a déjà été fait.

    if v.Vide()
        return( 0 )

    if v.NombreLumières() ≤ 4
        return( IlluminationNormale( p, v.sources ) )

    position = ÉvaluerPosition( v, P_N, P_Er, α_max )

    if position == extérieur
        return( 0 )
    else if position == intérieur
        return( IlluminationNormale( p, v.sources ) )
    else
        // position == intersecte
        return( IlluminationNiveauInférieur( p, v ) )
}
IlluminationNiveauInférieur( Point p, Voxel v )
{
    if v.Subdivisé()
        ∀ v.voxelEnfant
            IlluminationHiérarchique( p, v.voxelEnfant )
    else
        sourcesSélectionnées = SourcesBienPositionnées( v, P_N, P_Er, α_max )
        IlluminationNormale( p, sourcesSélectionnées )
}
```

FIG. 5.11: Algorithme de calcul de l'illumination à l'aide de la hiérarchie de lumières pour des surfaces spéculaires.

Scène	TR seul.	TR	Ward	HL	err.max	err.pixel
Guirlande 1  61 sources	10.7	12.7	26.8%	15.7%	3	0.002
Guirlande 2  261 sources	85.0	94.3	17.5%	10.9%	3	0.003
Guirlande 3  1056 sources	541.2	581.4	13.3%	11.0%	3	0.003
Spéculaire 1  62 sources	30.1	46.4	33.0%	5.6%	3	0.004
Spéculaire 2  257 sources	293.5	279.3	22.2%	3.3%	3	0.004
Spéculaire 3  1029 sources	1335.9	1514.9	18.4%	3.2%	3	0.003
Agrégat 1  64 sources	32.3	50.2	42.6%	6.6%	3	0.001
Agrégat 2  256 sources	135.8	196.5	48.5%	5.0%	3	0.001
Agrégat 3  1024 sources	611.6	779.3	56.0%	4.6%	3	0.001
Distribution 1  64 sources	28.0	57.8	45.2%	5.9%	3	0.001
Distribution 2  256 sources	109.4	225.5	47.8%	3.2%	3	0.000
Distribution 3  1024 sources	435.6	899.0	52.8%	2.6%	3	0.000

TAB. 5.2: Résultats du critère spéculaire.

Spéculaire 3).

Notre méthode donne d'encore meilleures performances. Dans tous les cas, elle est plus rapide que le tracé de rayon standard et que l'échantillonnage adaptatif des rayons d'ombre. Les réductions de temps de calcul se situent entre 84% et 97%. L'erreur maximale est encore une fois très faible et l'erreur moyenne est négligeable.

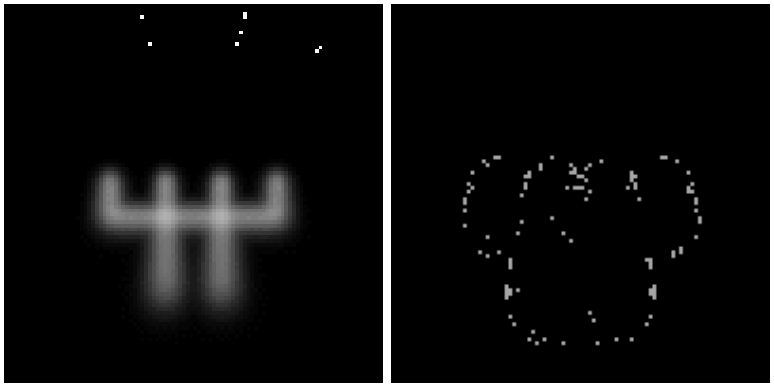
L'effet de l'utilisation du cône pour discriminer les sources est visible à la figure 5.12. Cette figure montre la différence amplifiée entre l'image de tracé de rayon et l'image calculée avec notre méthode pour les scènes Spéculaire 2 et Agrégat 1. Nous voyons que l'erreur faite par cette méthode se situe dans les régions moins intenses des *highlights*.

5.2.6 Borne sur l'illumination spéculaire

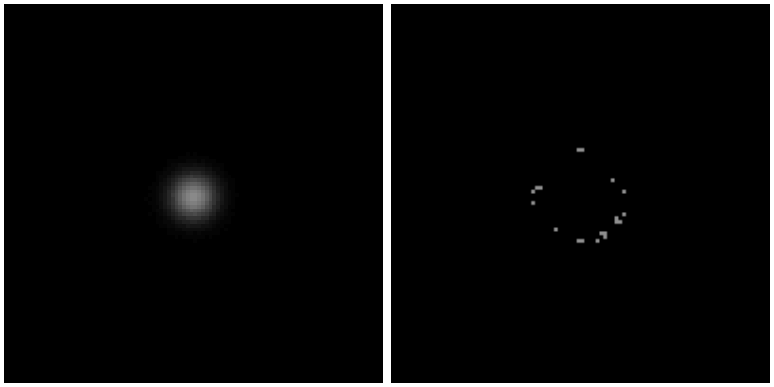
Nous avons présenté une façon de réduire le coût du calcul de l'apparence pour des surfaces spéculaires. Cette méthode donne de bonnes économies par rapport au tracé de rayon standard, mais souffre de défauts assez graves. Premièrement, elle ne garantit aucune borne sur l'erreur de l'illumination résultante. Comme nous l'avons vu, en ajustant le seuil convenablement, cette erreur est minime, mais demeure tout de même présente et non bornée. L'ajustement du seuil requis pour que l'erreur soit petite élargit le cône, ajoutant des calculs d'illumination qui peuvent être inutiles.

La méthode ne tient pas vraiment compte de la distance le long du vecteur de l'oeil réfléchi $\vec{E}_r$. Ce problème est flagrant quand une source est très près du point où nous calculons l'apparence. Si la distance est petite, le facteur $1/d^2$ fera que la contribution de la source est amplifiée. La contribution de la source peut donc être arbitrairement grande en fonction de la proximité à la surface. Une source près de la surface, mais légèrement à l'extérieur du cône, sera ignorée bien qu'elle ait une contribution importante. Inversement, une source très éloignée, mais proche de l'axe $\vec{E}_r$ sera prise en compte, même si sa contribution est infime. Finalement, la méthode néglige une partie de l'illumination en ignorant systématiquement la contribution des sources qui sont hors du cône. Le résultat du calcul de la réflexion spéculaire sera donc toujours plus foncé ou égal à la couleur calculée en utilisant toutes les sources.

Pour résoudre ces problèmes, il faut développer une méthode qui, comme pour la borne diffuse, garantit une erreur maximale sur le résultat du calcul d'illumination. Nous proposons ici une méthode donnant une borne d'erreur sur le calcul de l'illu-



(a) Scène spéculaire



(b) Scène agrégat

FIG. 5.12: Zone d'erreur du critère spéculaire. Les images de gauche représentent le rendu par la méthode hiérarchique. Les images de droite représentent la différence, pixel à pixel, entre l'image calculée par tracé de rayon standard et l'image calculée par la méthode hiérarchique. L'erreur étant infime, nous avons dû l'amplifier pour que la zone d'erreur soit bien discernable.

