Une synthèse des variantes du lancer de rayons et du lancer de faisceaux

Jean-Marc Hasenfratz, Djamchid Ghazanfarpour

Une synthèse des variantes du lancer de rayons et du lancer de faisceaux

J-M HASENFRATZ[*] *et D. GHAZANFARPOUR*[**]

[*] *iMAGIS- GRAVIR/IMAG*
220, rue de la Chimie
BP53, 38041 GRENOBLE Cedex 9
E-mail : Jean-Marc.Hasenfratz@imag.fr

[**]*Laboratoire MSI – Université de Limoges*
ENSIL Technopole
87068 LIMOGES Cedex
E-mail : ghazanfa@ensil.unilim.fr

RÉSUMÉ. L'algorithme de lancer de rayons est considéré comme l'un des plus intéressants pour créer des images de synthèse réalistes. Cependant, deux inconvénients majeurs le caractérisent : le problème de l'aliassage (marches d'escalier, disparition des petits objets et petites ombres) et celui des importants temps de calcul. Pour résoudre ces deux problèmes, de nombreuses voies ont été explorées. Certaines d'entre elles sont des échecs, d'autres améliorent la qualité des images et d'autres encore optimisent les temps de calcul. Nous nous proposons, dans cet article de passer en revue les différentes variantes du lancer de rayons : les solutions fondées uniquement sur des rayons d'épaisseur infinitésimale, celles fondées sur une approche volumique des rayons, appelés "faisceaux" et celles combinant ces deux types de rayons. Nous présentons aussi deux méthodes utilisant ponctuellement des rayons volumiques. L'une optimise le calcul des ombres portées et l'autre la visibilité entre deux carreaux dans un algorithme de radiosité.

ABSTRACT. Ray tracing is one of the most important rendering techniques used in computer graphics. However, two fundamental problems of classical ray tracers are aliasing (jaggies, disappearing small objects and small shadows) and important computational time. Many ways have been followed to solve these two problems. Some of them failed, others improve image quality and others optimize the computation time. In this paper, we present variants of ray tracing: solutions using only infinitesimal thick rays, solutions using volumetric rays called "beams" and solutions which mix these two types of rays. We propose two algorithms that occasionally use volumetric rays. One of them optimizes shadow determination and the other the visibility between two patches in the radiosity algorithm.

MOTS-CLÉS : lancer de rayons, lancer de faisceaux, cohérence de la scène, antialiassage.

KEY WORDS: ray tracing, beam tracing, antialiasing, image space subdivision, regular spatial subdivision.

1. Introduction

Il est admis que l'algorithme de lancer de rayons est parmi ceux qui produisent les images de synthèse les plus réalistes. Cependant, il possède deux inconvénients majeurs. Le premier est bien connu en synthèse d'images, il regroupe les problèmes de marches d'escalier sur les contours et la disparition des petits objets et petites ombres (***Figure 1***), on parle de problèmes d'aliassage. Dans le cas du lancer de rayons, l'aliassage est induit par l'approche discrète utilisée (on lance un rayon au centre de chaque pixel de l'écran). On arrive cependant à traiter efficacement les problèmes de marches d'escalier avec un suréchantillonnage adaptatif [GHAZ 92]. Par contre, on ne sait pas traiter efficacement les problèmes de disparition des petits objets et des petites ombres [GHAZ 92], ceci parce qu'il est impossible d'assurer que des objets petits ou allongés ne se glissent entre les rayons lancés (***Figure 1***).

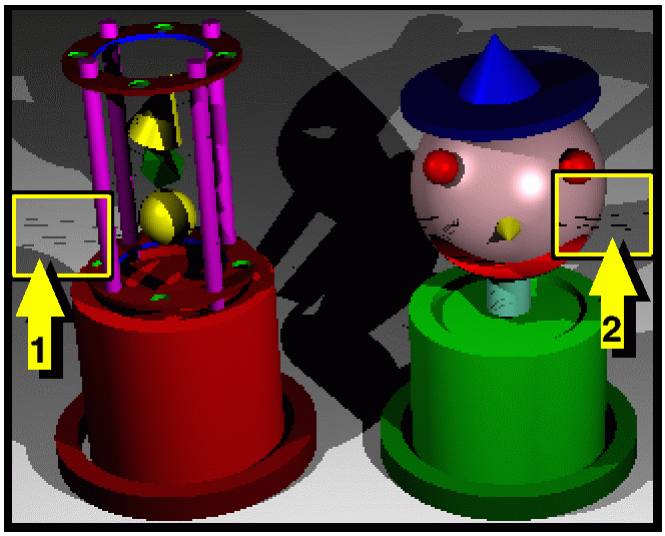


Figure 1 : Problème de disparition des petites ombres [1] et des petits objets [2] avec un lancer de rayon sans suréchantillonnage.

Le deuxième inconvénient est bien sûr l'important temps de calcul nécessaire lorsque l'on cherche à produire des images réalistes. Différentes solutions ont été proposées pour résoudre ces deux problèmes. Toutes tiennent compte de la cohérence de la scène, c'est-à-dire du fait que de nombreux rayons lancés suivent approximativement le même chemin. Nous présentons ici plusieurs algorithmes essayant d'exploiter au maximum cette cohérence des scènes pour résoudre nos deux problèmes. Nous les avons classés en trois grandes catégories :

1. Les algorithmes exploitant la cohérence de la scène pour traiter des groupes de rayons plutôt que de les considérer les uns indépendamment des autres. Les rayons sont ici d'épaisseur infinitésimale comme ceux utilisés par Whitted [WHIT 80]. De ce fait, ces algorithmes ne peuvent résoudre efficacement les

problèmes d'aliassage, en fait, le but est uniquement un gain de temps. Cette catégorie d'algorithme comporte seulement trois solutions [SPEE 85] [MULL 86] [SHIN 87].

2. Les algorithmes fondés exclusivement sur des rayons volumiques appelés faisceaux. Les formes utilisées pour ces faisceaux sont diverses ; on trouve, en particulier, des cônes [AMAN 84] et des pyramides de section polygonale quelconque [HECK 84].
3. Les algorithmes hybrides utilisant à la fois des rayons d'épaisseur infinitésimale et des faisceaux. Deux méthodes relativement proches sont décrites [MARK 90] [GHAZ 92] [GHAZ 98].

Il existe aussi des algorithmes qui n'utilisent que ponctuellement des faisceaux pour améliorer un traitement précis. Dans ce cadre, nous présentons la méthode proposée par Choi et al. [CHOI 92] pour optimiser le calcul des ombres portées et de celle de [HASE 98] pour optimiser le calcul de la visibilité entre deux carreaux dans un algorithme de radiosité.

2. Regroupement de rayons d'épaisseur infinitésimale

2.1 Prédiction des chemins empruntés par les rayons lancés

Speer et al. [SPEE 85] cherchent à limiter le nombre de calculs d'intersections rayon-objet. Pour cela, il profite de la cohérence de la scène pour construire des tunnels englobant un maximum de rayons très proches les uns des autres. Ces tunnels sont constitués de cylindres mis bout à bout. Les rayons contenus dans ces tunnels rencontrent tous les mêmes objets (***Figure 2***). On limite ainsi le nombre de calculs d'intersections.

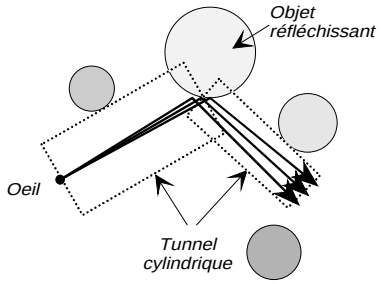


Figure 2 : Chemin emprunté par des rayons réfléchis englobés dans un tunnel

Malgré une construction rapide des tunnels et une logique simple, cette approche ne fournit pas les résultats escomptés. En effet, les tests effectués par Speer et al. sur des scènes contenant un nombre croissant de sphères montrent qu'à partir de neuf sphères le lancer de rayons conventionnel est plus rapide que la solution proposée. Ceci est principalement dû à un déséquilibre entre le temps nécessaire à la construction des différents tunnels et le nombre de rayons qui peuvent profiter de cette structure. L'auteur conclut ainsi : "*Nous avons montré qu'en dépit du degré élevé de cohérence dans une image, l'obligation de maintenir la validité des structures d'intersection fait obstacle à l'obtention de gains importants. Ces résultats donnent à penser que des méthodes algorithmiques plus fondamentales sont nécessaires pour réduire de façon substantielle les coûts de calcul du traçage de rayons lumineux*".

Cette démarche est donc globalement un échec, cependant elle met clairement en évidence la dualité entre les coûts de manipulation d'une structure de données et le gain de temps qui peut en découler.

2.2 Lancement simultané de tous les rayons à chaque niveau de la récursion

Müller [MULL 86] propose un algorithme de lancer de rayons regroupant les rayons en fonction de la profondeur de la récursion. Cette approche correspond à un parcours en largeur plutôt qu'en profondeur de l'arbre de récursion de Whitted [WHIT 80]. Ainsi, tous les rayons de vision sont traités en même temps, puis tous les rayons réfléchis et réfractés du premier niveau de récursion et ainsi de suite. Pour cela, pour chaque niveau de la récursion, l'espace de la scène est balayé à l'aide des six plans formant la boîte englobante de la scène. Supposons que l'on réalise cela pour le plan perpendiculaire à l'axe *z* et de plus petite coordonnée selon *z*. Cette face est déplacée dans la direction des *z* positifs. Durant ce balayage, seuls les rayons de visions, de réflexion et de réfraction ayant une composante positive suivant *z* sont considérés.

Le déplacement d'un de ces plans est en fait simulé à l'aide de deux listes *RE (“ray event”)* et *OE (“object event”)* d'événements discrets triés selon *z*.
RE contient les événements suivants (***Figure 3***) :

a) l'origine des rayons *r*, notée *Or* ;
b) les points d'intersection rayon-objet obtenus lors du balayage de l'espace, noté *Ir* pour la première intersection du rayon *r*.

