USTL

LIFL Laboratoire d'Informatique Fondamentale de Lille

THESE

présentée à

L'UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE

pour obtenir le titre de

DOCTEUR EN INFORMATIQUE

par

Christophe RENAUD

APPROCHES PARALLELES POUR LA RADIOSITE

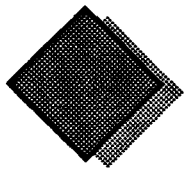
Thèse soutenue le , devant la commission d'examen composée de Messieurs :

Président de jury	J.L. DEKEYSER	LIFL (Lille)
Rapporteur	K. BOUATOUCH	INRIA (Rennes)
Rapporteur	P. GUITTON	LABRI (Bordeaux)
Directeur de thèse	M. MERIAUX	LIFL (Lille)
Examinateur	S. MIGUET	LIP (Lyon)
Examinateur	E. LEPRETRE	LIFL (Lille)

Soutenue le 29 OCTOBRE 1993

UNIVERSITE DES SCIENCES ET TECHNOLOGIES DE LILLE
U.F.R. d'I.E.E.A. Bât M3. 59655 Villeneuve d'Ascq CEDEX
Tél. 20.43.47.24 Fax. 20.43.65.66

Introduction



L'éclairement global d'une scène est au centre de nombreuses recherches actuelles, avec pour objectif d'obtenir la création d'images de synthèse réalistes. La prise en compte de phénomènes de plus en plus complexes amène, certes, une qualité de plus en plus élevée des images, mais nécessite en contrepartie des temps de calcul de plus en plus importants.

A l'inverse, les modèles d'éclairement locaux, qui ne s'occupent que des interactions directes entre une source de lumière et un objet, offrent une certaine simplicité qui a contribué à leur large diffusion. La preuve en est qu'il n'existe (presque) plus de machines graphiques qui ne possèdent, de base, les circuits spécialisés dédiés à l'affichage "temps réel" d'objets simples, dont l'illumination est calculée en tout point par le modèle d'interpolation de Gouraud.

Entre ces deux extrêmes existent des algorithmes qui permettent de calculer une grande partie des interactions lumineuses entre objets, et de résoudre le problème de l'illumination globale sous certaines hypothèses restrictives. Ces algorithmes sont plus connus sous les nom de *lancer de rayons* et de *radiosité*. Chacun d'entre eux pose des conditions très restrictives sur les phénomènes lumineux pris en compte, et ne permet à ce titre de résoudre que partiellement le problème de l'illumination globale.

Bien que simplifiées par rapport au modèle général, ces deux approches engendrent des coûts de calcul beaucoup plus importants que les modèles locaux, et sont bien loin, à ce jour, de pouvoir être utilisés de manière réellement interactive.

Parallèlement à cette croissance des coûts de calcul, ces dernières années ont vu le développement, d'une part de processeurs de plus en plus puissants, et d'autre part d'architectures parallèles disposant de centaines, voire de milliers de processeurs. Bien que leur utilisation s'avère en général plus délicate que celle des machines mono-processeur, la tentation est grande d'exploiter leur capacité de calcul élevées pour réduire les temps de génération d'une image.

De nombreux travaux ont permis depuis quelques années d'obtenir des accélérations très importantes de l'algorithme du lancer de rayons, en le parallélisant sur différents types d'architecture. Plus récemment, la technique de radiosité a, elle aussi, pu bénéficier des machines parallèles; les implantations de ces algorithmes ayant principalement eu pour cibles les machines à mémoire répartie, utilisant un mode de fonctionnement asynchrone.

Le but de ce travail est d'étudier les possibilités de parallélisme inhérentes à l'algorithme de radiosité, et d'étudier leur adéquation à diverses architectures parallèles. Notre travail s'articule autour de trois axes principaux, que nous allons à présent décrire.

Dans un premier temps, nous effectuerons une description des modèles d'illumination locaux, puis des algorithmes de lancer de rayonè et de radiosité (chapitre 1). Ceci nous permettra de spécifier les différences qui existent sur les types d'éclairement pris en compte par chacune de ces approches, et de définir précisément notre cadre de travail.

Les chapitres 2 et 3 sont une suite directe du premier chapitre, puisque leur objectif est de décrire précisemment des méthodes de calcul des facteurs de forme, quantité géométrique qui représente le "coeur" de l'algorithme de radiosité. Le chapitre 2 est consacré à la description des

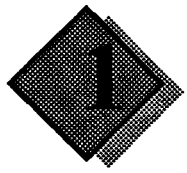
méthodes projectives et à leur comparaison, en fonction de différents critères. Le chapitre 3 a pour objectif de décrire les méthodes d'approximation des facteurs de forme issues de l'algorithme du lancer de rayons. Pour chacune de ces deux approches, nous proposerons un nouvel algorithme, permettant de diminuer le coût mémoire et d'augmenter la précision du calcul.

La seconde partie de ce mémoire permettra tout d'abord de définir les notions de base du parallélisme, et de décrire les deux architectures parallèles utilisées pour nos travaux : un calculateur à mémoire répartie (réseau de Transputers), et un calculateur massivement parallèle (la Maspar) (chapitre 4). Nous aborderons ensuite le chapitre 5, où les différentes possibilités de parallélisme seront analysées, et où nous étudierons les approches parallèles déjà envisagées dans un contexte asynchrone.

La dernière partie de ce travail a pour but d'étudier les possibilités de parallélisation des algorithmes de calcul des facteurs de forme sur un calculateur massivement parallèle, fonctionnant selon un mode SIMD. Différentes approches sont envisagées pour les méthodes projectives, afin de permettre d'utiliser de manière efficace l'ensemble des processeurs disponibles. Les aspects liés aux modes de communications propres à la Maspar seront étudiés, en vue d'obtenir une efficacité maximale lors des échanges de données entre processeurs.

Après avoir consacré un chapitre à l'analyse des résultats obtenus sur la Maspar, nous conclurons en évoquant les travaux ultérieurs qui feront suite à ce travail.

CHAPITRE



Modèles d'éclairement

La compréhension d'une image synthétique ne peut se faire qu'en tenant compte d'un certain nombre de paramètres, tels que les occultations entre objets proches et éloignés, l'existence d'ombres générées entre les objets ou encore la façon dont ceux-ci sont éclairés. Mais bien plus que la compréhension de l'image, c'est à présent le réalisme qui est recherché. Or, celui-ci ne peut être atteint qu'au prix de la modélisation de phénomènes lumineux toujours plus complexes et coûteux.

Dans ce chapitre, nous allons décrire un certain nombre de ces modèles d'éclairement, en les différenciant en deux catégories distinctes. D'une part, les modèles d'éclairement locaux, qui ne permettent d'obtenir que des informations d'éclairage propre à un objet, en négligeant volontairement les autres objets. Ils offrent néanmoins l'avantage de la simplicité. D'autre part, les modèles d'illumination globale, qui permettent de prendre en compte un grand nombre d'informations d'éclairage issues de l'ensemble de la scène considérée.

Nous décrirons donc, dans un premier temps, les modèles d'éclairement locaux, de manière à souligner leurs lacunes. L'algorithme du lancer de rayons, permettant de résoudre le problème de l'illumination globale sous certaines conditions sera ensuite introduit, puis nous détaillerons l'algorithme de radiosité, qui permet lui aussi de résoudre ce problème, avec des conditions différentes du lancer de rayons. Nous présenterons enfin différentes optimisations de cet algorithme, qui seront utilisées dans les chapitres suivants.

1.1 Modèles d'éclairement locaux

Les modèles d'éclairement locaux ne prennent en compte que les interactions directes entre une source et un objet donné. L'éclairage d'un point est calculé en déterminant la contribution directe de la source au point considéré. Nous allons décrire, dans un premier temps, ces modèles, puis nous présenterons deux modèles d'ombrage couramment utilisés pour limiter les coûts de calcul.

1.1.1 Modèles empiriques

Ces modèles distinguent deux types d'éclairement, qui dépendent des propriétés d'un objet :

- Si l'objet est *mat*, l'éclairement sera qualifié de *diffus*. Dans ce cas, la lumière parvenant sur l'objet est réfléchie uniformément dans toutes les directions. Ce type d'éclairement se produit dans le cas de surfaces rugueuses ou mates. Visuellement, cela se traduit par un aspect relativement "uniforme" de l'éclairage sur l'objet.
- Si l'objet est *brillant*, l'éclairement sera qualifié de *spéculaire*. Il existe alors une direction de réflexion privilégiée, qui correspond à la direction symétrique, par rapport à la normale à la surface au point considéré, de la direction d'incidence. Suivant la nature de l'objet, cette réflexion ne se produit que dans une direction unique (miroir), ou dans une portion de l'espace distribuée autour de cette direction privilégiée (objets "lisses"). Une tache brillante

apparaît alors sur l'objet, de taille plus ou moins importante selon que le matériau est faiblement ou fortement spéculaire.

Dans le cas général, l'éclairement "final" est calculé comme une combinaison de ces deux types d'éclairement. L'intensité en un point d'un objet est alors explicitée sous la forme :

$$I_{tot}(p) = \sum_{S} I_{diff}(p) + \sum_{S} I_{spec}(p)$$

où $I_{tot}(p)$ représente l'intensité au point p, $I_{diff}(p)$ l'intensité diffuse en p, $I_{spec}(p)$ l'intensité spéculaire en p, et S l'ensemble des sources éclairant la scène.

Eclairement diffus

L'éclairement diffus est régi par la *loi de Lambert*, aussi appelée *loi du cosinus*. Celle-ci s'exprime sous la forme :

$$I_{diff} = K_d \times \cos(\vec{N}, \vec{S}) \times I_S = K_d \times (\vec{N} \bullet \vec{S}) \times I_S$$

où (voir Figure 1.1) :

- K_d représente le coefficient diffus de l'objet (proportion de l'intensité incidente diffusée)
- $\vec{N}$ la normale à l'objet au point considéré
- $\vec{S}$ le vecteur reliant le point de l'objet à la source
- I_s l'intensité de la source

L'intensité diffusée est ainsi maximale dans la direction de la source, et apparait comme indépendante de la position de l'observateur, puisque celle-ci ne figure pas dans son expression.

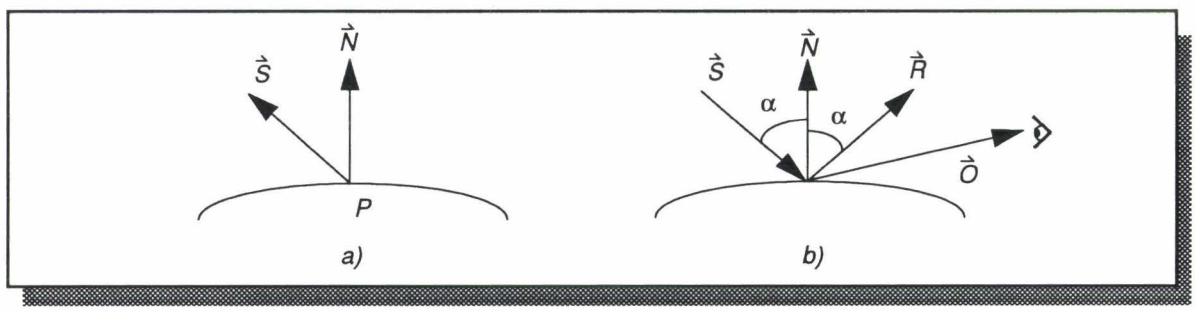


Figure 1.1 ***Géométrie pour le calcul de l'éclairement diffus (a) et spéculaire (b)***

Dans la mesure où cette loi ne tient pas compte de l'éloignement de l'objet à la source, Warnock [Warno 69] a proposé d'atténuer cette quantité, en la divisant par une valeur proportionnelle à la distance entre l'objet et la source.

Eclairement spéculaire

Phong a défini l'éclairement spéculaire sous la forme :

$$I_{spec} = W(\alpha, \lambda) \times \cos^s(\vec{R}, \vec{O}) \times I_S = W(\alpha, \lambda) \times (\vec{R}, \vec{O})^s \times I_S$$

avec (voir Figure 1.1) :

- $W(\alpha, \lambda)$ la fonction de réflectance spéculaire
- α l'angle d'incidence
- s l'exposant spéculaire de l'objet
- $\vec{R}$ le vecteur réfléchi (direction de réflexion privilégiée)
- $\vec{O}$ le vecteur d'observation (reliant le point de l'objet à l'observateur)
- I_S l'intensité de la source

La fonction de réflectance W est une fonction caractéristique du matériau dont est constitué l'objet. Elle dépend de l'angle d'incidence sous lequel l'intensité lumineuse d'une source parvient en un point de l'objet, et de sa longueur d'onde. A titre d'exemple, la Figure 1.2 présente la valeur de cette fonction pour l'or et l'argent.

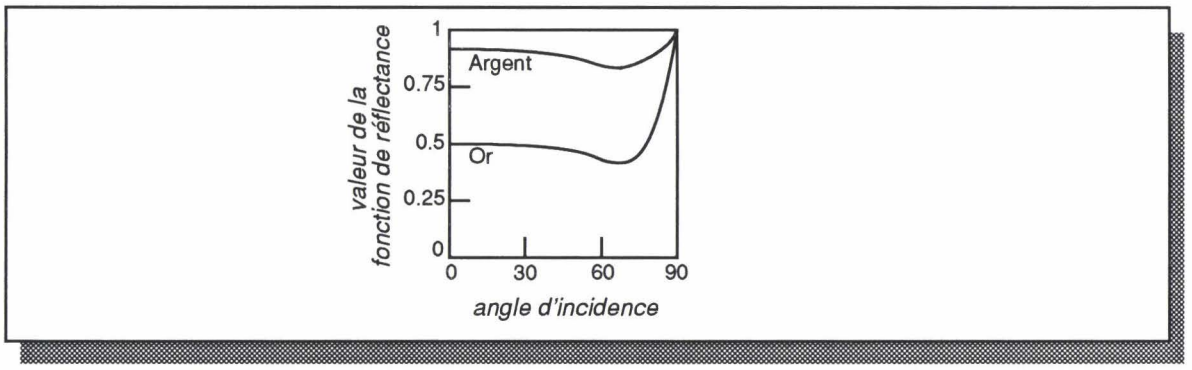


Figure 1.2 ***Exemples de courbes de réflectance spéculaire.***

En général, du fait de la difficulté de la calculer, la fonction W est remplacée par une constante K_s, appelée *coefficient spéculaire* de l'objet.

L'exposant spéculaire s représente l'indice de brillance de l'objet : lorsque s est petit (1-10), la surface est peu spéculaire, c'est à dire que la réflection se produit dans un grand nombre de directions; par contre, lorsque s devient grand, la surface est fortement spéculaire, et l'ensemble des directions de réflexion est distribué de manière beaucoup plus étroite autour de la direction privilégiée.

Notons que l'expression de l'intensité spéculaire fait intervenir la position de l'observateur, ce qui correspond à la réalité, dans la mesure où la tache de lumière résultant de l'éclairement spéculaire sur un objet se déplace à la surface de cette objet lorsque l'observateur bouge.

Eclairement ambiant

Les deux types d'éclairement précédents ne prennent en compte que les interactions entre les sources directement visibles. Cela signifie que les points d'un objet qui sont invisibles depuis la source ne recevront aucune intensité lumineuse. Pour éviter ce phénomène, un terme ambiant est en général introduit, qui permet d'associer un peu d'éclairage aux parties d'objets non éclairées. Il s'exprime sous la forme

$$I_{amb} = K_a \times I_{SA}$$

avec K_a le coefficient ambiant de l'objet, et I_{SA} l'intensité de la "source" ambiante.

La valeur de l'intensité de la source ambiante est calculée empiriquement, et correspond à l'ensemble de l'énergie lumineuse qui provient de l'environnement par l'intermédiaire des diverses réflexions entre objets.

Eclairement total

L'équation utilisée pour le calcul de l'éclairement local d'un point, prenant en compte chacun des effets précédents, est donc obtenue en additionnant chacun des éclairements précédents. Cependant, les valeurs d'éclairement qui viennent d'être définies, dépendent de la longueur d'onde considérée. Il est donc théoriquement nécessaire de les appliquer pour chaque longueur d'onde du spectre visible. En pratique, celles-ci ne sont appliquées que pour les trois primaires Rouge, Vert et Bleu, étant admis qu'une combinaison linéaire de ces trois couleurs permet de représenter une grande partie de l'ensemble des couleurs.

L'équation permettant de calculer l'éclairement local à un point, pour une source S et une longueur d'onde λ, s'écrit donc sous la forme :

$$I_{tot}(P,\lambda) = I_{amb}(\lambda) + I_{diff}(P,\lambda) + I_{spec}(P,\lambda) \qquad \textbf{(Eq. 1.1)}$$

Cette expression doit être calculée pour chaque point visible de l'observateur. De manière à limiter les calculs, en particulier dans le cas d'objets approximés par un ensemble de facettes planes, des méthodes d'approximation sont habituellement utilisées. Celles-ci sont décrites dans les paragraphes qui suivent.

1.1.2 Ombrage de Gouraud

De manière à éviter le calcul d'intensité en tout point, et pour obtenir une continuité (de degré C^0) d'intensité entre les facettes, Gouraud [Goura 71] propose d'interpoler cette intensité à partir de l'intensité effectivement calculée en quelques points seulement de la facette. Ces points, où un calcul complet est effectué, sont les sommets de chaque facette.

Dans la mesure où le calcul de l'intensité nécessite la connaissance de la normale au sommet considéré, celle-ci est calculée par moyenne des normales de chaque facette à laquelle le sommet appartient . Notons que, si la définition mathématique de la surface approximée par les facettes est connue, alors la vraie normale peut être utilisée.

Lorsque les valeurs d'intensité ont été calculées, une double interpolation linéaire est effectuée pour chaque point de la facette, de manière à assurer une continuité de l'intensité le long des arêtes communes entre facettes. Les équations d'interpolations , ainsi qu'un exemple de l'aspect visuel obtenu, sont représentés sur la figure ci-dessous (Figure 1.3).

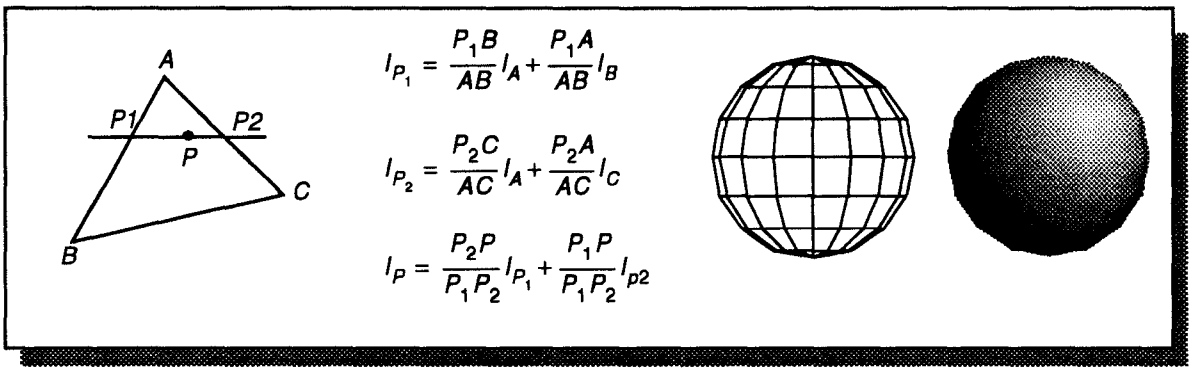


Figure 1.3 ***Principe de l'interpolation des intensités et aspect visuel sur une sphère facettisée***[1]

Le principal défaut de cette interpolation est qu'elle ne permet pas de rendre correctement les variations d'intensité spéculaire. En effet, celles-ci génèrent une forte variation d'intensité sur une petite surface. Cette variation locale entre en opposition avec le principe de l'interpolation des intensités, puisque celui-ci suppose une variation continue de l'intensité au sein d'une facette. La seule façon de lever ce problème est alors de diminuer la taille des facettes, ce qui a cependant pour inconvénient d'augmenter leur nombre.

1.1.3 Ombrage de Phong

L'équation (Eq. 1.1) montre que pour pouvoir calculer correctement la valeur de l'intensité spéculaire en un point, il est nécessaire de connaitre le vecteur normal à la surface en ce point. Phong [Phong 75] propose donc d'interpoler, non plus les intensités, mais les normales, à partir des valeurs de normale aux sommets de la facette. La formule de calcul de l'intensité décrite précédemment (Eq. 1.1) est alors calculée en tout point, en utilisant la valeur de la normale obtenue par interpolation (Figure 1.4).

1. Les images des sphères illuminées par les méthodes de Gouraud et Phong ont été aimablement fournies par Samuel Degrande.

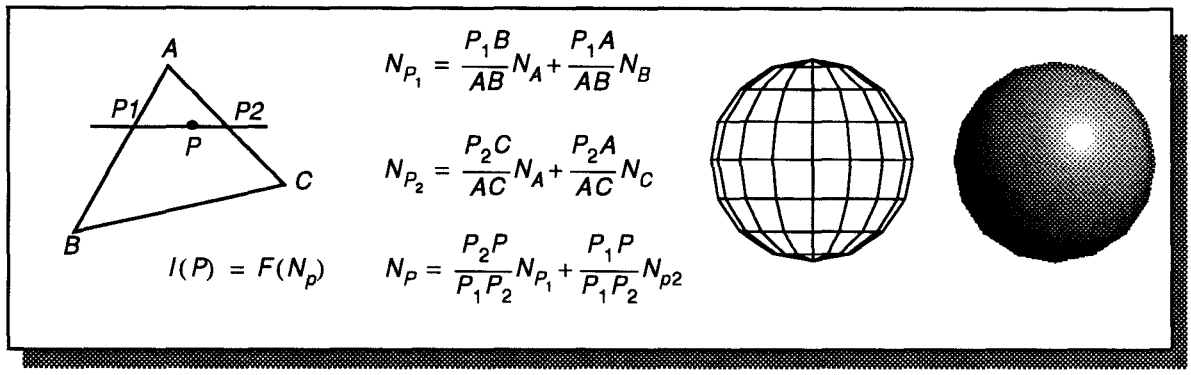


Figure 1.4 ***Principe de l'interpolation des normales et aspect visuel sur une sphère facettisée***

Ce procédé permet ainsi de capter de manière plus précise les variations spéculaires à la surface d'un objet, comme le montre la Figure 1.4 (l'aspect visuel est à comparer avec celui obtenu avec la méthode de Gouraud sur la Figure 1.3). Il pose néanmoins le problème du coût de calcul, puisqu'il nécessite un calcul d'éclairage complet par point visible.

1.1.4 Conclusion

Les modèles d'illumination locale que nous avons présentés dans ce paragraphe, bien qu'empiriques, offrent l'avantage de la simplicité. Les modèles d'ombrage associés sont très largement utilisés. En particulier, le principe d'interpolation de Gouraud est cablé dans la plupart des stations de travail graphiques. Différentes études sont en court pour intégrer l'algorithme d'interpolation de Phong au niveau physique de ces machines [Lefev 92].

Elle ne permettent cependant pas, de par leur principe, de tenir compte de deux phémomènes très importants en synthèse d'images : les ombres portées et les interactions entre objets. En effet, lorsque l'intensité est calculée en un point d'un objet, celui-ci est considéré comme visible de chacune des sources utilisées dans la scène. De ce fait, les objets ne peuvent se cacher les uns des autres et aucune ombre ne peut apparaître. D'autre part, l'obtention d'images de synthèse réalistes nécessite de pouvoir prendre en compte les interréflexions de lumière qui se produisent entre les objets, ce qui n'est pas possible avec ces modèles.

La prise en compte de l'ensemble de ces phénomènes conduit à une illumination globale de la scène. Nous allons à présent décrire deux algorithmes permettant d'offrir une solution à ce problème, chacun d'entre eux appliquant cependant un certain nombre de restrictions sur les types de réflexions prises en compte.

1.2 Le lancer de rayons

1.2.1 Principe

Le lancer de rayons est une généralisation de l'algorithme de *tracé de rayons* (*"ray casting"*), utilisé pour l'élimination des parties cachées [Appel 68], [Golds 71]. Dans cet algorithme, des rayons sont envoyés depuis la position de l'observateur, à travers chaque pixel de l'écran. Des calculs d'intersection sont alors appliqués avec les objets de la scène, de manière à trouver l'objet le plus proche, visible dans la direction du rayon (Figure 1.5).

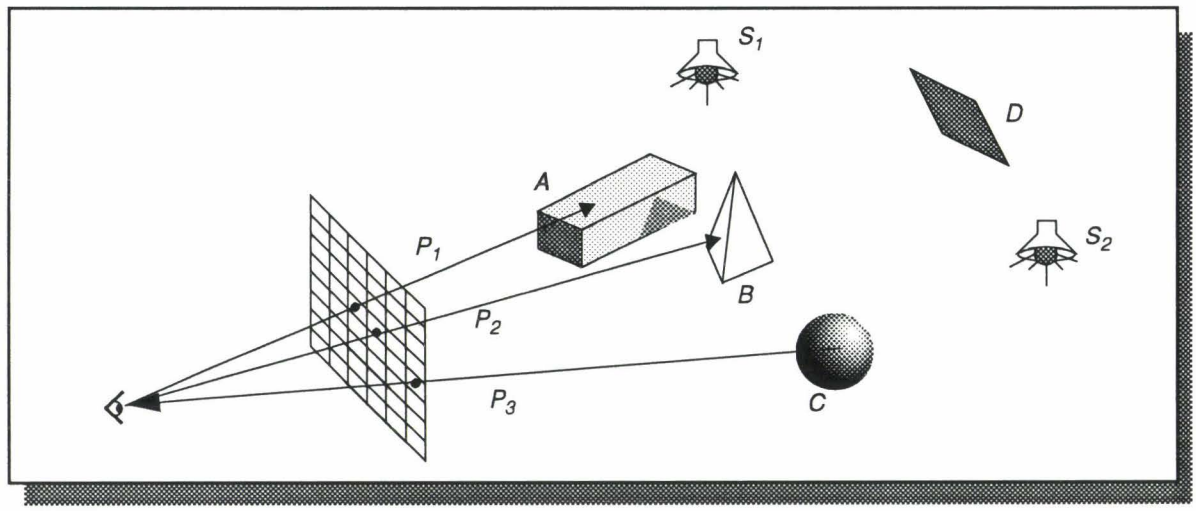


Figure 1.5 ***Principe du tracé de rayons***

Un calcul d'éclairement local, tel qu'il a été décrit au paragraphe 1.1.1, peut alors être appliqué aux points visibles, afin d'obtenir une scène illuminée.

De manière à utiliser un véritable modèle d'illumination, intégrant les phénomènes d'ombrage, de réflexion et de transmission de la lumière, Kay [Kay 79], puis Whitted [Whitt 80], généralisèrent le tracé de rayons pour obtenir l'algorithme de *lancer de rayons* (*"ray tracing"*), que nous allons présenter ci-dessous.

1.2.2 Algorithme

Afin de déterminer, d'une part l'objet visible en chaque pixel, et d'autre part son éclairage (la couleur du pixel), un rayon est lancé depuis la position de l'observateur à travers chaque pixel de l'écran. Pour chaque rayon, l'intersection avec l'objet le plus proche dans cette direction est calculée. En utilisant les notations de la Figure 1.6, nous allons décrire les différents types de rayons utilisés dans cet algorithme.

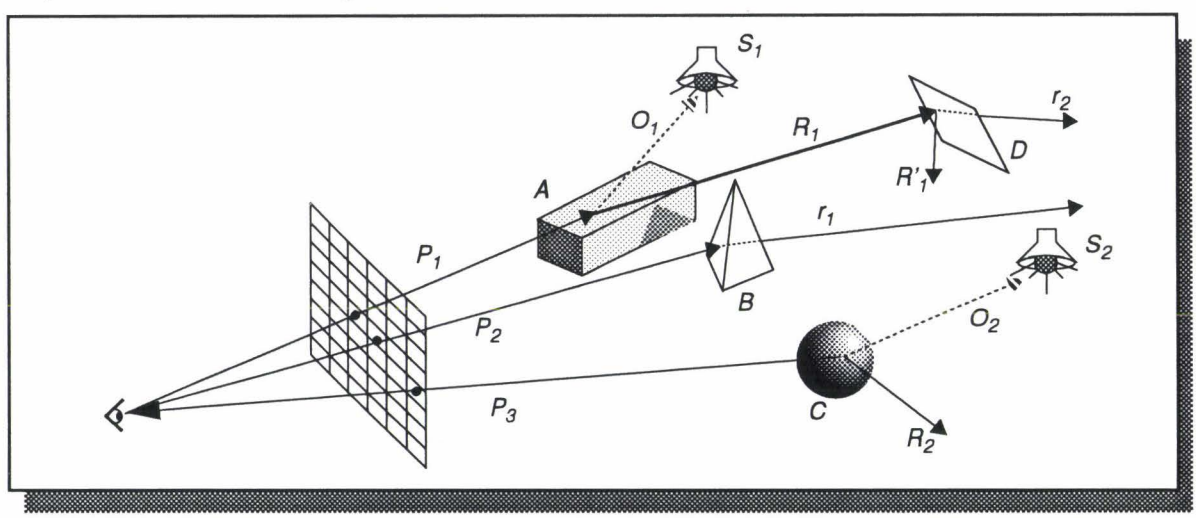


Figure 1.6 ***Principe du lancer de rayons***

Les premiers rayons lancés correspondent au même type de rayon que dans l'algorithme de tracé de rayons, et sont appelés rayons primaires (rayons P_1, P_2, P_3). Ils permettent de déterminer l'objet visible dans chaque pixel, et fournissent le point d'intersection entre le rayon et cet objet.

A partir d'un point d'intersection, trois types de rayons secondaires sont alors lancés :

- **les rayons d'ombrage** : ces rayons permettent de calculer l'intensité lumineuse reçue depuis chaque source au point considéré. Leur rôle est de déterminer si le point de départ du rayon est visible depuis une source donnée, en calculant les intersections des objets de la scène avec le rayon. Si aucune intersection n'est trouvée (rayons O_1 et O_2 de la Figure 1.6), le point est visible depuis la source, et l'intensité qu'il reçoit est calculée, en utilisant la formule (Eq. 1.1).

  L'utilisation des rayons d'ombrage permet ainsi de tenir compte des occultations entre objets, et de fournir la notion d'ombre portée.

- **les rayons réfléchis** : pour tenter de tenir compte des interactions entre les objets d'une scène (la lumière réfléchie par un objet peut éclairer un autre objet, visible de l'observateur), un rayon réfléchi est envoyé dans la scène (rayons R_1, R_2, R'_1 de la Figure 1.6). Ce rayon est calculé en supposant la surface des objets parfaitement spéculaire (une seule direction de réflexion), et en utilisant la loi de Fresnel, qui stipule que les rayons incidents et réfléchis sont coplanaires, et forment le même angle avec la normale à l'objet au point considéré (Figure 1.7).

- **les rayons réfractés** : ces rayons permettent de tenir compte de la transparence de certains objets. Dans le cas où un rayon primaire (ou secondaire) intersecte un objet non totalement opaque, un rayon réfracté est propagé dans la scène (rayon r1, r2 de la Figure 1.7). Ce rayon réfracté est calculé à l'aide de la loi de Descartes : les rayons incidents et transmis sont coplanaires, et vérifient la formulation donnée sur la figure ci-dessous.

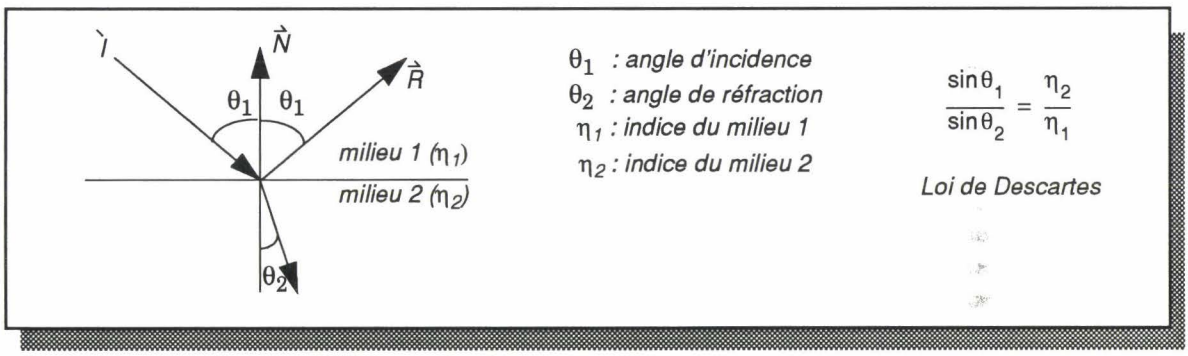


Figure 1.7 ***Géométrie pour les lois de Fresnel et Descartes***

Chaque rayon primaire engendre alors une arborescence de rayons, chaque rayon réfléchi ou réfracté générant à son tour des rayons d'ombrage, réfléchis et réfractés. L'intensité finale d'un pixel est calculée en fonction de l'intensité fournie par chaque rayon d'ombrage, à laquelle est ajoutée la contribution de toutes les branches inférieures de l'arborescence issue du pixel.

Le processus de génération de l'arborescence est poursuivi jusqu'à ce que la contribution de l'un des rayons soit jugée négligeable, ou que la profondeur atteinte soit supérieure à une profondeur maximale fixée. Cette arborescence est schématisée, pour un pixel, sur la figure ci-dessous (Figure 1.8), en fonction des notations de la Figure 1.6.

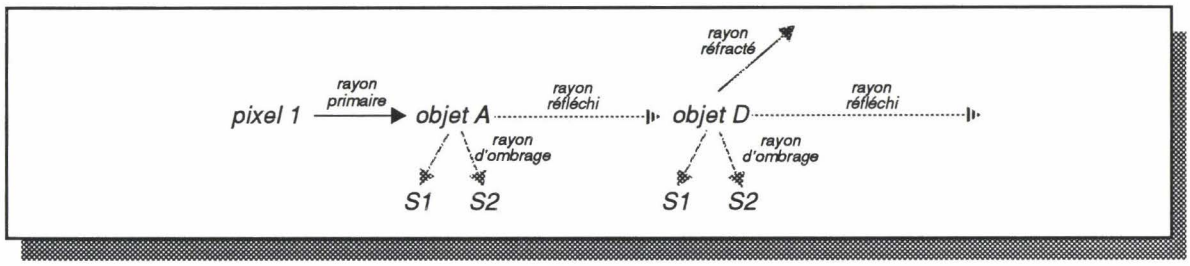


Figure 1.8 ***Exemple d'arborescence générée par le lancer de rayons***

Du fait du coût de calcul très important de cet algorithme, en particulier dû aux calculs d'intersection, un certain nombre de méthodes ont été envisagées pour accélérer son

traitement. Ces méthodes, font appel à des notions de cohérence entre rayons, ou de diminution du nombre d'intersections à calculer, et seront examinées au chapitre 3, paragraphe 3.3.

1.2.3 Conclusion

Le lancer de rayons est un algorithme puissant, permettant d'obtenir des images réalistes, en prenant en compte de nombreux phénomènes intervenant lors de l'illumination d'une scène. Il est cependant mal adapté à la prise en compte des réflexions diffuses, ou des interactions autres que ponctuelles entre objets. Bien que de nombreux travaux aient été effectués pour améliorer les phénomèmes pris en compte, tels que les réflexions ou transmissions non parfaitement spéculaires, l'obtention d'ombres moins tranchées, l'utilisation de sources non ponctuelles, etc... ([Cook 84], [Kajyi 86], [Ward 88], [Picot 92]), la prise en compte d'interactions diffuses reste difficile et très coûteuse. Nous allons donc décrire une seconde méthode d'illumination, la radiosité, dont le but est de permettre le calcul de l'illumination globale dans le cas de surfaces parfaitement diffuses.

1.3 La radiosité

La méthode de radiosité a été introduite par Goral [Goral 84] et Nishita [Nishi 85], dans le but de modéliser les interactions lumineuses entre des surfaces diffuses. En effet, les modèles de réflexion que nous avons décrits précédemment ne prennent pas en compte les réflexions d'objet à objet entre des surfaces diffuses, et ne permettent ainsi pas de calculer précisément l'illumination globale d'une scène.

Il faut cependant noter que la radiosité, qui est une adaptation à la lumière visible d'une technique utilisée dans l'étude des transferts radiatifs de chaleur[1][Gebha 61], [Farell 76], ne permet le calcul effectif de l'illumination globale que dans la mesure où l'environnement n'est constitué que de surfaces diffuses. Dans le cas contraire, l'illumination obtenue ne reflètera plus la réalité, du fait de l'oubli de certaines réflexions.