mination spéculaire. Cette méthode n'a pas été implantée et fait partie des travaux futurs. Cependant, nous jugeons importante la présentation de cette borne car elle complète celle pour les surfaces diffuses. Nous avons ainsi une borne sur toute l'erreur d'illumination à l'aide de la hiérarchie pour les réflexions diffuses et spéculaires.

La stratégie que nous présentons ici se base, tout comme celle du calcul adaptatif des rayons d'ombre, sur l'évaluation de contributions potentielles. Nous ne calculons cependant pas que des contributions potentielles de sources, mais aussi d'agrégats de sources. Nous calculons la contribution potentielle maximale d'un agrégat et, si cette dernière dépasse un certain seuil, nous forçons une évaluation plus précise.

Contribution spéculaire potentielle maximale

Nous calculons la contribution potentielle maximale d'un *voxel* ou d'une source lumineuse. Si celle-ci dépasse le seuil spéculaire, nous forçons une évaluation plus précise.

Pour un *voxel*, la contribution potentielle maximale est calculée par rapport à la distance minimale à l'axe $\vec{E}_r$ en utilisant les valeurs définies à la section 5.2.3 en page 59. La distance minimale à l'axe est $d_{axe,c} - \Delta d_{max}/2$ et la distance minimale le long de l'axe est $d_{axe} - \Delta d_{max}/2$. Avec ces données et I_v, nous pouvons évaluer la contribution potentielle maximale d'un *voxel*.

Pour une source, la contribution potentielle maximale est le calcul de l'illumination spéculaire dans lequel le calcul de visibilité de la source est ignoré.

Si la contribution potentielle maximale est plus grande que le seuil spéculaire, l'évaluation plus précise est forcée. Pour un *voxel*, cela signifie recommencer la procédure avec les enfants ou avec les sources si le *voxel* n'est pas subdivisé. Pour une source, sa visibilité est évaluée.

Les sources et *voxels* dont la contribution est plus petite que le seuil spéculaire sont ajoutés à une liste de contributions ignorées. Si la somme des contributions potentielles maximales ignorées dépasse le seuil spéculaire, l'évaluation plus précise des sources ou *voxels* associés aux plus grandes contributions est forcée, jusqu'à ce que la somme soit inférieure au seuil spéculaire. De cette façon, nous pouvons garantir une borne sur l'erreur de l'utilisation de la hiérarchie pour les surfaces spéculaires.

Pour ne pas biaiser les résultats en ignorant systématiquement les sources et agrégats que nous négligeons, une méthode similaire à celle utilisée dans le calcul adaptatif

des rayons d'ombre peut être utilisée. Une approximation de la contribution de ces sources et agrégats ignorés est alors évaluée et ajoutée à l'illumination déjà calculée. L'estimation de la contribution d'un agrégat est faite en utilisant la contribution de la source virtuelle.

5.3 Intégration des critères diffus et spéculaires

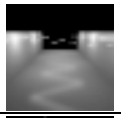
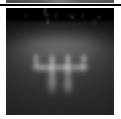
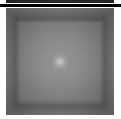
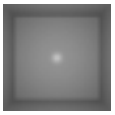
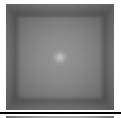
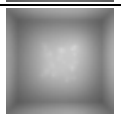
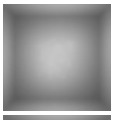
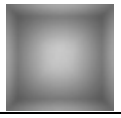
Nous présentons ici les résultats de l'intégration du critère diffus et du critère spéculaire de la section 5.2.3. L'intégration efficace des deux critères n'est pas simple. Nous faisons face à deux phénomènes différents et à deux approches. Il est difficile de trouver une façon efficace de regrouper les deux stratégies.

Dans notre implantation, la descente dans la hiérarchie se fait conjointement pour les deux critères. Évidemment, toute la hiérarchie n'est pas systématiquement traversée. Les deux critères cheminent ensemble tant que l'un d'eux n'a pas trouvé le niveau de la hiérarchie auquel il peut prendre une décision. Quand un des deux critères a terminé, l'autre continue la descente seul. Si les deux critères ont atteint des niveaux suffisants, la descente est arrêtée pour la branche correspondante de la hiérarchie.

5.3.1 Résultats des critères diffus et spéculaires

Comme nous venons tout juste de le voir, ces deux critères sont assez différents et difficilement intégrables. Il ne faut donc pas s'étonner que les temps de calcul soient plus grands que lors de l'analyse des critères individuellement. Nous lançons encore les rayons d'ombre pour fournir un indice sur les accélérations possibles par notre méthode. Nous utilisons les mêmes seuils que pour les tests séparés, soit $S_{diffus} = 0.01$ et $S_{spec} = 0.0001$. La table 5.3 montre les résultats de l'intégration des deux critères.

Tel que prévu, les temps de calcul de notre méthode sont plus élevés que lorsque nous testons les critères individuellement. Ils sont presque tous légèrement plus petits que la simple addition des temps de calcul pour les critères diffus et spéculaire. Ceci montre que, pour les scènes que nous utilisons, la traversée de la hiérarchie est peu coûteuse. Les hiérarchies de lumières de nos scènes ont entre trois et huit niveaux. En présence de scènes contenant plus de sources, la traversée de la hiérarchie serait plus importante. Ce problème pourrait cependant être contrebalancé par le fait que notre méthode parcourt seulement les branches nécessaires pour respecter la borne.

Scène	TR seul.	TR	Ward	HL	err.max	err.pixel
Guirlande 1  61 sources	10.7	12.8	73.4%	92.2%	3	0.001
Guirlande 2  261 sources	85.0	94.0	72.1%	78.1%	3	0.001
Guirlande 3  1056 sources	541.2	581.4	70.3%	51.7%	3	0.000
Spéculaire 1  62 sources	30.1	44.6	75.1%	66.8%	3	0.003
Spéculaire 2  257 sources	293.5	306.8	58.1%	39.2%	3	0.001
Spéculaire 3  1029 sources	1335.9	1535.1	62.2%	17.6%	3	0.001
Agrégat 1  64 sources	32.3	49.7	99.2%	77.9%	3	0.000
Agrégat 2  256 sources	135.8	210.7	97.1%	73.1%	3	0.000
Agrégat 3  1024 sources	611.6	811.5	107.9%	61.6%	3	0.000
Distribution 1  64 sources	28.0	59.1	84.6%	65.8%	3	0.000
Distribution 2  256 sources	109.4	230.3	87.5%	65.3%	3	0.000
Distribution 3  1024 sources	435.6	916.7	92.0%	64.5%	3	0.000

TAB. 5.3: Résultats des critères diffus et spéculaire.