OE contient les événements suivants :

c) la face avant des boîtes englobantes des objets, c'est-à-dire la première face rencontrée lors du balayage, notée ***f.***

Les événements de ces listes sont considérés dans l'ordre (les événements de même coordonnés en *z* sont traités dans n'importe quel ordre). On considère *R* comme une liste de rayons, vide au départ. Les traitements correspondant aux différents types d'événements sont les suivants :

d) Le rayon correspondant (en fait son point d'intersection avec le plan de balayage à cette coordonnée *z*) est inséré dans *R*. Cet événement est supprimé de *RE*.
e) Le rayon correspondant est supprimé de *R*.
f) On utilise la liste *R* pour déterminer les rayons qui traversent la face de la boîte englobante rencontrée lors de cet événement. Pour tous les rayons traversant cette face, un test d'intersection rayon-objet est effectué avec l'objet contenu dans la boîte considérée. Le point d'intersection avec l'objet, s'il existe, est ajouté à la place courante dans *R* avec *b)* pour type d'événement. S'il existe déjà un point d'intersection pour ce rayon dans *RE*, celui ayant la composante *z* la plus grande est supprimé.

L'algorithme prend fin, pour un niveau de récursion, lorsque les six plans de balayage ont traversé, dans les trois directions positives et les trois directions négatives du repère, la boîte englobante de la scène. On traite ainsi tous les rayons quelle que soit leur direction de propagation. Toutes les intersections avec les faces des boîtes englobantes sont déterminées et par conséquent toutes les intersections

possibles avec les objets. À chaque niveau de récursion, on dispose de l'ensemble des points d'intersections ainsi que des rayons incidents en ces points. On peut alors répéter l'algorithme de balayage des six plans pour le niveau de récursion suivant. Finalement, on aura parcouru l'arbre de récursion de Whitted [WHIT 80] en largeur.

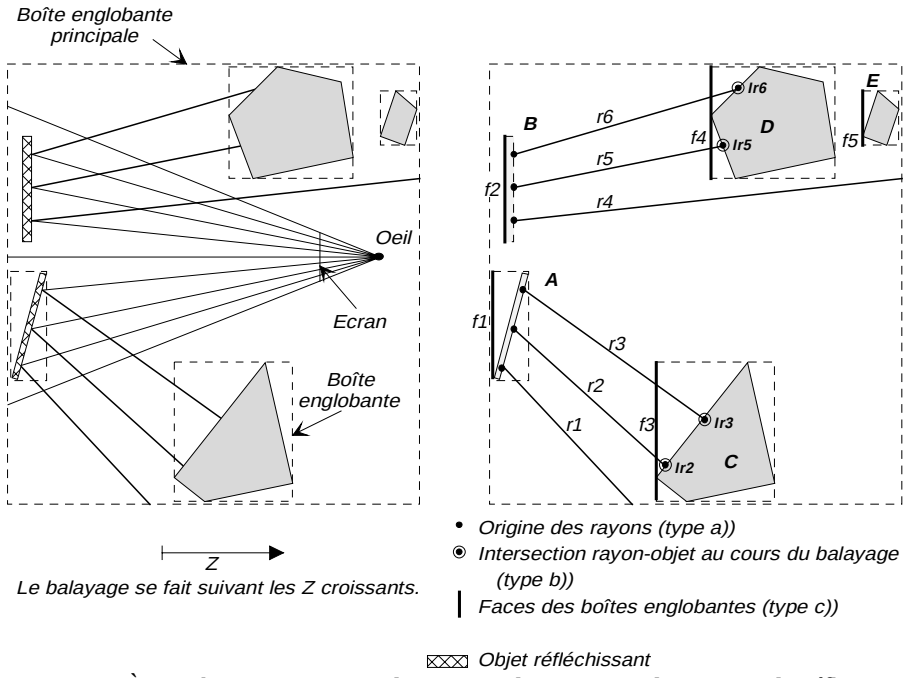


Figure 3 : À gauche : propagation des rayons de visions et des rayons de réflexion. À droite : événements de type a), b) et c) pour un niveau de récursion.

Le calcul du rendu de l'ïmage finale est proche de celui d'un lancer de rayons conventionnel. La principale différence est qu'au lieu de lancer un rayon après l'autre et de calculer la couleur pixel après pixel, on lance ici, tous les rayons en même temps et la couleur des pixels est affinée globalement. En effet, à chaque niveau de la récursion, les points d'intersection sont déterminés et mémorisés. Pour chacun de ces points, il ne reste plus qu'à lancer des rayons vers les sources lumineuses (comme dans un lancer de rayons conventionnel) pour déterminer s'il est à l'ombre ou non.

Le problème des marches d'escalier sur les contours est traité de manière très conventionnelle par un suréchantillonnage local déclenché par un contraste trop important entre des pixels voisins. Les autres problèmes d'aliassage, comme la disparition des petits objets, ne sont pas pris en compte.

Dans son article, Müller [MULL 86] ne conclut pas réellement que son algorithme a des performances meilleures, en temps de calculs, qu'un lancer de rayons conventionnel. Ceci était pourtant l'objectif principal. Par contre, l'auteur précise bien que le coût mémoire est très important. Cette approche parcourant en largeur l'arbre de récursion semble relativement limitée puisque le gain de temps

n'est pas établi mais que l'augmentation de l'espace mémoire nécessaire est importante.

2.3 Faisceau générique et quantification des approximations réalisées

Shinya et al. [SHIN 87] propose un *lancer de crayons* dont l'objectif est de diminuer les temps de calculs et d'augmenter le réalisme des images en restant fidèle aux lois de l'optique. L'idée est de développer des outils mathématiques permettant de quantifier les erreurs d'approximation commises lors de la propagation des rayons. L'évaluation de cette erreur permet, quand cela est nécessaire, d'appliquer des traitements adéquats pour améliorer la qualité du rendu.

Pour approcher l'erreur commise, les auteurs remplacent les rayons conventionnels par des ensembles de rayons ayant la même direction appelés *crayon*. Chaque crayon est composé d'un *rayon axial* et de *rayons paraxiaux* voisins du rayon axial. Le lancer d'un rayon axial permet de connaître les caractéristiques optiques (transmission, réflexion et réfraction) du milieu traversé par l'ensemble d'un crayon. L'auteur se fonde sur la théorie de l'approximation paraxiale [BORN 59], pour définir les contraintes à appliquer aux rayons paraxiaux afin que l'on puisse approcher les changements de direction du crayon par une transformation linéaire et obtenir un bon réalisme. Ces contraintes dépendent, entre autre, de la taille des pixels, de la dimension de la grille d'échantillonnage et du système optique traversé.

Dans l'absolu, cette approche est séduisante, malheureusement, la notion de *"bon réalisme"* est subjective et on ne peut donc pas quantifier cette notion. De plus, une hypothèse forte est nécessaire à l'approximation : les surfaces rencontrées par un crayon doivent être C1 continue.

Deux utilisations possibles du lancer de crayons sont présentées : un lancer de rayons fondé sur l'évaluation de l'erreur commise et une variante du *"lancer de faisceaux sur des objets polygonaux"* de proposée par Heckbert aet al. [HECK 84] traitant plus efficacement la réfraction.

2.3.1 *Lancer de rayons*

Dans cette méthode, on cherche à lancer simultanément le plus de rayons possibles. Pour cela, on détermine de manière dichotomique des ensembles de pixels pouvant être traités en même temps tout en assurant une bonne approximation. Le but est ici un gain de temps de calculs. L'algorithme utilisé est le suivant :

1. *Diviser l'écran en régions de n×m pixels (de l'ordre 5×5).*
2. *Lancer un rayon axial au centre de chaque région avec un lancer de rayons conventionnel. Calculer la matrice du système optique.*
3. *Vérifier les conditions de continuité (voir plus bas). S'il existe une arête dans cette région, utiliser un lancer de rayons conventionnel pour tous les pixels de la région.*
4. *S'il n'y a pas d'arête, calculer le facteur de tolérance et l'angle d'ouverture maximum possible du crayon pour cette région ([SHIN 87] propose une équation pour calculer cet angle maximum). En se fondant sur cet angle :*
   a. *Si l'angle d'ouverture maximum possible pour la région permet de couvrir toute la région, lancer tous les rayons paraxiaux dans la région en utilisant la matrice du système optique.*
   b. *Si l'angle d'ouverture maximum ne permet pas de couvrir au moins un pixel, utiliser un lancer de rayons conventionnel pour tous les rayons paraxiaux de la région.*
   c. *Sinon, diviser la région en sous-régions de sorte que l'angle d'ouverture maximum calculé puisse couvrir ces sous-régions individuellement. Répéter 2. à 4. pour chaque sous-région.*

Algorithme 1 : Lancer de rayons traitant simultanément plusieurs rayons.

La condition de continuité utilisée au point *3* de l'***Algorithme 1*** est très simple, elle est fondée sur la ou les surfaces traversées par le rayon axial de chaque région de l'écran. Deux cas interdisent l'utilisation du lancer de crayons :

1. Plusieurs surfaces sont traversées par des régions adjacentes : ces régions contiennent forcément une arête et la condition de continuité C1 n'est donc pas vérifiée ;
2. Des régions adjacentes traversent une même surface mais celle-ci contient une arête "brisant" la continuité C1.

Dans les autres cas, la condition de continuité est respectée et l'on peut utiliser la matrice système pour tous les rayons paraxiaux d'une même région de l'écran.

Notons qu'avec une telle condition, les petits objets peuvent facilement passer à travers les rayons. Ces cas se produisent plus fréquemment qu'avec un lancer de rayons classique où l'échantillonnage de l'écran est d'au plus un pixel alors qu'ici il correspond à des régions de 5×5 pixels.

2.3.2 Lancer de faisceaux sur des objets polygonaux et réfraction

Le lancer de faisceaux sur des objets polygonaux développé par Heckbert et al. [HECK 84] (voir ***Paragraphe 3.2***) est correct pour la réflexion. Par contre, pour la réfraction, il suppose que tous les rayons sont parallèles. Cette approximation entraîne des effets de réfraction peu réalistes. Shinya et al. [SHIN 87] proposent d'utiliser les matrices du système optique ainsi que son calcul d'évaluation de l'erreur commise pour améliorer les résultats. Malheureusement, peu d'informations sont données à ce sujet et aucune implémentation n'a été réalisée par l'auteur.

On peut supposer que le lancer de crayons intervient dans la deuxième étape de l'algorithme proposé par Heckbert et al., c'est-à-dire dans le calcul du rendu fondé sur la représentation arborescente de la scène. À ce niveau, l'ensemble des surfaces visibles est connu ainsi que le système optique pour y aboutir (il correspond aux sous-systèmes contenus dans la branche considérée). On peut donc lancer un crayon de section polygonale vers chaque surface à travers le système optique correspondant

pour calculer la couleur de la surface. Si l'erreur d'approximation pour chacun de ces crayons est trop grande, les crayons sont subdivisés afin de vérifier les conditions de tolérance.