Nous allons présenter, dans les paragraphes qui suivent, l'algorithme de radiosité et différentes amélioration qui y ont été apportées.

1.3.1 Système d'équations et résolution

Nous allons tout d'abord présenter les hypothèses et notations utilisées dans le cadre de la méthode de radiosité. Nous établirons ensuite le système d'équations régissant les échanges d'énergie lumineuse diffuse, puis nous détaillerons chacune des étapes de résolution de l'algorithme.

Hypothèses et notations

Le principe de la radiosité est d'établir un équilibre énergétique au sein d'une scène, entre tous les objets la composant. De manière à assurer la conservation de l'énergie au sein de l'environnement, celui-ci est clos. D'autre part, certaines hypothèses sont utilisées :

- Chaque objet émet ou réfléchit l'énergie lumineuse de manière parfaitement diffuse. Quelle que soit la direction incidente d'un flux lumineux, celui-ci sera réfléchi avec la même intensité dans toutes les directions de l'espace.
- L'énergie lumineuse est supposée émise uniformément sur toute la surface d'un objet. Pour assumer cette contrainte, les objets sont divisés en petits éléments de surface, appelés *facettes*, pour lesquels l'émission d'énergie par unité de surface est supposée constante.

Notons que, contrairement aux modèles précédents qui différenciaient les objets et les sources, tout objet est considéré comme une source. Une conséquence de cette remarque est que les

1. échanges de flux de chaleur entre différents objets

sources utilisées en radiosité ne sont plus ponctuelles, mais possèdent une surface, ce qui permettra d'ontenir un réalisme accru lors du calcul des échanges lumineux.

A partir de ces hypothèses, les différentes quantités utilisées en radiosité seront notées :

- B : la radiosité de la facette j. Elle représente la quantité totale d'énergie quittant la facette, par unité de temps et pas unité de surface (Watt/m^2).
- H_j : le flux d'énergie qui arrive sur la facette j issu de toutes les autres facettes de l'environnement (Watt/m2).
- E_j : la valeur d'énergie propre à la facette. Cette quantité est n'est différente de zéro que dans le cas où la facette appartient à une source lumineuse.
- ρ_j : la réflectivité de la surface. Cette quantité traduit la fraction de l'énergie parvenant sur la facette qui est réémise dans l'environnement.

Notons que ces différentes valeurs dépendent de la longueur d'onde utilisée. Nous allons à présent donner les différentes équations utilisées en radiosité.

Système d'équations

Chaque facette possède une énergie lumineuse, qui provient soit d'elle même (si la facette appartient à une source lumineuse), soit de l'énergie qu'elle reçoit du reste de l'environnement et qu'elle réfléchit totalement ou en partie. Il apparait alors que les différentes facettes de l'environnement sont liées par une relation de dépendance. Cette relation, qui permet de définir la radiosité d'une facette, s'exprime sous la forme :

$$B_j = E_j + \rho_j H_j \qquad \textbf{(Eq. 1.2)}$$

Le flux H_j correspond à la part de la radiosité qui est émise par chaque facette de l'environnement, et qui arrive sur la facette j. Cette quantité peut se réécrire :

$$H_j = \sum_{i=1}^{N} B_i F_{ij}$$

où N représente le nombre de facettes de la scène, et F_{ij} est appelé *facteur de forme*. Il correspond à *la fraction de l'énergie qui quitte la facette i pour arriver sur la facette j*. Dans la mesure où les surfaces sont toutes purement diffuses, ce terme est uniquement géométrique, et permet de traduire la façon dont deux facettes se perçoivent.

L'équation (Eq. 1.2) peut alors se réécrire sous la forme :

$$B_j = E_j + \rho_j \sum_{i=1}^{N} B_i F_{ij} \qquad \textbf{(Eq. 1.3)}$$

Il existe une équation de ce type pour chaque facette de la scène. Un système d'équations linéaire est donc obtenu, traduisant les relations energétiques qui existent entre toutes ces facettes :

$$B_j = E_j + \rho_j \sum_{i=1}^{N} B_i F_{ij} \text{ pour } j \in [1, N] \qquad \textbf{(Eq. 1.4)}$$

Ces équations se réécrivent, sous la forme matricielle suivante :

$$\begin{bmatrix} 1-(\rho_1 \cdot F_{1,1}) & -\rho_1 \cdot F_{1,2} & \cdots\cdots & -\rho_1 \cdot F_{1,N} \\ -\rho_2 \cdot F_{2,1} & 1-(\rho_2 \cdot F_{2,2}) & \cdots\cdots & -\rho_2 \cdot F_{2,N} \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ -\rho_N \cdot F_{N,1} & -\rho_N \cdot F_{N,2} & \cdots\cdots & 1-(\rho_N \cdot F_{N,N}) \end{bmatrix} \cdot \begin{bmatrix} B_1 \\ B_2 \\ \cdots\cdots \\ B_N \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \cdots\cdots \\ E_N \end{bmatrix} \qquad \textbf{(Eq. 1.5)}$$

Etapes de résolution

Nous allons à présent décrire les différentes étapes nécessaires à la résolution du système d'équations (Eq. 1.4), établi dans le paragraphe précédent. Dans ce système, seules les valeurs

de radiosité initiale (énergie lumineuse émise localement) pour chaque facette, ainsi que leurs coefficients de réflectance, sont connus. Les valeurs à calculer sont donc les facteurs de forme entre facettes, puis les radiosités. Trois étapes successives peuvent alors être distinguées :

- Calcul des facteurs de forme : la détermination du facteur de forme entre deux facettes nécessite de prendre en compte les occultations qui peuvent se produire entre les objets de l'environnement. De nombreux algorithmes ont été proposés dans ce but, et ils seront détaillés et analysés dans les chapitres 2 et 3.
- Résolution du système : la résolution du système nécessite théoriquement l'inversion de la matrice des facteurs de forme. Cohen [Cohen 85] propose d'utiliser une méthode rapide, la méthode de Gauss-Siedel, appelée aussi méthode des approximations successives. La méthode consiste, à partir d'une estimation initiale des radiosités (B_i), à calculer, par approximations successives, des valeurs de plus en plus proches de la solution exacte. Ce qui revient à poser la suite d'approximations suivantes :

$$B_j^{(0)} \text{ estimation initiale}$$

$$B_j^{(k+1)} = E_j + \rho_j \sum_{i=1}^{N} F_{ij} B_i^{(k)} \text{ pour } j \in [1,N]$$

  Le résultat est indépendant de l'estimation initiale choisie, ce qui n'est cependant pas le cas pour la vitesse de convergence. Il faut donc choisir convenablement les valeurs des $B_j^{(0)}$, et le choix qui semble le plus logique est de leur donner les valeurs des E_j correspondants. Dans ce cas, les approximations reviennent à prendre en compte les réflexions successives de la lumière entre les différentes facettes. Cohen estime qu'un faible nombre d'itérations (de l'ordre d'une dizaine) est nécessaire pour obtenir une bonne approximation de la convergence finale.
- Visualisation de la scène : après calcul de la radiosité de chaque facette, celle-ci est transformée en intensité lumineuse, par l'intermédiaire de la relation suivante qui relie les deux termes :

$$B_j = I_j \times \pi$$

  Cependant, la valeur de radiosité obtenue est une valeur au centre de la facette; pour obtenir des variations continues de couleur et de luminosité, il faut donc procéder à 2 étapes supplémentaires : obtenir une valeur de radiosité aux sommets de facettes (par moyenne des valeurs de radiosité des facettes adjacentes); puis effectuer une double interpolation linéaire de la couleur, en fonction des valeurs de radiosités en chaque sommet de facette. Cette seconde étape est similaire à l'étape d'interpolation utilisée par Gouraud, et décrite ci-dessus (paragraphe 1.1.2).

L'enchainement de ces trois étapes est représenté sur la Figure 1.9.

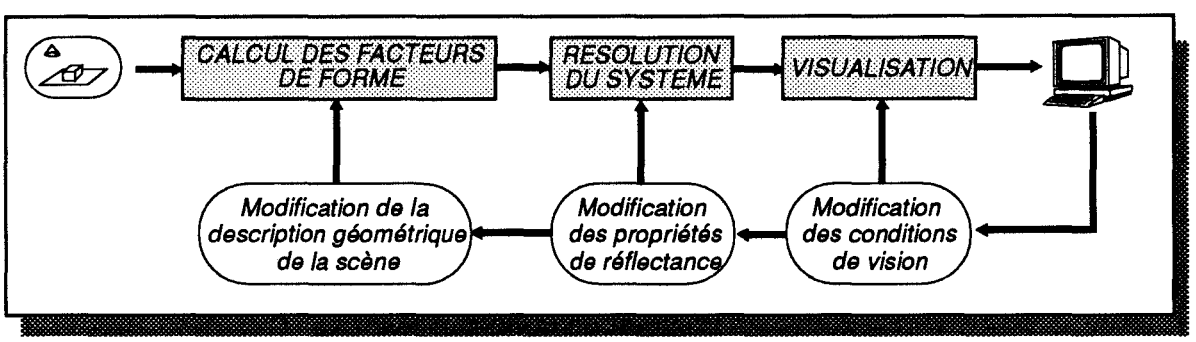


Figure 1.9 ***Etapes de calcul pour l'algorithme de radiosité***

Ce shéma montre l'indépendance des ces trois étapes, et met en valeur un certain nombre d'opérations applicables sur le schéma de résolution, sans que celui-ci ait à être repris entièrement.

Ainsi, après affichage de la scène, l'utilisateur peut modifier la position d'observation, sans que les calculs de facteurs de forme ou d'inversion de la matrice aient à être effectués une nouvelle fois, puisque par principe, la radiosité est l'indépendante du point de vue. L'utilisation de matériel spécialisé dans l'affichage de facettes (stations équipées d'accélérateurs graphiques câblés) permet alors de générer très rapidement des séquences d'images.

D'autre part, la modification des propriétés de réflectance des objets ne nécessite pas de recalculer les facteurs de forme, mais simplement de recommencer l'étape d'inversion de la matrice, puisque certains de ses coefficients peuvent alors avoir été modifiés.

Dans le cas où l'utilisateur désire modifier la géométrie de la scène, il est alors nécessaire de reprendre l'ensemble des calculs de facteurs de forme, puisque ceux-ci peuvent avoir changé en fonction du déplacement des objets.

Inconvénients

L'algorithme de radiosité permet de résoudre le problème de l'illumination globale, dans le cas de surfaces parfaitement diffuses. Tel qu'il vient d'être présenté, il pose cependant deux problèmes majeurs :

- Il nécessite un espace mémoire très important pour la repésentation de la matrice des facteurs de forme. Bien que ce coût mémoire puisse être divisé par un facteur 2, du fait de la relation de réciprocité qui existe entre les facteurs de forme de deux facettes, il croît en $O(N^2)$ pour N représentant le nombre de facettes de la scène. A titre d'exemple, la quantité de mémoire nécessaire à la représentation de la matrice pour une scène constituée de 35000 facettes est d'environ 2,5 Go.
- Une image ne peut être obtenue qu'après le calcul complet de la matrice, puis de son inversion. Les temps de calculs sont très importants, et interdisent toute interactivité avec cette approche.

Nous allons détailler, dans le paragraphe suivant, une reformulation de l'algorithme de radiosité qui permet de lever ces deux inconvénients.

1.3.2 Radiosité progressive

L'algorithme de radiosité progressive a été proposé par Cohen dans le but de résoudre les deux problèmes que nous venons d'évoquer. En reformulant les équations de radiosité, le coût de mémorisation des facteurs de forme est considérablement réduit, et un peu d'interactivité est introduit.

Principe

Dans l'algorithme précédent, le calcul de la radiosité d'une facette se fait par collecte de l'énergie diffusée par les autres facettes (*"gathering"*). De ce fait, une seule facette à la fois peut être éclairée. Cohen [Cohen 88] propose d'inverser ce procédé, et de calculer la distribution de l'énergie diffusée par une facette (*"shooting"*) sur l'ensemble des autres facettes (Figure 1.10).

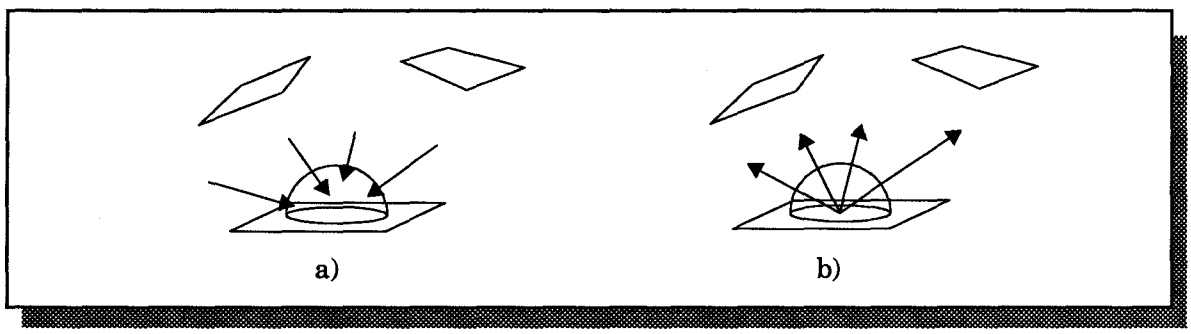


Figure 1.10 ***Radiosité par collecte (a) et par émission (b)***

De cette manière, à chaque étape d'émission, l'ensemble des facettes visibles depuis la facette émettrice reçoit une partie de l'énergie lumineuse diffusée. C'est donc l'ensemble de l'environnement qui est concerné par l'éclairage, ce qui permet, après chaque phase d'émission, de disposer d'une scène ayant reçue suffisamment d'énergie pour pouvoir être affichée, même si l'ensemble des itérations n'a pas été effectué. Ceci introduit donc une certaine interactivité pour l'utilisateur, qui peut dès lors obtenir des images rapidement, même si elles ne sont pas d'excellente qualité, dès les premières itérations. La qualité des images (de l'éclairage) *se raffine progressivement* au fur et à mesure des itérations.

Le système d'équations se réécrit alors sous la forme :

$$B_j = B_j + B_i \times (\rho_j F_{ji}) \text{ pour tous les } j \text{ avec } F_{ji} = F_{ij} \times \frac{S_i}{S_j} \text{ (cf paragraphe 2.1)}$$

Cette formulation appelle deux remarques :

- il est nécessaire de conserver une information supplémentaire pour chaque facette, qui permettra de mémoriser la quantité d'énergie que cette facette a reçue des autres facettes et qu'elle n'a pas encore diffusée, par réflexion, dans l'environnement. Cette énergie sera appelée *radiosité latente* d'une facette, puisqu'elle représente la part de la radiosité d'une facette qui est en attente de diffusion. Lorsqu'une facette est sélectionnée comme facette émettrice, sa radiosité latente est remise à zéro lorsque l'émission a eu lieu.
- chaque étape de raffinement consiste à calculer les facteurs de forme entre une facette émettrice et l'ensemble des autres facettes. Lorsque la mise à jour des radiosités a été effectuée, ces facteurs de forme, qui représentent alors une colonne de la matrice complète, peuvent être "effacés", puisqu'ils ne sont plus nécessaires pour les étapes suivantes. De ce fait, le coût de mémorisation, qui était en $O(N^2)$ lors du calcul de la matrice complète, devient en $O(N)$ avec cette approche.

Les deux principaux inconvénients de l'approche précédente, que nous qualifierons désormais de *"radiosité par calcul de la matrice complète des facteurs de forme"* sont donc éliminés par cette seconde approche, que nous appelerons *"radiosité par raffinement progressif"*.

Il est cependant important de noter que ces deux approches ne possèdent pas la même vitesse de convergence, puisque dans la méthode de raffinement progressif, une facette peut être sélectionnée plusieurs fois pour diffuser son énergie latente. Ceci s'explique par le fait qu'une facette est toujours susceptible de recevoir de l'énergie depuis d'autres facettes de la scène à un moment quelconque du processus de raffinement. De ce fait, la même colonne de facteurs de forme peut avoir à être calculée plusieurs fois.

Néanmoins, la méthode de raffinement progressif permet de ne pas calculer l'intégralité des colonnes de facteurs de forme, dans la mesure où des facettes qui possèdent une radiosité latente faible (facettes situées dans l'ombre de certains objets) peuvent ne pas être sélectionnées. Ces facettes peuvent se révéler particulièrememt nombreuses pour certaines scènes. Dans ce cas, elles seront effectivement éclairées (même faiblement), mais d'émettront jamais, allégeant d'autant le nombre d'itérations.

Choix de la facette émettrice

Le choix de la facette émettrice à considérer pour chaque nouvelle itération est important, puisqu'il conditionne la rapidité de convergence de la méthode. En effet, une facette ne disposant que d'une radiosité latente faible ne diffuse que peu d'énergie dans la scène. Les facettes de l'environnement ne reçoivent donc qu'une petite quantité d'énergie, la variation de radiosité étant trop faible pour être directement perçue au niveau de l'affichage. Par contre, si une facette disposant d'une énergie latente élevée est utilisée, sa contribution à l'illumination des autres facettes est importante, et la variation de radiosité qui en résulte permet d'obtenir une différence appréciable dans l'éclairement de la scène.

Cohen propose donc, à chaque nouvelle phase d'émission, de sélectionner la facette qui possède la plus grande radiosité latente. Ce choix permet d'assurer que, pour chaque itération, une

contribution maximale est apportée à l'illumination de chaque facette de la scène, conduisant à une convergence plus rapide.

Utilisation d'un terme ambiant

Cohen note cependant que, même en choisissant correctement la facette émettrice à chaque itération, des objets qui ne sont pas directement éclairés par celle-ci peuvent ne pas recevoir d'énergie lumineuse avant un grand nombre d'itérations.

De manière à obtenir une image exploitable dès les premières itérations, un terme d'éclairage ambiant est calculé à la fin de chaque émission. Il prend en compte la radiosité latente totale et la réflectivité moyenne de la scène. Lors de la phase d'affichage, cet éclairage ambient est distribué aux facettes, au prorata de leur surface.

Il est important de noter, cependant, que ce terme ambiant n'intervient en aucune façon dans le processus de résolution proprement dit : il ne sert qu'à afficher une solution plus précise de l'image très rapidement, et sa contribution décroît au fur et à mesure des émissions. Son utilisation ne permet donc pas une résolution plus rapide du système, puisqu'il n'est absolument pas utilisé dans le processus de résolution.

1.3.3 La subdivision adaptative

La méthode de radiosité nécessite de découper en un grand nombre de facettes les surfaces sur lesquelles sont effectuées les opérations d'éclairage. La précision des échanges lumineux, et donc du rendu final, dépend du nombre et de la taille de ces facettes. Il est cependant difficile d'évaluer, a priori, ces quantités. Nous allons préciser, dans un premier temps, les problèmes principaux que pose le découpage des objets en facettes, puis nous décrirons une méthode de découpage adaptatif de la scène qui permet de pallier ces problèmes. Nous préciserons, enfin, un certain nombre d'impératifs supplémentaires nécessaires pour assurer un rendu correct de l'image finale.

Le problème de la facettisation

La radiosité d'une facette est supposée constante sur toute la surface d'une facette. La conséquence de cette hypothèse est que, si la taille d'une facette est trop grande, les variations d'intensité qui s'y produisent ne peuvent pas être captées (Figure 1.11). Il est donc, a piori, nécessaire de disposer d'une facettisation suffisamment fine pour permettre de représenter ces variations, et d'obtenir une représentation aussi précise que possible des échanges lumineux qui se produisent en tout point de la scène.

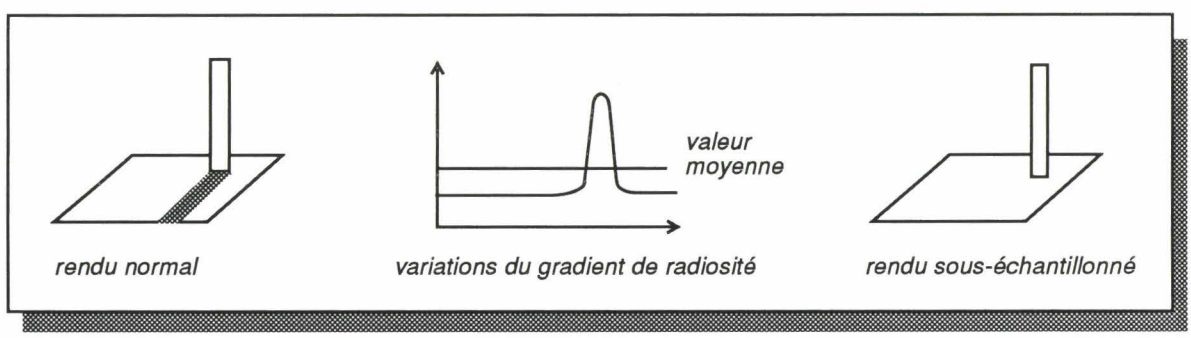


Figure 1.11 ***Le problème des variations locales du gradient de radiosité***

Cependant, il est important de remarquer que, plus le nombre de facettes augmente, plus le temps de calcul devient important. Dans le cas de la radiosité classique, le temps de calcul croît comme le carré du nombre de facettes, puisqu'il faut calculer l'ensemble des coefficients de la matrice. Dans le cas de la radiosité progressive, le même type de croissance est observable, puisque le nombre de facettes émettrices successives à considérer croît linéairement en fonction du nombre de facettes, de même que le nombre de facettes à considérer pour chaque émission.

D'autre part, il faut noter que les fortes variations du gradient de radiosité ne se produisent pas partout, mais uniquement en des endroits très localisés d'une scène. Typiquement, ces endroits se trouvent aux limites des ombres portées par un objet sur un autre, ou aux emplacements qui reçoivent directement une grande partie de l'illumination. Une grande partie d'une scène peut donc être découpée en facettes assez grandes, tandis que les parties posant problèmes doivent nécessairement être subdivisées plus finement.

Un découpage adaptatif automatique

Cohen [Cohen 86] propose une méthode qui permet de résoudre en partie ce problème, en introduisant un algorithme qui ne subdivise le découpage existant qu'aux endroits où cela s'avère nécessaire. Après chaque phase de mise à jour des radiosités, qu'il s'agisse d'une itération de la méthode de Gauss-Siedel, ou d'une émission de la radiosité progressive, les valeurs de radiosité entre les sommets voisins d'une facette sont examinées. Si ces valeurs présentent une différence trop importante, la facette à laquelle ils appartiennent est subdivisée en quatre facettes de même taille, et les calculs de facteurs de forme et de radiosité sont repris pour les nouvelles facettes générées. Ce processus est répété jusqu'à ce que la différence s'atténue, ou que la taille de la facette soit trop petite pour que le processus continue (ces deux seuils sont fixés par l'utilisateur). Chaque facette initiale qui doit être subdivisée est ainsi découpée, non pas en facettes, mais en *éléments*, ceux-ci pouvant à leur tour être eux-mêmes subdivisés en éléments plus petits.

Le nouveau niveau de structuration pour le découpage des objets réduit considérablement les calculs. Un élément est supposé posséder une radiosité constante sur toute sa surface, la radiosité des éléments d'une même facette pouvant bien entendu être différente. Tous les calculs d'éclairement portent alors sur ces nouvelles "entités", dans la mesure où ce sont elles qui doivent capter les variations de radiosité. De manière à limiter les coûts de calculs, les facteurs de forme ne sont pas évalués entre éléments , mais entre les éléments et les facettes de la scène (Figure 1.12.a), la radiosité d'une facette étant vue comme la moyenne des radiosités de ses éléments.

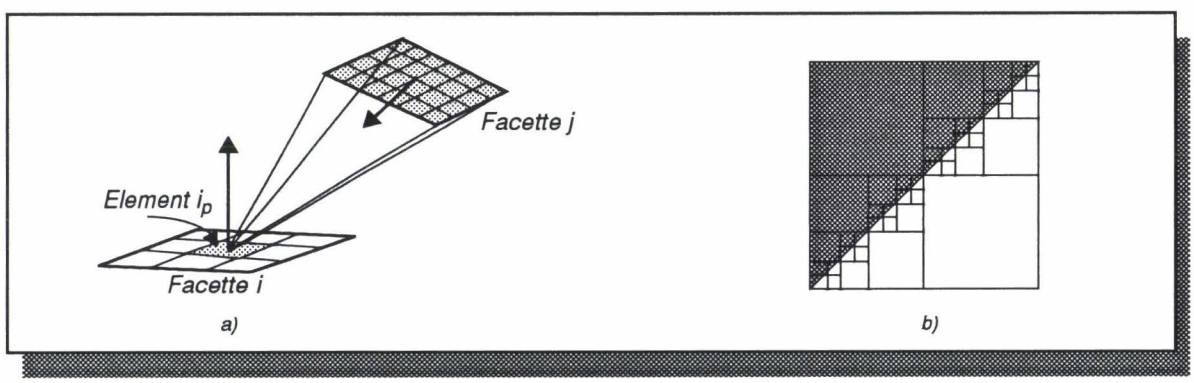


Figure 1.12 ***Facteur de forme élément-facette (a) et exemple de variation du gradient de radiosité générant un nombre élevé d'éléments (b)***

Notons que c'est la contribution énergétique des facettes de l'environnement sur chaque élément qui est calculée. La variation de radiosité au sein d'une facette est ainsi plus précise, puisque évaluée en plusieurs points de sa surface, tout en limitant les calculs à une solution intermédiaire entre le calcul des interactions entre facettes (moins coûteux mais moins précis) et celui des interactions entre éléments (plus précis, mais plus coûteux).

Différents problèmes peuvent cependant se poser lors de la subdivision, ne permettant plus de capter correctement les variations du gradient de radiosité. Le premier d'entre eux intervient lorsque la subdivision initiale de la scène n'est pas suffisante, c'est à dire que les facettes sont de trop grande taille. Dans ce cas, les variations de radiosité peuvent ne pas être repérées, car n'apparaissant pas au niveau des sommets de facette. D'autre part, si la variation ne s'effectue pas selon un axe parallèle à l'un des cotés d'une facette, le nombre de subdivisions à générer risque de croître très rapidement avant d'obtenir une approximation suffisante de son gradient

(Figure 1.12.b). Différents critères sont proposés dans [Airey 90] pour découper le long du gradient de radiosité lorsque cela est possible, et dans [Arvo 91], en se basant sur des comparaisons de valeur de facteurs de forme.

Les contraintes liées à la radiosité

La radiosité pose un certain nombre de contraintes sur la façon de découper les objets en facettes. Parmi celles-ci, nous allons évoquer tout d'abord le problème de la continuité de l'éclairage lors de l'affichage des facettes, ce problème ayant des conséquences directes sur la façon d'appliquer la subdivision adaptative.

Lorsque deux facettes adjacentes sont affichées selon le principe d'interpolation de Gouraud, il est nécessaire que les valeurs calculées le long des arêtes communes soient les mêmes. Dans le cas contraire, la discontinuité résultante sera flagrante. Cependant, la phase de modélisation d'abord, puis les phases de facettisation et de subdivision adaptative, peuvent générer des surfaces ou des facettes adjacentes, pour lesquelles la notion de continuité le long des arêtes communes est brisée. Ceci est schématisé sur la Figure 1.13.a. Dans ce cas, pour la facette *F*, la valeur d'intensité au point *b* est calculée, lors de l'interpolation, comme la valeur médiane des valeurs présentes aux points *a* et *c*. Cette valeur peut être fort différente de celle réellement calculée au point *b*, lorsqu'il appartient aux facette *f* et *f'*.

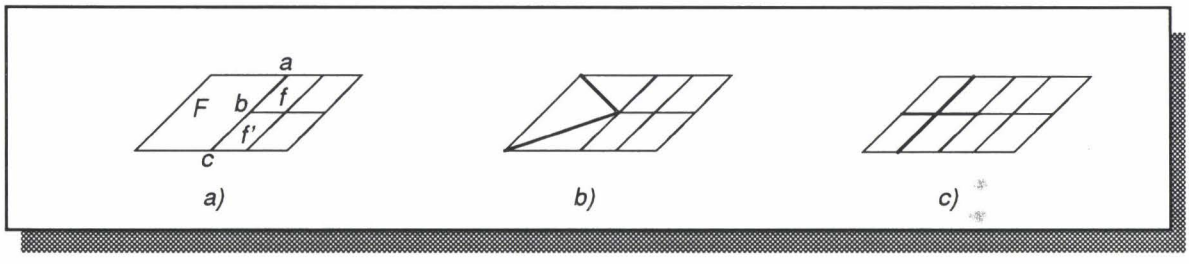


Figure 1.13 ***Discontinuité le long d'une arête (a) et notion d'ancrage (b) et désancrage (c)***

Pour résoudre ce problème, Baum [Baum 91] propose les notions d'*ancrage* et de *désancrage*. L'analyse des causes de ce problème montre qu'il survient du fait que le point *b* n'appartient pas à la facette *F*, et que l'interpolation de cette facette n'utilise par conséquent pas sa valeur. Lorsque ce cas survient, par exemple lors de la subdivision d'une facette en éléments, Baum propose d'ancrer la facette adjacente qui n'est pas subdivisée. Cet ancrage a pour but d'intégrer un point créé dans la description des deux facettes qui se partagent l'arête où il se trouve. De manière à ne pas créer un surplus de facettes et une propagation du processus d'ancrage, celui-ci est effectué de manière à ne pas créer de nouveau sommet. La facette ancrée est alors divisée en trois éléments, comme le montre la Figure 1.13.b.

Cependant, chacun des éléments ainsi créé peut alors avoir à se modifier à nouveau, selon qu'une autre facette adjacente se subdivise ou qu'il ait lui même à se subdiviser. Dans ce cas, avant toute autre opération la facette se désancre, avant d'entamer sa subdivision, ceci dans le but, d'une part de ne pas générer un nombre trop important de facettes, et d'autre part de conserver, autant que faire se peut, des facettes de même forme (Figure 1.13.c).

Le problème du découpage initial des objets en facettes, puis en éléments donne lieu à de nombreux travaux de recherche. Ils sont essentiellement basés sur l'analyse des problèmes liés au principe d'interpolation des radiosités lors de l'affichage. En effet, cette interpolation interdit la prise en compte des discontinuités d'éclairage qui peuvent se produire au sein d'une facette. La subdivision adaptative est l'une des solutions envisagée pour résoudre ce problème, mais nécessite souvent une subdivision très élevée des facettes, et ne peut être déclenchée que lorsque la subdivision initiale est suffisement fine pour détecter ces discontinuités. Un exemple de ces discontinuités est donné pour un cas simple sur la Figure 1.14.

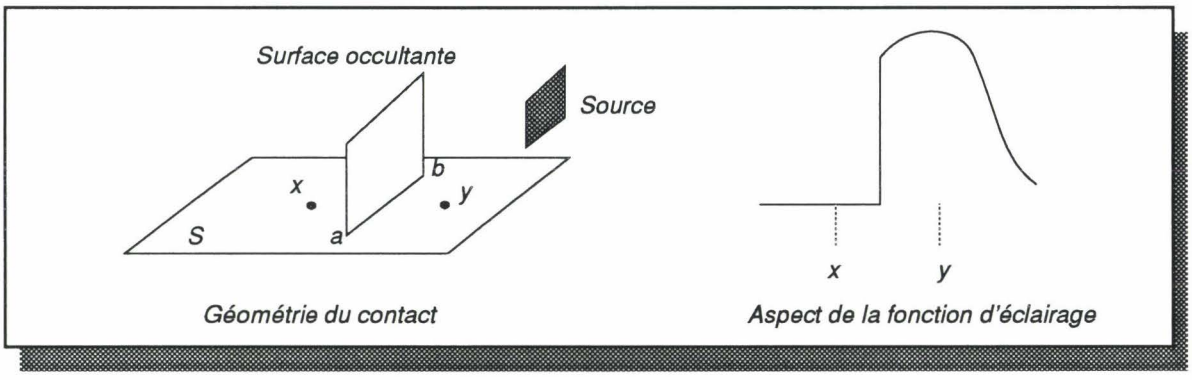


Figure 1.14 ***Exemple de discontinuité de contact (d'après [Lisch 92])***

Le segment (*a*,*b*) est en contact avec la surface horizontale *S*. Le point *x* est dans l'ombre de la surface verticale, tandis que le point *y* est éclairé. La fonction d'éclairage possède donc une discontinuité le long du segment (*a*,*b*). Si celle-ci n'est pas prise en compte, des facettes peuvent être générées de telle manière qu'elles soient recouvertes par une partie du segment. Lors du calcul de radiosité, une partie de chaque facette est à l'ombre, alors que l'autre partie est éclairée. Lors de la phase d'interpolation pour le rendu, une "fuite" de lumière sera observée du côté ombré, les facettes coupées par le segment se trouvant en partie éclairées du mauvais côté.

Ce type de discontinuité peut être pris en compte par la subdivision adaptative, mais risque de générer un très grand nombre de subdivisions, sans que la "fuite" soit nécessairement "colmatée". D'autres type de discontinuités apparaissent à la limite des ombres crées, sur des objets, par d'autres objets occultant.

Les études sur ce sujet se portent sur la génération d'un prédécoupage en fonction des ombres générées par les sources primaires, c'est à dire les objets qui possèdent une énergie initiale [Campb 90], [Heckb 92], [Lisch 92]. Un pré-processus est alors utilisé, et permet d'appliquer un premier découpage de la scène selon les lignes de discontinuité repérées. Bien que ces approches permettent de traiter correctement les discontinuités crées par les sources primaires, elles restent très coûteuses en temps de calcul, et ne permettent pas, à ce jour, de déterminer les discontinuités issues des réflexions secondaires.

1.4 Conclusion

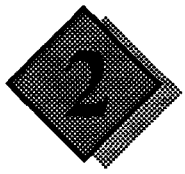
Les méthodes d'illumination locale, si elles offrent l'avantage de la simplicité, ne permettent pas la prise en compte des phénomènes complexes qui régissent les interactions lumineuses entre les objets d'une scène. En particulier, les phénomènes d'ombrage, de réflexion et de réfraction ne sont pas considérés.

Des méthodes comme le lancer de rayons ou la radiosité, permettent chacune, en fonction d'hypothèses restrictives assez fortes, de résoudre le problème de l'illumination globale. Dans le cas du lancer de rayons, celui-ci est résolu en ne considérant que des surfaces purement (ou fortement) spéculaires, et pour un point de vue donné. La génération d'une nouvelle image, due au déplacement de l'observateur, nécessite cependant un recalcul complet de l'éclairage.

La méthode radiosité ne considère que des surfaces purement diffuses, dont l'éclairement de ène est indépendant du point de vue. Le calcul de l'équilibre énergétique entre les objets permet donc la génération dynamique de séquences d'images lorsque l'observateur se déplace dans la scène.

Des modèles d'illumination globale beaucoup plus évolués ont été développés ces dernières années, permettant d'intégrer à la fois les phénomènes diffus et spéculaire au sein du même modèle. Ils ne sont cependant pas étudiés dans cette thèse, puisque nous nous limitons strictement au modéle purement diffus que représente la radiosité.

CHAPITRE



Méthodes projectives pour le calcul des facteurs de forme

La détermination des échanges lumineux par l'algorithme de radiosité est étroitement liée à la notion de facteur de forme. Celui-ci représente en fait le coeur de l'algorithme, tant par les quantités de calculs qu'il génère, que par l'importance que revêt la précision de son évaluation.

Le coût de l'évaluation des facteurs de forme représente plus de 90% du coût total de l'algorithme. Ceci est en particulier dû à la nécessité d'effectuer l'élimination des parties cachées depuis différents points de la scène, afin de permettre la prise en compte des occultations entre objets. D'autre part, cette évaluation se doit d'être aussi précise que possible, afin d'éviter l'apparition d'artefacts visuels sur les images générées, tels que la présence d'ombre ou de lumière là où elles n'existent pas.

De nombreuses méthodes ont été proposées pour approximer les facteurs de forme, en tentant à la fois de limiter les temps de calcul et d'augmenter la précision de l'évaluation des échanges lumineux. Nous nous attacherons, dans ce chapitre, à détailler un certain nombre de ces méthodes, ayant en commun le même principe de fonctionnement. Nous préciserons leurs avantages et inconvénients, puis nous établirons une comparaison entre les différentes méthodes.

2.1 Notion de facteur de forme

L'algorithme de radiosité nécessite de pouvoir déterminer la quantité d'énergie émise et/ou réfléchie par une facette qui atteint chacune des autres facettes de la scène. Le modèle de réflexion utilisé ne prenant en compte que les phénomènes purement diffus, la distribution d'énergie sera la même quelque soit la direction considérée. La quantité de lumière émise par une facette qui parviendra sur une facette quelconque dépendra donc directement de l'angle solide sous lequel la facette réceptrice est visible depuis la facette émettrice.

