Vers un rendu réaliste interactif

Gabriel Fournier et Bernard Péroche

LIRIS, Laboratoire d'InfoRmatique en Image et Systèmes d'information
CNRS / INSA de Lyon / Université Lyon 1 / Université Lyon 2 / Ecole Centrale de Lyon
Bâtiment Nautibus
8 boulevard Niels Bohr
69622 Villeurbanne Cedex
fournier@liris.cnrs.fr, bperoche@liris.cnrs.fr

Résumé : *Dans cet article sont présentés des travaux préliminaires dont l'objectif est de parvenir à la synthèse d'images en temps interactif tout en prenant en compte l'éclairement global. Pour préserver l'interactivité, la méthode effectue un rendu progressif qui converge vers une solution exacte, en ajoutant et raffinant l' éclairement direct puis indirect. L'algorithme repose sur la réutilisation des valeurs d'éclairement déjà calculées (vecteurs lumineux) qui sont stockées aux sommets de plusieurs maillages. Pour accélérer les calculs, l'algorithme s'adapte à la répartition de l'éclairement dans la scène. Il est également fait appels aux extensions des cartes graphiques pour accélérer le rendu.*

Mots-clés : Synthèse d'images, rendu interactif, éclairement global, maillage.

1 Introduction

Beaucoup de progrès ont été réalisés dans le domaine de la synthèse d'images réalistes depuis les travaux de Whitted [Whi79] introduisant le lancer de rayons. Aujourd'hui, l'effort de recherche porte sur l'accélération des algorithmes pour permettre à l'utilisateur d'interagir avec la scène sur laquelle il travaille. Cet article présente une architecture logicielle pour calculer des images prenant en compte l'éclairement global en temps interactif. L'ensemble des idées présentées dans cet article n'a pas encore été implanté dans le logiciel qui ne remplit pas encore son objectif, mais s'en approche. Le choix a été fait de privilégier l'interactivité par rapport à la qualité mais, pour le confort de l'utilisateur, on a choisi de ne dégrader que la qualité du calcul de l'éclairement. L'utilisateur peut ainsi au moins visualiser la géométrie de la scène qui reste exacte comme le propose [TPWG02], contrairement à de nombreux autres algorithmes progressifs qui fournissent des premières images très floues car sous-échantillonnées ou ayant des parties manquantes (comme le "render cache" par exemple). Le calcul de l'éclairement est progressif et converge vers la solution exacte du problème. Notre méthode consiste à séparer le rendu de la couleur de celui de la luminance. Cette dernière est calculée progressivement aux sommets de plusieurs maillages qui permettent de réutiliser des valeurs déjà calculées. La combinaison de la couleur et de la luminance est fait sur la carte graphique au moment de la génération de l'image affichée à l'utilisateur.

Dans le paragraphe 2 de cet article, nous allons présenter les travaux antérieurs aux nôtres dont nous nous sommes inspirés. Nous allons ensuite, dans le paragraphe 3, donner un aperçu global de l'architecture de notre logiciel avant d'en préciser un certain nombre de détails dans le paragraphe 4. Puis, nous comparerons, dans le paragraphe 5, nos travaux avec les autres méthodes de rendu interactif existantes et finirons par donner quelques idées que nous n'avons pas encore pu implémenter dans le paragraphe 6.

2 Travaux antérieurs

Les travaux présentés dans cet article reposent sur plusieurs idées et algorithmes :

- le lancer de rayon et les méthodes de Monte-Carlo ;
- l'utilisation d'un cache d'échantillons ;
- la progressivité du calcul de rendu ;
- la réutilisation d'échantillons des images précédentes ;
- une meilleure exploitation des possibilités du matériel.

Dans ce paragraphe, nous allons faire quelques rappels sur les méthodes dont nous nous sommes inspirés dans nos travaux.

2.1 Les vecteurs lumineux

L'idée d'un cache de luminance a été introduite par Ward et Heckbert [WH92] puis développée par Zaninetti et al. [ZSP98] qui proposent la méthode des vecteurs lumineux. Cette méthode consiste à ne calculer les luminances qu'en un nombre limité de points de l'espace, pour les stocker sous forme de vecteurs lumineux dans un kd-arbre. Ces vecteurs sont constitués essentiellement de la direction d'une source de lumière "virtuelle" et de sa magnitude. La méthode distingue trois formes de luminance : directe, indirecte et caustique, pour chacune desquelles elle calcule des vecteurs lumineux spécifiques. Pour obtenir la luminance en un point quelconque de l'espace, on essaye d'interpoler une valeur à partir des vecteurs lumineux déjà calculés dans le voisinage du point d'intérêt. En permettant de ne calculer la luminance qu'en un faible nombre de points, cette méthode accélère considérablement le calcul des images. L'interpolation entre les vecteurs pose quelques problèmes. En effet, les vecteurs susceptibles d'intervenir dans le calcul en cours sont pondérés selon de multiples critères. Cette pondération va aboutir dans certains cas à des artefacts sur les images générées qui semblent constituées d'un ensemble de petites taches. Malgré les améliorations apportées à la méthode [SP01], celle-ci est loin d'être interactive. Si elle permet de réutiliser des valeurs de luminance au sein d'une image à travers l'interpolation de vecteurs, elle ne permet pas de les réutiliser d'une image à une autre au sein d'une animation de type "walkthrough", car les vecteurs lumineux calculés sont dépendants de la position de l'observateur.