Notons que la condition de continuité est toujours vérifiée sur les surfaces visibles. De plus, le problème des petits objets est résolu, uniquement dans l'algorithme de Heckbert et al., au moment de la construction de l'arbre représentant les surfaces visibles de la scène indépendamment de l'utilisation des crayons. Par contre, le problème des marches d'escalier n'est pas pris en compte.

2.3.3 Discussion

L'étude d'un faisceau générique permet de considérer toutes les formes possibles de faisceaux. On peut ainsi définir des critères de qualité, en particulier concernant les effets de réfraction, applicable aux algorithmes de lancer de faisceaux. Ainsi, le problème de la réfraction dans le lancer de faisceaux sur des objets polygonaux [HECK 84] peut être atténué par une meilleure approximation fondée sur les matrices du systèmes optique traversé.

Néanmoins, le lancer de crayons est uniquement possible sur des régions continues, en particulier sur des régions ne contenant pas d'arêtes. Il est donc nécessaire de faire appel à d'autres techniques le long de ces arêtes. De plus, le problème des petits objets et des petites ombres n'est pas pris directement en compte. Notons que dans le cas de l'approche de Heckbert et al. [HECK 84], la détection des petits objets et petites ombres est effectuée par les faisceaux lancés et non lors de l'utilisation de crayons.

3. Rayons volumiques

3.1 Lancer de rayons avec des cônes

Une des premières approches volumiques d'un lancer de rayons est proposée par Amanatides et al. [AMAN 84] en remplaçant les rayons conventionnels par les rayons volumiques en forme de cône. Le but annoncé de cet algorithme est, entre autres, de résoudre les problèmes d'aliassage et d'améliorer les temps de calculs. L'idée est de lancer vers la scène un cône à travers chaque pixel de l'écran afin d'englober ce pixel. On peut ainsi, en théorie, calculer la proportion des différentes surfaces visibles à travers chaque pixel et effectuer un traitement de l'aliassage efficace. Nous verrons dans ce paragraphe que les différents buts ne sont pas atteints et que de nombreux problèmes subsistent.

3.1.1 Scènes opaques et non réfléchissantes

Pour chaque pixel, le rayon conventionnel d'épaisseur infinitésimale est remplacé par un rayon volumique en forme de cône appelé *faisceau conique de vision*. Le sommet de ce faisceau conique est l'œil. Son *angle au sommet*, c'est-à-dire l'angle formé par l'axe de symétrie du cône et son bord, est tel que le rayon *R* du cône au niveau de l'écran soit au moins égal à la diagonale *D* d'un pixel (***Figure 4***). Le faisceau englobe donc la totalité du pixel.

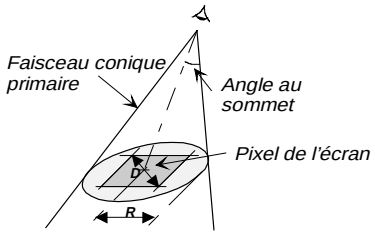


Figure 4 : Dimension d'un faisceau conique de vision.

Pour chaque faisceau conique de vision lancé, on étudie sa position par rapport à l'ensemble des objets. On en déduit l'existence d'une intersection faisceau conique-objet et uniquement la proportion de surface du cône retenue par l'objet. Pour les objets considérés (sphères, polygones et plans infinis), les calculs sont assez simples.

Une liste des objets rencontrés, triée en profondeur, est associée à chaque faisceau. Ces listes permettent le traitement de l'aliassage pour chaque pixel. Si l'objet le plus proche (le premier objet de la liste) ne couvre pas la totalité du faisceau, l'objet suivant dans la liste participe à la couleur du point. Puisqu'on ne considère que la proportion de surface retenue, le calcul de la couleur est exact pour des objets qui ne se superposent pas. Pour tenir compte de ces derniers (***Figure 5***), des informations supplémentaires peuvent être stockées dans les listes triées des objets rencontrés. Les auteurs ne donnent pas davantage de précisions sur le calcul de la couleur d'un pixel, en particulier la nature des informations supplémentaires. Kirk [KIRK 87], qui a aussi travaillé sur cet algorithme, propose une méthode pour calculer la couleur des pixels. Elle est simple mais très approximative et donne le plus souvent des résultats erronés.

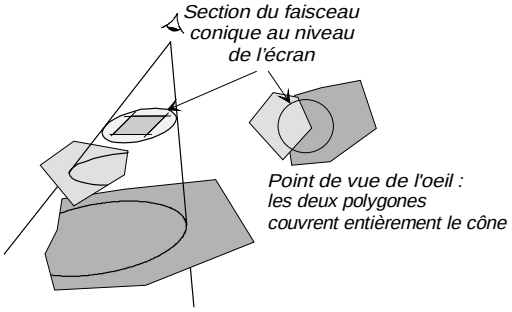


Figure 5 : Cas particulier : on ne tient pas compte des ensembles d'objets (ici deux polygones) couvrant entièrement le faisceau conique.

3.1.2 Cas de la réflexion et de la réfraction

Les effets de réflexion et de réfraction sont complexes dans le cas du lancer de cônes. L'auteur précise uniquement l'utilisation de la courbure de la surface rencontrée par le faisceau de vision pour construire et lancer des faisceaux coniques réfléchis et réfractés. Aucune précision supplémentaire n'est donnée concernant le calcul du sommet des faisceaux ni leurs angles au sommet. En particulier, on ne sait

pas toujours quelle doit être la section du faisceau conique réfléchi ou réfracté. En effet, supposons une sphère réfléchissante ou réfractante qui coupe partiellement le faisceau de vision (***Figure 6*** et ***Figure 7***). Dans ces cas, il est difficile d'imaginer la forme du faisceau conique réfléchi ou réfracté.

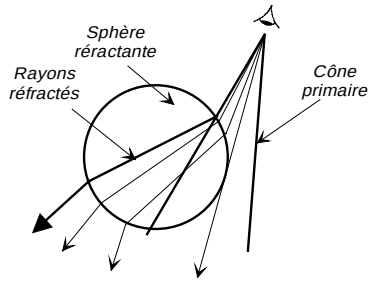


Figure 6 : Réfraction du faisceau de vision sur une partie d'une sphère.

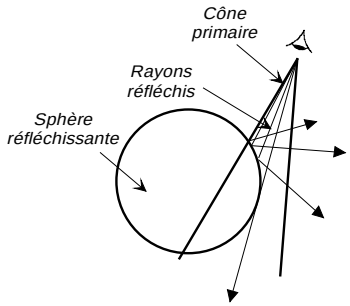


Figure 7 : Réflexion du faisceau de vision sur une partie d'une sphère.

Les effets de réflexion et de réfraction semblent donc pour le moins limités à quelques cas particuliers où les bords des objets réfléchissants ou réfractants ne sont pas à l'intérieur des cônes.

3.1.3 Discussion

Remplacer les rayons conventionnels d'épaisseur infinitésimale par des rayons volumiques en forme de cône permet de tenir naturellement compte de la cohérence de la scène. Malheureusement, cette forme en cône entraîne de nombreux problèmes. Le calcul de l'intensité lumineuse telle que Kirk [KIRK 87] le présente est très approximatif et semble pouvoir donner de bons résultats uniquement pour des scènes très (trop) simples. L'explication de la construction des faisceaux coniques réfléchis et réfractés n'a été trouvée dans aucun article. En particulier, on ne sait pas toujours quelle section doit avoir le faisceau au niveau de l'objet réfléchi ou réfracté. De manière générale, aucune équation caractérisant la variation de l'angle au sommet d'un cône traversant un système optique n'est proposée.

L'étude de cet algorithme permet donc de constater que l'utilisation d'un faisceau volumique en forme de cône est plutôt mal adaptée aux scènes complexes contenant des objets réfléchissants et réfractants. D'autres formes de faisceaux, en particulier des formes pyramidales, ont été utilisées. Le paragraphe suivant présente l'un de ces algorithmes.

3.2 Lancer de faisceaux sur des objets polygonaux

Heckbert et al. [HECK 84] proposent un *lancer de faisceaux sur des objets polygonaux*. Le but annoncé de cet algorithme est de diminuer les temps de calculs. Comme nous le verrons, des réserves à ce sujet peuvent être émises. Néanmoins, pour atteindre cet objectif, les auteurs se limitent aux scènes polygonales pour lesquelles les algorithmes de découpage ("clipping") sont bien connus. L'algorithme développé comporte deux parties distinctes :

1. la recherche des surfaces directement visibles, par réflexion ou par réfraction ;

2. le calcul de la couleur des pixels de l'ïmage.

La première étape est réalisée à l'aide d'un algorithme récursif de lancer de faisceaux. La deuxième utilise un algorithme séquentiel de remplissage de polygones de type balayage par lignes.

Dans cet algorithme, on construit une structure arborescente qui décrit le parcours, dans la scène, d'un faisceau pyramidal issu de l'œil et traversant exactement l'écran (***Figure 8***). Chaque objet qui se trouve sur le chemin de ce faisceau en retient une partie. Le faisceau est alors découpé selon la forme de l'objet (***Figure 8***) et la surface du polygone à l'intérieur de la pyramide est stockée dans l'arbre.

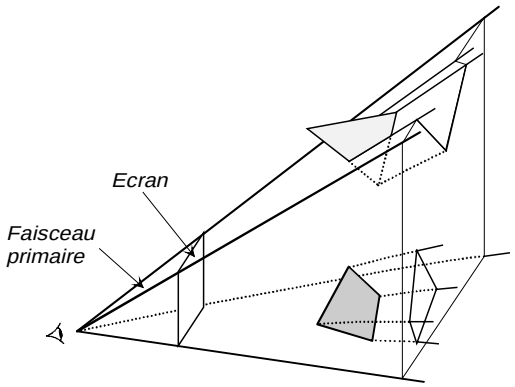


Figure 8 : Propagation du faisceau de vision dans la scène. (La pyramide possède une base uniquement pour une meilleure lisibilité de la figure)

3.2.1 Cas de la réflexion et de la réfraction

Si un objet rencontré est réfléchissant, on construit un *faisceau réfléchi* (***Figure 9***) issu de l'œil virtuel et ayant pour base la partie de la surface rencontrée. Si l'objet est réfractant, Heckbert et al. [HECK 84] généralisent l'utilisation de l'œil virtuel et proposent de placer le sommet du *faisceau réfracté* (***Figure 10***) sur la perpendiculaire au plan réfractant passant par l'œil. La position exacte dépend principalement des angles de réfractions des arêtes du faisceau.