La proportion d'énergie qui quitte cette facette émettrice, d'indice i, pour aboutir directement sur une facette réceptrice d'indice j, sera alors appelée facteur de forme entre les facettes i et j, et notée F_{ij}. En suivant les conventions de la Figure 2.1a, l'expression mathématique du facteur de forme est donnée par :

$$F_{ij} = \frac{1}{A_i} \cdot \int_{A_i} \int_{A_j} \frac{\cos\theta_i \cdot \cos\theta_j}{\pi \cdot r^2} dA_j dA_i \qquad \text{(Eq. 2.1)}$$

où

- F_{ij} est le facteur de forme entre la facette i et la facette j,
- A_i et A_j représentent les aires des facettes i et j,
- dA_i et dA_j sont deux éléments de surface de chacune des deux facettes,
- r représente la distance entre 2 éléments de surface dA_i et dA_j.

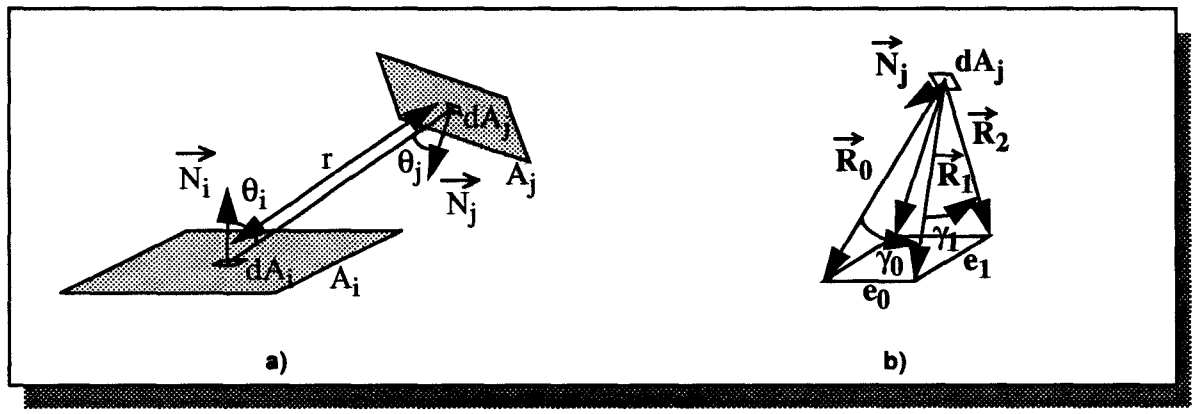


Figure 2.1 ***a) Géométrie des facteurs de forme b) Evaluation d'une intégrale de contour***

Par définition, nous avons les propriétés suivantes:

- ✓ $F_{ij} \in [0,1]$
- ✓ $\sum_j F_{ij} = 1$
- ✓ $F_{ij}A_i = F_{ji}A_j$

Les deux intégrales de surface présentes dans l'expression du facteur de forme (Eq. 2.1) peuvent être remplacées par deux intégrales de contour, en utilisant le théorème de Stokes [Goral 84]. Diverses formulations analytiques de la double intégrale des facteurs de forme peuvent être trouvées dans [Walto 87]. Dans la mesure où cette expression nous sera utile par la suite, nous pouvons réécrire l'expression de l'intégrale intérieure, en utilisant les conventions de la Figure 2.1 b, sous la forme suivante [Baum 89], [Picot 92]:

$$F_{dA_jA_i} = \frac{1}{2\Pi} \sum_{g \in G_i} \vec{N}_j \bullet \vec{\Gamma}_g \qquad \textbf{(Eq. 2.2)}$$

avec

- $F_{dA_jA_j}$ facteur de forme entre l'élément de surface dA_j et la facette A_j
- $\vec{N}_j$ vecteur normal à dA_j
- G_i l'ensemble des arêtes de la facette i
- $\vec{\Gamma}_g$ = vecteur de magnitude γ_g et de direction $R_g \wedge R_{g+1}$

L'utilisation d'une approche analytique pour l'évaluation des facteurs de forme nécessite de connaître exactement la visibilité entre les deux facettes. Dans la grande majorité des cas, il est bien évident que cette visibilité n'est pas connue, du fait des nombreuses occultations possibles entre les différents objets composants la scène. Ceci peut être exprimé, au niveau de la double intégrale de l'équation (Eq. 2.1), par l'introduction d'un terme de visibilité, *VIS*, valant 1 ou 0, qui traduit la visibilité entre deux éléments de surface dA_i et dA_j. L'intégrale se réécrit alors sous la forme:

$$F_{ij} = \frac{1}{A_i} \cdot \int_{A_i} \int_{A_j} \frac{\cos\theta_i \cdot \cos\theta_j}{\pi \cdot r^2} VIS(dA_i, dA_j)\, dA_j dA_i \qquad \textbf{(Eq. 2.3)}$$

La détermination de la visibilité entre facettes est analogue au problème bien connu en infographie de l'élimination des parties cachées. De nombreux algorithmes ont été proposés dans la littérature pour résoudre ce problème complexe. Un certain nombre d'entre eux ont donc été adaptés à l'évaluation des facteurs de forme, en tentant de concilier à la fois la précision de cette évaluation et la limitation des temps de calcul. Nous allons détailler, dans la suite de ce chapitre, certaines de ces méthodes, basées sur une notion de projection.

2.2 Principe des approches projectives

Les méthodes que nous appelerons projectives sont basées sur l'équivalent de Nusselt [Nusse 28], qui permet d'exprimer le facteur de forme sous forme d'une aire projetée. Ainsi, le facteur de forme entre l'élément de surface dS_i et la facette S_j de la Figure 2.2.a) est égal à l'aire de la projection orthogonale sur le plan support de dS_i de la région SP_j de la sphère unité obtenue en projetant radialement la facette S_j. La conséquence directe de cet équivalent est que toutes les facettes qui occupent le même angle solide, et qui ont donc la même aire projetée, possèdent le même facteur de forme avec l'élément de surface situé au sommet de cet angle. La facette S_k recevrait ainsi la même quantité d'énergie que la facette S_j si celle-ci ne la cachait pas de dS_i.

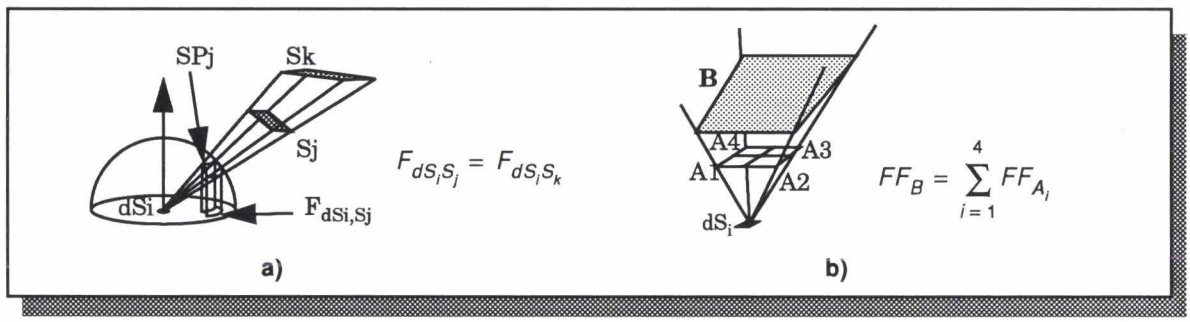


Figure 2.2 *a) Equivalent de Nusselt b) Somme d'angles solides*

Partant de cette remarque, il est possible d'envisager le calcul du facteur de forme d'une facette quelconque en fonction de celui d'une ou plusieurs autres facettes. Supposons, par exemple, que le facteur de forme des facettes A_i soit connu (voir Figure 2.2 b). Si une facette B occupe le même angle solide que celui défini par l'ensemble des facettes A_i, alors:

$$F_{dS_i, B} = \sum_i F_{dS_i, A_i}$$

Les approches projectives utilisent ce principe pour évaluer le facteur de forme d'une facette quelconque, en utilisant une surface particulière $\mathcal{S}$, découpée en angles solides élémentaires pour lesquels le facteur de forme sera précalculé. Pour déterminer l'angle solide occupé par une facette, il suffit de connaître l'ensemble des angles solides élémentaires approchant au mieux l'angle solide de la facette. Pour ce faire, une projection sur la surface $\mathcal{S}$ en direction du point avec lequel le facteur de forme est recherché est effectuée.

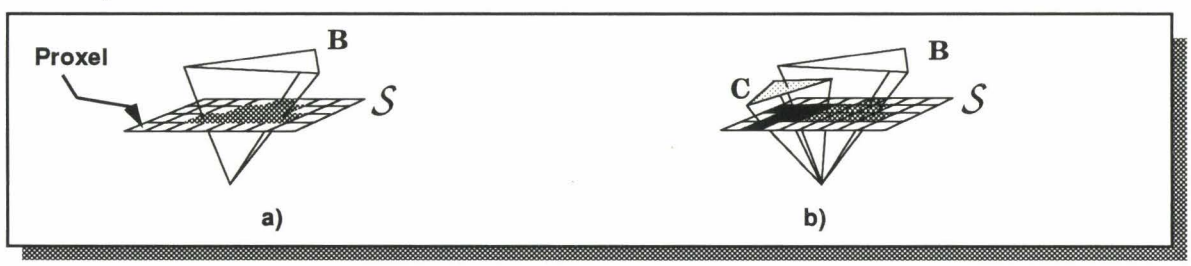


Figure 2.3 *Principe de la projection a) discrétisation de S b) tampon de profondeur*

L'ensemble des angles solides élémentaires recouverts par cette projection permet alors de déterminer le facteur de forme de la facette (voir Figure 2.3.a). Par analogie avec le terme pixel (Picture Element), nous appellons proxel (Projected Element) chaque angle solide élémentaire de la surface de projection.

Le second point à considérer pour le calcul des facteurs de forme est la prise en compte des occultations entre facettes. Les approches projectives résolvent simplement ce problème en utilisant l'algorithme du tampon de profondeur [Catmu 75]. Lorsqu'une facette est projetée en un proxel, son "identité" est mémorisée, de même que la distance à laquelle elle se trouve du centre de projection. Si une autre facette se projette en ce proxel, une comparaison de distance est effectuée de manière à ne conserver que la facette la plus proche (Figure 2.3.b)

Lorsque toutes les facettes ont été projetées, le facteur de forme de chaque facette est déterminé en effectuant la somme des facteurs de forme élémentaires des proxels recouverts par leur projection.

2.3 Approximation de l'hémisphère

L'énergie émise par une facette est répartie dans l'hémisphère des directions situé au dessus de la facette. La surface de projection idéale est donc une demi-sphère. Cependant, effectuer des projections sur une sphère et y implanter un algorithme d'élimination des parties cachées n'est pas une chose aisée. Différentes approches ont donc été proposées pour approximer la surface hémisphérique par un ou plusieurs plans.

2.3.1 L'hémicube

L'hémicube, proposé par Cohen en 1985 [Cohen 85], approxime l'hémisphère par cinq plans de projection formant un demi-cube. Cette approximation permet de se ramener à des algorithmes de projection planaires, plus simples à aborder. Chaque face de l'hémicube est découpée régulièrement en proxels carrés de même taille (Figure 2.4). L'hémicube est appliqué sur la facette depuis laquelle les facteurs de forme avec le reste de la scène sont à déterminer.

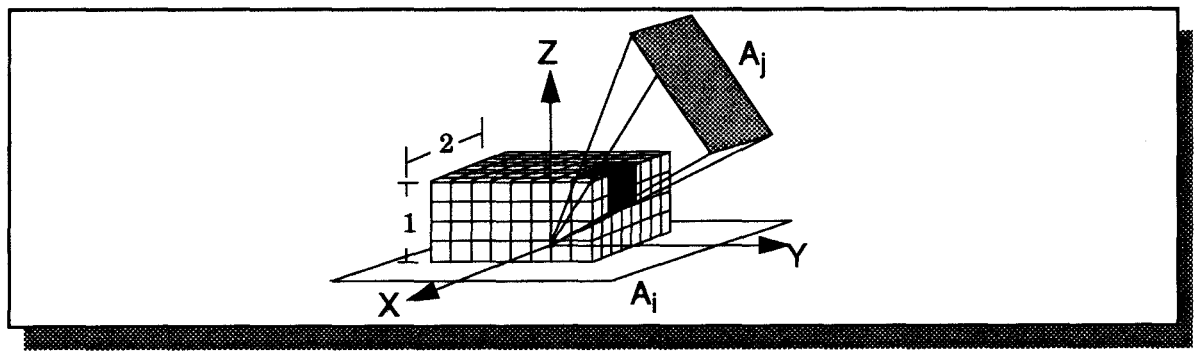


Figure 2.4 *L'hémicube*

Toutes les facettes de la scène sont projetées successivement sur les cinq faces de l'hémicube, les proxels recouverts par une facette projetée étant déterminés à l'aide d'algorithmes classiques de remplissage de polygônes (suivi de contours, godet ...).

Chaque proxel représente un petit angle solide auquel est associé un facteur de forme élémentaire. La valeur de ces facteurs de forme élémentaires est déterminée une fois pour toutes au début de l'algorithme de radiosité, puisque elle ne dépend pas de la facette d'application, mais de la taille des proxels. Ces valeurs sont données, selon le plan de projection considéré, par les formules suivantes:

- $\Delta F_p = \frac{1}{\Pi(x^2+y^2+1)^2}\Delta S$ pour la face supérieure

- $\Delta F_p = \frac{z}{\Pi(y^2+z^2+1)^2}\Delta S$ pour les faces latérales

avec

- ΔF_p facteur de forme élémentaire d'un proxel p,
- x, y, z, coordonnées du centre du proxel,
- ΔS la surface du proxel

Notons qu'il n'est pas nécessaire de mémoriser la valeur du facteur de forme élémentaire de chaque proxel. En effet, de par leur expression, il existe des symétries pour chacune des faces de l'hémicube. Il suffit donc de calculer 1/8 des facteurs de forme élémentaires de la face supérieure et 1/4 de ceux d'une des faces latérales pour avoir l'ensemble des valeurs possibles

sur l'hémicube complet. Ces valeurs seront mémorisées à part, un proxel n'étant alors constitué que d'une valeur de distance et d'une valeur d'identité, permettant de mémoriser la facette la plus proche visible en ce proxel.

Nous avons représenté sur la figure ci-dessous (Figure 2.5) l'évolution de la valeur des proxels selon leur emplacement sur l'hémicube.

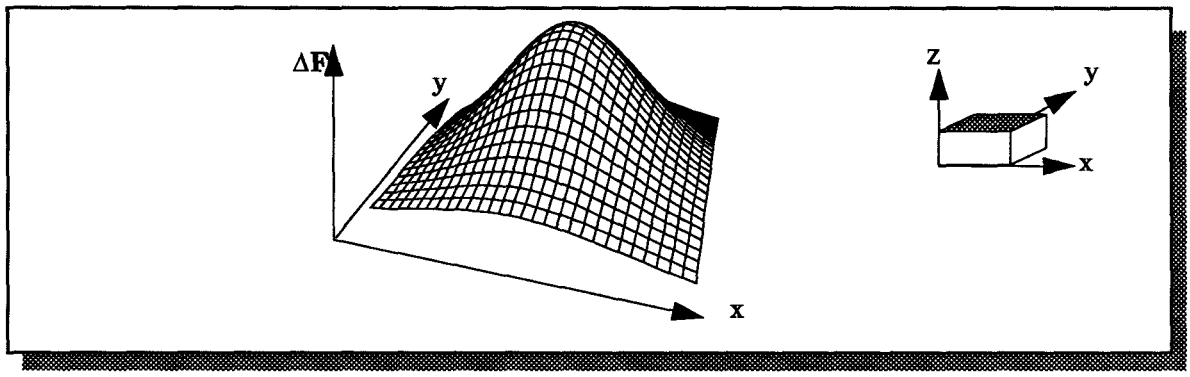


Figure 2.5 ***Evolution de la valeur des facteurs de forme pour la face supérieure de l'hémicube***

Il apparait clairement que les valeurs des facteurs de forme élémentaires sont très différentes selon l'emplacement des proxels auxquels ils sont rattachés. La valeur des facteurs de forme élémentaires est beaucoup plus importante au centre de la face supérieure de l'hémicube que pour les régions périphériques. La précision de l'échantillonnage des facteurs de forme par cette méthode déprend donc de la valeur des facteurs de forme élémentaires au centre de la face supérieure de l'hémicube. De ce fait, l'hémicube génère un sur-échantillonnage important des directions de projection qui s'écartent de la normale à la facette. Ce sur-échantillonnage est d'autant moins souhaitable que le nombre de proxels intervient directement sur les temps de calcul d'un hémicube.

2.3.2 Les plans de projection unique

L'hémicube approxime la surface de projection sphérique par cinq plans. Il est donc nécessaire d'effectuer cinq mises en perspectives successives de la scène pour déterminer les facteurs de forme entre celle-ci et la facette sur laquelle l'hémicube est appliqué. Afin de limiter le nombre de projections à effectuer, Sillion [SillA 89], [SillB 89], puis Recker [Recke 90] ont proposé une approximation de l'hémicube par un plan unique, parallèle au plan de la facette d'application.

Dans la mesure où il n'est pas possible d'utiliser un plan infini, la zone de projection se limite à un carré suffisamment grand pour approximer au mieux l'hémisphère des directions supérieures à la facette. Ceci génère néanmoins des pertes lors du calcul des facteurs de forme, puisque les directions rasantes à la facette sont ignorées. La relation suivante permet d'estimer les valeurs respectives de la taille du carré de projection et de sa distance par rapport à la facette, en fonction de l'erreur tolérée [SillA 89] :

$$\frac{D}{H} \approx \sqrt{\frac{2}{E}}$$

Dans cette expression, H représente la distance du plan de projection au plan de la facette, D la longueur du demi côté du carré de projection, et E l'erreur commise. A titre d'exemple, une tolérance de 1% nécessite un carré de côté 28 placé à une hauteur 1 du plan de la facette.

Le plan de projection unique de Sillion

La taille du carré de projection est beaucoup plus importante que celle de l'hémicube. Un découpage uniforme générerait donc un grand nombre de proxels, et par conséquent un suréchantillonnage très important. Sillion propose donc un découpage original du plan de projection, ayant pour but l'obtention de facteurs de forme égaux pour tous les proxels. Ceci est difficile à obtenir avec un découpage en proxels carrés, répartis régulièrement. Le découpage

adopté a donc pour objectif de générer des facteurs de forme élémentaires ayant la même valeur selon les axes principaux du plan. La forme du motif d'échantillonnage est obtenue, pour l'axe x, à l'aide de l'expression suivante, le motif étant symétrique pour l'axe y :

$$x_i = \frac{i}{N}\frac{\xi_N}{\sqrt{1-\frac{i^2}{N^2}\xi_N^2}} \qquad \text{avec} \qquad \xi_N = \sqrt{\frac{D^2}{1+D^2}}$$

Les x_i représentent les axes de découpe, N le nombre de proxels à générer, et D la longueur du demi côté du carré de projection. L'aspect de ce découpage, ainsi que la répartition des valeurs des facteurs de forme élémentaires, sont représentés sur la Figure 2.6.

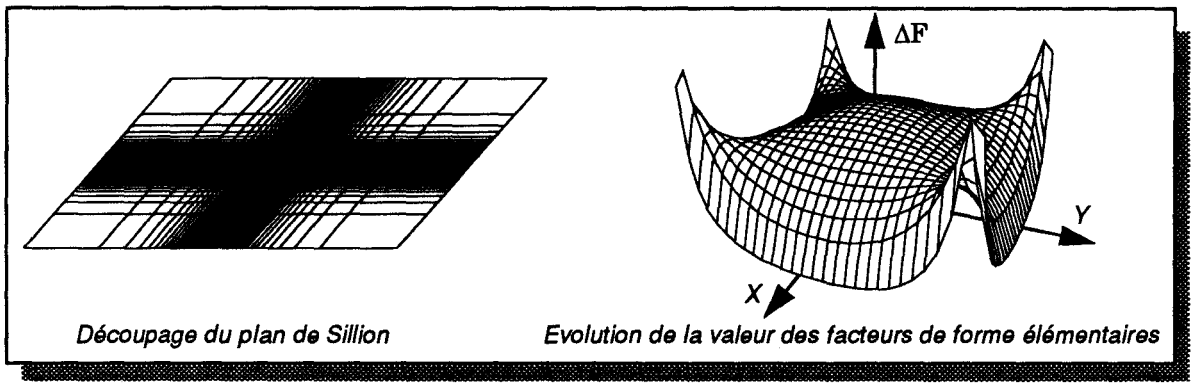


Figure 2.6 ***Plan de Sillion***

L'élimination des parties cachées est effectuée en utilisant l'algorithme de Warnock [Warno 69], afin d'exploiter la cohérence de l'image projetée.

Cet algorithme cherche à déterminer si, pour une fenêtre rectangulaire contenant un certain nombre de proxels, il est possible de se ramener à l'un des deux cas simples suivants: soit la projection de la facette recouvre entièrement la fenêtre, soit elle n'est pas présente dans cette fenêtre. Dans le premier cas, une zone complète est recouverte par la projection, sans avoir à analyser tous les proxels qui s'y trouvent. Dans le second cas, il n'est pas non plus nécessaire de poursuivre l'analyse de cette fenêtre, puisque aucun proxel ne peut être recouvert par la projection. Dans les autres cas, la fenêtre est subdivisée en quatres nouvelles fenêtres de même taille, et l'analyse est reprise dans chacune de celles-ci.

Cet algorithme est adapté de manière à prendre en compte la taille non uniforme des proxels, la subdivision d'une fenêtre s'effectuant de manière à laisser autant de proxels dans chaque sous-fenêtre.

D'autre part, afin d'éviter la projection de l'ensemble des facettes, celles-ci sont triées par rapport à leur distance au centre de projection. Lorsque la projection dans une fenêtre donnée doit être effectuée, la liste des facettes est parcourue par ordre croissant de distance. Toutes les facettes se projetant dans la fenêtre considérée sont marquées, de manière à générer une nouvelle liste. Lors de la projection, c'est cette liste qui sera considérée pour cette fenêtre et toutes ses sous-fenêtres éventuelles, sur lesquelles le même procédé est appliqué. L'avantage d'une telle approche est qu'elle permet d'éliminer la projection d'un certain nombre de facettes. En effet, lorsqu'une facette recouvre entièrement une fenêtre, toutes celles qui se trouvent derrière elle peuvent être éliminées, puisqu'elles ne pourront pas être visibles.

Enfin, cette approche diffère aussi de l'hémicube par son mode de calcul des facteurs de forme. En utilisant le tri décrit ci-dessus, une facette recouvrant une fenêtre reçoit obligatoirement le facteur de forme correspondant à la somme des facteurs de forme élémentaires des proxels contenus dans cette fenêtre, puisque aucune autre facette ne pourra y être visible. Pour tirer partie de cette propriété, les facteurs de forme élémentaires ne sont pas mémorisés comme

représentant la contribution de chaque proxel de coordonnées (i, j) , mais comme la somme des contributions de l'ensemble des proxels (x, y) vérifiants $x \leq i, y \leq j$ (voir Figure 2.7 a)

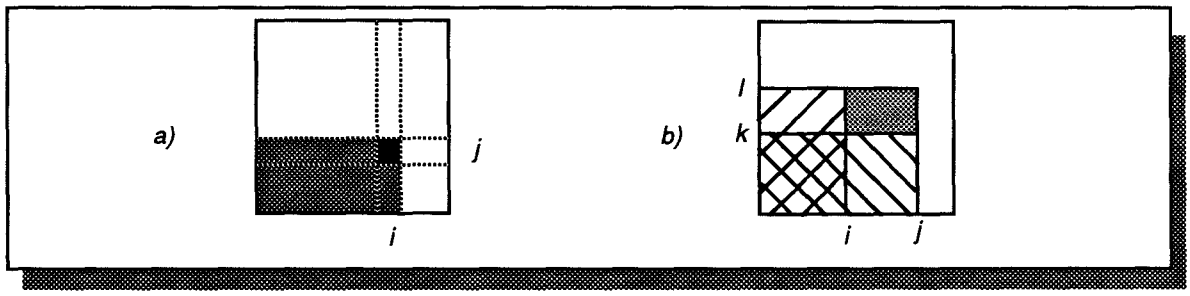


Figure 2.7 ***Représentation des facteurs de forme élémentaires (a) et calcul de la contribution d'une fenêtre quelconque (b)***

De cette manière, la contribution d'une fenêtre rectangulaire quelconque peut être calculée rapidement, en n'utilisant qu'une addition et deux soustractions. Si les contributions décrites ci-dessus sont mémorisées dans un tableau C, le facteur de forme d'une fenêtre délimitée par les sommets (x_i, y_k) et (x_j, y_l) est donné par (voir Figure 2.7.b) :

$$\Delta F_{x_i, x_j, y_k, y_l} = C[j, l] - C[j, k] - C[i, l] + C[i, k]$$

Ce procédé permet ainsi d'éviter une sommation sur l'ensemble des proxels de la fenêtre.

L'hémicube modifié

Recker propose lui aussi une approche n'utilisant qu'un seul plan de projection [Recke 90]. Son algorithme se différencie du précédent d'abord par le découpage adopté. Le but est d'obtenir une précision suffisante au centre du plan de projection, là où les facteurs de forme élémentaires possèdent les valeurs les plus élevées, tout en limitant le nombre total de proxels. Le découpage possède deux niveaux (voir Figure 2.8.a) : la zone centrale du carré de projection est découpée finement, tandis que la zone périphérique conserve des proxels de grande taille.

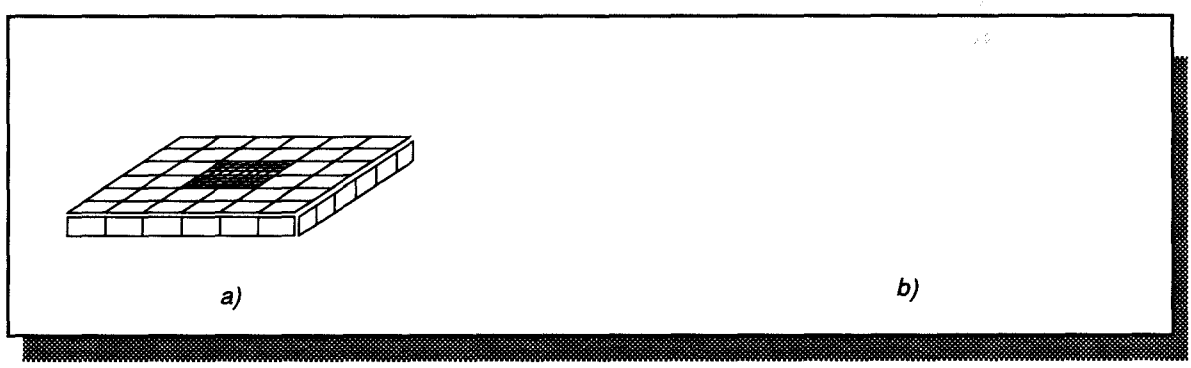


Figure 2.8 ***Plans de Recker a) Découpage b) Evolution des facteurs de forme élémentaires***

Le calcul des facteurs de forme se déroule en deux temps. Les facettes sont d'abord projetées sur la partie du plan possédant le découpage le plus large. Toutes les opérations d'élimination des parties cachées y sont appliquées, puis le calcul des facteurs de forme est effectué en ignorant la zone centrale. Les facettes sont ensuite à nouveau projetées, mais sur la zone centrale cette fois. Les mêmes opérations d'élimination des parties cachées sont appliquées, et les facteurs de forme calculés sur cette zone sont ajoutés à ceux obtenus lors de la première étape. Il est à noter que le découpage adopté, quelle que soit la zone considérée, est un découpage uniforme, à l'image de celui de l'hémicube. La valeur des facteurs de forme suit donc une évolution similaire à celle de l'hémicube pour chacune des deux zones (Figure 2.8.b).

Nous avons vu que l'utilisation d'un plan de projection unique pour l'évaluation des facteurs de forme génère une perte d'énergie, puisqu'il est impossible d'approximer l'hémisphère de projection par un plan fini. Recker note que, même en limitant ces pertes par l'accroissement de

la taille du carré de projection, leur effet reste perceptible sur les images obtenues, par comparaison avec celles fournies par l'algorithme de l'hémicube, où aucune perte n'est générée. D'autre part, l'accroissement de la taille du carré de projection augmente fortement les temps de calculs.

Afin de conserver l'intérêt du plan unique tout en réduisant les pertes au niveau des directions rasantes, Recker propose d'utiliser "occasionnellement" quatre petites parties de plans selon le même principe que les faces latérales de l'hémicube (voir Figure 2.8.a). A chaque application du plan de projection, l'énergie non diffusée par les directions rasantes est mémorisée au sein de la facette. Lorsque cette énergie dépasse un certain seuil, les faces latérales sont utilisées en plus du plan supérieur pour déterminer la répartition correcte de l'énergie en récupérant l'énergie précédemment "oubliée".

2.4 Utilisation de l'hémisphère

Les algorithmes présentés ci-dessus approximent la surface de projection sphèrique par un ou plusieurs plans. Bien que les projections planaires soient bien maitrisées, elles réclament beaucoup d'opérations géométriques et posent le problème de l'obtention d'un échantillonnage uniforme. Des travaux ont donc été menés dans le but d'utiliser l'hémisphère comme surface de projection. Ceux-ci ont débouchés sur deux types de solutions, selon que l'hémisphère ou le disque support de l'hémisphère est utilisé. Nous allons tout d'abord décrire une approche utilisant l'hémisphère proprement dit, puis nous détaillerons une solution qui utilise le disque support de l'hémisphère comme surface de projection.

2.4.1 L'hémisphère

L'idée est ici d'utiliser la demi-sphère des directions situées au dessus d'une facette, comme surface de projection [Spenc 90]. Une demi-sphère unité est construite et appliquée sur la facette considérée. Les facettes de l'environnement sont ensuite projetées radialement sur cette surface, chaque arête de la facette de départ formant une géodésique (arc d'un grand cercle) sur la demi-sphère (Figure 2.9.a).

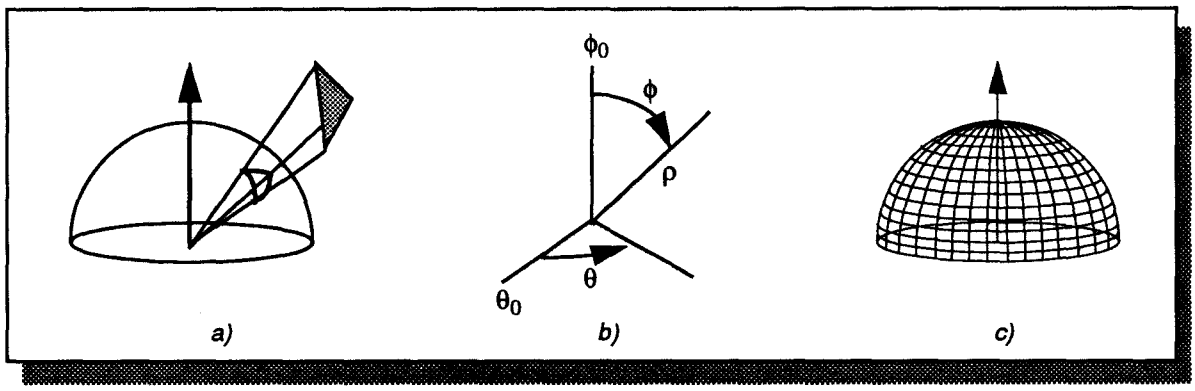


Figure 2.9 ***Utilisation de l'hémisphère a) Projection radiale b) Coordonnées sphériques c) Découpage à θ et ϕ constants***

Découpage uniforme

Dans la mesure où le système de coordonnées cartésiennes est assez mal adapté à ce genre de projection, Spencer propose d'utiliser le système de coordonnées sphèriques, défini par les angles θ et ϕ (longitude et latitude) et la distance à l'origine ρ (Figure 2.9 b). Il en déduit un découpage régulier en proxels, en divisant l'hémisphère par pas de $\Delta\theta$ et $\Delta\phi$ constants (voir Figure 2.9.c). Comme pour l'hémicube, chaque proxel ainsi généré représente une direction d'échantillonnage et contient l'identité de la facette qui s'y projette ainsi que sa distance à l'origine, afin d'appliquer l'algorithme du tampon de profondeur.

De manière à simplifier le processus de projection, seules des facettes triangulaires sont considérées. Les sommets de la facettes sont d'abord projetés sur l'hémisphère. La sphère étant unitaire, cette projection de réduit à une normalisation des coordonnées de ces points. Ces projections sont ensuite transformées dans le système de coordonnées sphériques et triées par ordre croissant en θ et ϕ. Après découpage par le plan support de la demi-sphère, l'élément de surface projeté est classé selon sa position sur la surface :

1. l'élément couvre le sommet de l'hémisphère ($\phi = 0$, θ quelconque);
2. l'élément couvre la ligne de référence $\theta = 0$ (fin de la zone 2D représentant le tampon de profondeur);
3. l'élément est une projection ordinaire, c'est à dire qu'il ne se trouve pas dans l'un des deux cas précédents.

Le "rectangle" englobant l'élément de surface projeté est alors calculé, en fonction de son classement dans l'une des trois catégories ci-dessus. Tous les proxels contenus dans cette zone sont ensuite parcourus, de manière à déterminer s'ils se trouvent à l'intérieur de l'élément de surface projeté. Ceci est fait en utilisant le principe du tracé de rayons: le point d'intersection entre un rayon issu du centre de l'hémisphère et passant par le centre du proxel, et le plan support de la facette est calculé, et une vérification est effectuée pour déterminer si ce point se trouve à l'intérieur du triangle définissant la facette. Si c'est le cas, la distance du point d'intersection au centre de l'hémisphère est calculée et une comparaison de distance est effectuée avec la facette déjà présente dans ce proxel.

L'hémisphère n'utilise qu'une seule surface de projection pour l'échantillonnage de l'ensemble des directions situées au dessus de la facette d'application. Ceci simplifie les transformations géométriques nécessaires à la projection. Néanmoins, un découpage régulier ne fournit pas des facteurs de forme égaux pour chaque proxel (Figure 2.10).

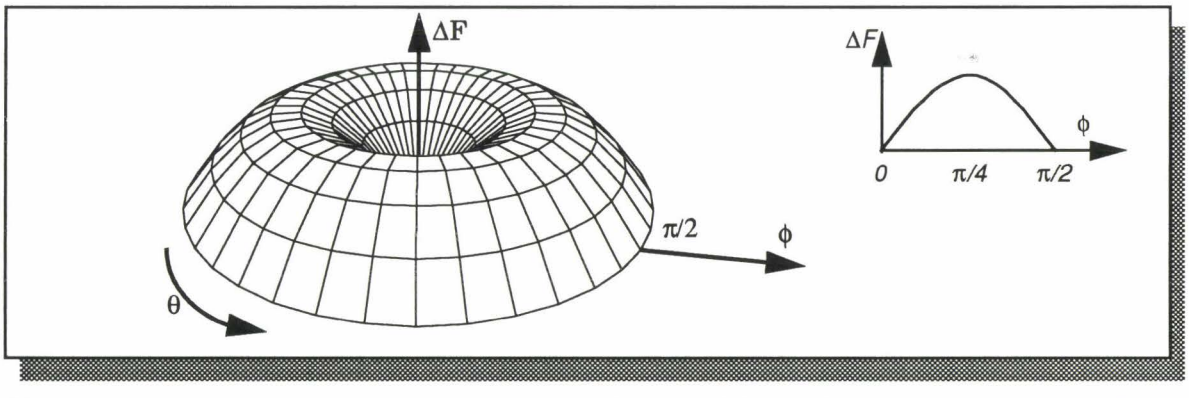


Figure 2.10 ***Répartition de la valeur des facteurs de forme élémentaires pour un découpage régulier de l'hémisphère***

En effet, si les facteurs de forme élémentaires sont égaux pour tous les proxels situés à la même latitude (ϕ=constante), ils varient lorsque celle-ci change. Ceci génère un sur-échantillonnage, qui reste cependant limité par rapport aux approches précédentes, comme nous le verrons au paragraphe 2.5. La valeur des facteurs de forme élémentaires est obtenue à l'aide de l'expression suivante :

$$\Delta F = \frac{2}{M} \sin\left(\frac{\phi 1 + \phi 2}{2}\right)(\sin\phi 2 - \sin\phi 1)$$

où M représente le nombre de bandes circulaires, et $\phi 1$ et $\phi 2$ les valeurs des angles englobants la bande circulaire où se trouve le proxel considéré.