Bien que notre méthode soit un peu moins performante que lors de l'analyse individuelle des critères, les réductions de temps de calcul restent considérables, surtout en présence de beaucoup de sources. De plus, notre méthode est toujours plus rapide que le tracé de rayon et souvent plus rapide que le calcul adaptatif des rayons d'ombre. Les erreurs maximales et moyennes sont encore très petites.

Notre méthode donne de meilleurs résultats pour les surfaces qui ont une réflexion uniquement diffuse ou spéculaire. Retrouver l'information pertinente dans la hiérarchie peut demander un temps considérable en présence de propriétés de réflexion plus complexes. Il faudrait développer des critères qui permettent de déterminer si l'utilisation de la hiérarchie peut réduire le temps de calcul, pour un point de vue, une surface et un agrégat donné. Toutes les configurations seraient ainsi traitées par la méthode la plus efficace.

La méthode de calcul adaptatif des rayons d'ombre donne de bons résultats pour les scènes Guirlande et Spéculaire. Comme pour notre approche, les résultats sont moins intéressants que lors de l'analyse des résultats pour les surfaces diffuses seulement et pour les surfaces spéculaires seulement. Pour les scènes Distribution et Agrégat, les temps de calcul sont très proches de ceux du tracé de rayon standard.

Chapitre 6

Traitement de la visibilité

Comme nous l'avons vu dans le chapitre précédent, l'utilisation d'une hiérarchie est intéressante. Les économies en temps de calcul sont appréciables, permettant de produire efficacement des images de scènes avec énormément de sources lumineuses.

Dans ce chapitre nous ne nous restreignons plus aux scènes pour lesquelles aucun bloqueur ne fait d'ombre visible. L'utilisation de la hiérarchie complexifie beaucoup le traitement de l'occlusion des sources. Nous ne pouvons pas simplement lancer un rayon d'ombre vers la source virtuelle, car la visibilité de chacune des sources n'est pas assez corrélée à celle de la source virtuelle. Nous avons besoin d'une façon plus complète de déterminer la visibilité d'un agrégat de sources.

Dans la prochaine section, nous regardons le problème de visibilité que nous devons résoudre. Nous voyons ensuite comment la visibilité volumétrique nous permet de résoudre efficacement notre problème. Nous introduisons finalement une méthode approximative pour calculer la visibilité des agrégats de sources. Le traitement de la visibilité, bien qu'il soit prometteur, n'a pas été implanté et fait partie des extensions de cette recherche.

Pour éviter toute confusion et alléger le texte, nous utilisons maintenant "agrégat" pour spécifier un *voxel* de sources et "*voxel*" quand il s'agit d'un *voxel* d'objets.

6.1 Problème de la visibilité d'un agrégat

Dans cette section, nous discutons de la visibilité des sources qui nous permet de déterminer si un point est illuminé ou dans l'ombre. Nous commençons par décrire le

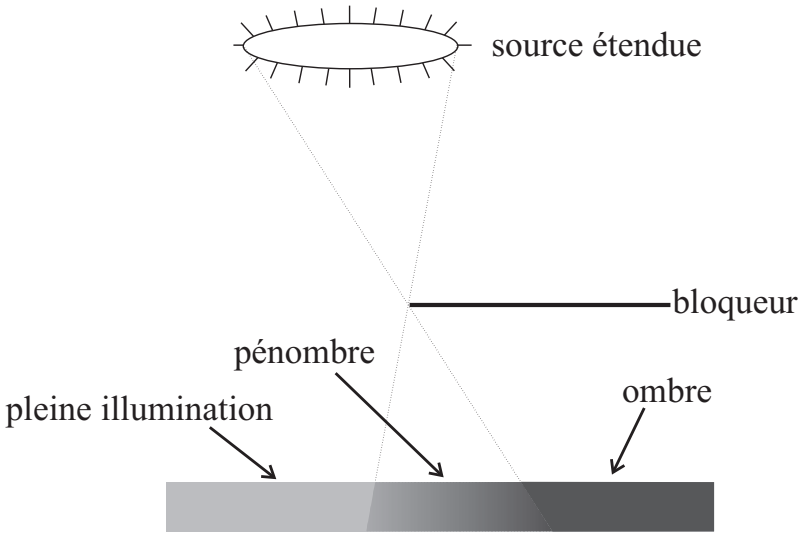


FIG. 6.1: Différentes zones d'illumination faites par un bloqueur.

problème de visibilité de sources en tracé de rayon, puis nous exposons le problème de visibilité d'un agrégat.

6.1.1 Visibilité de sources

Nous avons déjà parlé du problème de visibilité des sources (ombres) en tracé de rayon. Les sources ponctuelles sont souvent utilisées et, dans ce cadre restreint, les requêtes de visibilité se limitent à la *visibilité point à point*. Il suffit de vérifier si le segment reliant le point d'intérêt et la source intersecte un objet.

Lorsque des sources lumineuses étendues sont présentes dans les scènes, un nouveau type de visibilité doit être considéré. La visibilité entre un point et une surface (ou un segment) doit maintenant être calculée pour déterminer la contribution de la source. Cette visibilité est beaucoup plus complexe. Dans le cas de la visibilité point à point, la réponse est binaire : visible (illuminé) ou cachée (ombre). Pour la *visibilité point à surface*, il se peut aussi que la surface soit cachée seulement en partie (pénombre). Il faut alors calculer la portion de la source qui est visible du point et en déterminer la contribution à l'illumination. La figure 6.1 montre les différentes zones d'illumination produites par un bloqueur.

Le calcul de la portion visible d'une surface est beaucoup plus complexe que de déterminer si un rayon intersecte des objets. Il faut trouver les objets qui intersectent le *volume de visibilité* défini par le point et le contour de la surface. Le contour de la

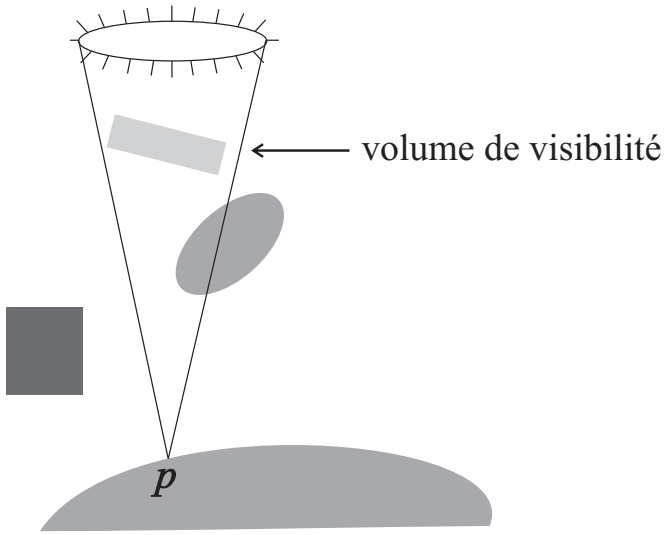


FIG. 6.2: Visibilité point à surface.