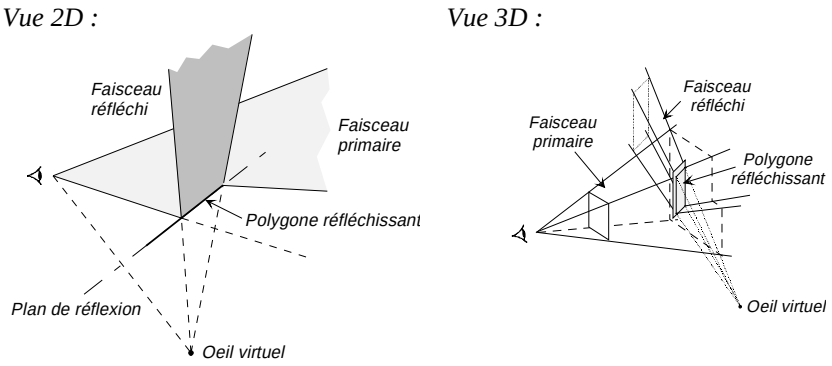


Figure 9 : Construction d'un faisceau de réflexion.

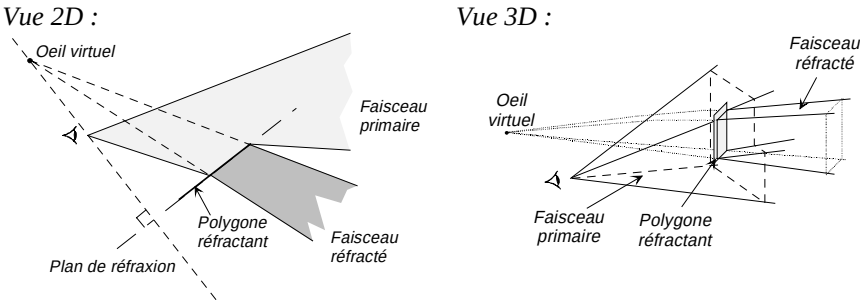


Figure 10 : Construction d'un faisceau réfracté.

3.2.2 Calcul du rendu

Pour calculer la couleur des pixels, on parcourt l'arbre décrivant la propagation du faisceau dans la scène. Au niveau de chaque nœud, on considère les polygones qui lui sont associés et on leur applique un algorithme de remplissage. On peut ainsi, avec une récursion terminale, calculer la couleur de tous les pixels de l'écran.

3.2.3 Calcul de l'ombrage

Dans le cas du lancer de faisceaux présenté par Heckbert et al. [HECK 84], le calcul des ombres fait partie d'un prétraitement. Les auteurs proposent une extension récursive de l'algorithme de Atherton-Weiler [ATHE 78]. Elle consiste à lancer un *faisceau d'ombre*, à partir de chaque source lumineuse de façon à envelopper toute la scène. Si un faisceau par source est insuffisant et qu'il ne couvre pas l'ensemble de la scène, plusieurs faisceaux, ne se superposant pas, seront utilisés. Ces faisceaux peuvent être réfléchis et réfractés, simulant ainsi les éclairages indirects. Chaque faisceau est associé à la liste des polygones qui y sont entièrement visibles. On utilise pour cela l'algorithme de calcul des surfaces visibles de Weiler-Atherton [WEIL 77]. Ces polygones éclairés sont transformés dans le repère de la scène pour être plaqués sur leurs parents respectifs. Le calcul des surfaces visibles est alors effectué grâce, de nouveau, à l'algorithme de Weiler-Atherton. Le calcul des ombres est réalisé en

utilisant les polygones “d'ombre”. Les surfaces sur lesquelles sont plaqués les polygones “d'ombre” sont éclairées et les autres sont à l'ombre.

3.2.4 Antialiassage

L'antialiassage fait partie des extensions proposées. La solution consiste à construire un graphe d'adjacence des faces à partir de la structure arborescente obtenue (l’auteur ne précise pas l’algorithme de construction). Ce graphe permet alors de trouver les contours des objets, ombres, réflexions et réfractions. Ces contours correspondent aux pixels devant être traités contre l’aliassage. Aucune précision n'est donnée quant à la manière de construire ce graphe. On peut supposer que les surfaces de polygones contenues dans l'arbre sont projetées dans le plan de l'écran en suivant le chemin inverse parcouru par le faisceau dans la branche de la surface. Un algorithme de “découpage” de polygones permet alors de construire le graphe.

3.2.5 Discussion

La représentation de la scène sous la forme arborescente correspond à une description continue dans R^2 des contours de l'image. En effet, c'est seulement au moment du calcul de la couleur des pixels qu'une définition de l'image est choisie. Cette représentation continue des contours de l'image permet alors un traitement efficace et simple des problèmes d'aliassage.

Le fait de lancer des faisceaux d’ombre vers la scène et de tenir compte, à ce niveau, des réflexions et réfractions permet de produire des éclairages indirects, très difficiles à obtenir avec un lancer de rayons classique. Des éclairages de type spot sont aussi possibles en limitant la section des faisceaux d’ombre.

Curieusement, aucun temps de calculs n'est donné alors que le but annoncé est un gain de temps. On peut supposer qu'il n'était pas optimal puisque Dadoun et al. [DADO 85] proposent une version optimisée du même algorithme. Là encore, aucune comparaison en temps n'est proposée. On peut logiquement penser que les temps de calculs sont relativement importants. En effet, de nombreux découpages entre polygones quelconques (concaves, troués,...) sont effectués, or un tel algorithme est très coûteux.

4. Algorithmes hybrides

Ce paragraphe est consacré à deux solutions hybrides utilisant à la fois un lancer de rayons conventionnel et un lancer de faisceaux. Ces solutions ont de nombreux points en commun tout en ayant été élaborées de façon totalement indépendante.

4.1 Choix dynamique entre les rayons et les faisceaux

Marks et al. [MARK 90] proposent un algorithme hybride alliant un lancer de rayons conventionnel et un lancer de faisceaux. Le but avoué est d'améliorer les temps de calculs du rendu d'une scène composée de polygones convexes. Pour cela, on lance un premier *faisceau pyramidal de vision* issu de l’œil à travers la totalité de l’écran. Si les quatre arêtes rencontrent une même face *F* et qu’aucun objet ne se

trouve dans le faisceau et entre l'écran et le plan de *F* (***Figure 11***), alors la couleur des pixels de l'écran correspondant à *F* est calculée directement pour l'ensemble de l'écran. Plus aucun test d'intersection n'est alors nécessaire.

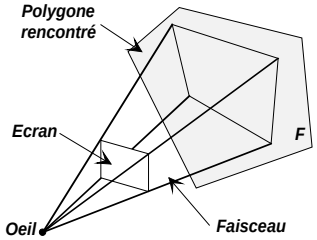


Figure 11 : Faisceau traversant la totalité d'un polygone.

Dans les autres cas, comme ceux de la ***Figure 12***, les auteurs proposent deux solutions :

1. soit on utilise un lancer de rayons conventionnel et on calcule de manière conventionnelle la couleur des pixels,
2. soit on subdivise l'écran en quatre à la manière de Warnock [WARN 69] (***Figure 13***) et on considère chacun des quatre sous-faisceaux et on réitère le processus.

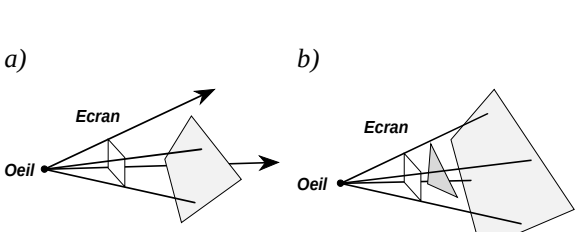


Figure 12 : a) Polygone bloquant seulement une partie du faisceau. b) Polygone "coincé" entre les arêtes du faisceau.

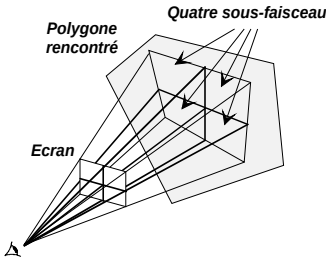


Figure 13 : Subdivision de l'écran et du faisceau.

Le critère de choix *C* entre le lancer de rayons conventionnel et le lancer de faisceaux est traduit par l'équation suivante :

Equation 4-1 :
$$C = F\frac{16 * F_w - D}{E_W^2 D}$$

où *F* est le nombre total de polygones dans la scène, *D* est la *profondeur* de la scène, F_W le nombre de polygones dans le faisceau et E_W la largeur de l'écran en pixels. La profondeur de la scène est une valeur (un peu vague) introduite par Marks et al. [MARK 90] qui peut être interprétée comme le nombre de coordonnées différentes des objets suivant l'axe *z*. Si l'expression *C* est plus grande qu'un seuil fixé empiriquement, on conclut que le niveau de cohérence de cette partie de la scène est faible et on utilise alors un lancer de rayons conventionnel. Sinon, on subdivise l'écran et les faisceaux correspondants.

Concernant la distinction entre le cas de la ***Figure 11*** et celui de la ***Figure 12***, on pourra regretter l'absence d'explication de la part de l'auteur. Néanmoins, une solution plus performante est proposée par Ghazanfarpour et al. [GHAZ 98].

4.1.1 Réflexion et réfraction

Lorsqu'une face *F* est réfléchissante, un *faisceau réfléchi* est lancé. Le concept de l'œil virtuel tel qu'il est suggéré entre autres par Heckbert et al. [HECK 84] est utilisé pour construire ces faisceaux. L'œil virtuel est le point *V* symétrique de l'œil par rapport au plan du polygone réfléchissant (***Figure 14***). Pour chacun des faisceaux réfléchis, on cherche les objets susceptibles d'être partiellement ou entièrement dans le faisceau. Si on se trouve dans un cas de figure semblable à celui de la ***Figure 11***, on calcule directement la couleur des pixels correspondant. Dans les autres cas, l'évaluation du critère *C* permet de choisir entre le lancer de rayons et le lancer de faisceaux afin de poursuivre la récursion.

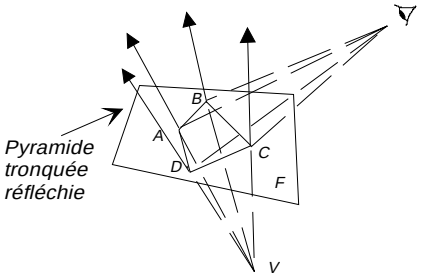


Figure 14 : Faisceau réfléchi.