Découpage non uniforme

Il est possible d'obtenir un découpage en proxels de même valeur de facteurs de forme élémentaires sur la surface sphérique. Il suffit en effet, en supposant le découpage constant selon l'angle θ, que les bandes sphèriques (découpage selon ϕ) apportent la même contribution au facteur de forme élémentaire. Afin de déterminer la façon d'effectuer ce découpage, nous allons utiliser le principe de l'équivalent de Nusselt (paragraphe 2.2.1). Chaque bande sphérique recherchée se projette sur le disque support de l'hémisphère sous forme d'un anneau. Selon l'équivalent de Nusselt, pour que chaque bande sphérique corresponde au même facteur de forme, il suffit que chaque anneau projeté ait la même surface (Figure 2.11 a).

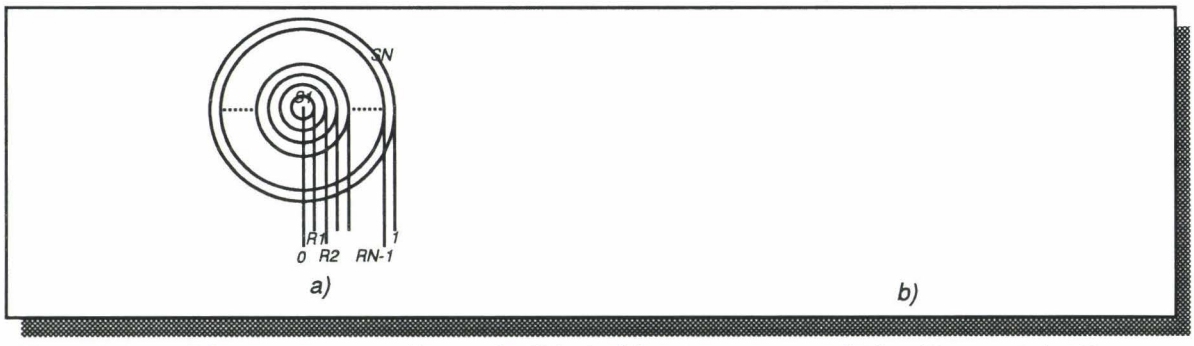


Figure 2.11 ***Découpage non uniforme de l'hémisphèrea) anneaux projetés b) aspect du découpage***

En notant N le nombre d'anneaux, A_i l'anneau d'indice i, S_{Ai} la surface A_i, et R_i le rayon extérieur de l'anneau i (voir Figure 2.11 a), nous avons les relations suivantes:

$$S_1 = \Pi R_1^2 = \frac{\Pi}{N} \Rightarrow R_1 = \frac{1}{\sqrt{N}}$$

$$S_2 = \Pi R_2^2 - S_1 = \Pi (R_2^2 - R_1^2) = \frac{\Pi}{N} \Rightarrow R_2 = \sqrt{\frac{2}{N}}$$

De manière générale, $R_i = \sqrt{\frac{i}{N}}$

Les différentes valeurs ϕ_i du découpage en ϕ peuvent donc être obtenues par la relation:

$$\phi_i = \text{acos}\,(R_i)$$

L'aspect du découpage obtenu apparaît sur la Figure 2.11 b . Nous montrons, au paragraphe 2.5, le gain obtenu, en terme de nombre de proxels générés, avec un tel découpage. Si N est le nombre de proxels sur la demi-sphère, alors la valeur du facteur de forme élémentaire est donnée par l'expression $\Delta F = \pi/N$. L'élimination des parties cachées sur un tel découpage peut être effectuée par l'algorithme de Spencer.

2.4.2 Le disque de projection

Une manière simple d'obtenir des proxels possédant la même valeur de facteur de forme élémentaire est d'utiliser le disque support de l'hémisphère comme surface de projection. En effet, selon l'équivalent de Nusselt (paragraphe 2.2.1), le facteur de forme entre une facette et l'élément de surface situé au centre du disque est proportionnel à la surface occupée par la projection de cette facette sur le disque. Un découpage en proxels de même taille de ce disque de projection assure une contribution identique de chaque proxel au facteur de forme. Le découpage proposé par Spencer et Goldfeather [Goldf 89], que nous utiliserons par la suite, est un découpage régulier en proxels carrés, selon les axes principaux du disque (voir Figure 2.12.a). Par mesure de simplicité, ce découpage est en fait généré sur le carré englobant le disque, mais seuls les proxels situés à l'intérieur du disque sont considérés pour les calculs.

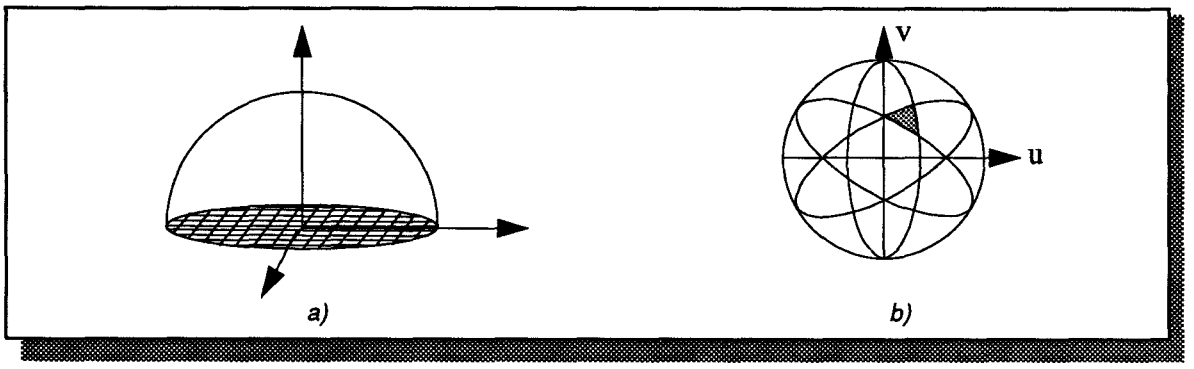


Figure 2.12 ***Disque de projection a) découpage du disque b) contours de projection elliptiques***

Les contours de la facette projetée sur le disque sont plus délicats à utiliser. En effet, la première étape de projection (projection radiale sur la surface sphérique) génère des contours dont chaque partie est un arc de grand cercle (géodésique). Ces grands cercles se projetent orthogonalement sur le disque sous forme d'ellipses dont le centre est le centre de la sphère. Le contour projeté sur le disque est défini par un ensemble d'arcs elliptiques (voir Figure 2.12.b). Nous allons détailler les spécificités de cette nouvelle approche.

Calcul des projections dans l'espace objet

Pour déterminer les proxels recouverts par le contour projeté, Spencer propose de travailler sur les arêtes de la facette plutôt que sur les arcs elliptiques projetés. Chaque arête est découpée en un certain nombre de points, en fonction de la mesure de l'angle formé par les deux sommets de l'arête. Les deux étapes de projection sont appliquées à chacun de ces points, fournissant la liste de proxels appartenant au disque de projection. La liste globale fournie par la projection de chacune des arêtes de la facette représente le contour de la projection. L'ensemble des proxels recouverts est obtenu à l'aide d'un algorithme de remplissage standard. Notons que pour cette approche, Spencer simplifie les calculs de distance nécessaires à la gestion du tampon de profondeur, en supposant tous les points de la facette équidistants du centre du disque.

Calcul des projections dans l'espace image

Nous nous proposons ici de décrire une approche travaillant directement dans l'espace image, c'est à dire utilisant explicitement la représentation sous forme d'arcs elliptiques pour déterminer les proxels appartenant au contour projeté. Il est nécessaire de connaître l'équation des ellipses supportant chaque arc, exprimée dans le repère associé au disque de projection.

Pour ce faire, nous allons utiliser un changement de repère introduit par Goldfeather dans le cadre de l'implantation sur la machine Pixel-Planes 5 [Fuchs 89] d'un algorithme de radiosité par projection hémisphérique.

Les coordonnées des arêtes définies dans le repère global (x,y,z) sont transformées dans le repère (u,v,ρ) sous la forme:

$$\begin{cases} \rho = \sqrt{x^2+y^2+z^2} \\ u = \dfrac{x}{\rho} \\ v = \dfrac{y}{\rho} \end{cases} \Leftrightarrow \begin{cases} x = \rho u \\ y = \rho v \\ z = \rho\sqrt{1-u^2-v^2} \end{cases}$$

Dans ce changement de repère, u et v correspondent aux coordonnées d'un point projeté sur le disque, tandis que ρ correspond à la distance de ce point au centre du disque.

Chaque arête de la facette définit un plan passant par les deux sommets de l'arête et le centre du disque. L'intersection de ce plan et de l'hémisphère se fait selon un grand cercle, dont la projection est l'ellipse recherchée. L'équation du plan est de la forme:

$$\Pi: \quad Ax + By + Cz = 0$$

En remplaçant x, y et z par leur expression en fonction de u, v et ρ, l'équation de l'ellipse représentant la projection orthogonale sur le disque du grand cercle précédent, revêt la forme suivante :

$$E(u,v): A\rho u + B\rho v + C\rho\sqrt{(1-u^2-v^2)} = 0 \qquad \textbf{(Eq. 2.4)}$$

Dans la mesure où ρ peut être considéré comme non nul, l'équation se simplifie en:

$$E(u,v): \quad Au + Bv + C\sqrt{(1-u^2-v^2)} = 0 \qquad \textbf{(Eq. 2.5)}$$

Cette équation ne décrit pas l'ellipse complète, mais seulement une moitié de celle-ci (voir Figure 2.13 a). C'est sur cette partie que se trouve l'arc représentant la projection de l'arête de départ. L'équation de l'ellipse complète est obtenue en élevant l'équation (Eq. 2.5) au carré. Dans ce qui suit, nous noterons cette demi-ellipse la Moitié Positive de l'Ellipse (MPE), car l'arête qui s'y projette a une coordonnée z positive dans le repère (x, y, z).

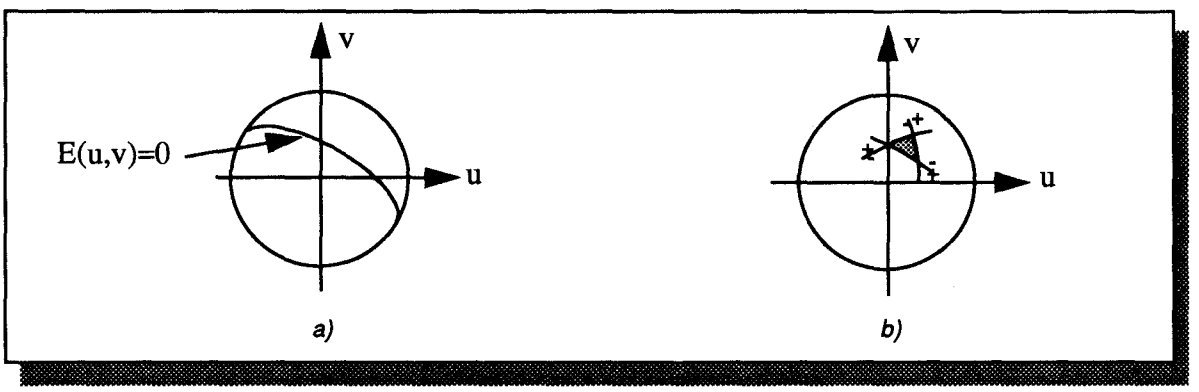


Figure 2.13 ***La MPE a) Découpe du disque par la MPE b) Intersection de plusieurs MPE***

La MPE divise le disque de projection en deux parties différentes, selon que le signe de E(u,v) est positif ou négatif. L'une de ces régions correspond à l'ensemble des proxels sont à l'intérieur de la projection de la facette par rapport à l'une de ses arêtes, tandis que l'autre représente les proxels situés à l'extérieur. Les proxels situés à l'intérieur de la projection sont ceux appartenant à l'intersection des différentes régions intérieures définies par chacune des arêtes. Il reste à présent à préciser comment distinguer la région intérieure de la région extérieure. Rappelons que, si le contour des facettes est orienté, l'évaluation des équations des droites correspondant à la projection d'une facette sur un plan, fournit des valeurs de même signe pour les points à l'intérieur de la projection, alors qu'à l'extérieur, au moins un des signes diffère. Cette propriété reste vraie dans le cas des MPE qui nous intéressent: un proxel (u0,v0) sera à l'intérieur de la projection sur le disque si l'évaluation des équations correspondants à chaque MPE fournit un résultat de même signe. Ce signe dépend de l'orientation fixée pour le contour des facettes lors de la création de la base de données contenant les facettes (cf Figure 2.13.b).

Une première approche consiste donc à déterminer, pour chaque proxel du plan de projection, son appartenance éventuelle à un contour projeté, en évaluant les différentes MPE. Cependant, la projection d'une facette ne recouvre qu'un faible nombre de proxels, par rapport à l'ensemble des proxels du disque. Il est donc inutile de les parcourir tous, et il est beaucoup plus avantageux de se limiter à la zone réduite au rectangle englobant la projection.

Détermination du rectangle englobant

Le calcul du rectangle englobant une ellipse complète peut être effectué en utilisant les propriétés des tangentes. En effet, le point le plus haut et le point le plus bas de l'ellipse

possèdent la caractéristique d'avoir une tangente horizontale. Déterminer ces points correspond à trouver les deux valeurs de ligne V_{min} et V_{max} telles que l'intersection de l'ellipse avec ces deux lignes ne fournit qu'un seul point d'intersection. Ces deux valeurs sont :

$$V_{max} = \sqrt{\frac{A^2 + C^2}{A^2 + B^2 + C^2}} \qquad V_{min} = -\sqrt{\frac{A^2 + C^2}{A^2 + B^2 + C^2}}$$

où A, B et C représentent les coefficients de l'ellipse définis dans l'équation (Eq. 2.5). Les valeurs correspondantes pour u sont obtenues en remplaçant ces valeurs dans l'équation de l'ellipse, ce qui permet d'obtenir les deux points P_{min} et P_{max} pour lesquels la tangente est horizontale (Figure 2.14 a):

$$P_{max}\left(\frac{-ABV_{max}}{A^2 + C^2}, V_{max}\right) \qquad P_{min}\left(\frac{-ABV_{min}}{A^2 + C^2}, V_{min}\right)$$

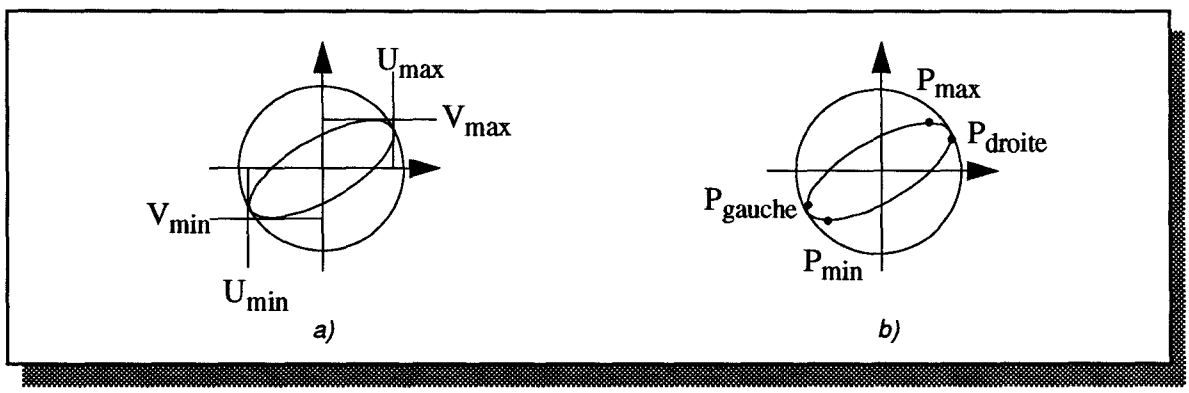


Figure 2.14 ***Rectangle englobant une ellipse a) Tangentes horizontales et verticales b) Extrema***

Le même procédé est utilisé pour obtenir les valeurs U_{max} et U_{min}, qui représentent cette fois les tangentes verticales, et en déduire les points P_{gauche} et P_{droite} correspondants.

$$U_{max} = \sqrt{\frac{B^2 + C^2}{A^2 + B^2 + C^2}} \qquad U_{min} = -\sqrt{\frac{B^2 + C^2}{A^2 + B^2 + C^2}}$$

$$P_{gauche}\left(\frac{-ABU_{max}}{B^2 + C^2}, U_{max}\right) \qquad P_{droite}\left(\frac{-ABU_{min}}{B^2 + C^2}, U_{min}\right)$$

Ces quatres valeurs U_{min}, U_{max}, V_{min} et V_{max} fournissent le rectangle englobant l'ellipse complète. Les arêtes projetées ne nous fournissent cependant qu'un arc d'ellipse, et utiliser le rectangle englobant la totalité de l'ellipse serait totalement inefficace. Nous allons donc calculer le rectangle englobant uniquement un arc, le plus petit rectangle englobant la projection de la facette sur le disque étant obtenu par l'union des différents rectangles englobants chaque arc de la projection.

Le rectangle englobant un segment de droite peut être trouvé simplement en déterminant la valeur des extrema. Ces extrema correspondent aux coordonnées des extrémités du segment. Dans le cas d'un arc d'ellipse, par contre, ceci n'est plus vrai, puisque un arc possède une concavité qui peut générer de nouveaux extrema qui ne correspondent plus forcément aux coordonnées des extrémités de l'arc. Nous allons décrire les deux cas qui peuvent alors se présenter, en ne raisonnant que sur la détermination de la valeur de V fournissant le haut du

rectangle englobant l'arc. Le raisonnement est identique pour les trois autres valeurs recherchées.

Dans le cas d'un arc d'ellipse, l'extremum en V est soit la plus grande des deux valeurs de la coordonnéeV de chacun des deux points d'extrémité (cas de l'arc 1, Figure 2.15.a), soit la valeur V_{max} correspondant à la tangente horizontale supérieure de l'ellipse (cas de l'arc 2, Figure 2.15.b). Pour déterminer de quel cas il s'agit, il est nécessaire de calculer le point P_{max} défini ci-dessus, puis de tester sa présence à l'intérieur de l'intervalle défini par les abscisses des deux points d'extrémité de l'arc. Si P_{max} est à l'intérieur de cet intervalle, alors V_{max} est la valeur recherchée. Dans le cas contraire, cette valeur correspond à la plus grande des deux ordonnées des points d'extrémité.

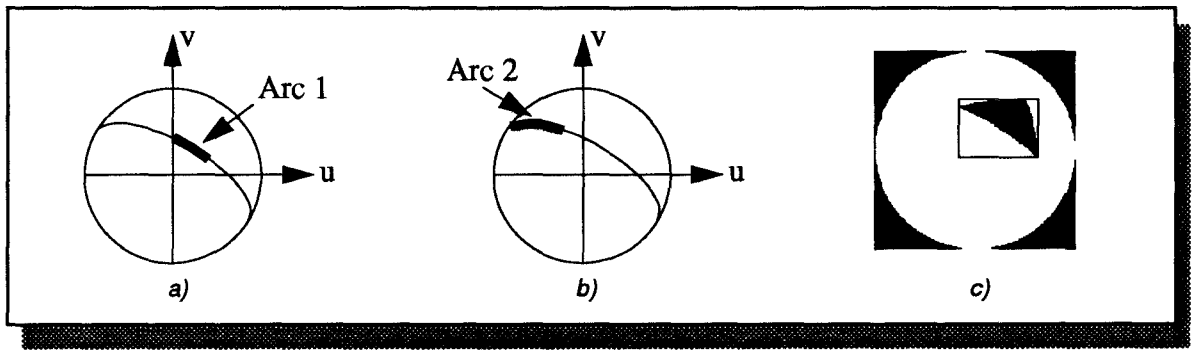


Figure 2.15 ***Rectangle englobant un arc d'ellipse: présence (a) ou absence (b) d'un extremum local; rectangle obtenu (c)***

L'algorithme de projection d'une facette sur le disque se ramène à l'algorithme suivant:

```
Evaluation de l'équations de chaque MPE
Evaluation du rectangle englobant
Pour chaque proxel du rectangle
|    Evaluer les MPE
|    Si tous les signes sont identiques
|    |    Calcul de la distance entre le disque et la facette
|    |    Application du tampon de profondeur
|    FinSI
FinPour
```

De manière à limiter les calculs à effectuer, la racine carrée de l'équation (Eq. 2.5) est précalculée pour chaque proxel du disque de projection. Elle ne dépend en effet que de la position du proxel. De plus, du fait de l'élévation au carré de U et V, et du rôle symétrique qu'ils jouent, seules 1/8 de ces valeurs sont mémorisées.

Lors de l'application du tampon de profondeur, il est nécessaire de connaître la distance à laquelle se trouve le centre du disque du point projeté dans le proxel. Une simplification consiste à supposer cette distance constante pour tout point de la facette. Elle n'est cependant valable que dans le cas de facettes suffisamment petites et éloignées. Une approche plus rigoureuse consiste à calculer la vraie distance. La distance au centre du disque d'un point (x,y,z) quelconque s'exprime sous la forme :

$$\rho = \sqrt{x^2 + y^2 + z^2}$$

Ce point appartient au plan de la facette, dont l'équation est de la forme $A'x + B'y + C'z + D' = 0$. En utilisant le changement de coordonnées introduit précédemment, cette équation devient :

$$A'\rho u + B'\rho v + C'\rho\sqrt{1 - u^2 - v^2} + D' = 0$$

où u et v représentent les coordonnées du proxel pour lequel la distance doit être calculée. Cette distance s'exprime, à présent, uniquement en fonction de u, v et des coefficients A', B', C' et D' :

$$\rho = -\frac{D'}{A'u + B'v + C'\sqrt{1-u^2-v^2}}$$

Dans la mesure où une distance relative suffit, sous réserve qu'elle croisse lorsque ρ croît, il est possible d'utiliser l'expression simplifiée suivante:

$$\lambda = -\frac{1}{\rho} = \frac{A'}{D'}u + \frac{B'}{D'}v + \frac{C'}{D'}\sqrt{1-u^2-v^2} = A''u + B''v + C''\sqrt{1-u^2-v^2}$$

où la racine carrée est précalculée, et les coefficients A", B" et C" sont constants pour tout proxel recouvert par une facette donnée.

2.5 Comparaison des méthodes

Nous nous proposons, dans ce paragraphe, de comparer les différents algorithmes projectifs présentés dans ce chapitre. Différents critères seront introduits selon le type de comparaison envisagée.

2.5.1 Critères de comparaison

Différents critères sont envisageables pour la comparaison des algorithmes décrits dans les paragraphes précédents, selon que la rapidité des calculs ou la qualité des images est privilégiée.

Le temps de calcul des facteurs de forme à partir d'une facette donnée est important. Il intervient d'une part sur l'interactivité attendue de l'algorithme par raffinement progressif, et d'autre part, sur le temps de calcul global nécessaire à la convergence de la méthode. La complexité de l'étape de projection dépend à la fois du nombre de proxels et du type de surface de projection choisie. Plus le nombre de proxels sera grand, plus le temps nécessaire à la projection sera élevé, d'où l'intérêt d'éliminer autant que faire se peut le suréchantillonnage. Cependant, les surfaces et les algorithmes réduisant ce suréchantillonnage, nécessitent souvent des opérations plus complexes, rendant le choix d'un algorithme moins évident.

Evaluer les algorithmes uniquement sur leurs aspects quantitatifs ne suffit pas. Obtenir des images très rapidement est sans grand intérêt si elles sont de trop mauvaise qualité. Juger de la qualité d'une image est cependant difficile, dans la mesure où celle-ci dépend de nombreux paramètres intervenant à la fois dans l'espace objets (précision des échanges lumineux, qualité de la modélisation des objets et de la représentation de leurs propriétés, ...) et dans l'espace image (aliassage, contraintes technologiques liées aux écrans, ...).

Nous comparons ci-dessous divers algorithmes projectifs d'abord d'un point de vue quantitatif, puis d'un point de vue qualitatif, en précisant à chaque fois le critère choisi pour cette comparaison. Au préalable, nous allons établir un critère d'équivalence entre les algorithmes, qui servira de base dans les comparaison ultérieures.

2.5.2 Critère d'équivalence

La comparaison des différents algorithmes projectifs ne peut être effectuée que sur la base d'un critère commun à tous. Nous proposons d'utiliser la valeur des facteurs de forme élémentaires (FFE) de certains proxels comme critère de comparaison. La plupart des algorithmes étudiés travaillent avec des proxels dont la contribution au facteur de forme diffère selon leur emplacement sur la surface de projection. De ce fait, la précision d'un algorithme est déterminée par les proxels qui possèdent la plus grande valeur de FFE. C'est en ces proxels que l'erreur commise dans l'approximation d'un facteur de forme est la plus importante. La plus grande précision fournie par les autres proxels peut alors être considérée comme "inutile", dans la mesure où une erreur plus grande est commise en certains endroits. Nous proposons donc de

comparer les différents algorithmes en déterminant le nombre de proxels nécessaires pour obtenir la même valeur maximale de FFE pour chacun.

Emplacement des FFEs de plus grande valeur

Pour l'hémicube, la valeur des FFEs est déterminée pour les proxels situés au centre de la face supérieure, de même que pour le plan de Recker. De plus, nous supposerons, pour ce dernier, que la taille des deux plans et leur découpage sont tels que la valeur des FFEs du carré extérieur, situés sur le bord commun avec le carré intérieur, possèdent eux aussi la valeur maximale de facteur de forme élémentaire.

Pour l'hémisphère découpé de manière uniforme, ces valeurs sont calculées pour un angle ϕ égal à $\pi/4$, puisque c'est à cet endroit que les FFEs sont les plus importants. Le choix du proxel de référence pour le disque ou l'hémisphère possédant un découpage non uniforme n'est évidemment pas important, puisqu'ils apportent tous la même contribution au facteur de forme.

Le cas du plan de Sillion est plus délicat à traiter. Le découpage particulier utilisé implique en effet que la valeur des FFEs des proxels situés aux quatres coins du plan est beaucoup plus importante que partout ailleurs. L'application de notre critère nécessite donc d'effectuer la comparaison à partir de ces proxels. Ceci conduit néanmoins à générer un nombre beaucoup trop élevé de proxels. A titre d'exemple, le nombre de proxels nécessaires dans le cas d'une valeur de FFE maximal de $1.27\ 10^{-6}$ serait d'environ 120 millions, alors qu'il n'est que de 3 millions pour l'hémicube. Nous avons donc choisi d'effectuer la comparaison par rapport aux proxels situés au centre du plan de projection, ceux-ci possédant la plus forte valeur de FFE, abstraction faite des coins du plan.

Détermination des résolutions équivalentes

Les valeurs représentant la résolution pour chacun des algorithmes apparaissent dans le tableau (Table 2.1). Le terme résolution représente le facteur de découpage de la surface de projection. Une résolution R signifie que la face supérieure de l'hémicube est découpée en RxR proxels, et chacune des faces latérales en RxR/2 proxels. De même, le plan de Sillion est découpé en RxR proxels. Recker utilise deux découpages différents, selon l'endroit du plan où se trouvent les proxels. Nous avons fait figurer les deux résolutions dans le tableau, sous la forme: R1 / R2. La première résolution R1 représente le découpage du carré intérieur, tandis que R2 représente celui du carré extérieur. A noter que les petites faces latérales ne sont pas prises en compte dans cette comparaison.

Facteur de forme de référence	Hémicube	Plan de Sillion	Plan de Recker	Hémisphère (découpage uniforme)	Hémisphère (découpage non uniforme)	Disque
$1.27\ 10^{-4}$	100	98	180 / 67	56	45	100
$3.18\ 10^{-5}$	200	197	360 / 134	112	89	200
$7.96\ 10^{-6}$	400	393	720 / 267	223	178	400
$3.54\ 10^{-6}$	600	590	1080 / 401	334	266	600
$1.99\ 10^{-6}$	800	786	1440 / 535	445	355	800
$1.27\ 10^{-6}$	1000	983	1800 / 668	556	444	1000

Table 2.1 ***Résolutions équivalentes pour les différents algorithmes projectifs***

La résolution indiquée pour le disque représente le découpage du carré qui l'englobe. Une partie des proxels ainsi générés n'est donc pas utilisée. Dans le cas des hemisphères, la résolution R représente le découpage selon la latitude (angle ϕ). Un découpage de même valeur angulaire est appliqué selon la longitude. Le nombre total de proxels correspond alors à l'expression Rx4R.

Les résultats fournis par ce tableau seront utilisés pour toutes les comparaisons qui sont effectuées dans les prochains paragraphes.

2.5.3 Comparaison quantitative

Nous avons vu, lors de la description des différents algorithmes, que l'un des soucis majeurs est de limiter le nombre de proxels utilisés pour l'échantillonnage des facteurs de forme. Dès lors, il est intéressant de pouvoir quantifier les gains, ou les pertes, obtenus par chacun de ces algorithmes par rapport à un algorithme de référence, et par là même d'obtenir une première comparaison basée sur le nombre de proxels utilisés. Une seconde comparaison est ensuite effectuée, en terme de place mémoire totale utilisée par chaque approche pour la représentation des proxels. Enfin, les temps d'évaluation des facteurs de forme selon l'algorithme sont comparés.

Nombre de proxels utilisés

Nous avons représenté dans le tableau (Table 2.2) le nombre de proxels nécessaires pour chaque approche à résolution équivalente. Les résultats sont présentés à partir d'une résolution de base de l'hémicube, les résolutions équivalentes pouvant être obtenues dans la Table 2.1.

Résolution de référence (hémicube)	Hémicube	Plan de Sillion	Plan de Recker	Hémisphère (découpage uniforme)	Hémisphère (découpage non uniforme)	Disque
100	30000	9604	36291	12544	8100	10000
200	120000	38809	145158	59536	31684	40000
400	480000	154449	580628	198916	126736	160000
600	1080000	348100	1306411	446224	283024	360000
800	1920000	617796	2322508	792100	504100	640000
1000	3000000	966289	3628917	1236544	788544	1000000

Table 2.2 ***Nombres de proxels utilisés par les algorithmes projectifs***

Les résultats des comparaisons montrent clairement l'intérêt de travailler avec la surface hémisphèrique et le disque de projection (environ 70% de gain par rapport à l'hémicube). En utilisant un découpage non uniforme de l'hémisphère, tel qu'il a été introduit en page 34, le suréchantillonnage qui existe dans l'approche de Spencer est éliminé, fournissant un gain d'environ 36% par rapport au nombre total de proxels obtenus dans le cas d'un découpage uniforme.

Le disque de projection semble nécessiter un peu plus d'échantillons. En fait, une partie des proxels générés lors du découpage du carré englobant ne sont jamais utilisés, puisqu'ils se trouvent à l'extérieur du disque. Le pourcentage des proxels inutiles est d'environ 21%, ce qui correspond à la différence qui apparaît dans le tableau entre le disque et l'hémisphère utilisant un découpage non uniforme. Il est tout à fait normal que ces deux approches utilisent le même nombre de proxels, puisque chacune d'entre elles apporte la même contribution au facteur de

forme. Notons qu'il est tout à fait possible, lors de l'implantation d'un algorithme basé sur le disque, de ne générer que les proxels réellement utiles.

Les deux approches utilisant un carré de projection unique posent quelques problèmes. Il semble évident que le découpage proposé par Recker est très coûteux (environ 21% de proxels en plus que l'hémicube), alors que le découpage obtenu par Sillion apparaît attrayant. Il faut cependant rappeler que la comparaison que nous avons faite pour ce découpage n'utilise pas correctement le critère que nous avions introduit. Si la comparaison avait été effectuée sur les valeurs des facteurs de forme élémentaires des proxels situés aux coins du carré de projection, le nombre de proxels à utiliser aurait été beaucoup plus important.

Coût mémoire

Les coûts correspondant à la mémorisation des proxels pour chaque algorithme sont représentés dans le tableau ci-dessous (Table 2.3). Ces coûts ont été évalués avec une résolution équivalente pour chaque surface de projection, et en tenant compte à la fois du nombre de proxels et du nombre de valeurs de FFEs effectivement mémorisées pour chaque surface (les symétries de chacun des algorithmes permettent de ne pas mémoriser autant de FFEs que de proxels).

Résolution de référence (hémicube)	Hémicube	Plan de Sillion	Plan de Recker	Hémisphère (découpage uniforme)	Hémisphère (découpage non uniforme)	Disque
100	0.25	0.04	0.31	0.10	0.06	0.06
200	1	0.16	1.25	0.40	0.25	0.25
400	4	0.72	5.01	1.59	1.01	1.03
600	9	1.44	11.28	3.57	2.26	2.29
800	16	2.56	20.05	6.83	4.03	4.09
1000	25	4.00	31.33	9.89	6.30	6.40

Table 2.3 ***Coûts mémoires des différents algorithmes projectifs (en Mo)***

La quantité mémoire nécessaire pour un **hémicube** de résolution R correspond au coût de représentation de chaque proxel ($3R^2$), auquel est ajouté celui de la mémorisation des valeurs de FFEs, soit 1/8 des FFEs de la face supérieure, et 1/4 des FFEs de l'une des faces latérales. En supposant que chaque proxel est codé sur 8 octets (4 pour l'identité de la facette projetée, et 4 pour sa distance au centre de projection), la quantité mémoire requise est :

$$Q_{hc} = 25R^2$$

Dans le cas du **plan unique de Sillion**, seuls les FFEs sont mémorisés, une facette recouvrant une fenêtre complète ne pouvant plus être occultée. De ce fait, il est inutile de conserver, ni l'identité de la facette, ni sa distance. Le coût de cette approche est de :

$$Q_{Sillion} = 4R^2$$

en supposant chaque FFE codé sur 4 octets.

Le **plan unique de Recker** utilise deux niveaux de découpage, symbolisés par deux résolutions R_1 et R_2. Nous supposerons que les deux grilles de proxels sont toutes deux représentées entièrement, même si la partie centrale de la grille externe (environ 10 % de la surface totale) n'est pas utilisée. Chaque proxel contient l'identité d'une facette et sa distance

au centre de projection. Du fait des symétries, seul 1/8 des FFEs pour chacune des deux grilles peut être représenté. Avec un codage des différentes valeurs sur 4 octets, la quantité mémoire totale nécessaire à représenter le plan de projection est :

$$Q_r = \frac{17}{2}(R_1^2 + R_2^2)$$

Dans le cas des algorithmes utilisant la surface de **projection hémisphérique**, la différence intervient essentiellement au niveau du nombre de proxels utilisés, et des FFEs à mémoriser. Le nombre de proxels est obtenu par le produit $R \times 4R$, où R représente la résolution selon la latitude (pas de découpage identique pour la longitude et la latitude). Chaque proxel contient l'identité d'une facette, et sa distance par rapport au centre de l'hémisphère. Dans le cas d'un découpage uniforme, il est nécessaire de mémoriser R FFEs, un pour chaque ligne de découpage, les proxels situés sur la même ligne possèdant la même valeur de FFE. L'expression permettant le calcul de la quantité mémoire nécessaire est donc :

$$Q_{hsu} = 32R^2 + 4R$$

en supposant que chaque composante (identité, distance, FFE) est codée sur 4 octets. Dans le cas de l'hémisphère à découpage non uniforme, le terme $4R$ disparait (les FFEs ont même valeur), fournissant l'expression:

$$Q_{hsnu} = 32R^2$$

La quantité mémoire nécessaire pour la représentation du **disque de projection** doit prendre en compte la mémorisation, pour chaque proxel, de l'identité de la facette visible, de sa distance, mais aussi la valeur de $\sqrt{1-u^2-v^2}$ précalculée en chaque proxel. Du fait des symétries apparaissant dans cette racine, son calcul peut être limité à 1/8 des proxels. D'autre part, la résolution R représente le découpage du carré englobant le disque. Il est donc possible de réduire le nombre total de proxels nécessaires à la représentaion du disque, d'un facteur $\pi/4$, où π représente la surface du disque unitaire, et 4 la surface du carré englobant. En tenant compte de ces différents paramètres, et en considérant que chaque valeur est mémorisée sur 4 octets, l'expression donnant la taille mémoire nécessaire au disque est donnée par:

$$Q_d = \left(\frac{17}{2}R^2\right) \times \frac{\pi}{4}$$

Le plan de Sillion est la surface de projection qui occupe, de loin, le moins d'espace mémoire. Ceci provient du fait que les proxels n'ont pas à mémoriser les informations concernant la facette qui s'y projette. En effet, la méthode particulière de projection de Sillion permet d'attribuer directement la bonne facette (celle qui est visible) aux bons proxels, ceci au prix d'un tri en profondeur des facettes à projeter. Cette approche évite de mémoriser les facettes projetées et permet d'affecter directement l'ensemble des proxels d'une fenêtre entièrement recouverte. Bien que cela n'ait été effectué dans aucune des autres approches, ce procédé est valable pour les autres surfaces de projection planaires, et doit pouvoir être appliqué aux techniques de projection hémisphériques.