Pour contourner ce problème, Crespin [CP03] propose d'étendre les vecteurs lumineux. Au lieu de stocker la luminance pour un point de vue donné, il propose de la calculer pour un certain nombre de directions et de la stocker sous forme de vecteur lumineux dans une structure rassemblant ensemble plusieurs vecteurs lumineux de même origine vus sous des directions différentes. Les calculs supplémentaires, n'impliquant que le filtrage de l'éclairement par la BRDF (Fonction de Distribution de la Reflectance Bidirectionnelle), sont peu coûteux par rapport au coût de l'envoi des mêmes rayons plusieurs fois pour chaque direction. Ainsi en interpolant entre les directions d'observation de vecteurs lumineux de même origine et entre divers vecteurs d'origine différente, on peut obtenir une image en un point quelconque de la scène. Malgré tout, cette méthode n'est pas encore réellement interactive : elle demande un certain temps de pré-calcul.

2.2 Le rendu progressif

L'approche des vecteurs lumineux consiste à stocker des échantillons dans l'espace objet et à réaliser les interpolations dans cet espace. Une autre solution consiste à travailler dans l'espace image. Ainsi la méthode de Painter et Sloan [PS89] consiste à diviser l'image selon un kd-arbre dont on subdivise progressivement les feuilles selon le nombre de pixels qu'elles couvrent et la variance des échantillons qu'elles contiennent. L'idée de travailler dans l'espace image a été reprise par Pighin [PLS97] qui crée une triangulation à partir d'un ensemble d'échantillons de l'image. Ces échantillons, calculés complètement, sont placés autour des discontinuités pour que celles-ci soient correctement rendues. Les triangles générés peuvent être affichés par le matériel qui effectue les interpolations. Ces techniques ne permettent pas encore l'interactivité, mais proposent un rendu progressif utile pour visualiser rapidement une scène.

La méthode "tapestry" de Simmons et Séquin [SS00] permet un début d'interactivité en utilisant, selon les termes des auteurs, un maillage 2D et demi. Leur maillage est projeté sur une sphère autour de l'observateur, ce qui permet de ne pas tout recalculer lorsque l'observateur bouge un peu. Cependant les discontinuités au niveau de la géométrie et des ombres restent floues au début du rendu.

2.3 L'interactivité grâce au découplage du rendu

Pour améliorer l'interactivité, Walter et al. [WDP99] proposent de séparer le calcul de l'éclairement de la boucle d'affichage et d'interaction avec l'utilisateur. En stockant les échantillons dans un "render cache", ces échantillons peuvent être réutilisés dans les images suivantes en les re-projetant à l'écran. Si cette méthode permet de combiner un rendu de très bonne qualité avec un affichage interactif, les images qu'elle génère contiennent de

nombreuses zones totalement absentes lorsqu'un utilisateur se déplace dans une partie de la scène qu'il n'a pas encore visualisée et dont les échantillons sont absents du "render cache".

En séparant le rendu de la géométrie et des textures des calculs d'éclairement effectués aux sommets d'un maillage hiérarchique de la scène appelé "shadow cache", Tole et al. [TPWG02] permettent un rendu avec une géométrie exacte. Lorsque l'observateur se déplace ou lorsque la scène est modifiée, les échantillons du "shadow cache" sont progressivement mis à jour, améliorant petit à petit la qualité de l'image. De cette façon, l'utilisateur peut interagir avec la scène en conservant une qualité d'image bien supérieure à celle du "render cache".

Dmitriev et al. [DBMS02] atteignent également l'interactivité en séparant le rendu de l'éclairement direct fait sur le matériel du calcul de l'éclairement indirect par suivi de chemins. Les photons lancés dans la scène sont stockés aux sommets d'un maillage dense de la scène. Pour parvenir à l'interactivité, ils utilisent des photons pilotes. Les photons sont regroupés par paquets selon leur origine et leur direction, dont un photon caractéristique est dit pilote. Lorsque la scène est modifiée, seuls les photons pilotes, dans un premier temps, sont re-émis afin de détecter les changements dans la scène pour ne re-émettre, ensuite, que les paquets de photons affectés par le changement dans la scène. L'ordre d'émission des photons a pour but de minimiser à moindre coût l'erreur perceptible par l'observateur. Pour cela, ils s'appuient sur la propriété de masquage du système visuel.