Pour traiter la réfraction, [MARK 90] introduit la notion de *périmètre intérieur*. Ce périmètre est constitué d'un ensemble de pixels de l'écran correspondant aux rayons qui heurtent le polygone *P*, à l'intérieur du quadrilatère *Q* (voir ***Figure 15***). Les rayons n'ayant pas pour origine un pixel situé à l'intérieur du périmètre ne traversent pas le polygone *P* dans *Q* : pour ces rayons, on utilise un lancer de rayons conventionnel pour évaluer leurs contributions à l'image finale. Les rayons traversant le périmètre sont assurés d'aboutir sur *P* dans *Q*, le calcul du rendu est donc simplifié.

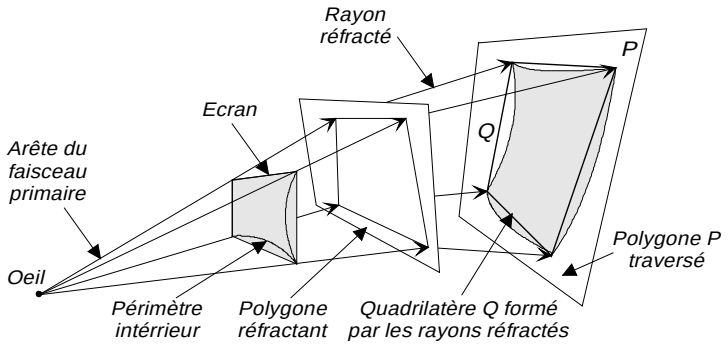


Figure 15 : Périmètre intérieur.

Notons que la détermination des périmètres intérieurs pour chaque surface réfractante est réalisée par raffinement progressif en lancant un grand nombre de rayons réfractés. De plus, cette détermination doit être réalisée pour chaque niveau de réfraction. On peut alors penser que plus le nombre de niveaux de réfraction est important et moins on a intérêt à utiliser le concept de périmètre intérieur.

4.1.2 Ombre

Les effets d'ombre sont traités par un lancer de rayons conventionnel. Néanmoins, les auteurs proposent une extension fondée sur des *faisceaux d'ombre*. Supposons qu'un faisceau de vision soit bloqué par un polygone et soit vide. On construit un faisceau pyramidal d'ombre ayant pour sommet une source lumineuse ponctuelle et pour base le polygone bloquant. Si ce faisceau est vide (***Figure 16.a***), la région du polygone considérée est entièrement éclairée. Si les quatre arêtes du faisceau d'ombre traversent un même polygone (convexe) avant de heurter la base du faisceau, la région est totalement à l'ombre (***Figure 16.b***). Dans ces deux cas, les conditions d'éclairement sont connues. Dans tous les autres cas (***Figure 16.c***), ces conditions sont évaluées, soit en subdivisant le faisceau de vision en quatre sous-faisceaux et en construisant quatre sous-faisceaux d'ombre, soit en utilisant un lancer de rayons conventionnel.

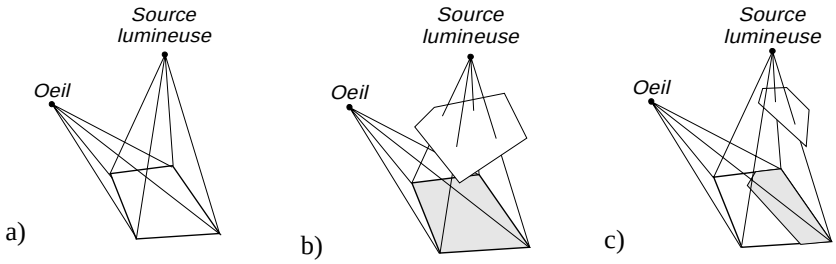


Figure 16 : Faisceaux d'ombre.

4.1.3 Discussion

Ce lancer de faisceaux hybride est intéressant par le choix dynamique entre un lancer de rayons et un lancer de faisceaux. Ce choix permet de diminuer les temps de calculs en évitant de construire un grand nombre de faisceaux très fins. Un autre aspect intéressant est une réfraction sans approximation. Cette réfraction est néanmoins coûteuse et une solution plus élégante est proposée par Ghazanfarpour [GHAZ 92]. Notons de plus que l'utilisation possible d'un lancer de rayons interdit le traitement efficace de l'aliassage, en particulier la disparition des petit objets et des petites ombres.

4.2 Lancer de faisceaux pyramidaux et subdivision adaptative

Ghazanfarpour et al. [GHAZ 92] [GHAZ 98] proposent un algorithme dont le but principal est le traitement efficace de tous les problèmes d'aliassage. De plus, contrairement aux autres méthodes, il évite les calculs coûteux d'intersection faisceau-objet ce qui autorise l'utilisation de scènes non-polygonales.

Cette approche comprend deux étapes, la première utilise des faisceaux pour détecter les régions avec des problèmes d'aliassage et la deuxième utilise un lancer de rayons pour calculer la couleur des pixels de l'écran.

La première étape, utilisant des faisceaux, ressemble à l'approche proposée par Marks et al. [MARK 90], on lance une *pyramide de vision* à travers la totalité de l'écran (***Figure 11***). On positionne les objets par rapport à la pyramide pour détecter d'éventuels problèmes d'aliassage (marches d'escalier sur les contours, petits objets ou petites ombres). S'il y a des problèmes (***Figure 17.a-d***), la région de l'écran est dite *ambiguë*, elle est divisée en quatre à la manière de Warnock [WARN 69] et quatre sous-pyramides de vision sont lancées (***Figure 13***). S'il n'y a pas de problème (***Figure 17.e-f***), la région est dite *uniforme* et aucune subdivision n'est nécessaire. Cette récursion prend fin lorsqu'on a atteint une région uniforme ou la limite de subdivision en sous-pixels fixée a priori (de l'ordre du $64^{ème}$ de pixel).

Pour positionner les polygones par rapport aux faisceaux, Ghazanfarpour et al. [GHAZ 98] ont développé des algorithmes simples et rapides fondés sur la notion de *plan séparateur*. Un plan est dit séparateur s'il divise l'espace en deux sous-espaces, l'un contenant entièrement le faisceau et l'autre entièrement le polygone. Si l'on arrive à construire un tel plan, on est sûr qu'il n'y a pas d'intersection faisceau-polygone, dans le cas contraire, on suppose qu'il y a intersection. Notons que conclure qu'il y a intersection alors que ce n'est pas le cas ne fausse aucunement l'algorithme, au plus, on effectuera des subdivisions inutiles.

Une fois les régions ambiguës détectées, on utilise un lancer de rayons conventionnel pour calculer la couleur des pixels. À ce niveau, les tests d'intersection rayon-objet ne sont plus nécessaires. En effet, les faisceaux lancés dans la première étape permettent de connaître très précisément le chemin parcouru par un rayon. Ce dernier traitement est donc très rapide.

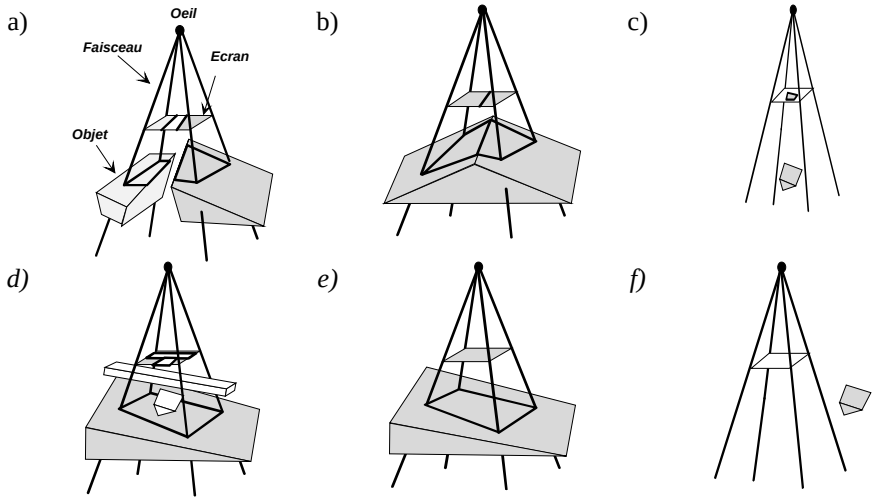


Figure 17 : Exemples de régions ambiguës (a-d) et uniformes (e-f).

4.2.1 Faisceaux d'ombre

Pour traiter les ombres, on utilise la même approche que Marks et al. [MARK 90] en construisant un faisceau d'ombre issu d'une source lumineuse *S* et ayant pour base le quadrilatère *ABCD* comme indiqué sur la ***Figure 18***. La visibilité de *ABCD* à partir de *S* est déterminée avec des tests similaires à ceux utilisés pour le faisceau de vision. Ces tests sont cependant plus simples que les précédents, car il ne faut distinguer par rapport à *S* que l'un des trois cas suivants :

1. *ABCD* est entièrement visible de *S*, il est donc directement éclairé ;
2. *ABCD* est entièrement caché de *S* par un polygone bloquant, il est donc dans l'ombre ;
3. *ABCD* est partiellement visible de *S*.

Dans les deux premiers cas, il n'y a pas d'ambiguïté et aucune subdivision n'est nécessaire. Une nouvelle source lumineuse avec la pyramide correspondante sera examinée. Dans le troisième cas, la région correspondante à *ABCD* sur l'écran est subdivisée. Notons que cette subdivision n'entraîne pas d'autre test sur les quatre nouvelles sous-pyramides de vision. Nous devons uniquement tester les sous-pyramides d'ombre afin de déterminer la visibilité de leurs intersections à partir de chaque source lumineuse. Finalement, une région est considérée comme uniforme si la visibilité dans son faisceau de vision et dans toutes ses pyramides d'ombre est sans ambiguïté.

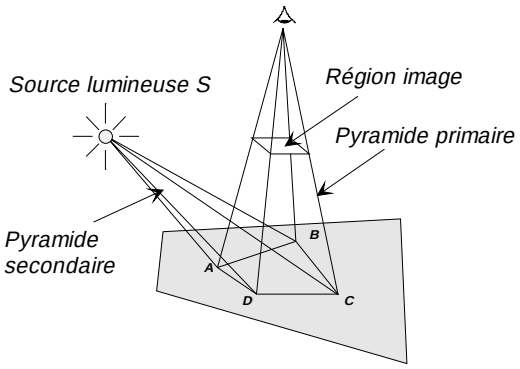


Figure 18 : Pyramide de vision et pyramide d'ombre.