Le tableau (Table 2.3) montre d'autre part le faible coût mémoire des surfaces de projection sphériques. Du fait des valeurs identiques des FFEs pour le disque et l'hémisphère utilisant un découpage non uniforme, la quantité mémoire est quasiment identique, la différence provenant de la mémorisation, pour chaque proxel du disque, de la racine carrée de l'expression (Eq. 2.5). Bien que cela ne soit pas indispensable, cette mémorisation permet de limiter les temps de calcul de l'algorithme.

Enfin, le plan de Recker montre le plus mauvais résultat, avec une quantité mémoire supérieure à celle de l'hémicube. Cela provient de l'importance du suréchantillonnage qui est obtenu en utilisant un découpage régulier de plan, même appliqué à deux niveaux différents.

Temps de calcul

Nous n'avons analysé ci-dessus que les paramètres dépendants du nombre d'échantillons utilisés par chaque algorithme. Il est important de pouvoir aussi les comparer en fonction de

leurs performances temporelles, les algorithmes les moins coûteux en terme de nombre de proxels à calculer pouvant se révéler plus longs à exécuter.

Nous avons implanté deuxde ces algorithmes, l'hémicube et le disque, et nous les avons exécutés sur plusieurs scènes de complexités différentes. Une description détaillée de ces scènes se trouve dans l'annexe A. La machine sur laquelle ont été effectuées les comparaisons a été la même dans tous les cas: il s'agit d'une station de travail Sparc LX, équipée d'un processeur microSparc à 50 MHz et de 48 Mo de RAM, et délivrant des puissances de crête d'environ 59 MIPs et 4,6 MFlops.

Nous nous sommes servi du tableau (Table 2.1) pour effectuer les calculs à résolution équivalente. Les résolutions indiquées sont celles de l'hémicube.

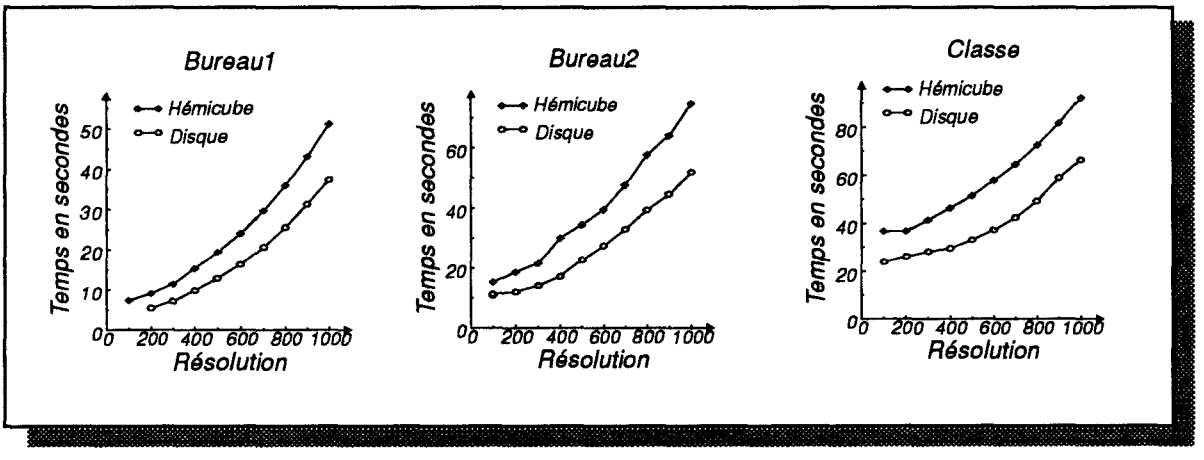


Figure 2.16 ***Evolution des temps de calcul en fonction de la résolution***

La comparaison des temps de calcul entre l'hémicube et le disque de projection montre un gain certain, mais qui reste limité, malgré le fait qu'une seule projection est effectuée dans le second cas, contre 5 dans le premier. Ceci provient de deux facteurs :

- les calculs à appliquer dans le cas du disque sont plus complexes que dans le cas de l'hémicube, puisqu'ils ne sont pas uniquement incrémentaux.
- l'algorithme de remplissage du contour projeté sur le disque, tel que nous l'avons écrit, nécessite le calcul de proxels inutiles. En effet, tous les proxels du rectangle englobant une projection sont pris en compte lors de la phase d'élimination des parties cachées, alors qu'un algorithme de suivi de contour est utilisé dans le cas de l'hémicube (ce qui permet de ne traiter que les proxels réellement à l'intérieur du contour projeté). Pour résoudre ce problème, nous travaillons actuellement sur la recherche d'un algorithme permettant de ne considérer que les proxels internes à un contour basé sur des arcs d'ellipse.

2.5.4 Comparaison qualitative

Nous nous proposons, dans ce paragraphe, de comparer la précision de l'échantillonnage fourni par différents algorithmes, à des résolutions équivalentes, au sens défini au paragraphe "Critère d'équivalence", page 39. Pour ce faire, nous avons évalué la valeur du facteur de forme fourni par les différents algorithmes pour des facettes identiques. Nous avons, de plus, fait varier la distance de ces facettes par rapport à la surface de projection, de manière à modifier la taille des projections (en terme de nombre de proxels recouverts). La géométrie des facettes est décrite sur la Figure 2.17. Les 4 facettes sont de même taille, et leur position est légèrement décalée par rapport à l'origine, de manière à éviter qu'elles ne se projettent de manière symétrique par rapport à chacun des axes.

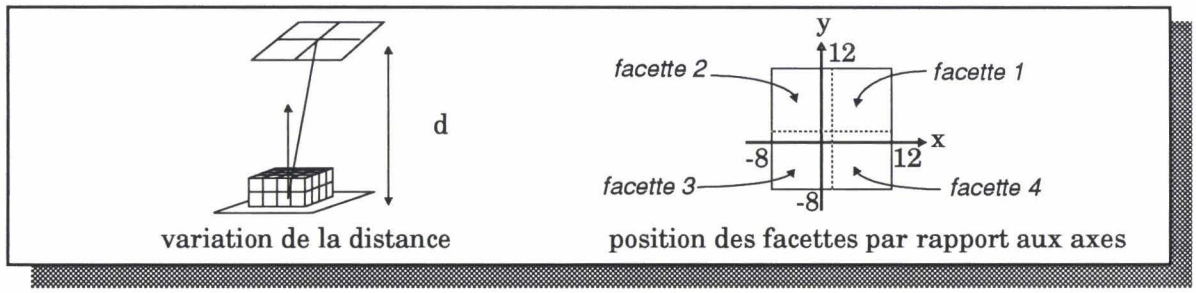


Figure 2.17 ***Géométrie des facettes pour les mesures de précision***

Nous représentons, sur la Figure 2.18, l'évolution des valeurs de facteurs de forme pour les facettes 2 et 4 (cf Figure 2.17), en fonction de la résolution de la surface de projection, et de la distance par rapport au centre de cette surface.

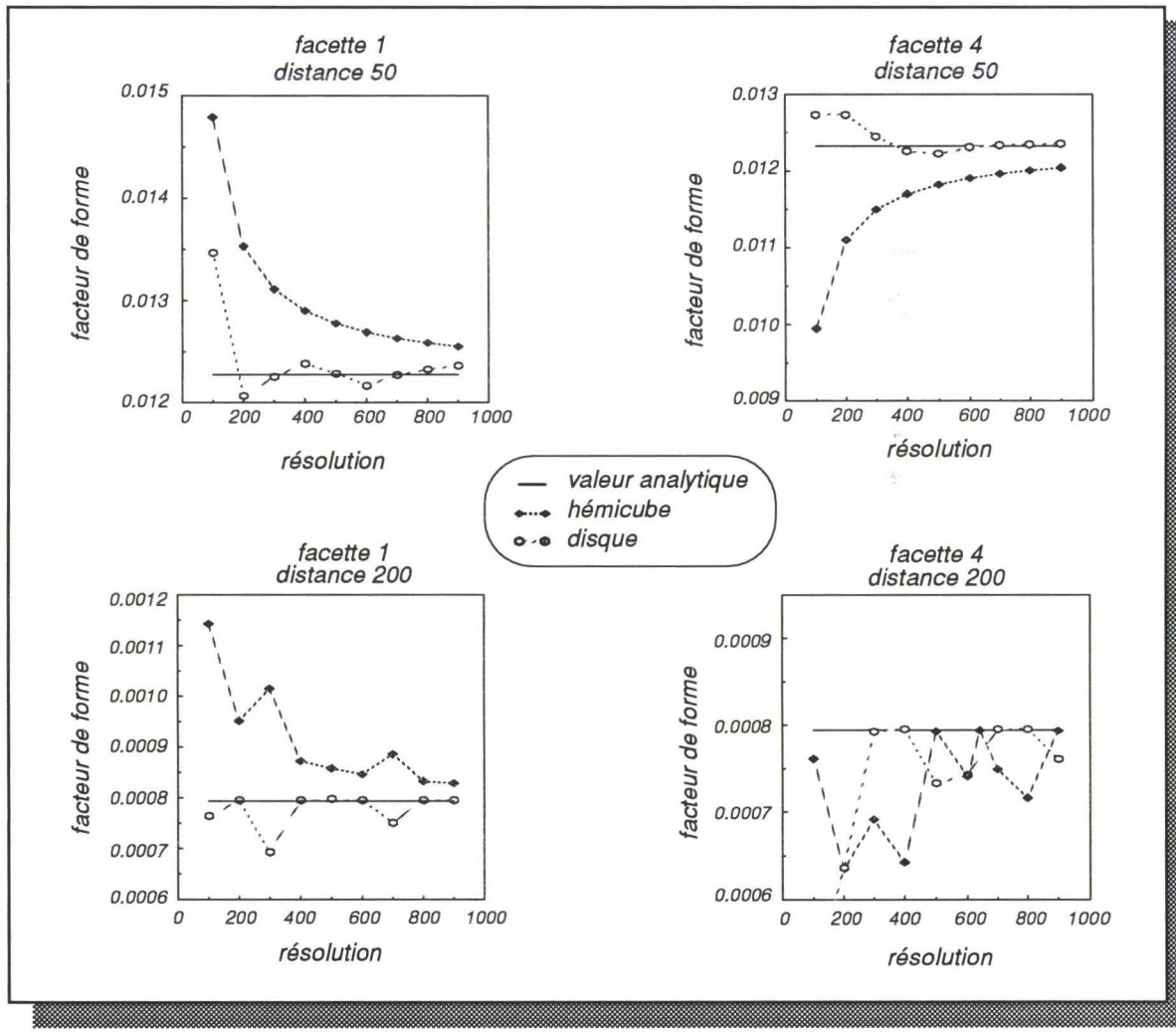


Figure 2.18 ***Evolution de la valeur des facteurs de forme selon la facette et la distance***

Les courbes présentées ont un aspect plus uniforme lorsque la facette projetée est relativement proche de la surface de projection. Ceci s'explique par le fait qu'elle recouvre alors un plus grand nombre de proxels, et que les défauts dûs à l'aliassage sont moins flagrants. Lorsque la distance augmente, la taille d'une facette projetée diminue, et les problèmes d'aliassage génèrent des erreurs plus importantes, qui varient fortement selon la résolution adoptée.

Il est intéressant de noter que, même pour une distance plus faible, l'évolution du facteur de forme pour le disque présente un aspect discontinu, contrairement à la courbe représentative de l'hémicube, qui semble continue. Cela provient vraisemblablement d'un aliassage plus important, du fait du tracé d'arcs d'ellipses sur un quadrillage régulier.

Il apparait aussi une différence notable entre l'évolution des facteurs de forme pour la facette 1 et pour la facette 2. En effet, dans le premier cas (facette 1), le facteur de forme est toujours surévalué (courbe au dessus de la valeur analytique). Dans le second cas (facette 4), par contre, il est toujours sous-évalué. Du fait que la facette 4 est la dernière facette projetée, les proxels communs le long des arêtes communes avec les facettes 1 et 3 ne lui sont pas attribués, puisqu'une facette située à la même distance y est déjà présente. Ce phénomène apparait de manière flagrante dans le cas de l'hémicube, puisque les arêtes sont relativement alignées avec les lignes et colonnes de la grille de projection. Dans le cas du disque, ce phénomène n'apparait pas, l'aspect présenté par les courbes étant beaucoup plus irrégulier.

2.6 Avantages et inconvénients de ces méthodes

2.6.1 Enoncé des problèmes

Les approches projectives offrent l'avantage d'utiliser des algorithmes d'élimination des parties cachées bien connus, tels que le tampon de profondeur. Néanmoins, elles posent un certain nombre de problèmes, dus à la fois à la technique d'échantillonnage et aux approximations qu'elles nécessitent. Baum [Baum 89] recense 3 hypothèses qui doivent être vérifiées, pour pouvoir obtenir une bonne approximation des facteurs de forme avec une méthode projective:

- Hypothèse de proximité : la distance entre deux facettes i et j est grande comparée à la taille de la facette i, sur laquelle est appliquée l'algorithme d'échantillonnage.
- Hypothèse de visibilité : la facette i est visible de la même manière, de tout point de la facette j (la fonction de visibilité possède une valeur constante).
- Hypothèse d'aliassage : la projection de chaque facette sur la surface d'échantillonnage peut être représentée précisément en utilisant un nombre fini de proxels.

Les deux premières hypothèses permettent de simplifier l'équation (Eq. 2.3), en supposant que le facteur de forme F_{ij} entre deux facettes i et j ne varie selon l'emplacement où il est évalué sur la facette i. Il peut donc être approximé par :

$$F_{ij} = \frac{1}{A_i} \cdot \int_{A_i}\int_{A_j} \frac{\cos\theta_i \cdot \cos\theta_j}{\pi \cdot r^2} VIS(dA_i, dAj)\, dA_j dA_i \cong \int_{A_j} \frac{\cos\theta_i \cdot \cos\theta_j}{\pi \cdot r^2} VIS(A_i, dA_j)\, dA_j$$

en n'appliquant qu'un hémicube pour l'échantillonnage (ou toute autre surface de projection).

La troisième hypothèse, quant à elle, permet de remplacer le calcul de l'intégrale simplifiée précédente par la sommation:

$$F_{ij} \cong \sum_{p \in P_{ij}} \Delta F_p$$

sur les proxels effectivement recouverts par la facette j (P_{ij}), en supposant que ceux-ci représentent exactement la projection de la facette j.

Baum montre que, lorsque l'une de ces hypothèses est violée, le facteur de forme résultant est erroné. Ainsi, dans le cas de la Figure 2.19.a, les surfaces 1 et 2 sont très proches l'une de l'autre. Le facteur de forme F_{21} entre la surface 2 (S_2) et la surface 1 (S_1), calculé à partir de l'hémicube est à peu près correct (comparé à sa valeur analytique), alors que le facteur de forme F_{12} entre S et S_2 est totalement faux. Ceci provient du fait que l'hémicube est appliqué enun seul point (le centre) des deux surfaces. Si la distance entre tout point de S_2 et un point de S_1 peut être considérée comme pratiquement identique, ce n'est pas le cas entre les points de S_1 et

un point de S_2. Dans la mesure où le facteur de forme dépend du carré de la distance, le résultat de F_{12} évalué depuis un seul point de S_1 est faux.

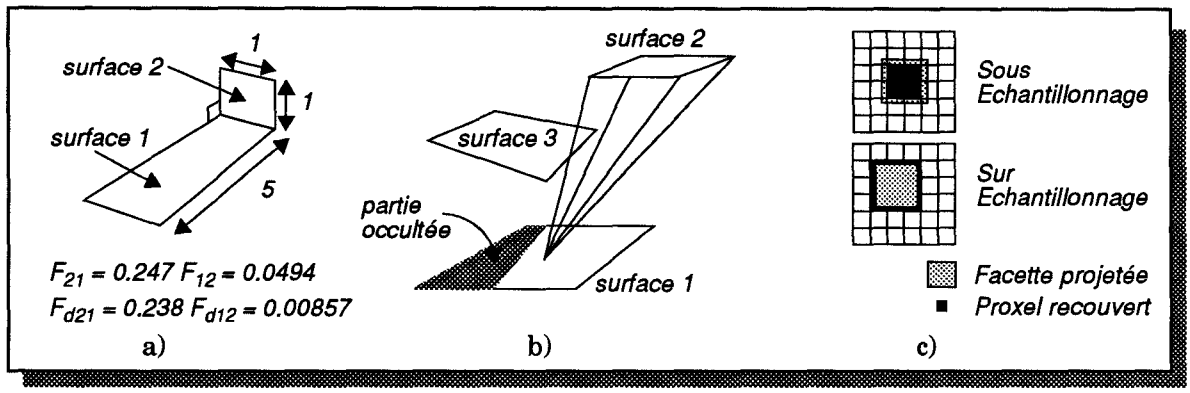


Figure 2.19 ***Violation des hypothèses de proximité (a), de visibilité (b) et d'aliassage (c)***

De même, selon que l'occultation par une surface est prise en compte ou non lors de l'application de l'hémicube, le facteur de forme peut être sous ou sur-évalué. Dans le cas schématisé sur la Figure 2.19.b, l'hémicube est appliqué au centre de S_1. S_2 est jugée entièrement visible, alors que S_3 occulte une partie de S_1 pour S_2. L'hémicube ayant été appliqué uniquement au centre de S_1, cette occultation n'est pas prise en compte, et le facteur de forme entre S_1 et S_2 est sur-évalué.

Enfin, le problème de l'aliassage se pose pour l'algorithme d'échantillonnage utilisé. Lorsqu'une facette est projetée sur l'hémicube, son appartenance à un proxel est déterminée par rapport au centre du proxel, ce qui revient à le considérer comme une quantité ponctuelle. De ce fait, selon la position de la facette projetée par rapport à ce centre, le facteur de forme pourra être sous-échantillonné ou, inversement, sur-échantillonné (Figure 2.19.c). Il peut aussi se produire, dans le cas de petites facettes, ou de facettes très éloignées de la facette d'application, que celles-ci soient totalement ignorées, car absentes de tout proxel.

Baum note que ces hypothèses sont plus souvent violées dans le cas de la radiosité progressive que dans le cas de l'algorithme utilisant la matrice complète des facteurs de forme. Dans ce dernier cas, lors de la décomposition d'une facette en éléments (chapitre 1), la surface de projection est appliquée sur chaque élément. De ce fait, l'hypothèse de proximité est rarement violée, puisque l'hémicube est appliqué sur de petits éléments. De même, en ce qui concerne l'hypothèse de visibilité, puisque la distance qui sépare tout point de l'élément de son centre est plus faible que dans le cas d'une facette plus grande. Dans le cas de la radiosité progressive, par contre, ces deux hypothèses sont beaucoup plus fréquemment violées, puisque la surface de projection est appliquée sur les facettes et que les éléments qui y sont projetés sont de taille inférieure. Les mesures effectuées par Baum montrent que, dans le plus mauvais cas, les deux facettes doivent être éloignées d'au moins 5 fois la taille de la facette d'application, pour que l'erreur de proximité commise soit inférieure à 2,5 pourcent.

L'aliassage est présent dans les deux approches, et son effet se fait particulièrement sentir lorsque la surface de projection est appliquée sur une source (radiosité progressive) ou lorsque la facette projetée est une source (matrice complète). Dans ce cas, l'erreur commise devient plus importante, puisque proportionnelle à l'énergie de la source.

Dans le cas de la radiosité progressive, lorsque l'intensité de la source est forte, la différence de facteur de forme entre éléments voisins projetés sur des proxels voisins, se traduit par une forte différence de radiosité entre ces facettes. De même, dans le cas de l'algorithme utilisant la matrice complète, le facteur de forme entre deux éléments voisins et une même source, risque d'être très différent, puisque l'aliassage se produisant sur la projection de la source ne se fera pas forcément de la même manière. Lors du rendu, les objets apparaissent alors avec des taches plus sombres aux endroits où le facteur de forme a été sous-évalué, et plus claires aux endroits où il a été sur-évalué.

Il faut enfin souligner le problème du calcul des radiosités aux sommets de facettes tel qu'il est effectué dans les approches projectives. En effet, dans ces approches, la radiosité est calculée au centre de chaque facette. Or, les informations de radiosité qui sont nécessaires lors de l'étape d'affichage, sont celles des sommets de facette, puisque c'est à partir des sommets que s'effectuent les interpolations de couleur. Ces valeurs de radiosité sont obtenues par pondération des radiosités des facettes auxquelles appartient un sommet. Ceci génère une certaine imprécision, en particulier pour les sommets qui n'appartiennent qu'à une ou deux facettes (coins ou bords d'objets).

2.6.2 Les solutions proposées

Différentes solutions ont été proposées pour résoudre les problèmes de violation des 3 hypothèses décrites dans le paragraphe précédent. La première d'entre elles concerne le problème de l'aliassage, inhérent à la méthode d'échantillonnage. Une solution permettant de réduire les erreurs produites par les facettes ne vérifiant pas les deux autres hypothèses sera ensuite décrite.

Réduction de l'aliassage

L'augmentation du nombre de proxels pour l'échantillonnage des facteurs de forme est l'une des solutions qui permet de réduire les erreurs, puisqu'elle permet d'approximer de manière plus précise la surface projetée de chaque facette. Néanmoins, le coût de calcul d'un hémicube, ou de tout autre surface de projection, dépend directement du nombre de proxels.

Une solution proposée par Airey [Airey 89] en vue de réduire le coût de calcul peut cependant être utilisée. En radiosité progressive, les erreurs ayant la plus grande importance se produisent lors des phases d'émission, lorsque l'intensité à distribuer est forte (cas des sources primaires, et éventuellement secondaires). Il faut alors un maximum de précision, et donc un grand nombre de proxels, afin de limiter les différences de radiosité trop fortes entre facettes voisines. Par contre, lors des phases d'émission où la radiosité latente d'une facette est faible, il est tout à fait possible de diminuer le nombre de proxels à utiliser, dans la mesure où l'erreur causée par l'aliassage a une moindre influence puisqu'elle est proportionnelle à la radiosité à distribuer. Ceci permet, de plus, de réduire le coût de calcul élevé généré par une surface de projection utilisant un grand nombre de proxels.

Meyer [Meyer 90] propose une alternative à l'accroissement du nombre de proxels, en partant du principe qu'une grande part de l'aliassage provient de la régularité de l'échantillonnage (répartition régulière des proxels sur la surface de projection) et de la régularité du découpage en facette.

Pour "casser" cette régularité, la surface de projection doit être appliquée de manière perturbée (jittering), afin que l'aliassage ne se produise pas de manière similaire sur les mêmes facettes à chaque nouvelle application. Différentes possibilités de perturbation sont examinées, telles que le déplacement aléatoire de la position d'application de la surface de projection sur une facette, ou la rotation de la surface de projection autour de son axe vertical (Figure 2.20).

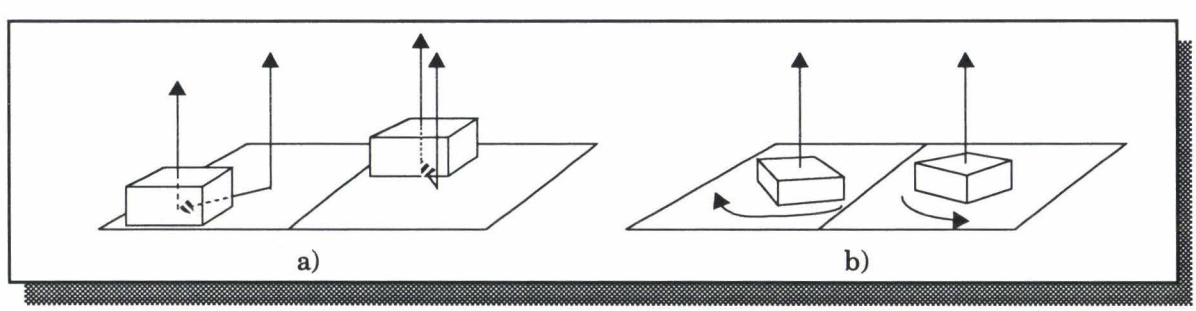


Figure 2.20 ***Perturbation de la position de l'hémicube (a) et de son orientation (b)***

Les résultats obtenus montrent qu'il est possible d'obtenir de bons résultats en conjuguant simultanément ces deux techniques, c'est à dire en appliquant à la fois une perturbation sur la position de la surface de projection, ainsi que sur son orientation.

Calcul analytique des facteurs de forme

La résolution des erreurs générées par la violation des deux premières hypothèses est plus difficile à résoudre. Baum propose une solution hybride, conjuguant les avantages respectifs de l'hémicube et du calcul analytique, pour obtenir une meilleure approximation des facteurs de forme.

Dans cette solution, l'hémicube est utilisé uniquement comme moyen de déterminer les facettes visibles dans l'ensemble des directions d'échantillonnage représentées par les proxels. Le calcul analytique des facteurs de forme est ensuite effectué à l'aide de ces informations.

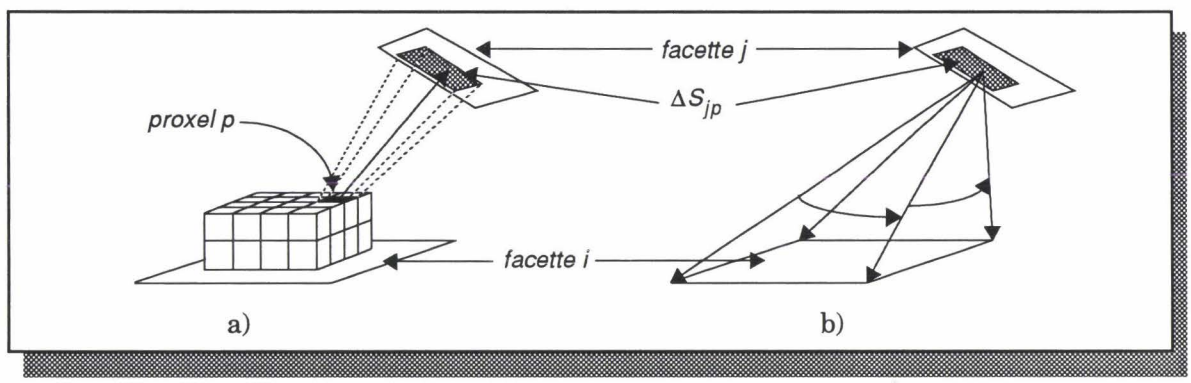


Figure 2.21 ***Calcul analytique: détermination du point d'application et de l'aire projetée (a) et calcul analytique (b)***

Lorsque chaque facette de la scène a été projetée sur l'hémicube, chaque proxel est traité à son tour, afin de calculer analytiquement la valeur du facteur de forme élémentaire de la facette qui y est visible. Pour ce faire, le point d'intersection entre un rayon, issu du centre de l'hémicube et passant par le centre du proxel, et la facette visible est calculé. La formule analytique vue au paragraphe 2.1 est ensuite appliquée sur le contour de la facette supportant l'hémicube. De manière à effectuer une pondération pour la totalité de la facette visible dans ce proxel, le facteur de forme obtenu est multiplié par l'aire de la facette qui est projetée dans le proxel.

Le facteur de forme entre une facette émettrice et une facette projetée est ensuite calculé comme la somme des facteurs de forme élémentaires calculés en chaque proxel où la facette est visible, divisée par la surface projetée totale de la facette. Cet algorithme est résumé ci-dessous, en utilisant les notations de la Figure 2.21.

```
Projeter toutes les facettes sur l'hémicube

Pour chaque proxel p
|
|     Déterminer un point sur la facette j visible à travers p
|     Calculer le facteur de forme analytique en ce point (F_ij^anal)
|     Calculer l'aire de la facette j projetée sur p (ΔS_jp)
|     Mettre à jour le ff de la facette j(F_ji += F_ji^anal x ΔS_jp)
|
FinPour

Pour toutes les facettes
|     Normaliser les facteurs de forme (F_ji /= ΣΔS_jp)
FinPour
```

Cet algorithme utilise un calcul purement analytique pour l'ensemble des facteurs de forme. Dans la mesure où il est assez coûteux, et que son application ne se justifie que dans le cas de

facettes proches, Baum ne l'applique que lorsque la distance d'une facette projetée en un proxel est inférieure à une distance minimale fixée, les autres facteurs de forme étant calculés par addition des facteurs de forme élémentaires associés à chaque proxel.

Cette approche permet de calculer précisément une grande partie des facteurs de forme, mais pose toujours le problème de l'hypothèse de visibilité. En effet, lors du calcul analytique du facteur de forme, la facette émettrice est supposée entièrement visible depuis le point d'application situé sur la facette projetée. Si cela n'est pas le cas, le facteur de forme est surestimé, ce qui conduit à des erreurs visibles lors de la phase de rendu. Pour résoudre ce problème, la somme de tous les facteurs de forme obtenus est calculée. Si celle-ci est supérieure à l'unité, alors cela signifie que des facettes ont vu leur facteur de forme surestimé. Dans ce cas, la facette émettrice est subdivisée, et l'ensemble des calculs (hémicube et calcul analytique) est réappliqué, cela jusqu'à ce que la somme des facteurs de forme soit proche de l'unité. A ce moment, les calculs sont arrêtés, et chaque facteur de forme est normalisé par rapport à la somme précédemment calculée, de manière à obtenir une somme exactement égale à 1, évitant ainsi d'émettre plus d'énergie que n'en possède la facette émettrice.

Notons enfin que cette technique, en plus de son coût, reste très sujette aux problèmes d'aliassage, puisque la détermination des facettes visibles est effectuée par l'intermédiaire de l'hémicube, ou de tout autre algorithme projectif.

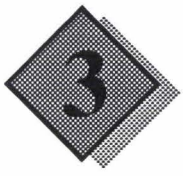
2.7 Conclusion

Différentes techniques d'échantillonnage pour le calcul des facteurs de forme ont été présentées. Elles sont toutes basées sur un principe projectif, issu de l'équivalent de Nusselt. Leur utilisation permet d'obtenir de bons résultats quant à l'approximation des facteurs de forme, sous réserve que le nombre d'échantillons qu'elles utilisent soit suffisamment important.

Nous avons introduit un critère de comparaison, basé sur la valeur maximale de facteur de forme élémentaire pour chaque algorithme. Ce critère nous a ensuite permis de comparer les différentes surfaces de projection, d'un point de vue quantitatif et qualitatif. Bien que l'ensemble des surfaces n'aient pas été entièrement comparées, il apparait que la méthode du disque de projection que nous avons introduite représente un bon compromis entre la quantité mémoire nécessaire à sa représentation, et la qualité de l'échantillonnage effectué. Elle pose néanmoins le problème du coût de calcul qui reste élevé dans la version actuelle, des optimisations étant à l'étude. Il serait d'autre part intéressant de poursuivre l'ensemble des comparaisons sur tous les algorithmes projectifs, en particulier en ce qui concerne les coûts de calcul et la précision obtenue.

Il faut cependant souligner que ces techniques posent un certain nombre de problèmes liès aux points d'application, et qu'ils sont très sensibles aux problèmes de l'aliassage. Des solutions qui ont été envisagées dans le cas de l'hémicube peuvent cependant être étendues aux autres algorithmes. Notons cependant que ces problèmes ne font que croître lorsque la complexité (niveau de détail) des scènes augmente, puisque celle-ci nécessite alors la création de facettes très petits.

CHAPITRE



Calcul des facteurs de forme par tracé de rayons

Dans le chapitre précédent, nous avons étudié un certain nombre d'algorithmes permettant l'approximation des facteurs de forme, basés sur une approche projective. Une seconde catégorie d'algorithmes a été introduite vers la fin des années 80, dont le premier objectif était d'éviter les problèmes de calcul des radiosités au sommet tels qu'ils sont effectués dans les méthodes projectives, de même que les problèmes d'aliassage qui leur sont inhérents. Ces algorithmes sont basés sur la technique du tracé de rayons. Cette technique a été utilisée de deux manières différentes, selon que les rayons sont tracés depuis les éléments à éclairer vers la source, ou depuis la source vers le reste de l'environnement. Nous allons décrire, dans ce chapitre, chacune de ces deux classes d'algorithmes, en précisant leurs motivations et leurs principaux atouts et inconvénients. Nous étudierons ensuite les différentes techniques d'optimisation applicables sur ces algorithmes, et nous introduirons une nouvelle technique, basée sur le tracé de faisceau, dont le but est d'obtenir une meilleure précision lors du calcul des échanges lumineux.

3.1 Echantillonnage de la source

3.1.1 Principe de l'approche

L'un des problèmes qui apparait, lors de l'utilisation des méthodes projectives dans l'algorithme de radiosité, et que nous avons évoqué dans le chapitre précédent, est que les valeurs de radiosité sont calculées, non pas en chaque sommet de facette, mais pour chaque facette. Dès lors, un niveau d'imprécision est introduit pour la phase d'affichage des images, puisque la valeur au sommet, qui est utilisée pour la phase d'interpolation, est calculée comme une moyenne des radiosités des facettes communes au sommet.

Wallace propose , dans le cadre d'un algorithme de radiosité progressive, de calculer directement la valeur de radiosité reçue en chaque sommet [Walla 89]. Il est alors nécessaire de déterminer la visibilité de la source courante depuis chaque sommet de l'environnement, puis, en fonction de la partie de la source visible depuis un sommet donné, de calculer la quantité d'énergie qu'il reçoit.

Les approches projectives ne peuvent être utilisées, puisqu'elles n'effectuent les opérations de visibilité que sur des surfaces finies, et non pas des éléments ponctuels. Wallace propose donc d'utiliser l'algorithme du tracé de rayons pour effectuer cette opération de calcul de visibilité, puisque, par principe, cet algorithme ne travaille que sur des quantités ponctuelles.

Ainsi, pour chaque sommet de l'environnement, plusieurs rayons de visibilité sont lancés vers la source, de manière à approximer la proportion de la source qui est visible depuis un sommet. Ce principe est schématisé sur la Figure 3.1.

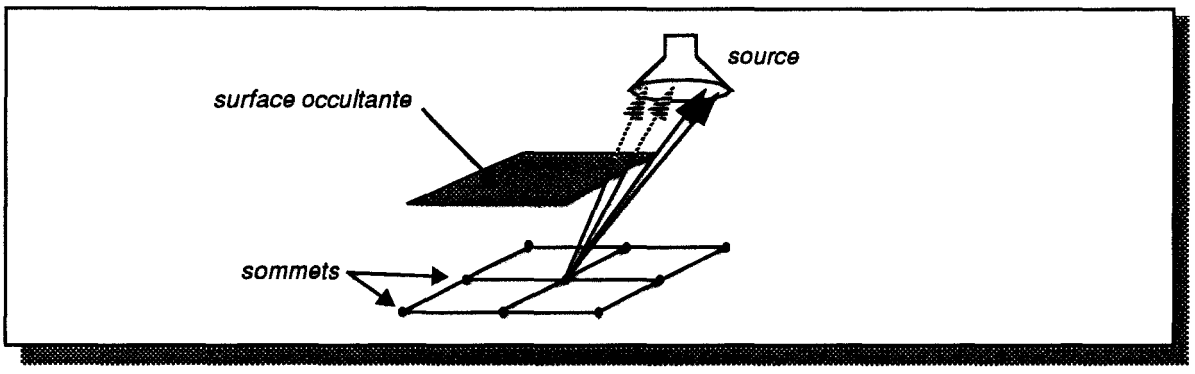


Figure 3.1 ***Principe de l'échantillonnage d'une source depuis les sommets de la scène***

La quantité d'énergie reçue par le sommet depuis la source peut être évaluée en fonction du nombre de rayons ayant atteint la source, et aussi en tenant compte de la distance et de l'orientation de la source par rapport au sommet.

3.1.2 Algorithme

Ces opérations de tracé de rayons vers la source sont appliquées pour chaque sommet de la base de données, et pour chaque nouvelle source sélectionnée à chaque itération. L'algorithme général peut alors être écrit de la manière suivante, sous forme de pseudo-langage:

```
Pour chaque phase de shooting
|
|     Pour chaque sommet de la base de données
|     |
|     |     déterminer le nombre de rayons parvenant sur la source
|     |     en déduire le facteur de forme avec la facette émettrice
|     |     mettre à jour la radiosité
|     |
|     FinPour
|
|     choisir une nouvelle facette émettrice
|
FinPour
```

A noter que, contrairement aux approches projectives, la mise à jour de la radiosité d'un sommet peut être effectuée dès que les informations d'intersection entre les rayons issus d'un sommet et les objets de la base de données sont connues. Il n'y a en effet aucune interaction entre les calculs de visibilité depuis deux sommets de la scène.

3.1.3 Calcul de la contribution d'une source sur un sommet

Le point le plus délicat de l'algorithme est de pouvoir déduire des rayons ayant atteint la source, une information énergétique, servant à la mise à jour de la radiosité d'un sommet.