surface est déterminé par rapport au point. La figure 6.2 montre le volume utilisé pour calculer la visibilité.

À partir des objets qui intersectent le volume de visibilité, il faut déterminer comment ils bloquent la surface. Ceci peut être fait en projetant les objets et en calculant la surface qui est touchée par ces projections. Ce calcul est très coûteux car il nécessite la projection de beaucoup d'objets et l'union de ces projections pour déterminer la surface qu'elles recouvrent.

Des méthodes approximatives sont souvent utilisées pour réduire la complexité du calcul de visibilité. Plutôt que de calculer exactement la visibilité d'une source étendue, cette visibilité peut être échantillonnée en traçant un certain nombre de rayons entre le point et la surface [CPC84]. La proportion des rayons qui atteignent la source sans être bloqués par un objet donne une approximation de la visibilité de la source. Plus le nombre de rayons est important, meilleure est l'approximation. Le temps de calcul augmente cependant avec le nombre de rayons à tracer.

6.1.2 Visibilité d'un agrégat

Avec l'utilisation de la hiérarchie, la visibilité n'est plus point à point. Un agrégat remplace plusieurs sources et est utilisé pour calculer l'apparence de la surface. Il faut déterminer comment est bloquée la lumière qui arrive de cet agrégat. Ce calcul s'apparente à une visibilité point à surface.

La visibilité que nous devons calculer est tout de même différente de la visibilité point à surface pour les sources étendues. Ce n'est pas la surface de l'agrégat tel que vu du point p qui éclaire, mais seulement certains points sur cette surface. Ces points correspondent aux sources projetées sur la surface de l'agrégat par rapport au point p. La surface de l'agrégat peut être approximée par la boîte minimale contenant les sources du *voxel*. Déterminer la répartition de ces points sur la surface ne peut être fait à partir de l'information conservée dans l'agrégat.

Pour faire un traitement correct de la visibilité d'un agrégat, nous ne pouvons prendre une décision que lorsque la visibilité est complète ou nulle. Si un agrégat est complètement visible, nous n'avons plus qu'à calculer l'apparence de la surface à partir de la source virtuelle. Si au contraire il est complètement caché, aucun calcul n'a à être fait puisque sa contribution à l'illumination est nulle. Lorsqu'un agrégat est partiellement visible, nous ne pouvons conclure quoi que ce soit quant à l'illumination qui arrive en p à partir des informations de l'agrégat. Il faut donc raffiner la visibilité à un niveau plus bas dans la hiérarchie des sources. Si un agrégat est subdivisé, nous calculons la visibilité avec ses enfants, sinon, il ne nous reste plus qu'à calculer la visibilité des sources qu'il contient.

Déterminer si un agrégat est complètement, partiellement ou pas du tout visible, est complexe par rapport au tracé des rayons d'ombre pour les sources qu'il contient. Notre méthode doit être plus rapide que le tracé de rayon, et ce même en considérant les techniques d'optimisation vues au chapitre 3. Nous ne pouvons donc pas nous permettre d'utiliser une méthode exacte.

6.2 Visibilité volumétrique

Comme la détermination de la visibilité doit être faite rapidement, nous optons pour une *visibilité volumétrique* [Shi92]. Les méthodes volumétriques [KV84] traitent l'espace comme un volume ayant une densité. La densité peut déterminer plusieurs caractéristiques telles la couleur, la réflexion et l'*extinction*. L'extinction est la caractéristique qui nous intéresse ici car nous voulons utiliser la méthode volumétrique pour calculer la visibilité. Elle va de nulle (complètement transparent) à totale (opaque) et détermine l'*atténuation* subie le long d'un rayon qui traverse le volume. L'atténuation est fonction de la longueur du segment du rayon qui est dans le volume.

La visibilité volumétrique a été utilisée dans le contexte de visibilité par texels [KK89, Ney96] et de radiosité [Sil94]. Bien qu'elle soit approximative, elle est plus rapide à calculer que la visibilité complète par plus d'un ordre de magnitude [Sil94]. Elle devrait donc offrir l'efficacité nécessaire pour que notre méthode soit plus rapide que le tracé des rayons d'ombre.

Pour une scène décrite par des objets et non des densités, il faut tout d'abord convertir ces objets en volumes et extinctions qui pourront ensuite être utilisés. Une méthode employée en radiosité hiérarchique [Sil94] permet de convertir des objets en une valeur d'extinction κ associée à un *voxel*. Cette extinction est calculée comme un rapport entre l'aire des objets contenus dans le *voxel* et le volume de ce dernier.

$$\kappa = \frac{A}{4V}$$

Cette méthode fait l'hypothèse d'une distribution uniforme de petits objets dont les orientations sont elles aussi distribuées uniformément.

Un inconvénient de cette technique est qu'elle perd toute l'information de spatialité et de directionnalité. Nous avons fait quelques tests avec cette méthode et avons conclu que l'hypothèse sur la distribution des objets est trop contraignante. De plus, lorsque l'hypothèse n'est pas respectée, les résultats sont décevants. Une pile d'assiettes peut alors donner la même extinction qu'un petit arbuste. La visibilité de ces objets est très différente, mais une fois qu'ils sont convertis en extinction, elle peut être la même.

Une autre façon de calculer la densité d'un *voxel* est de procéder par échantillonnage [RPV93]. Plutôt que de conserver une seule valeur par *voxel*, cette méthode en conserve plusieurs. La consommation en mémoire est évidemment accrue, mais l'information gardée pour un *voxel* est bien plus riche. Comme la méthode n'est utilisée que pour les *voxels* qui contiennent beaucoup d'objets, la représentation volumétrique est toujours plus simple que la géométrie qu'elle remplace. Il y a un compromis entre la richesse de l'information d'un *voxel* et la consommation en mémoire. L'évaluation des caractéristiques de la densité est faite en pré-calcul en traçant des rayons au travers des *voxels*.

6.3 Visibilité approximative d'un agrégat

Nous présentons dans cette section une approche qui permet de calculer rapidement une approximation de la visibilité d'agrégats de sources. Cette visibilité est dérivée de la visibilité volumétrique.

6.3.1 Conversion des objets en extinctions directionnelles

L'échantillonnage utilisé par Rushmeier *et al.* [RPV93] est employé pour connaître l'extinction associée à un *voxel*. Nous calculons trois extinctions, une selon chaque axe du *voxel*. Pour évaluer l'extinction d'un *voxel* selon un des axes, des rayons sont tracés à partir des deux faces ayant une normale parallèle à l'axe. Les rayons sont construits en reliant deux points choisis aléatoirement sur chacune des deux faces. La proportion des rayons qui intersectent un objet dans le *voxel* représente l'atténuation moyenne d'un rayon pour l'axe considéré.