4.2.2 Réflexions

Pour traiter les effets de réflexion, on construit une *pyramide réfléchie* comme dans Marks et al. [MARK 90] (***Figure 14***). Cette pyramide subit des tests similaires à ceux utilisés pour le faisceau de vision. Si plus d'un polygone est visible dans cette pyramide tronquée réfléchie (le fond de l'écran est considéré comme un polygone), on subdivise cette pyramide d'une façon similaire aux autres pyramides. Les résultats des subdivisions successives dues aux pyramides de vision, d'ombre et de réflexion sont montrés ***Figure 19***.

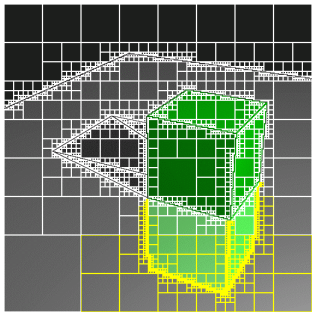


Figure 19 : Subdivisions des faisceaux de visions et d'ombre (en blanc) et subdivisions des faisceaux de réflexion (en gris).

4.2.3 Réfraction

L'approche proposée par Ghazanfarpour [GHAZ 92] pour résoudre le problème de la réfraction consiste à construire une *pseudo-pyramide de réfraction* englobant tous les rayons possibles réfractés par une région d'un polygone réfractant.

Soit *ABCD* l'intersection entre la pyramide de vision et le plan du polygone réfractant et r_1, r_2, r_3 et r_4 les rayons réfractés en *A*, *B*, *C* et *D* (***Figure 20***). Un bord de *ABCD* et un rayon réfracté forment un plan.

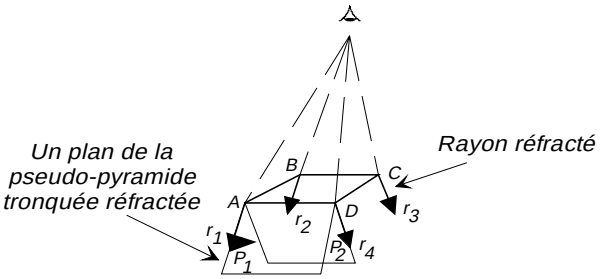


Figure 20 : Sélection des plans pour construire la pseudo-pyramide tronquée réfractée.

Considérons, par exemple, P_1 et P_2, les plans définis respectivement par le bord *AD* et le rayon r_1, le bord *AD* et le rayon r_4. Un de ces deux plans (P_2 dans notre exemple) se situe "entre" *ABCD* et l'autre plan. Utilisons le plan extérieur (P_1 dans notre exemple) ainsi que les trois autres définis de la même façon à partir des côtés *AB*, *BC*, et *CD* et les rayons réfractés pour construire un volume réfracté. Ce volume forme la *pseudo-pyramide tronquée réfractée* d'origine *ABCD*. Cette pseudo-pyramide contient tous les rayons réfractés par *ABCD*. Elle subit les mêmes tests de subdivision que ceux effectués dans le cas des autres pyramides secondaires. Ces pseudo-pyramides tronquées réfractées ne servent qu'à détecter les régions ambiguës. Le calcul de la couleur d'une telle région dépend uniquement des rayons réfractés et de la précision fournie par le lancer de rayons.

4.2.4 Temps de calculs

Pour comparer les lancer de faisceaux pyramidaux avec un lancer de rayons conventionnel, nous avons utilisé deux scènes tests : une ville virtuelle composée de 9114 polygones (***Figure 21***) et un ensemble de 2060 cubes (12360 polygones) de taille et position aléatoire (***Figure 22***).

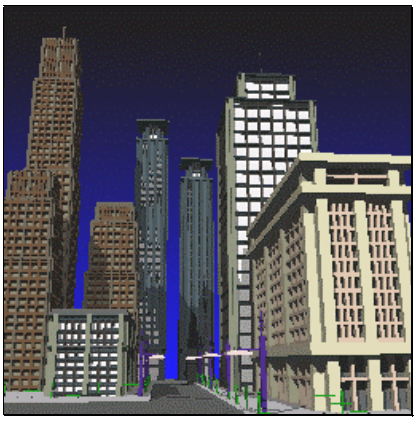


Figure 21 : Ville virtuelle.

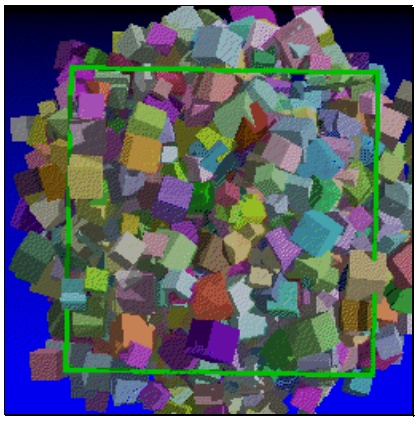
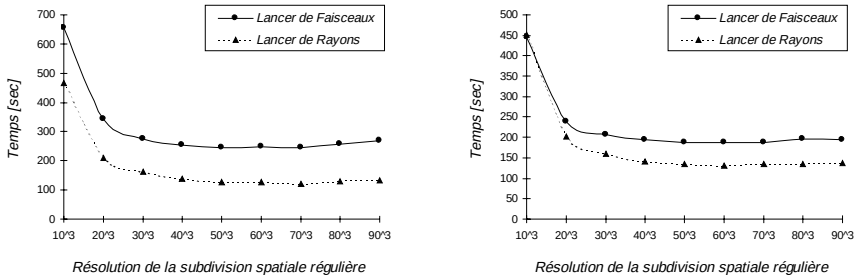


Figure 22 : Ensemble de cubes aléatoires.

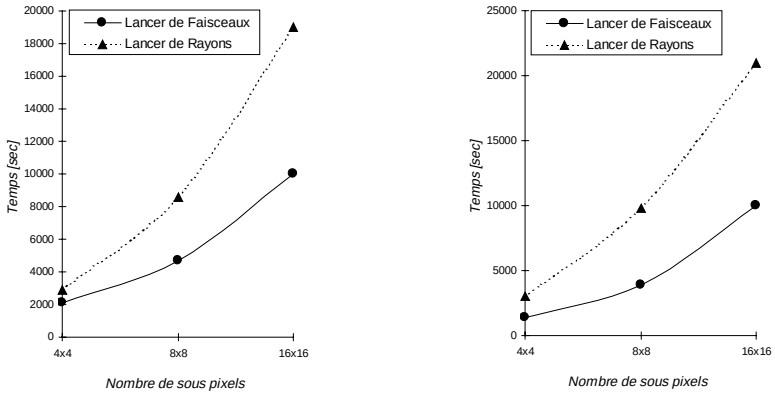
Nous avons comparé le lancer de faisceaux pyramidaux avec un lancer de rayons optimisé avec une subdivision spatiale régulière [AMAN 87]. En faisant varier la résolution de la subdivision spatiale de 10^3 à 90^3 (bornes représentant les résolutions habituellement utilisées), le lancer de faisceaux est moins rapide que le lancer de rayons (***Figure 23***). En fonction de la résolution de la subdivision et de la scène, le rapport de temps entre les deux approches varie entre 1 et 2.



Scène de la ville virtuelle. *Scène des cubes aléatoires.*

Figure 23 : Comparaison en temps entre le lancer de faisceaux et le lancer de rayons.

En général, on cherche à produire des images sans aliassage. Pour cela, on utilise un lancer de rayons avec une subdivision adaptative des pixels [WHIT 80]. Cependant, cette technique ne permet pas de détecter, à coup sûr, tous les petits objets et petites ombres. Il est donc difficile de comparer notre lancer de faisceaux avec un tel algorithme puisqu'il ne produit pas les mêmes images. Pour faire une comparaison plus réaliste, nous comparons le lancer de faisceaux à un lancer de rayons avec une subdivision systématique de tous les pixels. Les courbes de la ***Figure 24*** montrent clairement un gain de temps de l'ordre d'un facteur 2 lors de l'utilisation du lancer de faisceaux pour des images antialiassées à partir de 8x8 subdivisions au maximum.



Scène de la ville virtuelle. *Scène des cubes aléatoires.*

Figure 24 : Comparaison entre le lancer de rayons et le lancer de faisceaux pour le traitement de l'aliassage.

4.2.5 Extensions possibles

Notons que dans cette approche, il n'y a aucun calcul d'intersection entre les faisceaux et les objets. On cherche uniquement à savoir si des objets sont partiellement ou entièrement dans un faisceau. Cette solution permet donc le traitement de scènes non-polygonales comme les scènes de type CSG [HASE 96] (***Figure 25***).

D'autre part, l'utilisation de larges sources lumineuses pour produire des effets de pénombre homogène est possible [GHAZ 98]. Cette pénombre, contrairement aux méthodes généralement utilisées [COOK 84], ne fait pas appel à un suréchantillonnage stochastique. On peut ainsi éviter les phénomènes de "petits points" lorsqu'il y a de très larges sources lumineuses et que la zone de pénombre est très grande.

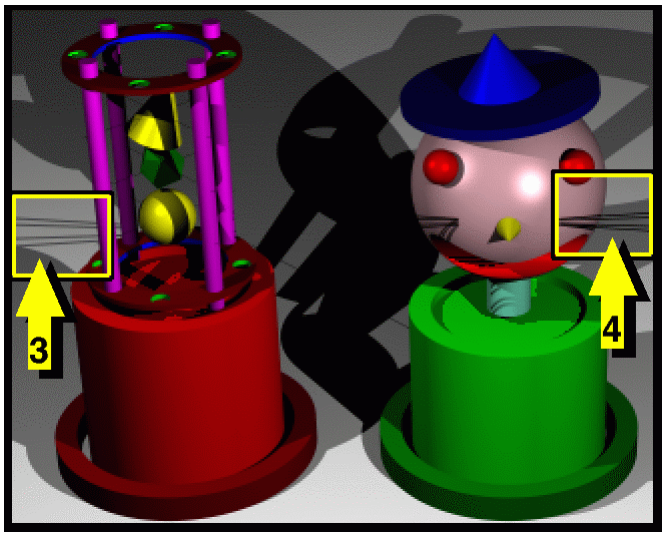


Figure 25 : Lancer de faisceaux sur une scène CSG. On constate que les problèmes de la disparition des petits objets [4] et des petites ombres [3] sont résolus.