La source est découpée en autant d'éléments de surface, que de rayons lancés depuis le sommet, possédant tous la même taille. Un rayon est lancé depuis un sommet vers chacun des ces éléments (Figure 3.2.a). De cette manière, chaque rayon est potentiellement porteur d'une partie de l'énergie issue de la source et parvenant au sommet. Le calcul de l'énergie transportée par un rayon est effectué de manière analytique, en approximant chaque élément de la source par un disque, surface pour laquelle une expression analytique simple existe.

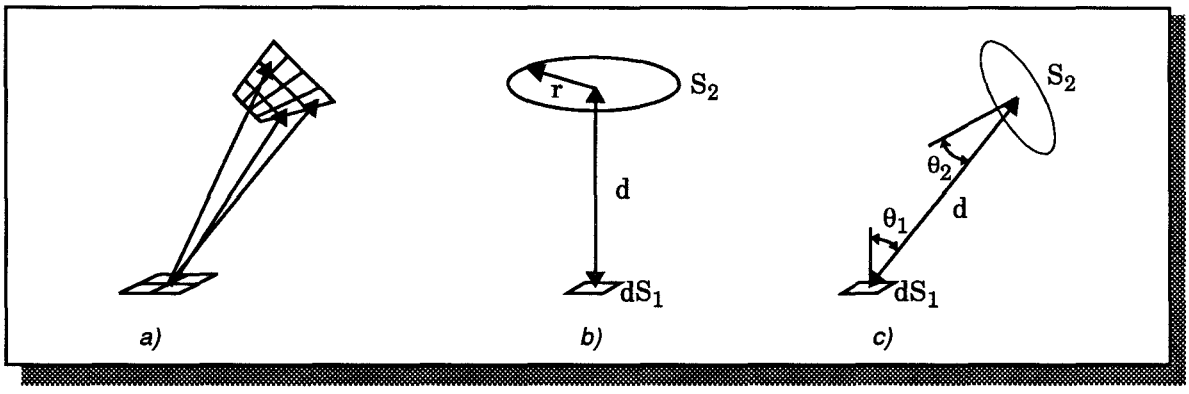


Figure 3.2 ***Découpage de la source en éléments de surface (a) et géométrie pour le calcul analytique du facteur de forme entre un élément de surface et un disque (b et c)***

Dans le cas d'un élément de surface dS_1 et d'un disque de surface S_2, situé à une distance d de dS_1, lui faisant directement face (Figure 3.2.b), le facteur de forme entre S_2 et dS_1 s'exprime sous la forme:

$$dF_{S_2, dS_1} = \frac{dS_1}{(\pi d^2 + S_2)}$$

De manière à tenir compte des effets des différentes orientations qui peuvent exister entre dS_1 et S_2, Wallace introduit les cosinus des angles entre les normales aux deux surfaces et la droite reliant le centre des deux surfaces (Figure 3.2.c). Le facteur de forme précédent se réécrit alors sous la forme:

$$dF_{S_2, dS_1} = \frac{dS_1 \cos\theta_1 \cos\theta_2}{(\pi d^2 + S_2)}$$

Le facteur de forme entre un élément de surface dS_1 et une surface quelconque S_2 peut alors être obtenu, en supposant que les rayons d'échantillonnage sont distribués uniformément sur la surface de la source S_2, comme la somme de tous les facteurs de forme élémentaires calculés à partir de l'expression précédente:

$$dF_{S_2, dS_1} = dS_1 \frac{1}{n} \sum_{i=1}^{n} \delta_i \frac{\cos\theta_{1i} \cos\theta_{2i}}{(\pi d_i^2 + S_2/n)}$$

où n représente le nombre d'échantillons utilisés, c'est à dire le nombre de rayons lancés, et δ_i un terme de visibilité valant 1 si la source est visible du sommet dans la direction du rayon d'indice i, et 0 sinon. Il est alors possible d'obtenir l'expression générale du calcul de la radiosité B_1 reçue par un sommet depuis une surface S_2, de radiosité B_2:

$$B_1 = \rho_1 B_2 S_2 \frac{1}{n} \sum_{i=1}^{n} \delta_i \frac{\cos\theta_{1i} \cos\theta_{2i}}{(\pi d_i^2 + S_2/n)}$$

C'est cette expression qui est utilisée pour le calcul de la radiosité reçue par chaque sommet, en tenant compte des rayons qui parviennent effectivement à la source, et de leur direction par rapport aux normales respectives du sommet et de la source au point d'impact.

3.1.4 Avantages et inconvénients

L'avantage immédiat de cette approche est qu'elle permet de résoudre de fait le problème du calcul de la radiosité directement au sommet des facettes, en éliminant les problèmes de discontinuité d'éclairage qui apparaissent avec les méthodes projectives. L'utilisation du tracé de rayons permet de plus de travailler sur la description exacte des surfaces, plutôt que sur leur approximation en facettes, fournissant ainsi une meilleure précision lors des calculs d'intersection. Enfin, le nombre de rayons à lancer peut être différent d'un sommet à l'autre, ce qui permet d'obtenir une meilleure précision des échanges lumineux là où cela s'avère nécessaire. Ainsi, dans le cas des sommets peu éloignés de la source, il est préférable d'avoir un échantillonnage important, tandis que dans le cas de sommets très éloignés, un nombre restreint de rayons est nécessaire. Rappelons que cette possibilité de faire varier la quantité d'échantillons selon l'emplacement des récepteurs n'existe pas avec les approches projectives.

Cette méthode de calcul soulève néanmoins de nouveaux problèmes, tels que le choix du nombre de rayons à lancer, la façon de répartir les échantillons sur la source, ou encore, l'augmentation du temps de calcul introduite par l'utilisation du tracé de rayons. Ce dernier point sera d'ailleurs l'objet du paragraphe 3.3, intitulé "Techniques d'optimisation".

En ce qui concerne le nombre de rayons à lancer, rappelons que les facteurs de forme sont calculés en approximant chaque région d'échantillonnage de la source par un disque. Cette approximation n'est valable que si la forme de chacune de ces régions est proche de celle du disque. Dans le cas contraire, il est nécessaire de subdiviser finement la source, de manière à réduire au maximum l'erreur commise. Wallace effectue ainsi un certain nombre de comparaisons sur la valeur exacte du facteur de forme pour un sommet, et la valeur calculée par son algorithme. Il montre ainsi que le nombre d'échantillons à utiliser, pour atteindre une même précision qu'un calcul analytique pur, augmente fortement (jusqu'à 16 pour les cas étudiés) lorsque la forme de la source s'éloigne de celle d'un disque, ou lorsque le sommet et la source ne se font plus directement face.

De même, lorsque la source est très proche du sommet à considérer (Figure 3.3.a), le nombre d'échantillons à évaluer devient très important. Utiliser une distribution uniforme des échantillons sur la source conduit alors à un grand nombre de calculs inutiles, puisque les directions les plus éloignées de la normale au sommet ne vont fournir qu'une faible contribution à l'éclairage du sommet.

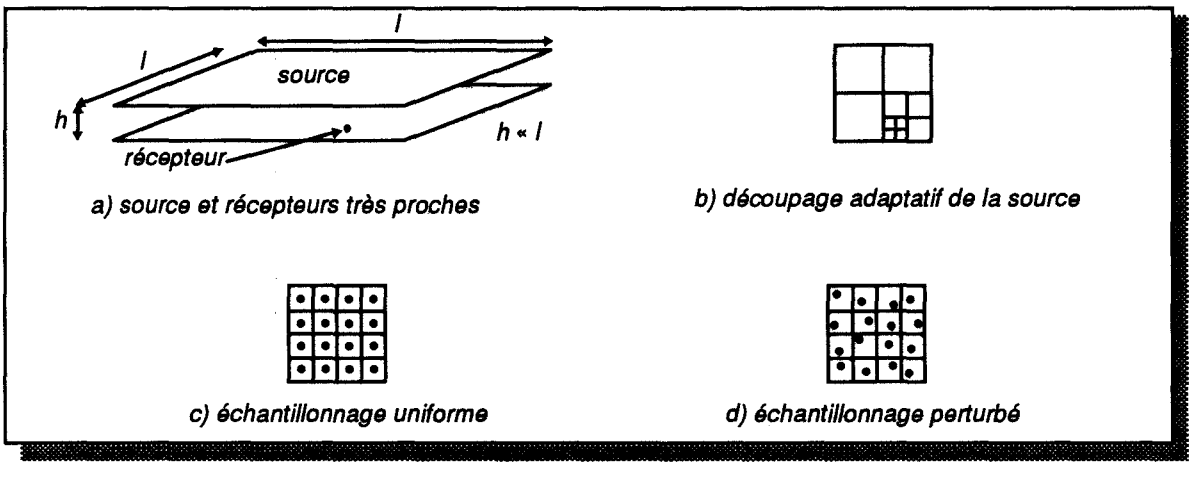


Figure 3.3 ***Différents problèmes d'échantillonnage de la source***

Dans ce cas, Wallace suggère d'utiliser un échantillonnage adaptatif de la source (Figure 3.3.b), le critère de subdivision de chaque zone étant un seuil d'énergie émise par chaque zone considérée: lorsque l'énergie émise par une zone devient inférieure à un certain seuil, le découpage de cette zone s'arrête; dans le cas contraire, la zone est subdivisée. Les rayons d'échantillonnage sont ensuite lancés, un par zone obtenue. La répartition uniforme des rayons sur la source introduit aussi un phénomène d'aliassage sur l'évaluation des facteurs de forme,

du fait de l'alignement des échantillons (Figure 3.3.c). Celui-ci se répercute sous forme d'aliassage visuel aux limites des ombres des objets. Pour pallier ce problème, Wallace propose d'utiliser, soit une pondération de la radiosité de chaque sommet avec celles des sommets voisins, soit d'effectuer une perturbation aléatoire de la position des échantillons sur la source (jittering) ayant pour effet de transformer l'aliassage en bruit (Figure 3.3.d).

3.2 Echantillonnage des directions d'émission

Cette seconde approche est l'inverse de la précédente: plutôt que de calculer l'énergie que chaque sommet de l'environnement reçoît depuis une source, un échantillonnage des directions d'émission depuis la source est effectué, de manière à déterminer les facettes de la scène qui reçoivent de l'énergie.

Languénou [Langu 91] propose de calculer les facteurs de forme entre la facette émettrice et les différentes facettes de la scène de manière analogue aux approches projectives. Les informations de visibilité depuis la source sont calculées par l'intermédiaire d'un hémisphère, placé au centre de la facette émettrice, et découpé en un grand nombre de proxels. Cette approche s'apparente donc fortement aux méthodes projectives utilisant un hémisphère [Spenc 90]. La différence essentielle provient de l'utilisation du lancer de rayons, plutôt que d'algorithmes basés sur la notion de tampon de profondeur, pour déterminer la facette visible en chaque proxel (bien qu'il soit impropre dans ce cas, nous conservons le terme proxel pour plus de clarté). En effet, pour chaque direction d'émission représentée par chaque proxel, un rayon est lancé dans la scène, porteur d'une valeur de facteur de forme élémentaire (Figure 3.4). Les intersections entre ce rayon et les objets de la scène sont calculées, et l'objet le plus proche de la source dans la direction du rayon reçoit la valeur de facteur de forme portée par le rayon.

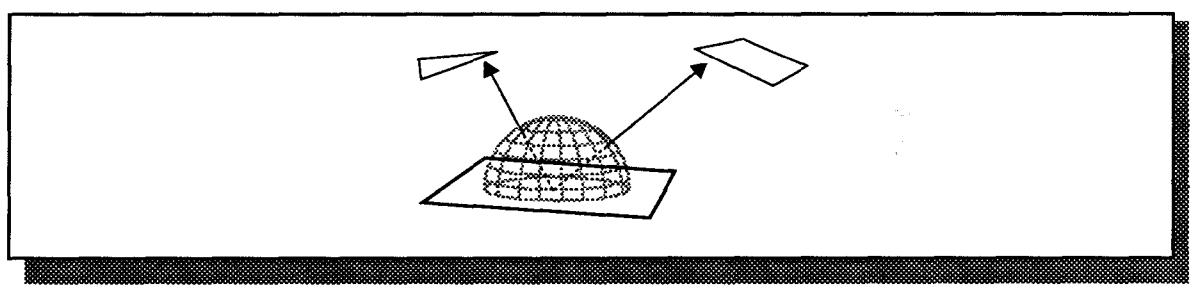


Figure 3.4 ***Principe de l'échantillonnage des directions d'émission par le tracé de rayons***

L'avantage principal qui apparait dans l'utilisation du tracé de rayons comme outil d'élimination des parties cachées, réside dans le fait qu'il n'est plus nécessaire de mémoriser la structure de projection utilisée dans les approches projectives. En effet, chaque rayon est traité séquentiellement, et la fin de son traitement indique que la facette visible dans la direction qu'il indique est connue. Il n'est donc pas nécessaire de mémoriser le proxel à travers lequel il passe. L'utilisation du lancer de rayons permet, d'autre part, de travailler sur la description exacte de la surface des objets, plutôt que sur l'ensemble des facettes qui les constituent. Ceci permet d'obtenir un calcul précis des points d'intersection avec un rayon. Enfin, l'utilisation de cette technique peut permettre la prise en compte de réflexions directionnelles.

En contrepartie, l'utilisation du tracé de rayons est plus coûteuse que l'application des méthodes projectives. Les auteurs notent cependant que l'utilisation des structures d'accélération qui sont décrites dans le prochain paragraphe, permet de réduire considérablement les écarts de coût entre les deux approches. D'autre part, ils utilisent une modélisation particulière, appelée texture de radiosité, qui permet d'effectuer les calculs d'intersection avec les surfaces représentant les objets, plutôt qu'avec l'ensemble des facettes approximant cette surface. La génération des facettes est alors effectuée dans le cadre d'une représentation paramétrique de leur position, ce qui permet, à partir du point d'impact sur la surface, de retrouver facilement la facette qui contient ce point d'impact. A noter que le même type d'algorithme et de modélisation est utilisé par Thomin [Thomi 93].

Dans la mesure où le principe d'échantillonnage est similaire à celui des approches projectives, cette approche souffre des mêmes problèmes que ceux exposés au paragraphe 2.6.1 (facettes manquées, aliassage, calcul de radiosité en chaque facette plutôt qu'au sommet des facettes, ...).

3.3 Techniques d'optimisation

L'un des problèmes communs aux deux approches est le temps de calcul important généré par l'utilisation du tracé de rayons. Celui-ci nécessite en effet de calculer les intersections avec un grand nombre d'objets de la scène, voire tous. Il est cependant possible d'utiliser les techniques d'optimisation qui ont été mises au point depuis de nombreuses années pour l'algorithme du lancer de rayons, le tracé de rayons n'étant en fait qu'une sous-classe de cet algorithme.

3.3.1 Les volumes englobants

Dans son article original sur le lancer de rayon, Whitted [Whitt 80] proposait l'utilisation de volumes englobants, de géométrie simple, qui permettent d'englober chaque objet de la scène. De cette manière, les calculs d'intersection entre un objet et un rayon sont tout d'abord effectué sur le volume englobant l'objet. S'il y a intersection avec l'englobant, alors un nouveau calcul d'intersection est lancé avec l'objet qui s'y trouve. Dans le cas contraire, il n'est pas nécessaire d'entreprendre un calcul d'intersection avec l'objet. Cette technique permet alors un gain de temps si le calcul d'intersection d'un rayon avec un volume englobant est plus simple à effectuer que le même calcul avec l'objet lui même. Pour cette raison, les volumes généralement employés sont des sphères ou des parallélépipèdes.

De manière à accélérer encore le calcul d'intersection, l'utilisation d'une hiérarchie de volumes englobants a ensuite été proposée [Rubin 80]. Dans ce cas, les volumes englobants un objet sont eux-mêmes regroupés successivement en volumes englobants de plus grande taille, en fonction de leur proximité dans la scène. La base de données est alors représentée sous la forme d'une arborescence, dont les noeuds sont des volumes englobants, et les feuilles les objets constituant la scène. Lors du traitement d'un rayon, celui-ci parcourt l'arbre depuis la racine jusqu'aux feuilles. Lorsqu'un test d'intersection avec un noeud de l'arbre se révèle négatif, toute la partie de l'arbre située sous ce noeud n'est pas testée. Ces deux notions sont représentées sur la Figure 3.5.

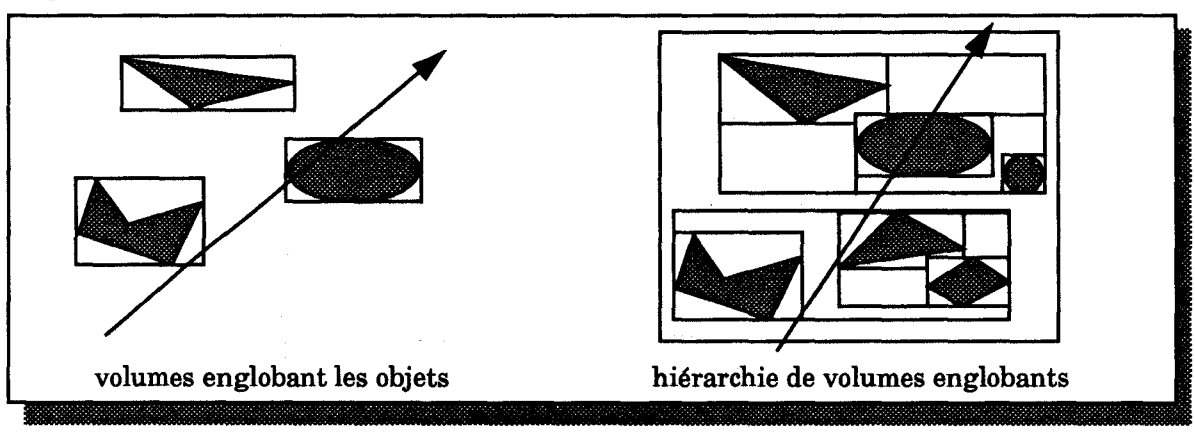


Figure 3.5 ***Principe des volumes englobants***

Nous avons implanté cette notion de hiérarchie de volumes englobants au sein d'un algorithme de radiosité utilisant l'approche de Wallace, détaillée au paragraphe 3.1. Les résultats obtenus montrent un gain certain de cette approche, qui dépend essentiellement de la complexité de la base de données. Ces résultats sont résumés dans le tableau ci-contre.

Scène	Temps sans V. E. (s)	Temps avec V. E. (s)	Gain
Bureau1	11,80	3,76	3,14
Bureau2	67,03	9,90	6,77
Classe	337,57	19,38	17,42

Table 3.1 ***Gain obtenu en utilisant les volumes englobants sur différentes scènes***

Les temps indiqués correspondent au temps moyen d'une phase d'émission, lorsque l'on n'utilise pas la notion de volume englobant, puis en l'utilisant. La troisième colonne du tableau représente le gain moyen que l'on obtient pour chacune des trois scènes en utilisant une hiérarchie de volumes englobants. Les volumes englobants utilisés sont des parallélépipèdes rectangles, pour lesquels chaque face est parallèle à l'un des axes de coordonnées du repère de la scène.

Le gain obtenu avec l'utilisation de cette technique d'optimisation dépend fortement de la scène utilisée. En effet, pour des scènes simples, c'est à dire possédant un nombre d'objets restreint, la hiérarchie obtenue ne possédera pas une profondeur suffisante pour permettre un gain très important, puisque celui-ci est avant tout basé sur la possibilité d'éliminer très rapidement des branches de grande taille de l'arborescence. Par contre, dès que le nombre d'objet augmente, la hiérarchie des volumes englobants permet d'obtenir des gains très appréciables, comme le montre le gain moyen obtenu dans le cas de la scène "Classe".

3.3.2 Découpage de l'espace objet

Le but de la méthode précédente est de diminuer le nombre d'objets à considérer pour les calculs d'intersection. Dans le cas des méthodes d'optimisation utilisant un découpage de l'espace objet, l'objectif est de réduire l'espace de recherche des intersections aux seules zones parcourues par un rayon.

Deux techniques de subdivision sont envisageables, selon qu'elles génèrent une subdivision régulière ou adaptative de l'espace de la scène:

- Dans le premier cas, l'espace est découpé en éléments de volumes de même taille, appelés *voxels* (Volume Elements). Chaque voxel contient la liste des objets qui se trouvent complètement ou en partie dans le volume d'espace qu'il représente [Fujim 86]. Lors de la recherche d'une intersection entre un rayon et les objets de la scène, seuls les objets situés dans les voxels traversés par le rayon sont considérés. Le choix des voxels à utiliser est effectué en suivant incrémentalement le rayon de voxel en voxel, par un algorithme du même type que les algorithmes utilisés pour le tracé de segments en 2 dimensions.
- Le but de la subdivision adaptative de la scène est de générer des partitions de l'espace de tailles différentes, en fonction de la répartition des objets dans la scène. Les zones qui possèdent un grand nombre d'objets seront subdivisées finement, alors que celles qui ne possèdent pas ou peu d'objets ne seront pas découpées. L'avantage de cette structure est d'optimiser le taux de remplissage de chaque partition, de manière à ce qu'il y ait approximativement le même nombre de calculs d'intersection à effectuer dans chaque volume. A l'inverse, cette différence de taille entre les partitions nécessite des algorithmes de suivi de rayons plus complexes.

  Différentes techniques de subdivision adaptative peuvent être citées, comme l'*octree* [Glass 84], où chaque zone est découpée en huit nouvelles zones de même taille, ou la partition *BSP* (Binary Space Partitioning) [Kapla 85], où chaque zone est découpée en deux zones de même taille.

Le découpage régulier de la scène en voxels de même taille offre l'avantage de la simplicité, tant au niveau de la structure de données, que des algorithmes de suivi de rayon de voxel en voxel. Il génère par contre un très fort coût en place mémoire, puisque l'intégralité des voxels doit être représentée. En revanche, ce coût est très largement atténué avec la subdivision de la scène de manière adaptative, puisque seules les zones où il existe effectivement des objets sont découpées. En contrepartie, les algorithmes de suivi d'un rayon entre voxels de tailles différentes sont plus complexes que les algorithmes incrémentaux de l'approche précédente. La figure ci-dessous (Figure 3.6) schématise ces deux types de découpage.

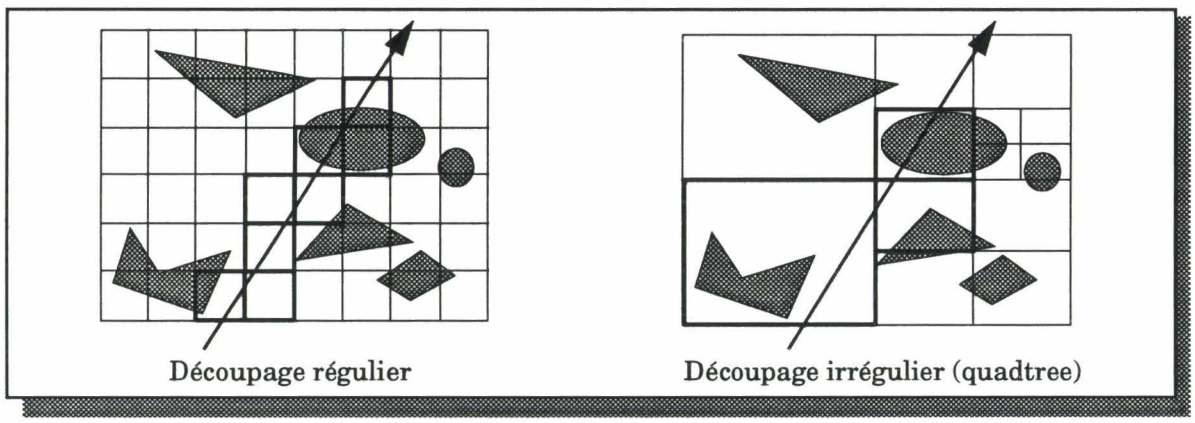


Figure 3.6 ***Découpages régulier et irrégulier d'une scène (représentation 2D)***

En définitive, le découpage régulier est bien adapté aux scènes qui présentent une répartition uniforme des objets, tandis que le découpage adaptatif tire pleinement partie des déséquilibres de charge entre les différentes zones d'une scène.

Degrande [Degra 93] effectue une comparaison théorique des accélérations maximales que l'on peut attendre de chacun de ces deux découpages. Les chiffres qu'il obtient montrent que le gain théorique qui peut être obtenu par ces méthodes d'accélération par subdivision est de 1000. Néanmoins, le gain réel est plus faible dans le cas général.

3.3.3 Utilisation de la notion de cohérence

Cohérence entre les rayons

Dans le cas de l'algorithme de lancer de rayons, il est possible d'exploiter la cohérence qui existe entre certains rayons, pour accélérer les calculs d'intersection. En effet, des rayons primaires issus de l'oeil de l'observateur et passant par des pixels voisins vont se diriger à l'intérieur de la scène avec des directions voisines. Il est donc possible de réduire le nombre d'objets à considérer en se limitant aux objets présents dans un volume "pyramidal" délimité par le point de départ des rayons, et les côtés de la zone de l'écran considérée [Amana 84], [Arvo 87].

Dans le cas de l'algorithme proposé par Wallace, où les rayons sont lancés depuis les sommets de la scène vers la source, deux types de cohérence sont utilisables:

- des rayons, issus de sommets voisins appartenant au même objet, ont une grande probabilité d'avoir le même comportement vis-à-vis des autres objets de la scène, car ils se dirigent vers la même source. En d'autres termes, si un rayon trouve une intersection avec l'un des objets de la scène, il existe de grandes chances pour que les rayons issus des sommets voisins possèdent aussi une intersection avec le même objet (Figure 3.7.a).
- les rayons issus d'un même objet se dirigent tous vers la même source. De ce fait, les objets qui peuvent potentiellement occulter ces rayons se trouvent tous dans un volume délimité par la source et la surface de départ des rayons (Figure 3.7.b).

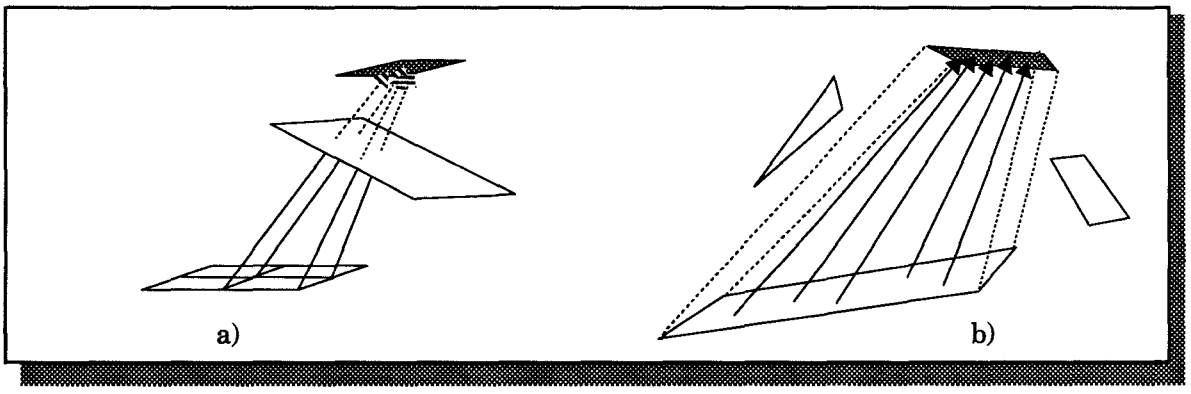


Figure 3.7 ***Schématisation des propriétés de cohérence entre les rayons issus d'un même objet***

Le premier type de cohérence est facilement exploitable, sous réserve de disposer de la notion de voisinage entre les sommets. Des résultats expérimentaux fournis par Haines montrent un gain de plus de 30% avec ce type de cohérence.

Haines [Haine 91] a exploité le second type de cohérence, en définissant une structure appelée *tunnel* (shaft), associée à l'utilisation d'une structure hiérarchique de volumes englobants. La construction de cette structure est effectuée en fonction des volumes parallélépipédiques englobant chacune des deux surfaces, la surface émettrice et la surface réceptrice (Figure 3.8.a).

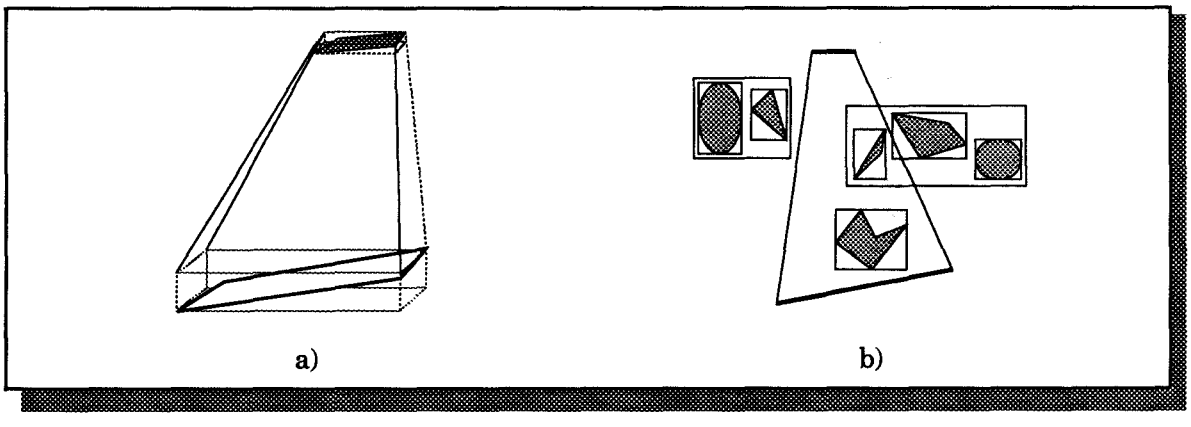


Figure 3.8 ***Construction d'un tunnel (a) et de la liste des candidats (b)***

Après construction du tunnel, une liste de volumes englobants, appelée "liste des candidats", est construite, de manière à ne contenir que les volumes englobants qui se trouve totalement ou partiellement à l'intérieur du tunnel. Ils sont alors les seuls à être testés par chacun des rayons issus de la surface réceptrice et se dirigeant vers la surface émettrice (Figure 3.8.b). Haines examine différentes stratégies pour effectuer la construction de la liste de candidats:

- "Always open": toute boite englobante, entièrement ou partiellement, présente dans le tunnel est ouverte, de manière à n'obtenir, en fin de création de la liste, que des objets primitifs entièrement ou partiellement présents dans le tunnel.
- "Overlap open": si un volume englobant est entièrement inclus dans le tunnel, il est ajouté à la liste des candidats. Dans le cas où il n'est que partiellement présent dans le tunnel, il est systématiquement ouvert, et les test sont à nouveau effectués sur ses volumes fils.
- "Keep closed": un volume englobant qui intersecte le tunnel est systématiquement ajouté à la liste des candidats. La liste des volumes englobants dont il est le père est donc implicitement ajoutée à la liste.

- "Ratio open": dans cette stratégie, le volume englobant testé est rajouté à la liste si il recouvre le tunnel avec plus d'un certain pourcentage. Dans le cas contraire, il est ouvert et les tests sont appliqués sur les volumes fils. La notion de pourcentage de recouvrement est calculée à partir du nombre de volumes fils qui se trouvent dans le tunnel.

Haines note que les meilleurs résultats expérimentaux (environ 20% d'intersections traitées en moins) sont obtenus avec la stratégie "ratio open", utilisant un seuil de recouvrement de 40%.

Il est aussi possible d'utiliser la cohérence entre les rayons issus de la source, dans l'algorithme proposé par Languénou. Celle-ci peut alors être exploitée de manière similaire à l'exploitation qui en est faite dans le cas du traitement des rayons primaires, dans l'algorithme de lancer de rayons. En effet, les rayons d'échantillonnage qui sont lancés au travers de proxels voisins ont une forte probabilité de toucher les mêmes objets (Figure 3.9).

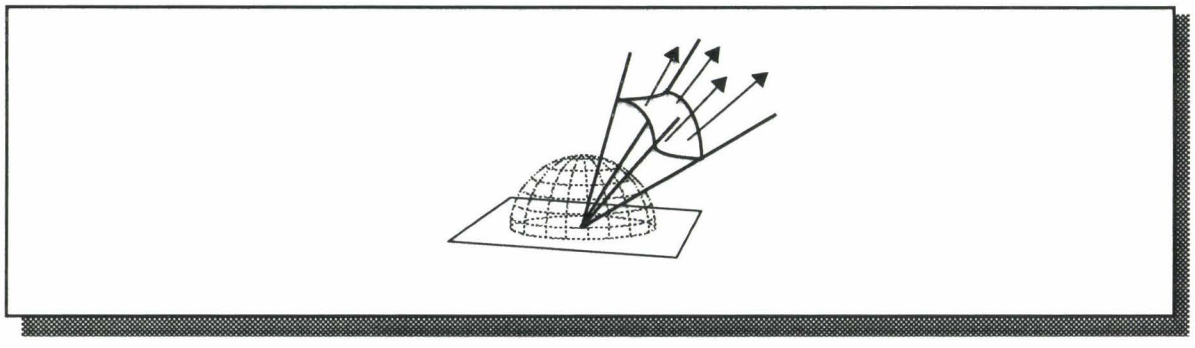


Figure 3.9 ***Schématisation des propriétés de cohérence entre proxels voisins.***

De ce fait, si la liste des objets visibles dans un ensemble de directions donné, défini par un angle solide, est connue, elle peut être utilisée par tous les rayons appartenant à cet angle solide.

Cohérence entre facettes voisines

Une seconde notion de cohérence est utilisable entre facettes émettrices voisines, pour lesquelles on peut considérer que la partie de la scène qu'elles percoivent varie peu. Ainsi, sous réserve que la distance entre une facette émettrice et les facettes de l'environnement soit suffisamment grande, Tellier [Telli 91] propose de calculer effectivement un hémisphère, puis d'approximer les hémisphères appliqués sur les facettes voisines en fonction de celui-ci. Pour cela, le déplacement appliqué sur l'hémisphère, pour l'appliquer sur une facette émettrice voisine, est répercuté sur les proxels de l'hémisphère réellement calculé. Ceci fournit donc une approximation des facettes visibles depuis la nouvelle position (Figure 3.10).

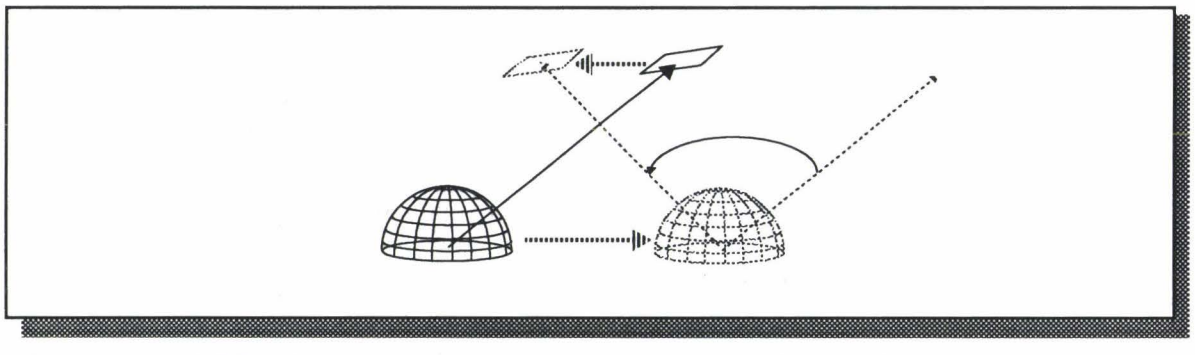


Figure 3.10 ***Schématisation des propriétés de cohérence entre facettes émettrices voisines.***

Différents problèmes surgissent néanmoins lors de cette approximation, tels que l'apparition de proxels non recouverts ou la nécessité de gérer l'apparition de nouveaux objets qui n'étaient pas présents sur l'hémisphère précédemment calculé. Pour résoudre le premier problème, un nouveau rayon est tracé à travers les proxels vides.

Le second point est quant à lui plus délicat à traiter, puisqu'il nécessite de détecter les objets qui peuvent devenir visible après un déplacement de l'hémisphère (Figure 3.11.a). L'apparition d'un nouvel objet, dû au déplacement de l'hémisphère, ne peut se faire que sur le bord des polygones projetés, sous réserve que le déplacement entre l'hémisphère calculé et l'hémisphère à approximer soit suffisamment faible. Pour éviter de rechercher les proxels recouverts par les bords de polygone après projection, les auteurs proposent de mémoriser des informations supplémentaires en chaque proxel, permettant de tenir compte du point d'impact sur un polygone visible avant projection. Ainsi, lorsqu'un rayon est lancé à travers un proxel, et que le point d'intersection avec le polygone visible à travers ce proxel est proche de l'une des arêtes du polygone, le trajet du rayon est poursuivi. Les polygones qui seraient visibles dans cette direction sont alors calculés, et mémorisés dans le proxel. En fait, le premier polygone rencontré suffit, sous réserve que le point d'impact ne se trouve pas trop près de l'une de ses arêtes. Le principe de cet algorithme est schématisé sur la Figure 3.11.b.