2.4 L'interactivité par la force brute

Une autre approche pour atteindre l'interactivité est celle de Wald et al. [WSBW01] qui n'utilisent pas de cache d'échantillons pour réduire le nombre de calculs, mais optimisent le lancer de rayons. Ils regroupent les rayons par quatre et les déplacent ensemble dans la scène. Le gain de temps est produit par l'amélioration de l'utilisation du cache du processeur et par l'emploi d'instructions SIMD. Les images sont ainsi générées en temps interactif et la méthode permet de travailler sur des scènes de plusieurs millions de triangles plus efficacement qu'avec un rendu matériel (à partir d'un certain nombre de triangles). Cependant, ces images ne prennent pas en compte l'éclairement global. Pour en tenir compte, Wald et al. [WKB+02] proposent d'utiliser la méthode baptisée "instant radiosity" de Keller [Kel97]. Cependant le temps interactif n'est atteint que sur une grappe de PC. De plus, cette méthode qui consiste à rajouter une cinquantaine de sources secondaires à la scène, ne converge pas vers la solution exacte, même si elle améliore la qualité des images. Enfin, certains effets de l'éclairement global comme le "color bleeding" ne sont pas pris en compte.

Le calcul de l'éclairement global en temps interactif, voire en temps réel, n'est pas encore une réalité, mais les derniers travaux présentés dans ce paragraphe s'en rapprochent.

3 Aperçu global de notre méthode

Notre stratégie pour atteindre le temps interactif est de limiter le nombre de points de l'espace où l'éclairement est complètement échantillonné, en réutilisant, au sein d'une même image et au sein d'une même séquence, les valeurs d'éclairement déjà calculés et en adaptant le calcul d'un éclairement, lorsqu'il a lieu, à sa contribution perceptible à l'image. Pour atteindre cet objectif, nous avons choisi de reprendre l'idée de découplage de l'affichage et du calcul de l'éclairement du "render cache" [WDP99]. Nous reprenons également les principes du "shadow cache" [TPWG02], en utilisant un maillage pour stocker la luminance dans l'espace objet et en séparant l'affichage de la géométrie et de la texture de l'affichage de la luminance qui est calculée progressivement. Pour effectuer nos calculs d'éclairement, nous utilisons les algorithmes d'échantillonnage proposés dans la méthode des vecteurs lumineux [ZSP98, SP01] que nous avons adaptée.

Notre système repose donc sur plusieurs maillages de la scène aux sommets desquels nous stockons des vecteurs lumineux résumant l'éclairement reçu en ces points. Ces maillages permettent d'interpoler une valeur de luminance en tout point de la scène. Pour que l'interpolation n'induise qu'une erreur minimum, le maillage est progressivement subdivisé dans les régions où la luminance est discontinue. Nous utilisons un maillage par source de lumière pour l'éclairement direct et un maillage pour l'éclairement indirect. Dans ces maillages, nous stockons l'éclairement avant réflexion par la BRDF des objets ; ainsi il pourra être réutilisé après déplacement de l'observateur pour générer de nouvelles images. Lors du calcul d'une image, il restera à réfléchir l'éclairement incident stocké selon la BRDF : ceci n'est malheureusement possible que pour l'éclairement direct de sources ponctuelles. Lorsque la source de lumière est étendue, le vecteur lumineux stocké dans le maillage résume tous les rayons

envoyés lors de son calcul, il n'est plus alors possible de réfléchir chacun de ces rayons par la BRDF. En ne réfléchissant que le vecteur lumineux, nous effectuons une approximation qui peut être non négligeable dans le cas de très grandes sources étendues qu'il vaut mieux alors subdiviser. De la même façon, nous sommes conduits à négliger la BRDF dans les calculs de l'éclairement indirect, car conserver la direction et l'énergie de chacun des rayons utilisés dans les calculs, afin de les filtrer à nouveau par la BRDF, serait trop coûteux en espace mémoire.

Pour générer une image, nous utilisons la carte graphique au moyen d'OpenGL. Chaque image est affichée progressivement. Pour préserver l'interactivité, l'image est tout d'abord dessinée avec des valeurs de luminance calculées uniquement aux sommets du maillage géométrique des objets, puis au fur et à mesure que le maillage de la luminance est raffiné, l'image est mise à jour. Pour ne pas redessiner l'image complètement à chaque mise à jour (triangle par triangle), elle est dessinée en plusieurs passes. Une première passe génère une image contenant en chaque pixel la couleur de l'objet vu en ce pixel, puis une seconde passe génère progressivement une image contenant la luminance de chaque pixel. Une dernière étape consiste à combiner les deux images en appliquant un opérateur de reproduction de tons pour obtenir l'image finale. Lorsque le maillage de luminance est mis à jour, l'image de luminance est mise à jour rapidement en ne dessinant que les quelques triangles qui ont changé, puis la dernière passe est exécutée pour afficher une image mise à jour. Pour ne pas appliquer trop souvent la dernière passe, qui, bien que rapide, a tout de même un certain coût, nous n'effectuons pas cette dernière à chaque mise à jour de l'image de luminance, mais à intervalles de temps fixe, ce qui laisse assez de temps pour effectuer plusieurs mises à jour de l'image de luminance.