6.3.2 Calcul de l'atténuation

Pour connaître l'atténuation d'un rayon qui traverse le *voxel*, il faut tout d'abord trouver le segment du rayon qui traverse le *voxel*. L'atténuation produite le long du rayon est calculée à partir de la projection du segment sur les trois axes du *voxel* et des extinctions correspondantes.

La visibilité que nous devons calculer n'est pas celle d'un rayon, mais du volume défini par p et le contour de l'agrégat. Ce volume est beaucoup trop complexe et nous en faisons l'approximation par un cône ayant comme sommet p et centré sur la boîte minimale des sources. L'ouverture du cône doit contenir entièrement la boîte minimale des sources.

Pour calculer la visibilité à l'intérieur du cône, nous devons tenir compte des *voxels* qui l'intersectent. Comme nos objets sont conservés dans un *octree*, nous avons une représentation de visibilité volumétrique sur différents niveaux. Nous choisissons les *voxels* qui intersectent le cône et qui sont presque entièrement à l'intérieur. Pour les *voxels* qui intersectent le cône, mais pour lesquels seulement une petite partie de leur volume se retrouve dans le cône, le calcul de visibilité sera fait avec leurs enfants. Évidemment, si un *voxel* n'est pas subdivisé, nous devons l'utiliser même si sa représentation n'est pas vraiment adéquate pour calculer la visibilité dans le cône.

6.3.3 Méthode hybride

L'atténuation attribuée au rayon par le *voxel* où se trouve p, ne doit pas être prise en compte car elle représente une ombre erronée que p ferait sur lui-même. Nous n'utilisons donc la visibilité volumétrique que pour la partie du cône qui est à une certaine distance de p. Pour calculer la visibilité dans la première partie du cône, nous utilisons le tracé de rayon standard en marquant les *voxels* pour ne pas les considérer ultérieurement dans le calcul de visibilité volumétrique. Une fois que nous sommes suffisamment loin de p, nous pouvons utiliser la visibilité volumétrique pour calculer l'atténuation du cône. Nous arrêtons graduellement le tracé des rayons d'ombre pour ne pas introduire de discontinuités dues à un changement brusque de stratégie.

Les rayons d'ombre tracés entrent dans le calcul de l'atténuation dans le cône. Chaque rayon d'ombre qui intersecte un objet entraîne une augmentation de l'atténuation du cône d'un facteur $1/m$ où m est le nombre de sources (et par conséquent le nombre de rayons d'ombre).

Dès que l'atténuation dépasse un certain seuil, nous considérons qu'il y a occlusion complète et arrêtons les calculs.

Une fois le calcul de visibilité terminé, l'illumination calculée à partir de la source virtuelle est multipliée par l'atténuation du cône pour donner l'illumination résultante en p.

Chapitre 7

Conclusion

Le réalisme des images de synthèse demande une bonne approximation de l'illumination de scènes réelles. Ces dernières renferment souvent un grand nombre de sources lumineuses. Le temps de calcul de l'illumination directe en un point est proportionnel au nombre de sources présentes dans la scène puisque l'illumination est calculée séparément pour chaque source.

Nous avons présenté une méthode qui permet de réduire les coûts du calcul de l'apparence des surfaces en présence de centaines de sources lumineuses. Notre méthode utilise le tracé de rayon, les sources ponctuelles et le modèle de réflexion de Phong. Elle se base sur une hiérarchisation des sources lumineuses en agrégats. Cette hiérarchisation utilise la structure d'*octree* et permet de représenter à plusieurs niveaux de précision l'information de l'illumination présente dans la scène. Des critères qui permettent de quantifier l'erreur faite par l'utilisation de la hiérarchie sont introduits pour guider le choix des différents niveaux de précision requis lors du calcul de l'apparence d'une surface. Pour la réflexion diffuse, le critère se base sur le point de la surface où nous voulons calculer l'apparence ainsi que sur la position et la dimension de l'agrégat de sources par rapport à ce point. Ce critère est indépendant du point de vue et donne une borne efficace sur l'erreur maximale due à l'utilisation de la hiérarchie pour les surfaces diffuses. La réflexion spéculaire est traitée par un critère qui détermine un cône en se basant sur le point où nous voulons calculer l'apparence et le point de vue. Ce cône est utilisé pour considérer seulement les sources qui ont une contribution multipliée par un facteur plus grand qu'un certain seuil. En utilisant la hiérarchie, nous pouvons déterminer rapidement les sources qui doivent être considérées.

Pour les scènes sans ombre, notre méthode donne de bons résultats et se compare avantageusement avec la méthode du calcul adaptatif des rayons d'ombre [War91]. Les réductions des temps de calcul à l'aide de la hiérarchie vont jusqu'à 97% pour nos scènes de test. De plus, lorsque notre méthode s'applique mal à une scène, la surcharge de calcul observée est d'environ 10% à 20%. L'aspect le plus intéressant est qu'en plus d'accélérer le rendu, notre méthode permet d'avoir une borne sur l'erreur commise lors de son utilisation.

La hiérarchie permet de traiter des scènes contenant des milliers de sources avec un temps de calcul espéré logarithmique par rapport au nombre de sources. Ce coût est nettement réduit comparativement aux techniques précédentes et notre méthode fournit une borne maximale sur l'erreur. Il est possible de rendre efficacement des scènes qui auraient difficilement pu être traitées auparavant. Notre méthode permet d'avoir des illuminations sophistiquées sans que la pénalité en temps de calcul soit trop grande.

Les coûts mémoire de notre méthode ne sont pas excessifs grâce à l'utilisation d'une structure hiérarchique. Une consommation accrue de mémoire de l'ordre de $O(\log(m))$, où m est le nombre de sources, doit cependant être escomptée.

Le niveau de représentation de l'illumination que nos critères permettent de choisir est indépendant du facteur de blocage. L'erreur introduite par l'utilisation d'un agrégat respectera la borne d'erreur fixée par l'usager peu importe la visibilité de l'agrégat. Notre méthode donne donc une base solide permettant de l'étendre aux scènes avec ombres.

Calculer la visibilité de l'agrégat peut cependant être très coûteux comparativement au tracé des rayons d'ombre des sources qu'il contient. C'est pourquoi nous proposons un calcul approximatif de la visibilité. Cette méthode permet de calculer efficacement la visibilité d'un agrégat en utilisant le tracé de rayon ainsi que la visibilité volumétrique. Le tracé de rayon est utilisé dans la région près de la surface, permettant de bien détecter les objets qui risquent de causer des ombres très nettes. À plus grande distance, la visibilité volumétrique prend la relève et approxime la visibilité à l'aide de l'information d'extinction pré-calculée pour les *voxels* d'objets. Bien que nous n'ayons pas encore de résultats, cette méthode nous semble très prometteuse.