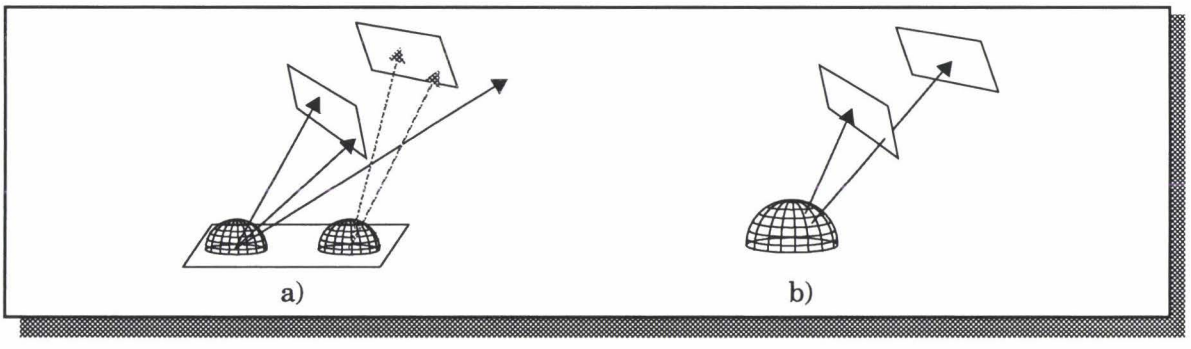


Figure 3.11 ***Le problème des objets devenant visible après déplacement de l'hémisphère (a) et prolongation des rayons pour les intersections sur les bords de polygones (b)***

Enfin, le choix des facettes émettrices, sur lesquelles peut être appliquée l'approximation, doit être fait avec précaution, de manière à minimiser au maximum les erreurs découlant de cette approximation. En effet, celle-ci n'est valable que pour des petits déplacements de la surface d'échantillonnage, des déplacements éloignés générant des modifications trop importantes des informations de visibilité. Les auteurs proposent donc de n'appliquer cet algorithme que sur des facettes directement voisines de la première facette de calcul. Dans ce cas, en supposant des facettes rectangulaires, huit hémisphères peuvent être approximés plutôt que directement calculés.

Les résultats obtenus dans [Telli 91] montrent un gain variant de 3 à 5 pour différentes scènes de test. Notons enfin que ce principe est applicable à une surface de projection quelconque, telle que l'hémicube.

3.4 Echantillonnage d'une source à l'aide du tracé de faisceaux

Nous expliquons, dans ce paragraphe, une méthode sur laquelle nous travaillons pour évaluer les facteurs de forme entre un sommet et une source, en utilisant le tracé de faisceaux, plutôt que le tracé de rayons utilisé par Wallace. Nous présentons , dans un premier temps, le principe du tracé de faisceaux, et les domaines dans lesquels il a été précédemment employé. Nous détaillons ensuite la façon dont nous envisageons son utilisation dans le cadre de la radiosité, en précisant ses avantages et inconvénients par rapport au tracé de rayons. Nous présentons enfin quelques résultats obtenus avec cette approche, et les perspectives de recherche qu'elle offre.

3.4.1 Principe du tracé de faisceaux

Les premiers travaux utilisant un faisceaux plutôt qu'un rayon unique (ou un groupe de rayons) ont été effectués par Amanatides [Amana 84] et Heckbert [Heckb 84]. Leurs principales motivations sont de réduire l'aliassage inhérent à l'échantillonnage ponctuel effectué avec

l'algorithme du lancer de rayons, et de profiter de la cohérence qui existe entre les rayons passant par le même pixel (suréchantillonnage) ou par des pixels voisins.

Amanatides utilise un faisceau conique, issu de l'oeil de l'observateur, dont l'ouverture correspond à la taille du pixel par lequel il passe. Lorsqu'un objet est rencontré, des cônes d'ombrage, de réflexion et de transmission sont calculés et envoyés dans la scène, selon le même principe que le lancer de rayons (Figure 3.12.a).

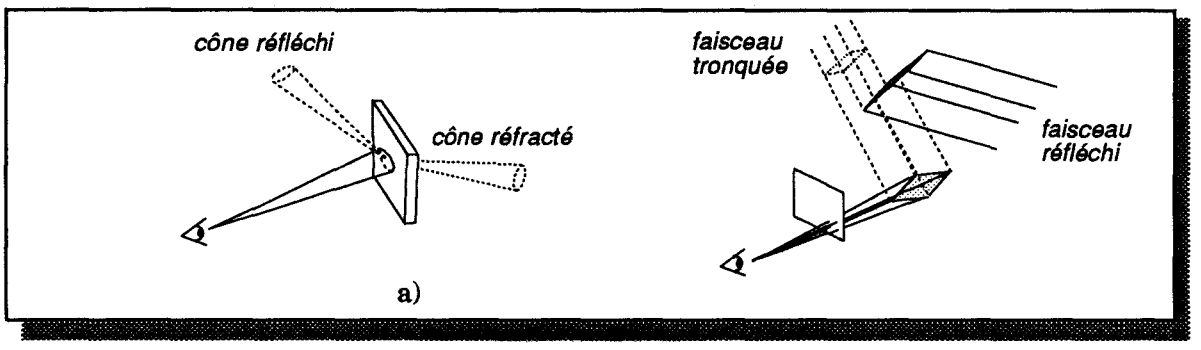


Figure 3.12 ***Principe du lancer de cône (a) et de faisceaux polygonal (b)***

De manière à pouvoir traiter correctement le problème de l'aliassage, lorsque l'objet rencontré ne recouvre pas entièrement la section du cône, celui-ci est prolongé afin de constituer une liste des objets qui y sont visibles, la taille maximale de la liste étant fixée à huit objets. Lors du calcul de l'intensité qui parvient en un point, tous les objets présents dans la liste sont considérés, au prorata de la surface de la section du cône qu'ils couvrent réellement.

Amanatides expose différentes méthodes permettant, d'une part, d'éliminer rapidement les objets qui ne peuvent pas avoir d'intersection avec un cône, et d'autre part, de calculer l'intersection d'un cône avec une sphère ou un polygone.

L'algorithme de Heckbert utilise des faisceaux de section polygonale. Pour chaque faisceau, les coordonnées des objets (des polygones) de la scène sont transformées dans le système de coordonnées du faisceau. Un tri en profondeur est ensuite appliqué sur les objets, puis ils sont testés les uns après les autres, afin de déterminer s'ils se trouvent ou pas à l'intérieur du faisceau. Lorsqu'un polygone est reconnu comme étant à l'intérieur du faisceau, il est mémorisé dans une liste des polygones visibles. Le faisceau est ensuite modifié, de manière à en retirer le contour du polygone précédemment rencontré. Le traitement du faisceau se poursuit alors jusqu'à ce qu'il devienne égal à un faisceau de section vide (la section est entièrement recouverte par les polygones qui se trouvent dans la liste des polygones visibles), ou qu'il n'existe plus aucun objet à traiter. Des faisceaux réfléchis (réfractés) sont alors envoyés, la section d'un faisceau réfléchi ayant la même forme que celle de l'intersection du faisceau père avec le polygone intersecté. Notons que cet algorithme est limité au traitement des surfaces polygonale, et qu'il génère un coût de calcul important pour la détermination du contour d'intersection exact d'une surface avec le faisceau.

Plus récemment, Ghazanfarpour [Ghaza 88] a proposé un algorithme basé sur la même notion, et utilisant des faisceaux pyramidaux (section carrée). Le but de cet algorithme est exclusivement d'améliorer la qualité des images, en réduisant l'aliassage, sans utiliser de suréchantillonnage. Lorsque le faisceau rencontre un objet, et que cet objet ne couvre pas totalement la section du faisceau, celui-ci est subdivisé récursivement en 4 sous-faisceaux. Le même traitement de recherche d'intersection est alors appliqué récursivement sur chaque sous-faisceau, de même que sur les faisceaux réfléchis et réfractés. L'avantage de cette approche réside dans le fait que la "section" du faisceau ne varie jamais, simplifiant ainsi le traitement par rapport à l'approche de Heckbert. D'autre part, du fait que le contour exact d'une intersection avec un objet n'est pas calculée explicitement, mais approximée par subdivisions successives, cette approche n'est pas limitée au traitement des polygones.

3.4.2 Utilisation dans le cadre de la radiosité

L'un des principaux problèmes qui apparait, lors de l'utilisation du tracé de rayons pour l'évaluation des facteurs de forme entre un sommet et une source, est de déterminer le nombre exact de rayons à lancer afin d'obtenir une valeur précise du facteur de forme. Wallace montre que ce nombre peut être élevé et varier selon différents paramètres, tels que la distance de la source au sommet, la taille de la source ou son orientation vis-à-vis du sommet. Le temps de calcul du facteur de forme entre un sommet et une source est bien entendu proportionnel au nombre de rayons lancés. Nous proposons donc de remplacer les rayons utilisés pour l'échantillonnage de la source, par un faisceau pyramidal de sommet le point de la scène pour lequel le calcul de facteurs de forme doit être effectué, et de base le contour de la source.

Principe

L'objectif de cette approche est d'obtenir un facteur de forme précis, quels que soient le sommet et la source considérés, en obtenant le contour exact de la source telle qu'elle est visible depuis le sommet. L'obtention de ce contour permet alors d'effectuer un calcul analytique de facteur de forme, plus précis que l'échantillonnage appliqué lors de l'utilisation du tracé de rayons. Nous nous limitons, dans un premier temps, à des scènes constituées uniquement de polygones convexes.

Le principe de notre approche est alors le suivant : pour chaque sommet de la scène, un faisceau est construit, en générant les plans de son contour. Les différentes surfaces de la scène sont ensuites testées par rapport à ces plans, de manière à éliminer les surfaces qui sont complètement à l'extérieur du faisceau (Figure 3.13.a). Ceci est effectué en appliquant l'algorithme de codage de Hodgman et Sutherland [Hodgm 74], permettant de spécifier la position de chaque surface par rapport à chaque plan formant le contour du faisceau.

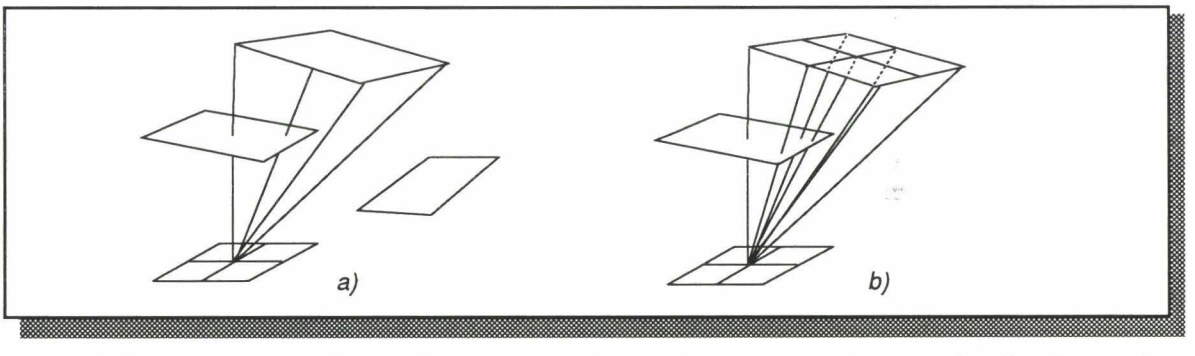


Figure 3.13 ***Elimination des surfaces externes (a) et subdivision pour l'approximation des surfaces internes (b)***

Les surfaces qui se trouvent à l'intérieur du faisceau, ou celles pour lesquelles il n'est pas possible de conclure immédiatement, sont ajoutées à une liste de candidats, chacun de ces candidats pouvant potentiellement masquer une partie de la source. Cette liste est triée, au fur et à mesure que des surfaces y sont ajoutées, en fonction de l'éloignement des surfaces par rapport au sommet du faisceau.

Lorsque toutes les surfaces ont été testées, deux cas se présentent :

- soit la liste des candidats est vide, ce qui signifie que la source est entièrement visible depuis le sommet d'application, et le calcul du facteur de forme peut être effectué, en appliquant la formule analytique décrite au paragraphe 2.1, page 25, puisque le contour exact de la source est connu.
- soit la liste des candidats n'est pas vide; auquel cas il faut considérer chacun d'entre eux pour déterminer la partie de la source qui est cachée, afin de pouvoir appliquer ensuite le calcul du facteur de forme sur la partie visible.

Pour calculer la partie occultée par chaque surface, il est nécessaire d'appliquer un algorithme de fenêtrage de chaque surface de la liste par la section du faisceau. Lorsque ceci est effectué, le

contour exact de la source, telle qu'elle est visible depuis le sommet, est obtenu. Cependant, ce calcul est très coûteux, du fait de la modification du contour de fenêtrage après qu'une intersection entre le faisceau et une surface ait été calculée [Ghori 93].

Nous nous sommes donc orientés vers l'utilisation de la méthode proposée par Ghazanfarpour, où, plutôt que de calculer le contour exact de la source vue depuis un sommet, celui-ci est approximé récursivement par des faisceaux de plus en plus fin (Figure 3.13.b). Ceci génère évidemment une perte de précision, puisque le contour exact n'est pas calculé, mais permet de simplifier considérablement le processus de calcul. En effet, lorsqu'un faisceau est lancé, une intersection est recherchée avec une surface de la liste. Trois cas peuvent alors se présenter:

- la surface occulte totalement le faisceau: les calculs sont stoppés sur ce faisceau, puisque la source n'y est pas visible.
- la surface est en dehors du faisceau: auquel cas la surface est éliminée, et la surface suivante est considérée.
- autres possibilités, c'est à dire que la surface occulte partiellement la source, ou qu'il n'est pas possible de déterminer simplement sa position par rapport au faisceau: le faisceau est subdivisé et les mêmes calculs reprennent sur chaque nouveau faisceau.

Il apparait ainsi qu'aucune opération de fenêtrage n'est appliquée, et que seules des opérations simples servant à déterminer si une surface occulte totalement ou pas du tout un faisceau sont utilisées. En fait, ceci n'est pas tout à fait vrai. En effet, dans le dernier cas, le fait de ne pas savoir si une surface occulte en partie ou pas du tout un faisceau peut conduire à une subdivision très forte du faisceau, sans que cela ne fournisse d'avantage de renseigments sur l'occultation potentielle du faisceau par la surface. Les deux cas qui peuvent se présenter sont schématisés sur la Figure 3.14 ci -dessous.

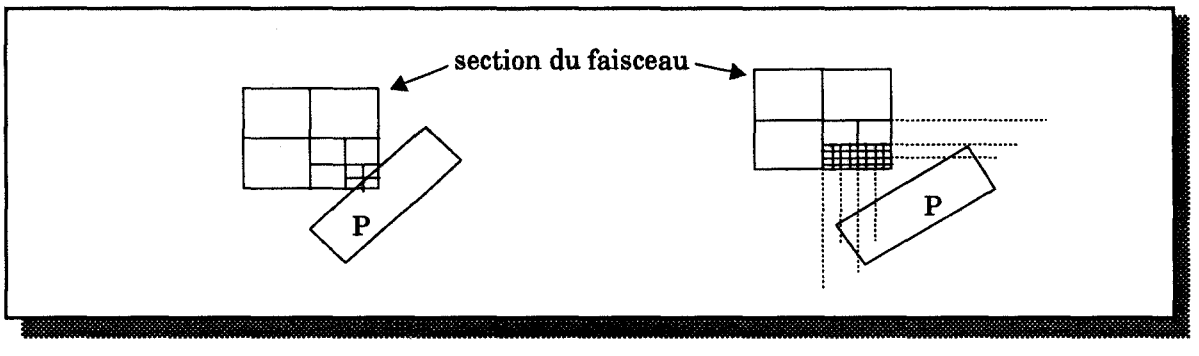


Figure 3.14 ***Deux positions générant un cas d'occultation indéterminé.***

Dans le premier cas, le polygone P coupe effectivement la fenêtre représentant la section du faisceau. Dans ce cas, la subdivision de la fenêtre est justifiée, et permet, à terme, d'obtenir une bonne approximation du contour visible. Dans le second cas, par contre, le polygone est extérieur à la fenêtre, mais le codage de la position du polygone par rapport à chaque côté de la fenêtre ne permet pas de le détecter et génère une subdivision à outrance. De plus, le même codage, appliqué à chaque nouvelle sous-fenêtre, ne permet pas non plus, pour une grande partie d'entre elles, de conclure sur la possibilité d'occultation par le polygone P.

Il est donc nécessaire d'appliquer une opération de fenêtrage, par rapport aux côtés du contour pour lesquels la position du polygone est indéterminée, destinée, non pas à obtenir le contour résultant, mais simplement à déterminer si le polygone est à l'extérieur du faisceau. Cette opération permet alors de limiter les subdivisions du faisceau aux seules surfaces qui possèdent effectivement une intersection avec le faisceau.

Lorsque toutes les surfaces de la liste des candidats ont été considérées, le facteur de forme final est calculé comme la somme des facteurs de forme associés à chaque sous-faisceau, chacun d'entre eux étant calculé analytiquement.

La récursion est stoppée lorsque celle-ci est descendue à un niveau supérieur à un seuil fixé, ou lorsque, pour un faisceau donné, la surface testée est soit totalement visible, soit totalement

invisible. Un niveau de récursion maximum de 4 (jusqu'à 256 sous-faisceaux) élimine les problèmes d'aliassage.

Quelques résultats

Cet algorithme a été implanté et appliqué sur diverses petites scènes de test, permettant de juger de la complexité temporelle de l'approche, de même que de la précision des résultats. Ces scènes sont représentées sur la Figure 3.15.

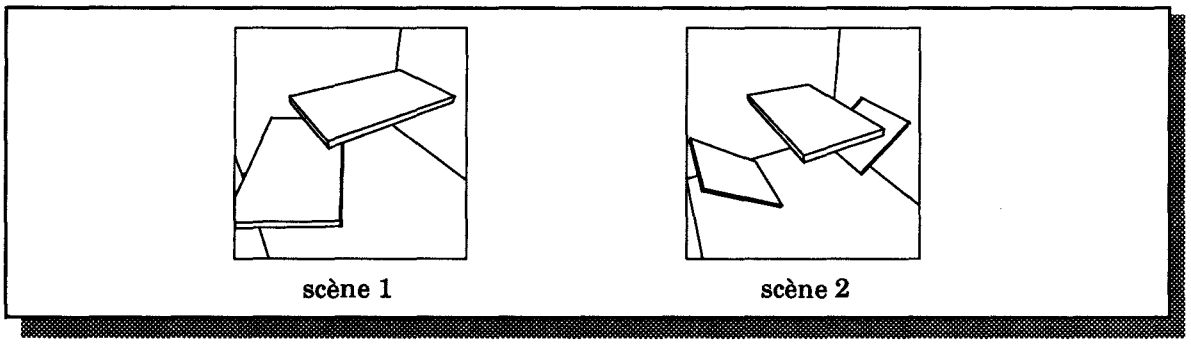


Figure 3.15 ***Scènes de test pour le tracé de faisceaux***

Comme il fallait s'y attendre, l'algorithme est évidemment plus coûteux que celui du tracé de rayons. Les chiffres présentés dans le tableau ci-dessous (Table 3.2) montrent effectivement l'importance de la différence qui existe entre les deux approches.

	Temps scène 1 (s)	Temps scène 2 (s)
tracé de rayon	0,9	1,1
tracé de faisceau	11,6	18,45

Table 3.2 ***Comparaison des temps de calculs entre le tracé de rayon et le tracé de faisceaux***

Il convient cependant de noter que, dans le cas du tracé de rayons, un seul rayon a été lancé vers la source, ce nombre étant évidemment insuffisant pour obtenir la précision requise pour le calcul des facteurs de forme. Ceci permet cependant de quantifier le rapport, en terme de coût de calcul, existant entre les 2 approches, le rapport moyen étant d'environ 17 dans le plus mauvais des deux cas (le tracé de 17 rayons requiert le même temps que le tracé d'un faisceau).

Une analyse de l'algorithme a permis de montrer que la plus grande partie du temps est consacrée à la constitution de la liste des surfaces pouvant avoir une intersection avec le faisceau. Le rapport moyen du coût de chaque étape de l'algorithme est présenté dans le tableau suivant (Table 3.3).

phase de calcul	proportion du temps de calcul global
constitution de la liste des candidats	71,8%
gestion des récursions	28,1%
Calcul du facteur de forme	0,1%

Table 3.3 ***Rapport du coût de chaque étape***

Notons que le nombre moyen de subdivisions est d'environ 0,7 pour la première scène et 1,2 pour la seconde. Ces valeurs peu élevées correspondent au fait que, pour la plupart des sommets considérés, la source est entièrement visible ou entièrement invisible. De ce fait, il est possible d'éliminer l'ensemble des surfaces très rapidement, ou d'arrêter le calcul dès que l'une d'entre elles est trouvée totalement occultante.

Les résultats visuels, qui figurent sur les deux images ci-dessous, permettent de juger de l'intérêt de cette technique.

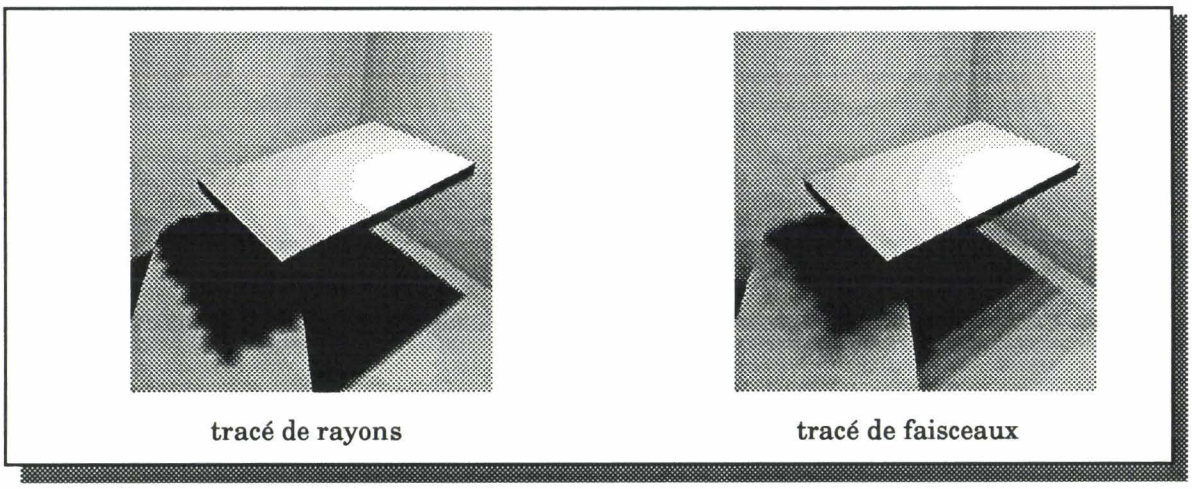


Figure 3.16 ***Comparaison du tracé de rayons et du tracé de faisceaux***

Les calculs utilisant l'algorithme du tracé de rayons ont utilisé un seul rayon pour échantillonner la source, de même qu'un unique faisceau a été tracé dans le cas du second algorithme. Il apparaît très nettement que le tracé d'un rayon, du fait qu'il ne fournit qu'une réponse binaire au test de visibilité de la source, offre un éclairage trop prononcé. Il n'existe ainsi quasiment aucune pénombre. Dans le cas du tracé de faisceaux, par contre, l'ombre de la planche supérieure sur le sol présente une pénombre correctement traitée, de même que cette pénombre est présente sur l'ombre dessinée sur la seconde planche.

L'obtention d'une image de qualité équivalente à celle fournie par le tracé de faisceaux, a nécessité l'utilisation de 25 rayons pour l'échantillonnage de la source. La différence de temps de calcul entre le tracé d'un rayon et celui d'un faisceau est quant à lui resté dans un rapport d'environ 20.

L'effet de crênelage qui apparaît sur la planche inférieur provient du fait que le maillage n'est pas aligné avec les axes principaux de la scène, et qu'aucune subdivision adaptative n'est appliquée pour corriger ces défauts.

Perspectives

La comparaison des temps de calcul obtenus entre le tracé de faisceaux et le tracé de rayons montre que ce dernier est beaucoup plus efficace dans le cas où peu de rayons sont nécessaires. Par contre dès que ce nombre augmente, il semble plus intéressant d'utiliser le tracé de faisceaux. En fait, les deux algorithmes peuvent être associés pour obtenir un maximum d'efficacité en terme de temps de calcul et de précision des échanges lumineux. En effet, lorsqu'une grande précision est demandée lors du calcul des facteurs de forme, un grand nombre de rayons doivent être lancés. Cela se produit essentiellement dans le cas où la surface émettrice possède une forte radiosité latente (source ou source secondaire). Dans ce cas, le tracé de faisceau semble tout indiqué, puisqu'il permet d'obtenir une grande précision, pour un coût de calcul identique, si ce n'est moins élevé. Dans tous les autres cas, le nombre de rayons est en général faible, variant entre 1 et 4, puisque le besoin de précision n'est pas aussi important lorsque la radiosité latente à distribuer est peu élevée. Le tracé de rayons est alors à préférer au tracé de faisceaux.

Un autre cas où le tracé de faisceaux peut être envisagé concerne les sommets très proches de la surface émettrice. Wallace a montré que, dans ce cas, le nombre de rayons à utiliser est élevé, et leur distribution nécessite une découpe récursive de la source. Le tracé de faisceaux permet alors de prendre en compte facilement l'ensemble de la surface émettrice, en ne nécessitant, a priori, que peu de subdivisions, puisque le nombre d'objets se trouvant entre les deux surfaces est nécessairement réduit dans ce cas.

Nous avons fait l'hypothèse, dans les paragraphes qui précédent, que les calculs d'intersection avec les rayons ou avec les faisceaux étaient appliqués sans structure d'accélération. Dans la mesure où le tracé de rayons peut bénéficier de nombreuses techniques d'optimisation, il est nécessaire que celles-ci soient applicables, avec le même gain, au tracé de faisceaux, afin que les rapports obtenus précédemment soient conservés.

Bien que l'application des techniques d'accélération, détaillées au paragraphe 3.3, au tracé de faisceaux n'ait pas encore fait l'objet de nos travaux, il nous semble qu'une grande partie d'entre elles puissent lui être appliquées :

- Le principe de la cohérence entre sommets voisins est très facilement applicable au tracé de faisceaux, puisque toute surface qui occulte totalement un faisceau donné conserve une grande probabilité d'occulter un faisceau issu d'un sommet voisin.
- La notion de tunnel de visibilité, introduite par Haines, semble tout aussi applicable au tracé de faisceaux, puisque les faisceaux se dirigent tous vers la même surface émettrice.
- Les volumes englobants sont eux-aussi applicables aux faisceaux, que ce soient les volumes sphèriques ou parallélépipédiques, bien que l'intersection d'un faisceau et d'une sphère semble plus simple à déterminer (rappelons que la seule information nécessaire n'est pas l'intersection en elle-même, mais simplement le fait de savoir si elle existe).
- Enfin, la notion de subdivision spatiale semble plus délicate à effectuer, car elle nécessite de pouvoir suivre le trajet du faisceau complet au travers d'une grille de voxels.

L'application de ces différentes techniques au tracé de faisceaux est actuellement à l'étude. D'autre part, nous étudions aussi la possibilité de calculer le contour exact de la source telle qu'elle est visible depuis un sommet. Si cette solution a été écartée dans un premier temps, il nous semble à présent, d'après les premiers résultats obtenus en utilisant le principe de la subdivision, qu'il serait moins coûteux de l'utiliser. En effet, le principe de la subdivision du faisceau nécessite l'appel à une fonction de fenêtrage partiel, qui serait elle aussi utilisée dans l'algorithme de calcul du contour exact.

Enfin, notons que cette première étude se limite à l'utilisation de surfaces polygonales convexes. Une étude complémentaire doit être effectuée, afin de pouvoir déterminer l'existence d'une intersection entre un faisceau et des primitives de plus haut niveau (quadriques, splines ...) dans le cas de l'algorithme par subdivision, ou de calculer explicitement cette intersection dans le cas de l'algorithme générant le contour exact.

3.5 Conclusion

Nous avons détaillé deux classes d'algorithmes utilisant la technique du tracé de rayons pour effectuer l'élimination des parties cachées et le calcul des facteurs de forme.

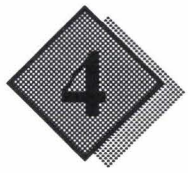
Les algorithmes de la première classe permettent de calculer les valeurs de radiosité directement aux endroits où ils seront nécessaires pour la phase d'affichage (les sommets), en offrant d'autre part la possibilité de faire varier, pour chaque sommet, le nombre de rayons d'échantillonnage utilisés.

Le seconde classe d'algorithmes présentée ressemble en de nombreux points aux techniques projectives et a les mêmes désavantages, en particulier les problèmes d'aliassage et de perte des petits objets. Néanmoins, la structuration sous forme paramétrique de la base de données permet un gain intéressant dans le calcul des intersections (par rapport au nombre total de facettes).

Ces algorithmes bénéficient de la précision du calcul d'intersection offerte par la technique du tracé de rayons, avec des primitives de plus haut niveau que les facettes planes. Ils subissent néanmoins le coût de calcul élevé de ces mêmes techniques. Différentes méthodes d'optimisation envisageables ont été exposées, et offrent un gain substantiel.

Enfin, pour remédier aux problèmes de précision qui apparaissent dans le cas du tracé de rayons depuis les sommets des facettes, nous avons introduit une technique basée sur le tracé de faisceaux. Si les résultats visuels montrent l'intérêt d'utiliser cette approche, il est important de pouvoir lui appliquer des techniques d'optimisation performantes, de manière à la rendre effectivement utilisable, avec des temps de calcul raisonnables. Nous envisageons un algorithme hybride, utilisant le tracé de faisceaux lorsque le besoin de précision se fait sentir, et le tracé de rayons lorsque cette précision peut être moindre.

CHAPITRE



Parallélisme et architectures parallèles

Les algorithmes que nous avons présentés au cours des chapitres précédents ont tous en commun des coûts de calcul élevés, et, par conséquent, des temps d'attente très importants pour l'utilisateur. Il existe deux moyens de diminuer les temps de calcul : utiliser des processeurs disposant de performances plus élevées, ou utiliser plusieurs processeurs simultanément pour réaliser le traitement complet des algorithmes. La première solution est tributaire des progrès techniques dans les domaines de l'intégration des circuits ou de leur complexification. La seconde permet, en utilisant des processeurs moins performants, d'accroître considérablement les performances des algorithmes.

Nous nous proposons, dans ce chapitre, d'introduire différentes notions concernant le parallélisme, telles que les types de parallélisme ou les types d'architectures parallèles existants. Les architectures parallèles sont ainsi classées par rapport à deux critères, qui sont leur mode de fonctionnement et le mode d'organisation de leur mémoire. Nous présenterons enfin deux architectures parallèles que nous avons utilisées pour nos travaux: un processeur à mémoire distribuée (le Transputer) et un calculateur massivement parallèle (la Maspar).

4.1 Les types de parallélisme

Il y a parallélisme dès que plusieurs processeurs traitent en commun un même problème, celui-ci étant alors découpé en plusieurs processus. Trois organisations différentes des processus sont décrites dans ce paragraphe, selon le type de parallélisme qui leur est appliqué.

4.1.1 Le parallélisme de contrôle

Le parallélisme de contrôle est basé sur la constatation qu'une application informatique est constituée d'un ensemble d'actions exécutables simultanément. De ce fait, il semble intéressant de pouvoir exécuter ces actions, appelées tâches ou processus, sur des processeurs différents, afin d'obtenir un gain de temps dans l'exécution de l'application complète. Dans le cas idéal, le gain est linéaire s'il y a autant de processeurs que de tâches à exécuter.

Dans le cas général, cependant, le fonctionnement des différentes tâches est lié à une certaine notion de coopération, provenant de la dépendance plus ou moins importante qui existe entre elles. Deux cas de dépendance peuvent se présenter :

- la dépendance de contrôle de séquence : celle-ci correspond au séquencement des tâches dans l'algorithme. Certaines d'entre elles ne peuvent être exécutées que lorsque d'autres ont été traitées, car elles ont besoin des résultats obtenus précédemment pour pouvoir fonctionner.
- la dépendance de contrôle de communication: cette dépendance s'exprime au travers de la nécessité d'échanger des informations entre les tâches qui s'exécutent parallèlemment. Bien que les tâches puissent être lancées simultanément, la poursuite de leur exécution peut être conditionnée par certains résultats obtenus par d'autres tâches, ou certaines données présentes uniquement sur d'autres processeurs que celui où elle s'exécute.

Ces dépendances vont générer des contraintes sur l'affectation des ressources disponibles aux tâches à exécuter, puisque celles-ci seront plus ou moins difficiles à utiliser efficacement, selon la complexité du graphe de dépendance liant les tâches.

Tout le problème du parallélisme de contrôle est alors de gérer les dépendances entre les actions à effectuer, afin d'obtenir une utilisation optimale des ressources disponibles.

4.1.2 Le parallélisme de données

Certaines applications reposent sur l'exécution d'un même groupe d'actions sur un grand nombre de données. Il peut donc être intéressant d'appliquer chaque action simultanément sur plusieurs données distinctes. La parallélisation consiste alors à associer les ressources de calcul aux données plutôt qu'aux actions, chaque processeur recevant une partie des données à traiter. Les actions à effectuer sont envoyées successivement aux processeurs, qui les exécutent sur leurs propres données. Ces données sont alors organisées sous forme de structures régulières (vecteur, matrices, ...), qui correspondent fortement à ce type de parallélisme.

Il est alors important que les actions envoyées à chaque processeur soient de complexité équivalente, afin de ne pas générer de déséquilibre de charge.

4.1.3 Le parallélisme de flux

Le parallélisme de flux est directement inspiré de la notion de travail à la chaîne : en entrée, un flux de données est disponible, sur lequel il faut appliquer successivement un certain nombre d'opérations. Les ressources disponibles sont alors allouées aux processus exécutant ces actions, et les données transitent successivement par chacun des étages du pipeline ainsi constitué. Une donnée D traitée par le processus P à l'instant T sera ainsi traitée par le processus P+1 à l'instant T+1, tandis que le processus P s'occupera de la donnée D+1.

Dans le cadre de données de type vectorielles, connues au départ de l'application, l'approche est comparable au parallélisme de données décrit ci-dessus. La différence provient simplement de la façon de traiter les données par rapport aux actions : dans le mode parallélisme de données, les données sont fixes et les différentes opérations leur sont successivement appliquées; dans le mode parallélisme de flux, ce sont les données qui transitent de processeur en processeur, pour y être utilisées par les différents processus. Le principal problème est ici de découper l'algorithme en tâches successives de complexité équivalente, de manière à éviter que l'étage le plus lent ne ralentisse par trop le débit du pipeline.

Dans le cas où les données ne sont pas connues au départ, cette approche est typique d'un environnement "temps réel", où des données saisies par un dispositif d'entrée sont envoyées dans le pipeline pour y être traitées. En plus du problème d'équilibrage des différents étages du pipeline, il est ici nécessaire de pouvoir l'alimenter en continu, afin de ne pas rompre le déroulement du cycle.

4.2 Architectures parallèles

Plusieurs classifications des architectures parallèles sont envisageables, selon le point de vue utilisé. Dans ce paragraphe, nous effectuerons une classification selon le type de parallélisme employé, en utilisant une classification proposée par Flynn. Nous détaillerons ensuite chacune des catégories.

4.2.1 Classification de Flynn

Une classification des architectures des machines parallèles a été proposée, à la fin des années 60, par Flynn [Flynn 72]. Elle est basée sur l'analyse des flots de données et des flots de contrôle, et permet de distinguer 4 classes, selon que chacun de ces flots est simple ou multiple. Ces classes sont représentées dans le tableau ci-dessous (Table 4.1).

		Flot de données	
		Simple	**Multiple**
Flot de contrôle	**Simple**	**SISD**	**SIMD**
	Multiple	**MISD**	**MIMD**

Table 4.1 *Classification de Flynn*

Dans cette classification, les machines de type **SISD** (*Single Instruction stream Single Data stream*) représentent l'ordinateur séquentiel traditionnel, fonctionnant selon le modèle de Von Neumann, où une seule instruction à la fois est exécutée sur une seule donnée.