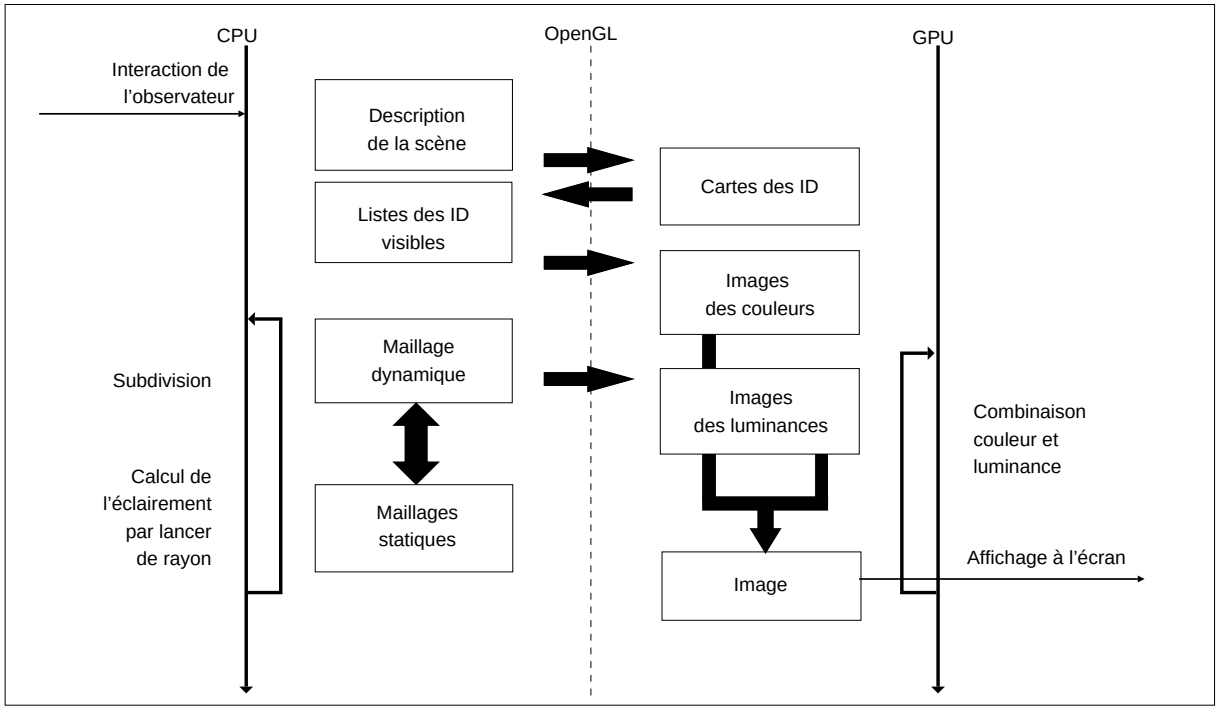

FIG. 1 – *Architecture du système.*

4 Détails de l'implémentation

Nous allons, dans ce paragraphe, décrire plus en détails le fonctionnement de notre cache d'éclairements, la méthode de calcul des vecteurs lumineux indirects et comment nous générons l'image finale à partir des images de couleurs et de luminances.

4.1 Cache d'éclairement

4.1.1 Maillages

Pour bénéficier de l'accélération des cartes graphiques comme des optimisations du lancer de rayon décrites par Wald et al. [WSBW01], nous avons fait le choix de restreindre la description de la scène à un ensemble de triangles. Ceci nous permet également de simplifier les interpolations de vecteurs de la méthode des vecteurs lumineux. En effet, dans cette dernière, les vecteurs étaient calculés en des points quelconques de la surface des objets. Lorsqu'une interpolation était nécessaire pour obtenir une nouvelle valeur de luminance, il fallait choisir un voisinage dans lequel chercher des vecteurs lumineux déjà calculés. Puis, parmi ces vecteurs, il fallait rejeter ceux qui ne correspondaient pas à la bonne surface, et il fallait également éliminer les vecteurs calculés en un point d'une surface ayant une normale trop différente de celle du point d'interpolation. Après avoir pondéré les différents vecteurs, on pouvait enfin réaliser l'interpolation. Cette méthode conduisait à des interpolations coûteuses et de qualité difficilement maîtrisable. Pour simplifier les interpolations, nous avons décidé de ne calculer nos vecteurs lumineux qu'aux sommets d'un maillage hiérarchique triangulaire. Ainsi, pour obtenir une luminance en un point, il suffit de retrouver à quelle maille triangulaire le point appartient et de réaliser une interpolation linéaire entre les trois vecteurs déjà calculés aux sommets de la maille.