7.1 Améliorations et extensions

7.1.1 Borne d'erreur relative

Notre méthode donne des bornes absolues sur l'erreur de l'illumination. Il serait intéressant et facile d'utiliser une erreur relative comme le mentionne Ward [War91]. L'oeil humain est plus sensible aux différences relatives d'intensités qu'aux différences absolues. Les modifications nécessaires à notre méthode pour utiliser une intensité relative sont mineures. Il suffit de négliger la somme des intensités des sources et la couleur de la surface dans la borne diffuse et spéculaire (si une borne efficace est dérivée pour la réflexion spéculaire, ce qui semble possible).

7.1.2 Représentation en agrégats

La représentation de l'illumination qui est gardée à chaque niveau dans les agrégats de sources pourrait contenir une information plus précise. Une boîte englobante minimale non alignée sur les axes serait intéressante. Elle pourrait donner une meilleure information sur la répartition spatiale des sources dans l'agrégat. Nous pourrions également conserver de l'information quant au degré d'uniformité de la répartition des sources dans l'agrégat.

La hiérarchie des sources pourrait être mieux adaptée aux sources présentes dans la scène. L'utilisation d'une hiérarchie de volumes englobants ou d'un *kd-tree* [FS88] permettrait d'avoir des agrégats mieux formés et moins dépendants de la régularité de la subdivision de l'*octree*.

La source virtuelle pourrait aussi être plus complexe qu'une seule source ponctuelle. Une source étendue, quelques sources ponctuelles ou une source avec fonction d'émission pourraient être de meilleures représentations dans certains cas.

7.1.3 Sources étendues

L'extension de ce travail à des sources étendues serait très intéressante. Elle demanderait cependant de développer d'autres critères d'erreur pour ces nouveaux types de sources. L'illumination d'une source étendue pourrait plus simplement être représentée par un agrégat de sources ponctuelles. Ceci serait particulièrement intéressant car la méthode pourrait s'adapter aux sources de toutes les formes. Il suffirait de trouver une

façon adaptée de générer une distribution de sources ponctuelles qui remplaceraient la source étendue [VG84].

7.1.4 Éclairage indirect

Adapter notre méthode à la simulation complète d'éclairage permettrait d'obtenir des images d'un réalisme encore plus grand. L'extension aux méthodes Monte Carlo devrait se faire sans trop de problèmes puisqu'elles utilisent le tracé de rayon. La partie difficile serait d'adapter le calcul des probabilités d'échantillonnage à l'utilisation de la hiérarchie.

7.1.5 Modèles de réflexion

Tel que nous l'avons mentionné, notre méthode pourrait être modifiée pour donner de meilleurs résultats pour les surfaces spéculaires tout en bornant l'erreur faite pour de telles surfaces. Il est sûrement possible d'étendre sans trop de difficultés la méthode à des modèles de réflexion relativement simples [LFTG97], mais plus puissants que le modèle de Phong.

7.1.6 Traitement des ombres

L'extension à des scènes contenant des ombres est primordiale pour bien démontrer l'intérêt de la méthode. Le traitement de la visibilité que nous avons présenté est intéressant, mais ne donnera sûrement pas de bons résultats pour toutes les configurations de bloqueurs possibles.

Le développement d'un critère de qualité sur la visibilité que nous calculons permettrait de décider de raffiner la visibilité lorsque c'est nécessaire. L'information d'atténuation d'un *voxel* pourrait aussi être plus riche et calculée seulement lorsque c'est nécessaire. Calculer un échantillonnage de la visibilité selon plus de trois axes pourrait permettre d'avoir une meilleure approximation. Faire l'échantillonnage selon des axes calculés à partir de la distribution et de l'orientation des objets dans un *voxel* pourrait aussi donner une approximation de qualité supérieure à coût moindre.

Les calculs de visibilité pourraient également être rendus plus simples avec l'utilisation de plusieurs sources d'informations comme le tracé de rayon, la cohérence et la visibilité volumétrique. Un traitement adéquat de ces différentes informations de

visibilité permettrait d'avoir une approximation de meilleure qualité.

Dans un même ordre d'idées, les agrégats qui contiennent des objets auront une illumination bien particulière et qui dépend de l'orientation et de la distribution des objets qu'ils contiennent. Échantillonner l'illumination résultante pourrait permettre de traiter convenablement des scènes comme celle des arbres présentée à la figure 1.1 en page 6.

Avec notre méthode, il est possible de calculer efficacement des images contenant énormément de sources lumineuses. Ceci donne beaucoup plus de liberté dans la création de scènes en réduisant la contrainte du temps de calcul proportionnel au nombre de sources. Des résultats plus réalistes peuvent ainsi être atteints beaucoup plus facilement. Le besoin d'utiliser des astuces pour avoir une image dont l'éclairage semble complexe, mais est en réalité très simple, est éliminé. L'éclairage résultant de beaucoup de sources est plus réaliste et nous connaissons la différence maximale entre l'image que nous avons et le calcul complet par tracé de rayon standard.

Bibliographie

[AW87] John Amanatides et Andrew Woo. « A fast voxel traversal algorithm for ray tracing ». In G. Marechal, éditeur. *Eurographics '87*, pages 3–10. North-Holland, août 1987.

[AWG78] P. Atherton, K. Weiler et D. Greenberg. « Polygon Shadow Generation ». In *Computer Graphics (SIGGRAPH '78 Proceedings)*, volume 12, pages 275–281, août 1978.

[BB84] L. S. Brotman et N. I. Badler. « Generating Soft Shadows with a Depth Buffer Algorithm ». *IEEE Computer Graphics and Applications*, volume 4, numéro 10, pages 71–81, octobre 1984.

[Ber86] P. Bergeron. « A General Version of Crow'S Shadow Volumes ». *IEEE Computer Graphics and Applications*, volume 6, numéro 9, pages 17–28, 1986.

[Cat74] Edwin E. Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces.* Thèse de doctorat, Dept. of CS, U. of Utah, décembre 1974.

[CF89] Norman Chin et Steven Feiner. « Near Real-Time Shadow Generation Using BSP Trees ». In Jeffrey Lane, éditeur. *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23, pages 99–106, juillet 1989.

[CF92] Norman Chin et Steven Feiner. « Fast object-precision shadow generation for areal light sources using BSP trees ». In David Zeltzer, éditeur. *Computer Graphics (1992 Symposium on Interactive 3D Graphics)*, volume 25, pages 21–30, mars 1992.

[CK92] H. K. Choi et C. M. Kyung. « Pysha : a Shadow-Testing Acceleration Scheme for Ray Tracing ». *Computer-aided design*, volume 24, numéro 2, février 1992.

[CMS87] Brian Cabral, Nelson Max et Rebecca Springmeyer. « Bidirectional Reflection Functions From Surface Bump Maps ». In Maureen C. Stone, éditeur. *Computer Graphics (SIGGRAPH '87 Proceedings)*, volume 21, pages 273–281, juillet 1987.