4.2.6 Discussion

La technique du *lancer de faisceaux pyramidaux* proposée par Ghazanfarpour et al. [GHAZ 92] [GHAZ 98] est une approche relativement simple et efficace pour la visualisation de scènes complexes qui, de plus, fournie un antialiassage robuste. Les faisceaux pyramidaux sont utilisés comme volumes englobants pour détecter les régions uniformes ou ambiguës. Contrairement aux méthodes similaires, aucun calcul d'intersection explicite faisceau-polygone n'est nécessaire ce qui autorise le traitement de scènes non polygonales comme les scènes "CSG". Le nombre de calculs d'intersection rayon-polygone est limité et les petits objets et les petites ombres sont détectés et traités. Un avantage important de cette méthode est l'absence

des approximations linéaires ou calculs complexes utilisés dans les méthodes analogues [HECK 84] [SHIN 87] pour le cas de la réfraction.

5. Les faisceaux comme outil ponctuel du lancer de rayons

Nous présentons dans ce paragraphe une utilisation des faisceaux pour résoudre des problèmes ponctuels. Dans un premier temps, il s'agit de diminuer les temps de calculs nécessaires pour produire les ombres portées en utilisant des faisceaux pyramidaux. Dans un deuxième temps, des faisceaux semblables seront utilisés pour déterminer la visibilité entre deux carreaux dans le cadre de la radiosité.

5.1 Accélération du calcul des ombres

Choi et al. [CHOI 92] présente une structure appelée *pyramide d'ombre* permettant de réduire le temps passé dans le calcul des ombres. Pour cela, une pyramide d'ombre est construite pour chaque objet (polygone convexe) rencontré pour la première fois par un rayon et pour chaque source lumineuse. Une liste des objets contenus dans chacune des pyramides est construite et associée aux différentes pyramides. Les pyramides sont elles-mêmes associées aux différentes faces des objets.

Pour savoir si un point d'une face est à l'ombre, un rayon issu de ce point est lancé vers chacune des sources, comme pour un lancer de rayons conventionnel. Par contre, les tests d'intersection ne sont effectués qu'avec les objets contenus dans les listes associées aux pyramides concernant la face rencontrée de l'objet.

Cette structure étant assez lourde, l'auteur propose la méthode *PYSHA*. C'est une méthode hybride pour tester si un point d'un polygone ou d'une sphère est à l'ombre. Elle combine des subdivisions spatiales avec sa structure de pyramide d'ombre.

5.1.1 Construction des pyramides et des listes

Les pyramides d'ombre ont pour base la face de l'objet rencontré et pour sommet une source lumineuse. Pour profiter de la cohérence de la scène, l'auteur propose une subdivision spatiale adaptative limitée à une profondeur de 3.

Pour construire la liste des objets susceptibles d'être dans une pyramide d'ombre, l'auteur cherche à trouver les objets clairement à l'extérieur de la pyramide. Par négation, il peut alors construire la liste. Pour cela, deux étapes successives sont proposées :

1. trouver les sous-espaces uniformes clairement à l'extérieur de la pyramide ;
2. trouver les objets contenus dans les sous-espaces n'ayant pu être écartés à l'étape *1* et clairement situés en dehors de la pyramide.

Les objets n'ayant pu être positionnés clairement à l'extérieur du faisceau sont ajoutés dans la liste des objets associés à la pyramide.

Pour affirmer qu'un objet ou qu'un sous-espace est en dehors d'une pyramide, on cherche un plan séparateur. Si un tel plan existe, on peut affirmer que l'objet est à l'extérieur de la pyramide. Choi et al. [CHOI 92] propose différents plans :

a) le plan formant la base de la pyramide (la face de l'objet rencontré) (***Figure 26.a***) ;
b) les plans formant les côtés de la pyramide (***Figure 26.b***) ;
c) des plans tangents aux arêtes de la pyramide (***Figure 26.c***).

Les plans tangents aux arêtes sont orientés de sorte que leur normale corresponde à la moyenne des normales des deux plans adjacents à l'arête considérée. Ces plans permettent d'écarter certains objets pour lesquels les plans de type *b)* n'ont pas suffit. C'est par exemple le cas de l'objet *C* dans la ***Figure 26.d***.

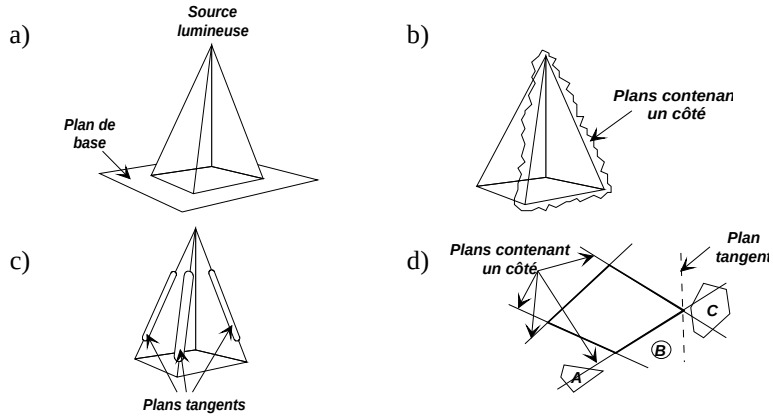


Figure 26 : Plans permettant d'écarter des objets visiblement hors de la pyramide. (dans la figure d), les objets A et B sont écartés grâce aux plans de type b) et l'objet C est écarté grâce à un plan de type c))

Pour améliorer les performances de l'algorithme, la liste est triée en se basant sur trois règles intuitives :

1. plus un objet est grand par rapport aux autres, plus la probabilité qu'il a de couper entièrement un faisceau d'ombre est grande ;
2. plus un objet est proche d'une source lumineuse, plus son ombre portée est grande ;
3. plus un objet est orienté perpendiculaire à la direction de la pyramide, plus son ombre portée est importante.

Ces trois règles sont rassemblées dans l'équation du *facteur de forme des ombres f* suivante :

$$f = (A/d)\ |N \bullet P|$$

où A est la surface du polygone, d la distance de la source au barycentre du polygone, N la normale du polygone et P la direction de la pyramide. La direction de la pyramide correspond à un axe passant par le barycentre du polygone et par la source lumineuse (***Figure 27***).

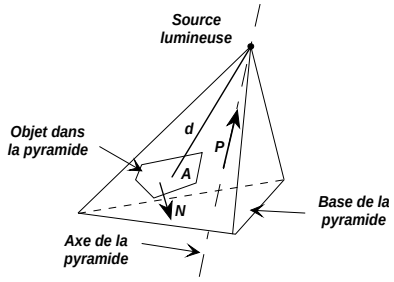


Figure 27 : Facteur de forme des ombres.

5.1.2 Raffinement de la liste

Une autre stratégie proposée par Choi et al. [CHOI 92] est fondée sur l'utilisation d'un algorithme de lancer de rayons travaillant par lignes de balayage et des objets constitués de polygones convexes. Il s'agit de construire des sous-listes des listes d'objets associées aux pyramides. Un objet est mis dans une sous-liste seulement si un rayon d'ombre l'a déjà rencontré pour la ligne courante et que le dernier rayon d'ombre ne l'a plus rencontré. Ceci correspond en fait à un objet qui a fait de l'ombre sur la ligne de balayage courante et qui n'en fait plus. Cette liste permet de ne plus tester d'intersection avec les objets ayant déjà produit une ombre portée.

5.1.3 PYSHA

Choi et al. [CHOI 92] étudie en détail l'évolution de son algorithme en fonction de différents paramètres tels que la taille des pyramides, la surface des objets dans la pyramide, leur nombre, … Il conclut que l'utilisation des pyramides d'ombre est optimale pour des pyramides ayant une large base et contenant un petit nombre d'objets. Dans le cas contraire, le surcoût de travail engendré par les différentes listes peut pénaliser l'algorithme. Après une comparaison de son algorithme avec un lancer de rayons optimisé avec une subdivision spatiale régulière, il propose PYSHA. PYSHA est une version hybride mélangeant un lancer de rayons avec subdivision spatiale régulière et la structure de pyramides d'ombre. En fait, pour chaque polygone rencontré, on va choisir d'utiliser l'une ou l'autre des deux méthodes. Le critère de choix est empirique et fondé sur la surface d'une éventuelle pyramide d'ombre et sur le nombre d'objets qu'elle contient. Ainsi modifié, l'algorithme proposé semble, sur la base de six scènes tests, plus rapide. Le gain varie entre 1.2 et 2.

5.1.4 Discussion

La solution proposée met en œuvre une optimisation classique du lancer de rayons : la subdivision régulière de l'espace. De plus, cette subdivision bénéficie aussi aux pyramides d'ombre lors du positionnement des objets. On peut regretter que ce positionnement ne soit pas optimum et n'utilise pas au maximum la subdivision spatiale régulière. Notons aussi que l'utilisation seule d'un lancer de

rayons pour calculer l'ombrage induit des problèmes d'aliassage, en particulier la disparition des petites ombres.

5.2 Optimisation du calcul de la visibilité

Hasenfratz [HASE 98] propose d'utiliser des faisceaux pour faciliter la détermination de la visibilité. Deux algorithmes utilisant cette optimisation ont été développés. Le premier construit sur l'écran les contours des objets, des ombres, des réflexions et des réfractions des objets visibles de la scène. Le deuxième repose sur le calcul de la radiosité de Wallace et al. [WALL 89] et facilite la détermination de la visibilité entre deux carreaux. Les méthodes utilisées dans ces deux approches sont très proches, nous ne présenterons donc ici que la solution appliquée à la radiosité.

Pour améliorer les temps de calculs lors de l'évaluation de la visibilité entre un carreau émetteur et un carreau récepteur, on cherche à éliminer les carreaux clairement non visibles depuis l'émetteur. Notons que le but n'est pas de calculer une visibilité exacte mais bien de diminuer le nombre de carreaux fournis au processus évaluant précisément cette visibilité. Dans ce but, on utilise la subdivision spatiale régulière et la notion de faisceaux de la manière suivante :

1. chercher des polygones susceptibles de cacher un grand nombre de carreaux qu'on appellera des "bons occulteurs" ;
2. construire, pour chaque occulteur, un faisceau issu du carreau émetteur et traversant exactement l'occulteur (Figure 28) ;
3. parcourir les voxels de chacun des faisceaux, positionner les carreaux associés à ces voxels par rapport aux faisceaux et les marquer "non visibles" lorsqu'ils sont entièrement dans un faisceau ;
4. éffectuer le test de visibilité uniquement entre le carreau émetteur et les carreaux qui ne sont pas marqués "non visibles".