Notre solution utilise plusieurs maillages. Un premier maillage, qui sert de base aux autres, est associé à la géométrie de la scène. Les valeurs d'éclairement indépendantes de la position de l'observateur, c'est-à-dire les composantes diffuses des éclairements directs et indirects, sont stockées dans d'autres maillages, dits "statiques", que l'on construit par subdivision progressive du maillage géométrique. Nous utilisons un maillage par source de lumière pour pouvoir réutiliser ces données dans les calculs des composantes spéculaires de la luminance pour lesquelles il est nécessaire de connaître le pourcentage d'occlusion de chaque source. Nous séparons également le maillage contenant l'éclairement indirect des maillages associés à l'éclairement direct, car ses discontinuités beaucoup moins marquées et son importance visuelle moindre permettent de le raffiner moins prioritairement que le maillage de la luminance directe. Nous utilisons un dernier maillage, dit "dynamique", qui est recréé à partir des maillages "statiques", à chaque fois que l'observateur se déplace. Ce maillage contient la luminance vue par l'observateur, c'est-à-dire la combinaison des éclairements direct et indirect, diffus et spéculaire, filtrés par la BRDF des objets. C'est ce maillage qui est dessiné pour générer l'image de luminance. Pour limiter les calculs et l'occupation mémoire, ces maillages partagent leurs sommets. Pour cela, nous subdivisons toujours les triangles de la même façon : en quatre triangles, par subdivision de chaque arrête en son milieu.

4.1.2 Détermination des objets visibles

Notre algorithme consiste à échantillonner l'éclairement aux sommets de nos différents maillages, puis à les raffiner. Dans le cas d'une scène complexe, le nombre d'objets présents dans la scène fait qu'il est impossible de calculer un éclairement pour chaque sommet des maillages non raffinés (qui reposent sur le maillage géométrique de la scène) avant de générer la première image. Pour gérer cela, nous ne calculons que l'éclairement des objets visibles à l'écran ; il ne peut pas, en pratique, y en avoir plus que de pixels. Pour déterminer la liste des objets visibles, nous faisons une première passe de rendu OpenGL, où nous dessinons les identifiants des objets (adresse mémoire sur 32 bits) dans un pbuffer. Cette première passe est peu coûteuse car aucun calcul d'éclairement (même matériel) n'est fait, seul le test de profondeur de la carte est effectué. Néanmoins, pour pouvoir utiliser l'information au niveau du processeur, il est nécessaire de transférer cette image d'identifiants en mémoire centrale ; cette opération est relativement coûteuse, mais le gain de temps qu'elle procure ensuite à notre algorithme la rend finalement intéressante. Nous profitons de la présence de cette information en mémoire centrale pour compter le nombre de pixels qu'occupe chaque triangle visible ; cela sera utile pour déterminer s'il est nécessaire de subdiviser ou non chaque triangle des maillages.

4.1.3 Subdivision des maillages

Le raffinement de chacun de nos maillages se fait progressivement jusqu'à ce que chaque triangle contenant une discontinuité n'occupe plus que trois pixels à l'écran. Les triangles qui ne contiennent pas de discontinuité immédiatement perceptible sont subdivisés jusqu'à ce que leur surface dans l'espace objet soit inférieure à un seuil fixé dépendant de la volonté que l'utilisateur a de détecter les petites ombres. Un seuil trop petit va demander une place en mémoire trop importante. Pour déterminer si l'arête d'un triangle contient une discontinuité, nous étudions

le contraste de Michelson $\frac{L_{max}-L_{min}}{L_{max}+L_{min}}$ entre les deux sommets. Pour déterminer le nombre de pixels visibles à l'écran d'un triangle appartenant à un maillage, nous utilisons l'extension GL_NV_occlusion_query disponible sur notre carte (Quadro FX de NVidia). Nous utilisons cette extension à chaque mise à jour de l'image de luminance. Cette image est mise à jour à chaque fois qu'un triangle du maillage dynamique est subdivisé. Cette mise à jour pose quelques problèmes : la carte fait des approximations pour le calcul des profondeurs qui ne sont pas les mêmes pour un triangle père que pour ses quatre triangles fils. Le dessin des fils en activant le test de profondeur conduit à des artefacts très visibles car une partie de l'image n'est pas mise à jour. Pour éviter ce problème, nous utilisons le stencil buffer et l'algorithme suivant :

- activation du test de profondeur et de l'écriture dans le stencil buffer ;
- dessin du père ;
- désactivation du test de profondeur, activation du stencil test et de l'écriture dans le zbuffer ;
- dessin de chaque fils avec remise à zéro du stencil test.

La subdivision des maillages est dirigée par la subdivision du maillage dynamique. Le choix de l'élément du maillage dynamique à subdiviser est fait aléatoirement. Pour obtenir la luminance en un sommet du maillage dynamique, nous allons chercher les valeurs d'éclairement stockées dans les maillages statiques sous-jacents. Si nécessaire, ces maillages seront subdivisés à ce moment-là.