[CPC84] Robert L. Cook, Thomas Porter et Loren Carpenter. « Distributed Ray Tracing ». In *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pages 137–45, juillet 1984.

[Cro77] Franklin C. Crow. « Shadow Algorithms for Computer Graphics ». In James George, éditeur. *Computer Graphics (SIGGRAPH '77 Proceedings)*, volume 11, pages 242–248, juillet 1977.

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

[EK89] K. S. Eo et C. M. Kyung. « Hybrid Shadow Testing Scheme for Ray Tracing ». *Computer-Aided Design*, volume 21, numéro 1, pages 38–48, janvier/février 1989.

[FS88] Donald Fussell et K. R. Subramanian. « Fast Ray Tracing Using K-D Trees ». Technical Report TR-88-07, U. of Texas, Austin, Dept. Of Computer Science, mars 1988.

[FTI86] Akira Fujimoto, Takayuki Tanaka et Kansei Iwata. « Arts : Accelerated Ray-Tracing System ». *IEEE Computer Graphics and Applications*, pages 16–26, avril 1986.

[Gla84] Andrew S. Glassner. « Space Subdivision For Fast Ray Tracing ». *IEEE Computer Graphics and Applications*, volume 4, numéro 10, pages 15–22, octobre 1984.

[Gla89] Andrew (editor) Glassner. *An Introduction to Ray Tracing*. Academic Press, 1989.

[HG86] E. A. Haines et D. P. Greenberg. « The Light Buffer : a Shadow Testing Accelerator ». *IEEE Computer Graphics and Applications*, volume 6, numéro 9, pages 6–16, 1986.

[HSA91] Pat Hanrahan, David Salzman et Larry Aupperle. « A Rapid Hierarchical Radiosity Algorithm ». In Thomas W. Sederberg, éditeur. *Computer*

Graphics (SIGGRAPH '91 Proceedings), volume 25, pages 197–206, juillet 1991.

[HTSG91] Xiao D. He, Kenneth E. Torrance, Francois X. Sillion et Donald P. Greenberg. « A Comprehensive Physical Model for Light Reflection ». In Thomas W. Sederberg, éditeur. *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 175–186, juillet 1991.

[HW91] Eric Haines et John Wallace. « Shaft culling for efficient ray-traced radiosity ». In *Eurographics Workshop on Rendering*, 1991.

[JC95] Henrik Wann Jensen et Niels J. Christensen. « Photon maps in Bidirectional Monte Carlo Ray Tracing of Complex Objects ». *Computers and Graphics*, volume 19, numéro 2, pages 215–224, 1995.

[Kaj86] James T. Kajiya. « The Rendering Equation ». In David C. Evans et Russell J. Athay, éditeurs. *Computer Graphics (SIGGRAPH '86 Proceedings)*, volume 20, pages 143–150, août 1986.

[Kap87] Michael R. Kaplan. « The Use of Spatial Coherence in Ray Tracing ». In David E. Rogers et Ray A. Earnshaw, éditeurs. *Techniques for Computer Graphics*, pages 173–193. Springer Verlag, 1987.

[Kel97] Alexander Keller. « Instant Radiosity ». In Turner Whitted, éditeur. *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 49–56. ACM SIGGRAPH, Addison Wesley, août 1997.

[KK89] James T. Kajiya et Timothy L. Kay. « Rendering Fur with Three Dimensional Textures ». In Jeffrey Lane, éditeur. *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23, pages 271–280, juillet 1989.

[KV84] James T. Kajiya et Brian P. Von Herzen. « Ray Tracing Volume Densities ». In Hank Christiansen, éditeur. *Computer Graphics (SIGGRAPH '84 Proceedings)*, volume 18, pages 165–174, juillet 1984.

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

[MS95] Paulo W. C. Maciel et Peter Shirley. « Visual Navigation of Large Environments Using Textured Clusters ». In Pat Hanrahan et Jim Winget, éditeurs. *1995 Symposium on Interactive 3D Graphics*, pages 95–102. ACM SIGGRAPH, avril 1995.

[Ney96] Fabrice Neyret. « Synthesizing Verdant Landscapes using Volumetric Textures ». In Xavier Pueyo et Peter Schröder, éditeurs. *Eurographics Rendering Workshop 1996*, pages 215–224, New York City, NY, juin 1996. Eurographics, Springer Wein.

[Pho75] Bui-T. Phong. « Illumination for Computer Generated Pictures ». *Communications of the ACM*, volume 18, numéro 6, pages 311–317, juin 1975.

[PJ91] Andrew Pearce et David Jevans. « Exploiting Shadow Coherence in Ray Tracing ». In *Proceedings of Graphics Interface '91*, pages 109–116, juin 1991.

[RPV93] Holly Rushmeier, Charles Patterson et Aravindan Veerasamy. « Geometric simplification for indirect illumination calculations ». In *Proceedings of Graphics Interface '93*, pages 227–236, Toronto, Ontario, Canada, mai 1993. Canadian Information Processing Society.

[RW80] S. M. Rubin et T. Whitted. « A 3-Dimensional Representation for Fast Rendering of Complex Scenes ». *Computer Graphics*, volume 14, numéro 3, pages 110–116, juillet 1980.

[SAG94] Brian Smits, James Arvo et Donald Greenberg. « A Clustering Algorithm for Radiosity in Complex Environments ». In Andrew Glassner, éditeur. *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 435–442. ACM SIGGRAPH, ACM Press, juillet 1994.

[Shi92] Mikio Shinya. « Hierarchical 3D texture ». In *Graphics Interface '92 Workshop on Local Illumination*, pages 61–67, mai 1992.

[Sil94] François Sillion. « Clustering and Volume Scattering for Hierarchical Radiosity Calculations ». In *Fifth Eurographics Workshop on Rendering*, pages 105–117, Darmstadt, Germany, juin 1994.

[SP94] François Sillion et Claude Puech. *Radiosity and Global Illumination*. Morgan Kaufmann, San Francisco, 1994.

[SS89] David Salesin et Jorge Stolfi. « The ZZ-Buffer : a Simple and Efficient Rendering Algorithm with Reliable Antialiasing ». In *Proceedings of the PIXIM '89*, pages 451–466, 1989.

[SS90] David Salesin et Jorge Stolfi. « Rendering CSG Models with a ZZ-Buffer ». In Forest Baskett, éditeur. *Computer Graphics (SIGGRAPH '90 Proceedings)*, volume 24, pages 67–76, août 1990.

[SWZ96] Peter Shirley, Chang Yaw Wang et Kurt Zimmerman. « Monte Carlo Techniques for Direct Lighting Calculations ». *ACM Transactions on Graphics*, volume 15, numéro 1, pages 1–36, janvier 1996.