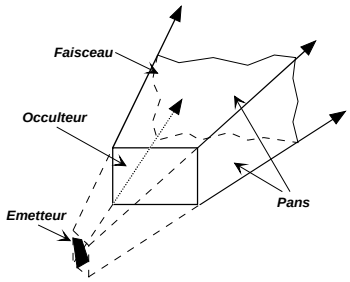


Figure 28 : Faisceau englobant l'émetteur et traversant un occulteur.

Cette solution, très simple à insérer dans n'importe quel processus de détermination de la visibilité est efficace. Par exemple, dans le cas de la radiosité de Wallace et al. [WALL 89] avec un maillage régulier, on a obtenu un gain de temps de l'ordre 25%.

Conclusion

Ce tour d'horizon a été consacré aux différentes variantes du lancer de rayons cherchant à profiter au maximum de la cohérence de la scène pour résoudre le problème de l'aliassage et celui du temps de calculs.

Dans la partie consacrée à l'utilisation de rayons d'épaisseur infinitésimale, Speer et al. [SPEE 85] cherchent à prédire le chemin que va parcourir un rayon en construisant un tunnel autour du dernier rayon lancé. Cette méthode est annoncée comme une voie à ne pas suivre. Elle apporte cependant une solution pour positionner un objet par rapport à un cylindre. La deuxième solution est celle de Müller [MULL 86] qui lance en même temps tous les rayons d'un même niveau de récursion. Ce parcours en largeur de l'arbre de Whitted [WHIT 80] permet une gestion efficace des rayons lancés. La troisième solution, et parmi les plus connues, est celle de Shinya et al. [SHIN 87] qui regroupe les rayons en crayons. Les équations proposées permettent de maîtriser la qualité de la réfraction, cependant ce regroupement n'est possible que sur des surfaces continues. En particulier, la solution n'est pas applicable sur les bords des polygones et ne traite donc pas les marches d'escalier.

Dans la partie consacrée aux algorithmes utilisant uniquement des faisceaux, Amanatides et al. [AMAN 84] proposent de lancer des cônes pour traiter les divers problèmes d'aliassage. Malheureusement, cette forme conique est peu appropriée même à des scènes polygonales et de nombreux problèmes en découlent. La solution de Heckbert et al. [HECK 84] semble nettement plus intéressante. L'utilisation d'un faisceau de section polygonale permet de profiter de tous les algorithmes de découpage en 2D. De plus, la représentation de l'image de la scène par ses contours peut autoriser un traitement efficace de l'aliassage même si cela n'était pas le but premier de cette solution. L'inconvénient majeur de cette solution est la complexité des calculs d'intersection faisceau-objet. De plus, les effets de réfraction sont approchés à une transformation linéaire.

Dans la partie qui présente les solutions hybrides, les deux solutions proposées sont relativement semblables mais les buts sont toutefois différents. Dans le cas de Marks et al. [MARK 90], le but est un gain de temps, il alterne donc l'utilisation d'un lancer de faisceaux avec l'utilisation d'un lancer de rayons. Dans le cas de Ghazanfarpour et al. [GHAZ 98], c'est le traitement de l'aliassage qui prime et l'on utilise le lancer de faisceaux pour détecter les régions à problèmes. Ce n'est que dans un deuxième temps que l'on lance des rayons pour le calcul du rendu proprement dit. En fait, chacun arrive à ses fins, les premiers obtiennent de meilleurs temps de calculs par contre ils ne traitent pas systématiquement les problèmes d'aliassage et les second résolvent tous les problèmes d'aliassage mais proposent une solution plus lente.

Enfin, la dernière partie a été consacrée à l'utilisation de faisceaux pour améliorer les temps de calculs des ombres ou de la visibilité. On a ainsi pu optimiser le calcul de la radiosité de Wallace et al. [WALL 89].

Une synthèse des méthodes proposées dans cet article est présentée dans le Tableau 1.

Méthodes	Objectifs		Scènes		Résultats qualitatifs								Résultats quantitatifs				Remarques
					Réfraction		Petits objets		Marches d'escalier		Pénombre						
	Gain de temps	Antialiassage	Polygonales	Non-polygonales	Approximée	Exacte	Non détectés	Detectés	Non traitées	Traitées	Non traitée	Traitée	Décevants	Mitigés	Bons	Très bons	
[SPEE 85]	●			●		●	●		●		●		●				*(1)*
[MÜLL 86]	●					●	●			●	●			●			*(2)*
[SHIN 87]	●		●	●		●	●		●		●			●			*(3)*
[AMAN 84]	●	●	●		●		●			●		●		●			*(4)*
[HECK 84]	●		●		●			●		●	●				●		*(5)*
[MARK 90]	●		●			●	●			●	●				●		*(6)*
[GHAZ92][HASE96][GHAZ98]	●	●	●	●		●		●		●		●				●	*(7)*
[CHOI 92]	●		●			●	●			●	●				●		*(8)*

(1) Les scènes utilisées ne comportent que des sphères.
(2) Nécessite beaucoup de mémoire.
(3) Aucun traitement des marches d'escalier n'est possible directement puisque la surface doit être C1 continue.
(4) Beaucoup d'imprécisions.
(5) Des effets d'éclairage indirecte sont possibles.
(6) Tous les polygones doivent être convexes.
(7) Il n'y a pas de calcul d'intersection faisceau-objet, on peut traiter des polygones concaves [GHAZ 92] et des scènes CSG [HASE 96].
(8) Tous les polygones doivent être convexes.

Tableau 1 : Synthèse des différentes méthodes proposées.

Remerciement

Nous remercions la Région Limousin pour son soutien financier.

Bibliographie

[AMAN 84] John AMANATIDES, "Ray Tracing with Cones", Computer Graphics (Proceedings of SIGGRAPH'84), July 1984, 18 (3), pp. 129-135.

[AMAN 87] John AMANATIDES, Andrew WOO, "A Fast Voxel Traversal Algorithm for Ray-Tracing", Proceedings of Eurographics Conference, August 1987, pp. 27-38.

[ATHE 78] Peter ATHERTON, Kevin WEILER, Donald GREENBERG, "Polygon Shadow Generation", Computer Graphics (SIGGRAPH'78), 12(3), 1978, pp. 275-281.

[BORN 59] M. BORN, E. WOLF, "Principles of Optics", 1959, pp. 190-196, New-York: Pergamon.

[CHOI 92] H. K. CHOI, C. M. KYUNG "PYSHA - A Shadow-Testing Acceleration, Scheme for Ray-Tracing", Computer-Aided Design, February 1992, Vol. 24(2) pp. 93-104.

[COOK 84] Robert L. COOK, T. PORTER, L. CARPENTER, "Distributed Ray Tracing", Computer Graphics (Proceedings of SIGGRAPH'84), July 1984, 18(3), pp. 137-145.

[DADO 85] Norm DADOUN, David G. KIRKPATRICK "The Geometry of Beam, Tracing", ACM Symposium on Computational Geometry, 1985, pp. 55-61.

[GANG 84] M. GANGNET, D. GHAZANFARPOUR, "Techniques for Perspective Mapping of Plane Textures", International Electronic Image Week, CESTA, Biarritz, May 1984, pp. 29-35.

[GHAZ 91] Djamchid GHAZANFARPOUR, Bernard PEROCHE, "A High Quality Filtering Using Forward Texture Mapping", Computers & Graphics, 15(4), 1991, pp. 569-577.

[GHAZ 92] Djamchid GHAZANFARPOUR, "Visualisation réaliste par lancer de pyramides et subdivision adaptative", acte du MICAD 92, PARIS, février 1992, pp.167-181.

[GHAZ 98] Djamchid GHAZANFARPOUR, Jean-Marc HASENFRATZ, "A Beam Tracing with Precise Antialiasing for Polyhedral Scenes", Computer & Graphics, January 1998 (à

paraître), 22(1).

[HASE 96] Jean-Marc HASENFRATZ, Djamchid GHAZANFARPOUR, "Rendering CSG Scenes with General Antialiasing", CSG 96 Set-theoretic Solid Modeling Techniques and Applications, Information Geometers, Winchester, 1996, pp. 275-289.

[HASE 98] Jean-Marc HASENFRATZ, "Lancer de faisceaux en synthèse d'image", Thèse de doctorat de l'Université de Limoges, janvier 1998.

[HECK 84] Paul S. HECKBERT, Pat HANRAHAN "Beam Tracing Polygonal Objects", Computer Graphics (SIGGRAPH'84 Proceedings), July 1984, 18(3), pp. 119-127.

[KIRK 87] David B. KIRK "The Simulation of Natural Features Using Cone Tracing", The Visual Computer, 1987, 3, pp. 63-71.

[MARK 88] Joseph MARKS, Robert J. WALSH, Mark FRIEDEL "Hybrid Beam-Ray Tracing", Harvard University - Center for Research in Computer Technology TR-03-88, 1988.

[MARK 90] Joseph MARKS, Robert J. WALSH, John CHRISTENSEN, Mark FRIEDEL "Image and Intervisibility Conherence in Rendering", Proceedings of Graphics Interface'90, May 1990, pp. 17-30.

[MULL 86] H. MULLER "Image Generation by Space Sweep", Computer Graphics Forum 1986, 5, pp. 189-196.

[SHIN 87] Mikio SHINYA, Tokiichiro TAKAHASHI, Seiichiro NAITO "Principles and Applications of Pencil Tracing", SIGGRAPH'87, July 1987, 21(4), pp. 45-54.

[SPEE 85] L. Richard SPEER, Tony D. DEROSE, A. BARSKY "A Theoretical and Empirical Analysis of Coherent Ray-Tracing", Computer Generated Images: The State of the Art, 1985.

[WALL 89] John R. WALLACE, Kells A. ELMQUIST, Eric A. HAINES, "A Ray Tracing Algorithm for Progressive Radiosity", SIGGRAPH'89, July 1989, 23(3), pp 315-324.

[WARN 69] J. WARNOCK, "A Hidden Surface Algorithm for Computer Generated Half-Tone Pictures", TR 4-15, NTIS AD-753 671, Computer Science Department, University of Utah, Salt Lake City, UT, June 1969.

[WEIL 77] Kevin WEILER, Peter ATHERTHON, "Hidden Surface Removal Using Polygon Area Sorting", Computer Graphics, 1977, 11(2), pp. 214-222.

[WHIT 80] Turner WHITTED "An Improved Illumination Model for Shaded Display", Communications of the ACM, June 1980, 23(6), pp. 343-349.