4.1.4 Gestion de la mémoire

Tous ces maillages sont très coûteux en espace mémoire, et il est impossible de les conserver complètement subdivisés. Ainsi, lorsqu'un triangle subdivisé jusqu'au seuil de détection des petites ombres ne contient pas de discontinuité, il est supprimé, et ses parents également, récursivement, jusqu'au premier parent qui contient une discontinuité. Ceci permet d'économiser le plus longtemps possible la mémoire. Mais, même en procédant ainsi, la mémoire arrivera à saturation et il faudra supprimer des éléments des maillages statiques. Nous faisons alors le choix de supprimer prioritairement les éléments des maillages d'éclairement direct de sources ponctuelles qui sont les plus rapides à re-calculer. Nous envisageons de stocker sur disque les valeurs d'éclairement des sources surfaciques et les valeurs d'éclairement indirect tant elles sont longues à calculer par rapport au temps nécessaire à leur écriture et relecture sur un disque dur. Toutes ces manipulations de mémoire, en utilisant des allocations dynamiques, sont relativement coûteuses dès que l'on commence à supprimer des éléments. Pour éviter de passer par le système d'exploitation pour la gestion de la mémoire, nous avons choisi de pré-allouer la totalité de la mémoire disponible sur le système (qui n'est dédié qu'à la synthèse d'image) et de gérer nous même les allocations et désallocations de mémoire au moyen de listes chaînées d'éléments.

4.2 Échantillonnage de l'éclairement indirect

En chaque sommet du maillage de l'éclairement indirect, nous échantillonnons cet éclairement au moyen d'un hémisphère que nous subdivisons en cellules à travers lesquelles nous envoyons des rayons. Nous pondérons leur contribution en fonction de l'angle solide de la cellule à travers laquelle ils passent. La méthode consistant à échantillonner uniformément l'hémisphère donne de bon résultats à partir d'un nombre de rayon important mais est malheureusement coûteuse en temps. Pour gagner du temps sans perdre en précision, il nous faut échantillonner l'éclairement plus finement autour de ses discontinuités et plus grossièrement les zones où il est homogène. Nous avons retravaillé l'algorithme basé sur la variance des rayons proposé par Serpaggi et Péroche [SP01] qui ne donnait pas toujours les résultats escomptés.

Notre algorithme adapte le nombre de rayons utilisés pour chaque hémisphère et, en chacun, il concentre les rayons dans les zones d'intérêt. Pour concentrer les rayons dans les zones où l'éclairement n'est pas homogène, nous testons, après chaque subdivision de cellule, si l'éclairement dans cette cellule est homogène. C'est-à-dire, si après subdivision, l' éclairement recueilli par chacun des rayons envoyés dans la cellule se trouve dans une certaine bande (de 5 à 20%) autour de la luminance recueillie avant subdivision. Si c'est le cas, la cellule est dite homogène et ne sera plus subdivisée. Pour savoir si assez de rayons ont été envoyés dans un hémisphère, nous utilisons deux critères. Il faut que l'angle solide formé par les cellules homogènes soit supérieur à un seuil (selon la qualité de l'image et le temps que l'on souhaite y passer, on peut fixer ce seuil entre 50% et 90% de l'angle solide de l'hémisphère). Il faut, ensuite, que l'estimation de l'éclairement sur l'hémisphère ait convergé, pour cela nous regardons si $\frac{\Delta_{lum}}{\Delta_{ray}} \leq \epsilon$ où Δ_{lum} est la différence entre l'éclairement à l'itération i et l'éclairement

à l'itération $i-1$ et où Δ_{ray} est la différence entre le nombre de rayons lancés à l'itération i et ce nombre à l'itération $i-1$. Ce critère de convergence permet d'interrompre le calcul quand l'envoi de nouveaux rayons n'a pratiquement plus d'effet sur la luminance globale de l'hémispère. Pour faire converger plus vite l'estimation de l'éclairement indirect, et pour ne pas sortir trop tôt de la boucle après satisfaction du critère de convergence, les cellules à subdiviser sont ordonnées selon leur homogénéité ; les cellules les moins homogènes seront subdivisées en premier. La subdivision initiale de l'hémisphère a une influence sur le calcul qui sera fait. Dans les scènes peu éclairées, où la lumière provient d'une zone assez limitée, il faudra que la subdivision initiale soit assez fine pour que cette direction soit repérée. Dans des scènes éclairées normalement (boîte de Cornell par exemple), nous avons utilisé une subdivision initiale en 12 cellules : 3x4. Chaque cellule subdivisée l'est à angle solide constant, et un seul rayon est envoyé par cellule.