[VG84] C. P. Verbeck et D. P. Greenberg. « A Comprehensive Light Source Description for Computer Graphics ». *IEEE Computer Graphics and Applications*, volume 4, numéro 7, pages 66–75, juillet 1984.

[WA90] Andrew Woo et John Amanatides. « Voxel Occlusion Testing : A Shadow Determination Accelerator for Ray Tracing ». In *Proceedings of Graphics Interface '90*, pages 213–220, mai 1990.

[War91] Gregory Ward. « Adaptive shadow testing for ray tracing ». In *Eurographics Workshop on Rendering*, 1991.

[Whi80] Turner Whitted. « An Improved Illumination Model for Shaded Display ». *Communications of the ACM*, volume 23, numéro 6, pages 343–349, juin 1980.

[Wil78] Lance Williams. « Casting Curved Shadows on Curved Surfaces ». In *Computer Graphics (SIGGRAPH '78 Proceedings)*, volume 12, pages 270–274, août 1978.

[Wil93] Nicholas Wilt. « Object–Oriented Ray Tracing in C++ ». In *Object–Oriented Ray Tracing in C++*. Wiley, New York, 1993.

[Woo93] Andrew Woo. « Efficient shadow computations in ray tracing ». *IEEE Computer Graphics and Applications*, volume 13, numéro 5, pages 78–83, septembre 1993.

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

Annexe A

Glossaire

Ce glossaire donne la définition utilisée dans le présent document pour certains termes. Comme l'utilisation du français en infographie peut parfois porter à confusion, les termes traduits ou empruntés de l'anglais se retrouvent ici.

agrégat *Cluster*

aliassage *Aliasing.* Artifices ayant la forme de marches d'escalier ou de Moirés, dus à l'échantillonnage insuffisant d'un signal.

apparence de la surface *Surface Shading.* Couleur de la surface après avoir calculé la contribution des sources et la transformation de cette illumination par la surface. Le terme illumination est parfois utilisé pour signifier apparence de la surface.

arbre *BSP* *BSP-tree*, *Binary Space Partition.*

atténuation Diminution de l'intensité lumineuse qui voyage entre deux points (régions). Elle est considérée entre 0 (aucun blocage) et 1 (toute la lumière est bloquée).

bloqueur Par rapport à un point, un bloqueur est un objet qui cache une source lumineuse. Un objet est considéré comme un bloqueur si le segment qui lie le point et la source l'intersecte. Il n'a pas besoin d'être le seul, le premier ou le dernier qui intersecte le segment.

BRDF *Bi-directional Reflectance Distribution Function.*

clipping De l'anglais *clip*. Procédure qui, à partir d'une certaine frontière, retranche les parties d'objet (ou de segment) qui se trouvent à l'extérieur. Par exemple, un

polygone en deux dimensions peut être restreint à une fenêtre d'affichage en lui retranchant les parties qui sont à l'extérieur de la fenêtre.

construction de solides *Constructive Solid Geometry, CSG.*

correction d'aliassage *Antialiasing.* Méthode réduisant les problèmes d'erreur dus à un sous-échantillonnage.

éclairage direct *Local illumination.* Éclairage causé par la lumière qui arrive directement des sources vers les surfaces (par opposition à éclairage indirect). Complément de l'éclairage indirect.

éclairage indirect *Global illumination.* Éclairage causé par la lumière qui, partant des sources, est réfléchie sur un ou plusieurs objets avant d'atteindre les surfaces. Complément de l'éclairage direct.

équation fondamentale du rendu *Rendering Equation.* Voir section 2.3 en page 12.

extinction Atténuation par unité de longueur. Représente le facteur de blocage d'un volume (ou objet). Elle est considérée comme entre 0 (transparent) et 1 (opaque).

facteur de forme *Form Factor.* Utilisé en radiosité, représente la proportion de l'énergie reçue par une surface j et en provenance d'une autre surface i. Voir [SP94, page 30–31].

facteur de forme pour ombre *Shadow-Form Factor.* Facteur de forme dérivé du facteur de forme utilisé dans les méthodes de radiosité.

highlight De l'anglais. Reflet très spéculaire s'apparentant aux reflets d'un miroir ou d'une surface métallique.

illumination Nous utilisons parfois le terme illumination pour remplacer succinctement le terme apparence de surface.

infographie Computer Graphics.

light buffer De l'anglais. Structure qui conserve la liste des objets qui sont des bloqueurs potentiels selon un échantillonnage de directions. Voir section 3.2.3 en page 21.

matériel graphique Graphics Hardware.

Monte Carlo Par abus de langage, nous utilisons Monte Carlo pour désigner le calcul de l'illumination par méthode d'intégration de Monte Carlo.

objets Par abus de langage, nous considérons comme objet toute primitive qui n'est pas une source lumineuse.

OORT *Object Oriented Ray Tracer* [Wil93]. Logiciel de tracé de rayon conçu par Nicholas Wilt.

octree De l'anglais. Méthode de subdivision hiérarchique de l'espace pour laquelle un élément, lorsqu'il est subdivisé, produit en huit sous-éléments disjoints. Ces sous-éléments sont de tailles égales et la subdivision se fait selon trois axes orthogonaux.

point de vue *View Point*. Nous parlerons aussi de caméra ou d'oeil.

rayon de l'oeil *Viewing Ray*.

rayon d'ombre *Shadow Ray*.

rendu *Rendering*.

RVB Représente une couleur ou une intensité sous la forme d'un triplet (Rouge, Vert, Bleu). Équivalent français du *RGB*.

shadow map De l'anglais. Un tampon-z est calculé à partir d'une source lumineuse et utilisé pour calculer si un point d'intersection sur un objet est dans l'ombre. Voir section 3.3.3 en page 26.

simulation d'éclairage *Global Illumination*.

sources lumineuses étendues Area Light Source.

tampon-z *Z-Buffer*.

tessellation Emprunt à l'Anglais *tessellation*.

tracé de rayon *Ray Tracing*. La traduction "lancé de rayon" est également utilisée.

visibilité point à point Détermination de la visibilité le long d'un rayon reliant deux points.

visibilité point à surface Détermination de la visibilité à l'intérieur du volume défini par le contour (projection) de la surface et un point.

visibilité volumétrique Calcul de visibilité qui se fait dans des volumes ayant une densité (contrairement à la visibilité retrouvée en tracé de rayon où se trouve un espace vide parsemé d'objets).

volume de visibilité Volume dans lequel se trouvent les occlusions potentielles d'une surface dans le cadre de la visibilité point à surface.

voxel Emprunt à la construction anglaise *voxel*, qui est dérivée de *volume element*. Analogue volumétrique du *pixel* (*picture element*). *Voxel* d'une grille régulière, d'un *octree*.