4.3 Génération de l'image finale

La génération de l'image finale nécessite de combiner l'image de couleurs et l'image de luminances, toutes deux stockées sur la carte graphique. Effectuer cette combinaison sur le processeur central est relativement coûteux car cela nécessite de transférer les deux images de la carte graphique en mémoire vive, de les combiner, puis de dessiner l'image résultante à l'écran. La dernière génération de carte graphique disposant de "fragment program" permet de réaliser la combinaison sur la carte, évitant les coûteux transferts d'images de la carte vers la mémoire vive. Le modèle de Phong permet de décomposer la luminance de la façon suivante :

$$L_r(x,\vec{\omega_r}) = [d\ k_d\ (N.\vec{\omega_i}) + s\ k_s\ F_s(\vec{\omega_i},\vec{\omega_r})]\ L_i(x,\vec{\omega_i})$$

Nous avons choisi d'utiliser cette même décomposition pour nos images. Nous utilisons la possibilité des cartes graphiques de gérer deux couleurs simultanément lors du rendu pour générer des images contenant à la fois la couleur spéculaire $(s\ k_s)$ et la couleur diffuse $(d\ k_d)$ des objets, codées chacune en flottant sur 3 fois 16 bits, que nous combinons dans une image de 128 bits stockée sur la carte sous forme de texture. Nous procédons de la même façon pour l'image de luminance contenant une composante spéculaire $(F_s(\vec{\omega_i},\vec{\omega_r})\ L_i(x,\vec{\omega_i}))$ et une composante diffuse $((N.\vec{\omega_i})\ L_i(x,\vec{\omega_i}))$. Ceci nécessite d'utiliser l'extension GL_NV_float_buffer. Pour générer l'image finale, nous dessinons à l'écran un rectangle auquel nous attribuons la texture constituée par les deux images de couleurs et de luminances. Nous utilisons le "fragment program" pour effectuer la combinaison pour chaque pixel de la carte. Nous pouvons en profiter pour appliquer dans la même passe un opérateur de reproduction de tons, pourvu qu'il ne nécessite pas d'information globale sur l'image.

5 Résultats et discussions

L'implémentation de notre méthode n'est actuellement pas terminée, si bien que nos résultats sont aujourd'hui limités. Nous avons développé l'architecture générale du programme et nous sommes capables de générer des images en temps interactif ; cependant ces images ne contiennent pas encore l'éclairement indirect, ce qui limite leur intérêt. Néanmoins, nous avons pu tester avec succès la séparation du rendu de la géométrie de celui de la luminance ainsi que leur recombinaison au moyen d'un "fragment program" sur la carte graphique. Nous avons également pu constater l'efficacité de notre algorithme de subdivision du maillage de l'éclairement direct qui arrive à détecter les ombres directes et à les dessiner sans crénelage en un temps très bref.

Notre méthode s'apparente à celle du "shadow cache" [TPWG02] mais nous pensons qu'elle sera plus efficace. En effet, dans la méthode du "shadow cache", les valeurs de luminance sont stockées dans un unique maillage mélangeant toutes les formes d'éclairement en une seule valeur. Au contraire, dans notre méthode, l'éclairement est stocké selon sa nature (direct ou indirect) et selon sa source, dans différents maillages sous forme de vecteurs lumineux. Il nous est ainsi possible de "mieux" réutiliser les valeurs d'éclairement. Nous pouvons prendre en compte la BRDF lorsque l'utilisateur se déplace. Nous pouvons également utiliser les valeurs d'éclairement direct, déjà calculées et présentes dans le maillage, dans le calcul de l'éclairement indirect. De la même façon, les valeurs d'éclairement direct calculées lors de l'échantillonnage de l'éclairement indirect peuvent être stockées dans le maillage pour être réutilisées plus tard.

Pour l'instant, notre méthode ne permet pas de gérer les reflets, les transparences et les caustiques. Pour ce qui est des réflexions et des transparences, deux solutions s'offrent à nous. Nous pouvons les calculer avec l'éclairement indirect, au risque de devoir attendre longtemps avant qu'elles n'aient la netteté des réflexions ou des

transparences générées par l'autre solution qui consisterait à les calculer dans un nouveau maillage. Cette dernière méthode serait beaucoup plus rapide car le maillage des réflexions pourrait être raffiné presque aussi rapidement que celui des ombres directes. En effet, le calcul d'une réflexion ou d'une transparence ne nécessite que l'envoi d'un seul rayon. Le calcul des caustiques semble plus problématique ou en tout cas plus coûteux.

Notre méthode souffre d'un problème important : les vecteurs lumineux ne sont pas calculés de façon progressive. Une fois le calcul d'un vecteur terminé, il n'est pas possible, pour l'instant, d'améliorer l'information contenue dans ce vecteur. Ceci est une limitation importante, car pour obtenir rapidement une image intéressante, on peut être tenté de limiter la précision des calculs des vecteurs lumineux. Ce manque de précision, générant du bruit, sera masqué dans les premières images générées par les nombreuses interpolations, mais, à mesure que le maillage portant les vecteurs lumineux sera raffiné, le bruit deviendra visible et la qualité de l'image diminuera. Pour que notre algorithme converge vers une solution exacte, il est nécessaire de pouvoir développer un calcul progressif des vecteurs lumineux.

6 Conclusion et perspectives

Les travaux présentés dans cet article ne constituent que le début de nos recherches. Pour obtenir le plus rapidement possible des images intéressantes, nous souhaitons favoriser à plusieurs niveaux de notre algorithme les calculs ayant un fort impact perceptuel. Nous envisageons d'améliorer l'algorithme de choix de l'élément du maillage à subdiviser pour raffiner l'image prioritairement dans les zones d'intérêt : ombres nettes, reflets, halos spéculaires,... Les travaux de Yee et al. [YPG01], qui adaptent leur rendu en fonction de l'attraction visuelle des éléments de la scène, nous semblent adaptables à notre méthode. Ensuite, en nous inspirant des travaux de Ramasubramanian et al. [RPG99], nous souhaitons pouvoir adapter la subdivision du maillage et le calcul de chaque vecteur lumineux à leur position dans la scène : limiter les calculs dans les zones déjà surexposées par d'autres sources de lumière ainsi que dans les zones où de hautes fréquences dans la couleur, la luminance directe, ou la géométrie peuvent masquer les erreurs dans la luminance indirecte. Pour l'instant, notre système travaille avec le modèle d'éclairement de Phong que nous avons décomposé pour en calculer des éléments sur le processeur central et d'autres sur la carte graphique où nous les recombinons pour obtenir l'image finale. Nous envisageons d'adapter rapidement un modèle d'éclairement plus avancé à notre décomposition pour produire des images plus réalistes. Pour finir, notre système, découplant le calcul de la luminance de son affichage, peut être, de façon intéressante, réparti sur plusieurs machines pour obtenir les valeurs de l'éclairement global plus rapidement. Ce sont ces pistes de recherche que nous souhaitons explorer dans un futur proche.

Références

[CP03] Rodolphe Crespin and Bernard Péroche. Vecteurs lumineux dynamiques. Technical Report RR10.03, LIRIS, Lyon, 2003.

[DBMS02] Kirill Dmitriev, Stefan Brabec, Karol Myszkowski, and Hans-Peter Seidel. Interactive global illumination using selective photon tracing. In *Rendering Techniques 2002: 13th Eurographics Workshop on Rendering*, pages 25–36, June 2002.

[Kel97] Alexander Keller. Instant radiosity. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, pages 49–56, August 1997.

[PLS97] Frédéric P. Pighin, Dani Lischinski, and David H. Salesin. Progressive previewing of ray-traced images using image plane discontinuity meshing. In *Eurographics Rendering Workshop 1997*, pages 115–126, June 1997.

[PS89] James Painter and Kenneth Sloan. Antialiased ray tracing by adaptive progressive refinement. In *Computer Graphics (Proceedings of SIGGRAPH 89)*, volume 23, pages 281–288, July 1989.

[RPG99] Mahesh Ramasubramanian, Sumanta N. Pattanaik, and Donald P. Greenberg. A perceptually based physical error metric for realistic image synthesis. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 73–82, August 1999.

[SP01] Xavier Serpaggi and Bernard Péroche. An adaptive method for indirect illumination using light vectors. *Computer Graphics Forum*, 20(3):278–287, 2001.

[SS00] Maryann Simmons and Carlo H. Séquin. Tapestry: A dynamic mesh-based display representation for interactive rendering. In *Rendering Techniques 2000: 11th Eurographics Workshop on Rendering*, pages 329–340, June 2000.

[TPWG02] Parag Tole, Fabio Pellacini, Bruce Walter, and Donald P. Greenberg. Interactive global illumination in dynamic scenes. *ACM Transactions on Graphics*, 21(3):537–546, July 2002.

[WDP99] Bruce Walter, George Drettakis, and Steven Parker. Interactive rendering using the render cache. In *Eurographics Rendering Workshop 1999*, June 1999.

[WH92] Gregory J. Ward and Paul Heckbert. Irradiance gradients. *Third Eurographics Workshop on Rendering*, pages 85–98, May 1992.

[Whi79] Turner Whitted. An improved illumination model for shaded display. In *Proceedings of the 6th annual conference on Computer graphics and interactive techniques*, page 14. ACM Press, 1979.

[WKB+02] Ingo Wald, Thomas Kollig, Carsten Benthin, Alexander Keller, and Philipp Slusallek. Interactive global illumination using fast ray tracing. In *Rendering Techniques 2002: 13th Eurographics Workshop on Rendering*, pages 15–24, June 2002.

[WSBW01] Ingo Wald, Philipp Slusallek, Carsten Benthin, and Markus Wagner. Interactive rendering with coherent ray tracing. *Computer Graphics Forum*, 20(3):153–164, 2001.

[YPG01] Hector Yee, Sumanta Pattanaik, and Donald P. Greenberg. Spatiotemporal sensitivity and visual attention for efficient rendering of dynamic environments. *ACM Transactions on Graphics*, 20(1):39–65, January 2001.

[ZSP98] Jacques Zaninetti, Xavier Serpaggi, and Bernard Péroche. A vector approach for global illumination in ray tracing. *Computer Graphics Forum*, 17(3):149–158, 1998.