Les machines de type **SIMD** (*Single Instruction stream Multiple Data stream*) correspondent à un mode de fonctionnement où chaque processeur exécute simultanément la même instruction, mais sur des données différentes. Les différentes unités de traitement fonctionnent selon un mode synchrone, et sont contrôlées de manière centralisée.

Dans le cas du mode de fonctionnement **MIMD** (*Multiple Instruction stream Multiple Data stream*), chaque processeur travaille indépendamment des autres, sur des données différentes. Les différents processeurs fonctionnent de manière asynchrone, et possèdent leur propre unité de contrôle.

Enfin, les machines utilisant un mode **MISD** (*Multiple Instruction stream Single Data stream*) correspondent aux machines pipeline, où chaque processeur applique des opérations différentes sur un même ensemble de données. Chaque unité possède un contrôle local, permettant un fonctionnement asynchrone, mais la structure pipeline nécessite une resynchronisation des processeurs pour le traitement successif des différentes données.

4.2.2 Granularité

Il est aussi possible de caractériser les machines parallèles en fonction de leur degré de parallélisme, appelé aussi *granularité*. Trois types de grain de parallélisme sont généralement proposés:

- les machines à *gros grain*, ou à faible parallélisme, caractérisées par un nombre restreint de processeurs (inférieur à 50).
- les machines à *grain moyen*, pour lesquelles le nombre de processeurs varie entre 50 et 200.
- et les machines à parallélisme massif, dites à *grain fin*, dont le nombre de processeurs est supérieur à 200.

Cette notion de granularité peut aussi être utilisée pour caractériser le type de parallélisme applicable à un algorithme. Le parallélisme est dit à grain fin lorsqu'il est exploité au niveau des instructions élémentaires (addition, par exemple) à appliquer au sein d'un algorithme. Il est dit à grain fort lorsque le parallélisme est exploité au niveau des différents processus qui composent une application.

4.3 Schémas d'organisation de la mémoire

La conception de machines parallèles soulève un certain nombre de problèmes, dont celui de l'organisation de la mémoire. En effet, comme pour les machines séquentielles, chaque processeur doit avoir accès à la mémoire pour pouvoir fonctionner correctement. Deux schémas

d'organisation sont alors envisageables, selon que tous les processeurs partagent un accès à une mémoire commune, où qu'ils disposent chacun de leur propre mémoire. Ces deux modes d'organisation sont décrits ci-dessous.

4.3.1 Mémoire partagée

Dans ce type d'architecture, la mémoire est partagée par tous les processeurs, et est perçue par chacun d'entre eux comme un unique espace d'adressage. Les processeurs sont reliés à la mémoire centrale par l'intermédiaire d'un bus unique (Figure 4.1.a). Puisque la mémoire est unique, son accès est sérialisé, c'est à dire qu'un seul processeur à la fois peut y avoir accès. Ce fonctionnement nécessite donc un bus fournissant un haut débit de communication, de manière à éviter les problèmes de contention d'accès à la mémoire.

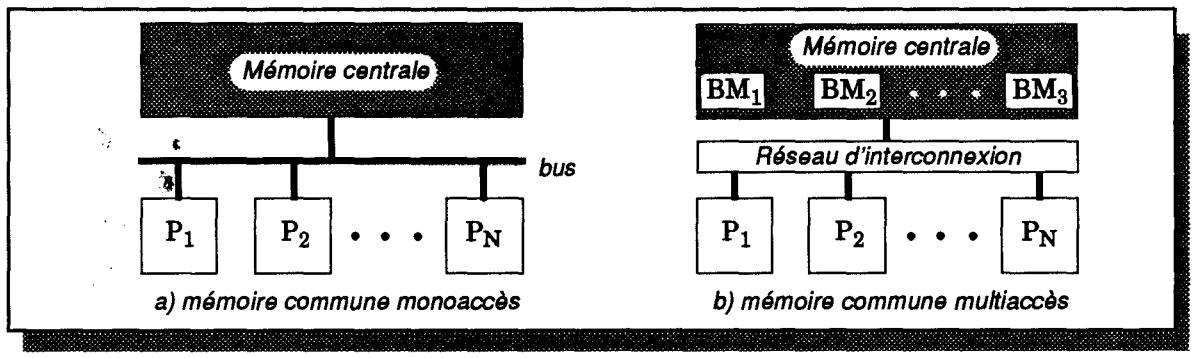


Figure 4.1 ***Architectures à mémoire partagée***

Les problèmes de contention ne peuvent cependant pas être évités lorsque le nombre de processeurs augmente. Pour limiter les conflits d'accès, tout en conservant le même type d'approche, des solutions de découpage physique de la mémoire ont été proposées. La mémoire est découpée en plusieurs bancs mémoire distincts, auxquels les processeurs sont reliés par l'intermédiaire d'un réseau d'interconnexion (Figure 4.1.b). Celui-ci peut être statique ou dynamique (reconfigurable), et posséder des topologies variées [Sanso 91]. Bien que la mémoire soit physiquement découpée, elle reste vue, logiquement, comme un espace d'adressage unique.

Plusieurs accès mémoire simultanés deviennent donc possibles avec cette organisation, en conservant cependant la contrainte d'accès séquentiel à un banc mémoire. Pour optimiser les accès parallèles, les données de grande taille, telles que les vecteurs ou les matrices, sont entrelacées sur les différents bancs, de manière à ce que des données consécutives se trouvent sur des bancs différents. Les communications entre processeurs, sur ce type d'architecture, sont effectuées par l'intermédiaire de variables partagées, dont chaque processeur connaît l'emplacement.

4.3.2 Mémoire distribuée

La seconde organisation possible pour la mémoire est de distribuer celle-ci sur chaque processeur. Chacun d'entre eux est donc relié de manière statique à un banc mémoire différent, et ne peut accéder qu'à celui-ci (Figure 4.2).

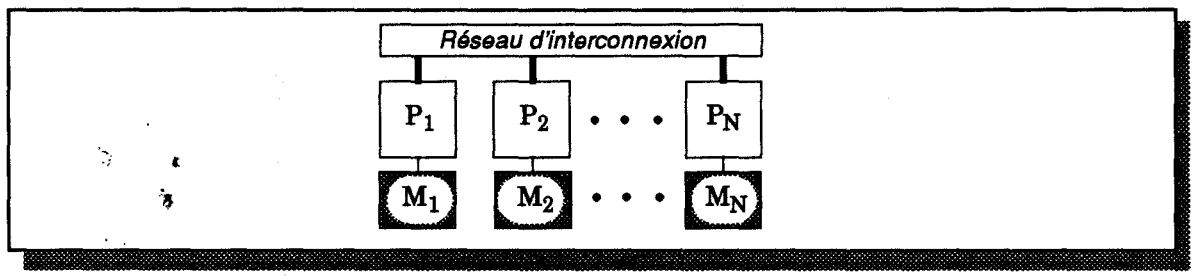


Figure 4.2 ***Architecture à mémoire distribuée***

Dans ce cas, il n'existe plus de problèmes de conflits d'accès à la mémoire, mais les échanges d'informations ne peuvent plus s'effectuer par l'intermédiaire d'une variable commune. Ces échanges s'effectuent alors par l'intermédiaire de messages, qui transitent dans un réseau d'interconnexion auquel chaque processeur est relié.

4.4 Définition de critères d'évaluation

Lorsqu'une application est exécutée en parallèle sur un nombre N de processeurs, l'idéal serait que cette exécution s'effectue N fois plus vite que si elle était exécutée sur un seul processeur. Cependant, de nombreux paramètres (temps de communications, délais d'attente, synchronisation des processus, ...) interviennent, et entraînent une perte de performance. Il est alors intéressant de disposer de critères qui permettent de juger des performances obtenues par une application parallèle. Nous détaillons, ci-dessous, divers critères couramment utilisés.

Efficacité

L'efficacité d'une implantation parallèle est définie comme le rapport entre le temps d'exécution d'un algorithme en séquentiel, et le temps d'exécution de ce même algorithme en parallèle. Elle s'exprime par:

$$Eff(n) = \frac{T(1)}{n \times T(n)}$$

avec

- n représentant le nombre de processeurs utilisés;
- $T(1)$ le temps d'exécution en séquentiel;
- $T(n)$ les temps d'exécution sur n processeurs.

L'efficacité maximale sera alors atteinte pour une valeur de 1.

Accélération

La notion d'accélération est très proche de la notion précédente d'efficacité. Elle est définie par :

$$Acc(n) = \frac{T(1)}{T(n)}$$

avec

- n représentant le nombre de processeurs utilisés;
- $T(1)$ le temps d'exécution en séquentiel;
- $T(n)$ les temps d'exécution sur n processeurs.

L'accélération maximale prendra la valeur n.

Rendement

Le rendement est défini comme le rapport entre le nombre de traitements utiles et le nombre total de traitements effectués. Il s'exprime sous la forme:

$$Rend(n) = \frac{TU(n)}{TT(n)}$$

avec

- n représentant le nombre de processeurs utilisés;
- $TU(n)$ le nombre de traitements utiles;
- $TT(n)$ le nombre de traitements total.

Le rendement maximal est atteint pour une valeur de 1.

4.5 Conclusion partielle

Nous avons présenté une analyse succincte des différents types de parallélisme, ainsi que des classifications habituellement utilisées pour les architectures parallèles. Il est difficile de pousser plus avant une telle analyse, dans la mesure où elle nécessiterait un approfondissement des différentes machines existantes, tant du point de vue architectural, que du point de vue du mode de fonctionnement.

Nous présentons, dans les deux paragraphes qui suivent, deux machines particulières, utilisant deux modes de fonctionnement radicalement différents : le Transputer, processeur dédié au traitement d'application selon un mode MIMD; la Maspar, calculateur massivement parallèle, dont le fonctionnement est assuré selon un mode SIMD.

4.6 Le Transputer

Le transputer est un processeur spécifiquement conçu pour l'exécution de programmes parallèles. Sa principale caractéristique réside dans ses 4 liens de communications bidirectionnels. Ceux-ci permettent d'effectuer des communications synchrones entre des transputers, ou entre un transputer et le monde extérieur. Leur utilisation autorise la construction de réseaux de grande dimension et de topologies variées, et offre une puissance de calcul élevée et un débit de communication important.

Nous présentons ci-dessous un élément de la famille des transputers, le T800, puis la machine Multicluster II que nous avons utilisée pour nos travaux.

4.6.1 Le T800

L'architecture des différentes versions de Transputers (T200, T400, T800) reste à peu près la même. Nous décrivons celle du T800 sur la figure ci-dessous(Figure 4.3).

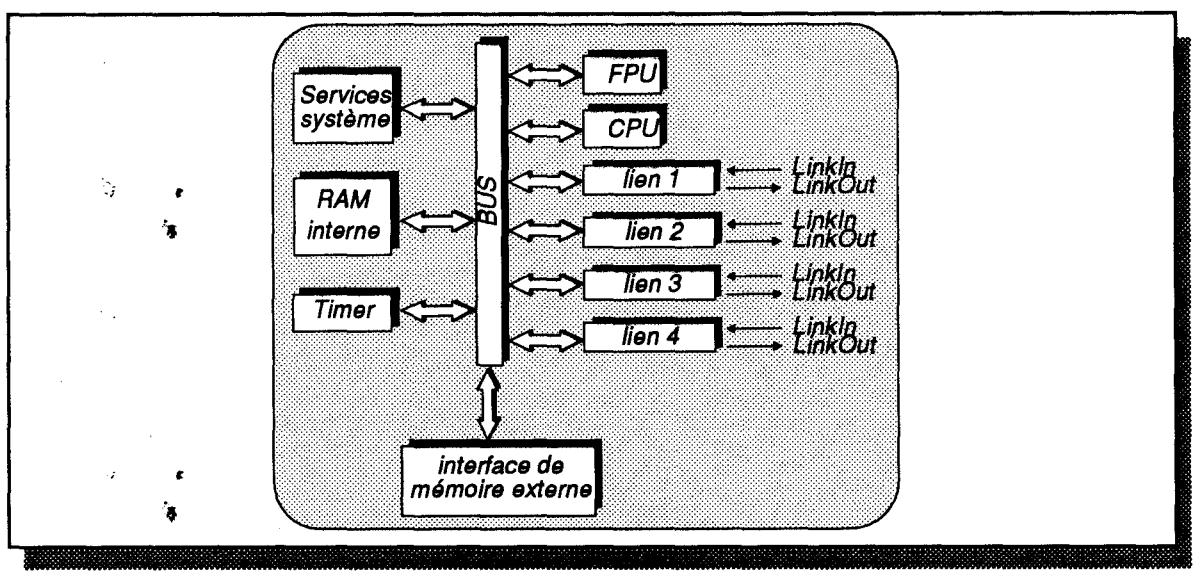


Figure 4.3 ***Architecture du T800***

Le T800 est un processeur 32 bits, fournissant une puissance de calcul moyenne de 15 MIPS (cette valeur varie selon la fréquence d'horloge utilisée), auquel a été adjointe une unité de calculs flottants, travaillant sur 64 bits, et fournissant une puissance d'environ 2.5 MFlops. Ces deux unités peuvent travailler en parallèle.

Il est équipé d'une mémoire interne de 4 Ko, dont l'accès s'effectue en un cycle d'horloge. Ce temps d'accès est à comparer aux 3 cycles nécessaires pour accéder à la mémoire externe, d'une

capacité maximale de 4 Go. Mémoire interne et mémoire externe font parties d'un même espace d'adressage linéaire.

La vitesse standard de transmission supportée par tous les transputers est de 10 MBits par seconde, le T800 pouvant fonctionner aussi aux vitesses de 5 et 20 Mbits par seconde. L'utilisation d'une vitesse standard, ainsi que l'indépendance de fonctionnement entre les liens et l'unité de calcul, permettent la connexion de transputers de familles différentes au sein d'un même réseau.

Une autre caractéristique du transputer est que le mécanisme de répartition des tâches (*scheduler*) a été intégré au sein du micro-code du processeur, ce qui permet d'effectuer des changements de contexte très rapides entre différents processus fonctionnant simultanément sur le même processeur (ce temps est inférieur à la micro-seconde).

4.6.2 Le Multicluster II

Cette machine est équipée de 32 transputers T800, disposant chacun de 2 Mo de mémoire RAM. Ces transputers sont connectés à deux crossbars (voir Figure 4.4), qui permettent de reconfigurer entièrement le réseau selon la topologie désirée.

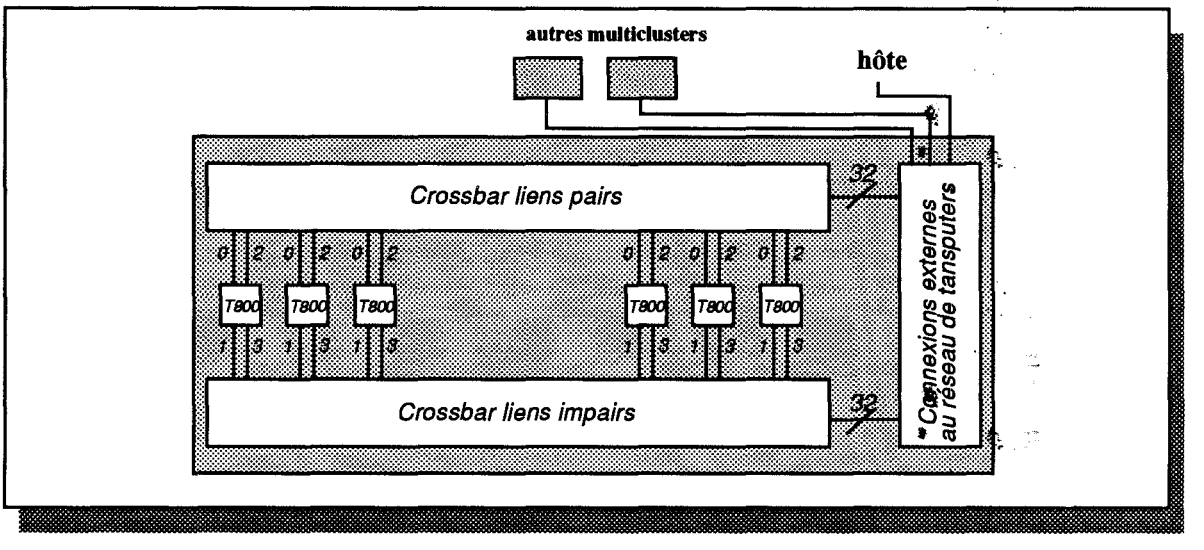


Figure 4.4 ***Aperçu de l'architecture du Multicluster II***

La machine est dotée de 8 transputers supplémentaires, servant de points d'entrée sur le réseau. Huit utilisateurs peuvent donc travailler simultanément sur la machine, et se partager les ressources : chaque utilisateur réserve les processeurs dont il a besoin tout au long de sa connexion, et ceux-ci ne seront à nouveau accessibles à un autre utilisateur qu'après les avoir libérés. Il est d'autre part possible de connecter d'autres Multiclusters au réseau, de manière à étendre les performances du système.

4.6.3 Le système Helios

Helios est un système d'exploitation distribué, développé pour les architectures à base de transputers [Helios 89]. L'interface et les différents outils utilisables sous Helios sont similaires à ceux offerts par le système d'exploitation UNIX.

Helios permet d'introduire une certaine souplesse au niveau de la gestion et de l'utilisation d'un réseau. La distribution de la couche système sur l'ensemble des noeuds du réseau lève la contrainte des 4 liens physiques, en introduisant la notion de liens virtuels, gérés par le système. De même, l'utilisateur est libre de placer explicitement ses processus sur le réseau ou de laisser le système se charger de ce placement, en fonction de la charge de chaque processeur.

Un compilateur C, auquel est ajouté un certain nombre de librairies de communication de plus ou moins haut niveau, est disponible sous Helios. Dans la mesure où il n'existe pas de véritable extension du langage permettant de gérer le parallélisme, une application Helios sera constituée de tâches fonctionnant en parallèle (sur le même processeur ou sur des processeurs différents), pouvant communiquer par l'intermédiaire de canaux virtuels, ou directement via les canaux physiques.

La description des tâches constituant une application, de même que les canaux logiques de communication qui existent entre elles, sont définies par l'écriture d'un fichier écrit en "langage" CDL (Component Description Langage). C'est ce fichier qui est lu lors du lancement de l'application, et qui permet au système de placer les tâches sur les processeurs adéquats, d'établir les connexions demandées et de lancer l'application.

4.7 Une machine massivement parallèle: la Maspar

Nous décrivons ci-dessous les principales caractéristiques de la machine massivement parallèle sur laquelle nos approches ont été développées. Cette machine est une Maspar MP-1, dont les différentes configurations matérielles peuvent comprendre de 1024 à 16384 processeurs (PEs). Les performances attachées à chaque configuration (puissance de calcul, quantité de mémoire, bande passante de communication) évoluent linéairement avec le nombre de processeurs. Cette machine possède un mode de fonctionnement de type SIMD.

4.7.1 Architecture de la Maspar

L'architecture générale de la Maspar [Dec 92] comprend les éléments suivants (Figure 4.5):

- Un ensemble de PEs disposés selon une grille à deux dimensions et rassemblés par groupes de 16 (sous-grilles 4x4).
- Un processeur de contrôle de la grille des PEs, appelé ACU (Array Control Unit), qui est chargé de l'exécution du programme parallèle. L'ACU est un processeur à part entière, disposant d'une mémoire locale (120 Ko), et capable d'exécuter les parties séquentielles d'un programme.
- Deux réseaux de communication permettant les communications: soit entre processeurs directement voisins (Xnet), soit entre processeurs quelconques (Global Router).
- Une machine hôte sert principalement d'interface entre l'utilisateur et l'unité parallèle (ACU et PEs),

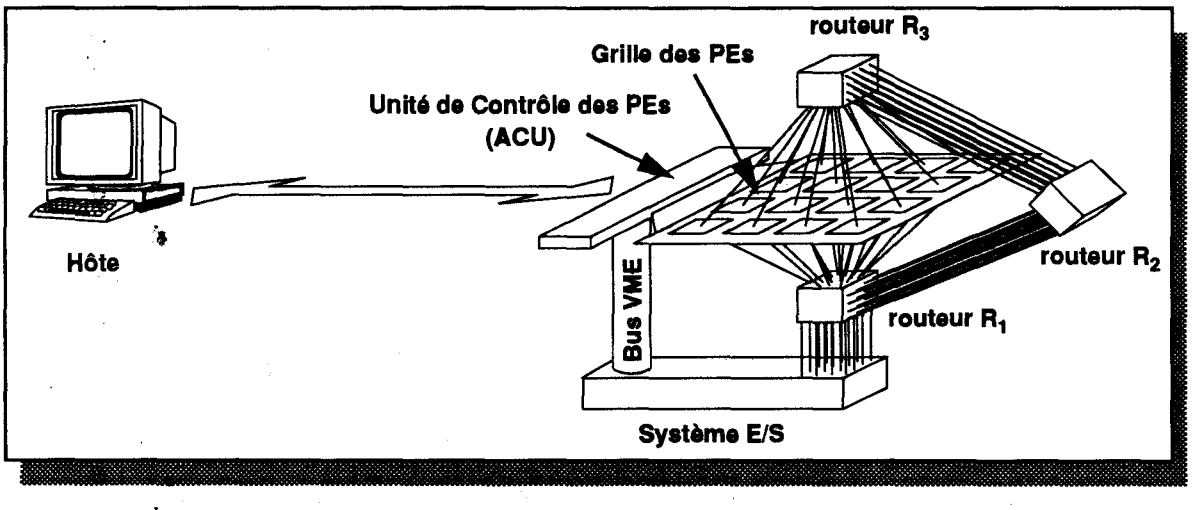


Figure 4.5 *Vue d'ensemble de l'architecture de la Maspar*

Cette architecture manipule deux types de variables: les variables scalaires et les variables multiples. Une variable scalaire n'existe qu'en un unique exemplaire dans la mémoire de l'ACU. Une variable multiple est présente dans la mémoire locale de chacun des PEs à la même adresse.

Tant qu'elles ne concernent que des données scalaires, les instructions du programme sont exécutées sur l'ACU. Lorsque l'une des données est multiple, l'ACU diffuse le micro-code des instructions à exécuter et les données éventuelles à l'ensemble des PEs. Des résultats peuvent être transférés des PEs vers l'ACU par l'intermédiaire d'un arbre de réduction auquel tous les PEs sont connectés.

Description des PEs

Chaque PE possède une ALU opérant sur des données 4 bits, 32 registres 32 bits, et une mémoire locale de 16 Ko dans la configuration actuelle. Les registres sont adressables par bit ou par octet, et sont directement accessibles au programmeur. Les PEs disposent d'un registre d'état, qui comprend, outre les drapeaux classiques, des bits utilisés lors des communications et un bit d'activité. Celui-ci permet de contrôler l'activité du processeur: les instructions ne sont exécutées que s'il est positionné à 1.

Les PEs sont regroupés par groupes de 16, configurés sous forme de grilles 4x4. Les PEs d'un même groupe partagent un accès commun au global router (voir paragraphe 4.7.2). Chaque groupe de PEs possède un bloc mémoire de 1 Mbits. Ce bloc est découpé en 16 zones différentes de 16 Ko chacune, chaque zone représentant la mémoire locale à un PE. L'accès à la mémoire est fait séquentiellement pour chaque PE d'un groupe. L'adressage sur 32 bits peut être effectué directement (la même adresse dans le bloc local de 16 Ko pour tous les PEs) ou indirectement (une adresse différente pour chaque PE).

4.7.2 Réseaux de communication

Deux types de réseaux sont prévus pour les communications entre PEs. Ceux-ci permettent, soit des communications locales entre voisins directs, soit des communications point à point entre processeurs quelconques.

Le réseau "Xnet"

Ce réseau est formé d'une grille à 2 dimensions qui permet de relier chaque PE à ses 8 voisins immédiats [Nicko 90], [Shers 93]. De manière à limiter le nombre de connexions par PE, chacun d'entre eux est relié à 4 connecteurs, chaque connecteur étant commun à 4 PEs (Figure 4.6 a). Les PEs situés sur un bord de la grille sont connectés aux PEs situés sur le bord opposé, le réseau complet formant un tore.

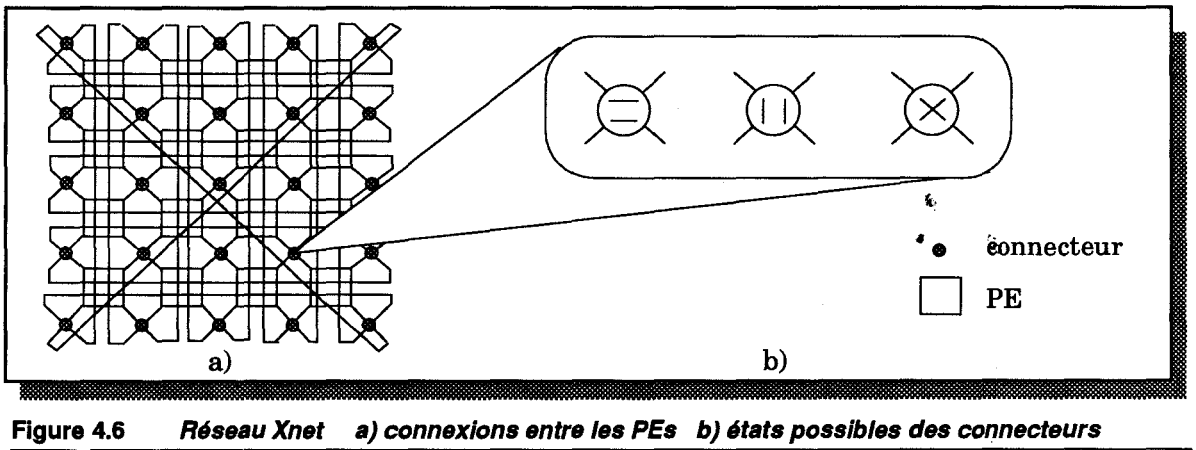


Figure 4.6 *Réseau Xnet a) connexions entre les PEs b) états possibles des connecteurs*

Chaque connecteur peut prendre trois états différents, shématisés sur la Figure 4.6 b, selon que les communications s'effectuent horizontalement, verticalement ou en diagonale. Tous les

connecteurs prennent le même état en même temps, les communications devant toutes être effectuées simultanément dans la même direction (par exemple, tous les PEs envoient une donnée vers le nord, et reçoivent donc simultanément une donnée par le sud).

Le réseau "Global Router"

Le "global router" permet d'effectuer des communications point à point entre des PEs quelconques du réseau [Nicko 90], [Blank 90]. Il utilise 3 étages de routeurs pour implémenter les fonctionnalités d'un crossbar 1024x1024 (Les PEs étant regroupés par groupes de 16, une MP-1 équipée de 16k PEs dispose de 1024 groupes. Un crossbar 1024x1024 est donc suffisant pour permettre les communications entre deux groupes de PEs). Chaque groupe possède une connexion de sortie vers le routeur R1 et une connexion d'entrée depuis le routeur R3 (Figure 4.5). Dans la mesure où ces connexions sont communes à tous les PEs d'un même groupe, les communications des PEs d'un groupe ne peuvent être effectuées que l'une après l'autre.

Un PE désirant effectuer des communications vers un ou plusieurs autres PEs transmet leur identité au routeur R1. Chaque étage du routeur détermine une connexion vers l'étage suivant, en fonction du numéro du PE destinataire. Une fois le chemin établi, la communication peut être effectuée de manière bidirectionnelle.

4.7.3 Performances

Un PE délivre une puissance de crête d'environ 2 MIPS et 40 KFLOPS en simple précision. Ces performances permettent d'obtenir, pour une machine disposant de 16384 processeurs, une puissance de crête totale d'environ 30 GIPS et 1,5 GFLOPS [Nicko 90].

La bande passante du réseau "Xnet" est supérieure à 20 Go par seconde, pour une configuration disposant de 16384 PEs, tandis que celle du "global router" ne dépasse pas 1.5 Go par seconde. Cette différence s'explique par le fait que les PEs sont regroupés par groupe de 16, chaque groupe partageant un accès unique au "global router". Plusieurs communications à partir d'un groupe, ou une communication à destination de plusieurs PEs d'un groupe, ne peuvent donc être faites que séquentiellement, multipliant d'autant le temps de communication total.

Le global router fournit donc un moyen de communication plus général que le réseau xnet, puisqu'il permet des communications entre processeurs quelconques, mais au prix d'un débit moyen plus faible.

4.7.4 Environnement de développement

L'environnement de développement comprend plusieurs langages, dont le Fortran et MPL, et divers outils graphiques de mise au point utilisables sous XWindow, tels qu'un debugger parallèle et un système de "profiling" (visualisation du coût proportionnel de chaque ligne de programme).

Le langage MPL que nous avons utilisé est basé sur le langage C. Il permet de prendre en compte le parallélisme de données utilisé sur la machine, par l'introduction d'un nouveau qualificatif de type de variable, le qualificatif *"plural"*. Celui-ci permet de représenter la même variable sur l'ensemble des processeurs de la machine, la valeur de cette variable pouvant être différente d'un processeur à l'autre. Cette variable sera alors appelée *"variable multiple"*.Par exemple, les lignes suivantes:

```
plural int i;

plural float f[3];
```

permettent de déclarer, sur chaque PE, une variable entière i et un tableau de 3 nombres réels f, chacune de ces variables étant située à la même adresse mémoire sur chaque PE. Les opérations appliquées sur ces variables multiples seront effectuées sur l'ensemble des PEs, par opposition avec les opérations sur les variables scalaires, qui ne seront exécutées que sur le processeur contrôlant la grille de PEs (ACU). L'utilisation du qualificatif *"plural"* s'étend à la notion de pointeur. La déclaration:

```
plural int *ptr_acu;
```

désigne une variable scalaire (résidant sur l'ACU) *ptr_acu* qui pointe vers une adresse de la mémoire des PEs (la même pour tous les PEs) contenant une valeur entière, tandis que la ligne:

```
plural int * plural ptr_pe
```

permet de déclarer une variable multiple *ptr_pe* (résidant dans chaque PE) qui pointe vers une adresse de la mémoire des PEs contenant une valeur entière, chaque adresse pouvant être différente. La représentation de ces différentes variables est shématisée sur la Figure 4.7

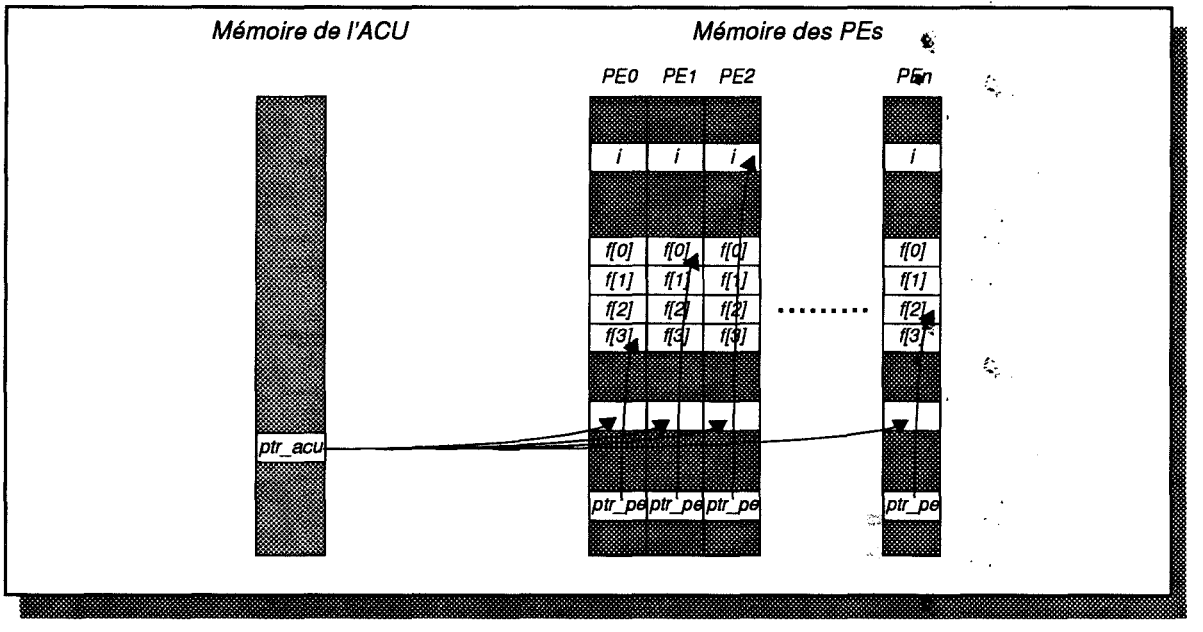


Figure 4.7 ***Représentation des variables multiples***

Les structures de contrôle sont elles aussi étendues de manière à prendre en compte des conditions sur les variables multiples. Ceci s'effectue par l'intermédiaire du bit d'activité. Les PEs qui obtiennent une réponse négative lors de l'évaluation d'une condition faisant intervenir des variables multiples, positionnent leur bit d'activité à 0. De cette manière, ils sortent de l'ensemble des PEs actifs, et n'y reviendront que lorsque la partie de programme qui dépend de la condition sera terminée.

Par exemple, dans le cas de la structure conditionnelle suivante,

```
if(<condition_plural>) {
|    bloc1;
}
else {
|    bloc2;
}
```

le bloc d'instruction 1 sera d'abord exécuté par tous les PEs obtenant une réponse positive à la condition, tandis que les autres PEs resteront inactifs. Dans un second temps, le bloc d'instructions 2 sera exécuté uniquement par les PEs précédemment inactifs (les bits d'activité étant inversés lors de l'entrée dans la branche *else*).

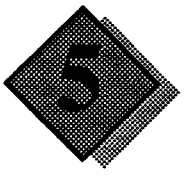
Le même type de fonctionnement est assuré pour les structures répétitives (while, for, do). Dans ce cas, les tests sont toujours effectués en parallèle, et la boucle complète ne se termine que lorsque l'ensemble des PEs verront le résultat de la condition devenir faux (tout PE voyant sa condition devenir fausse deviendra inactif, tandis que les autres PEs continuerons l'exécution répétitive de leur corps de boucle). Le langage MPL peut alors être perçu comme une surcouche du langage C, offrant des possibilités de gestion du parallélisme massif, tant au travers de ses bibliothèques que de ses structures de contrôle.

4.8 Conclusion

Les architectures parallèles présentent plusieurs caractéristiques dont il faut tenir compte lors de l'implantation d'un algorithme, quel que soit cet algorithme. Celles-ci concernent essentiellement le mode de fonctionnement (MIMD, SIMD) et la gestion de la mémoire (distribuée, commune). Ces deux aspects d'une machine conditionnent le découpage de l'algorithme séquentiel en vue d'obtenir des performances aussi élevées que possible lors de son implantation sur une machine parallèle.

Nous avons présenté deux machines, le Transputer et la Maspar, utilisant chacune un mode de fonctionnement différent. Les chapitres qui vont suivre vont étudier les possibilités de parallélisme qui existent sur chacune d'entre elles, dans le cadre de l'algorithme de radiosité. De manière plus générale, le chapitre 5 sera consacré à l'étude des solutions parallèles implantables sur des machines MIMD, tandis que, dans le chapitre 6, nous développerons différentes approches visant à utiliser un grand nombre de processeurs fonctionnant en mode SIMD.

CHAPITRE



Radiosité et parallélisme

Les chapitres précédents ont soulignés un aspect important des méthodes de radiosités, en l'occurrence le coût de calcul élevé généré par l'évaluation des facteurs de forme, et ce quelle que soit la méthode envisagée. Il apparaît donc souhaitable de proposer des schémas de parallélisation pour ces algorithmes, afin de diminuer les temps de calcul, et, par conséquent, améliorer l'interactivité avec l'utilisateur.

Dans un premier temps, nous analyserons les possibilités de parallélisme qui existent au sein de chaque approche de radiosité, ce qui nous permettra de dégager un certain nombre de schémas communs. Nous détaillerons ensuite les différentes implantations parallèles étudiées à ce jour dans un contexte de parallélisme MIMD. Les différentes approches seront classées en fonction de la représentation mémoire utilisée pour la base de données.

5.1 Les sources de parallélisme en radiosité

Les algorithmes de radiosité présentent un certain nombre de niveaux de parallélisme potentiels. Ceux-ci diffèrent légèrement selon l'approche choisie (radiosité progressive ou calcul de la matrice des facteurs de forme), et également en fonction de l'algorithme de calcul des facteurs de forme. Nous allons détailler, ci-dessous, les sources de parallélisme pour chacune de ces deux approches, en tenant compte du type d'algorithme utilisé pour le calcul des facteurs de forme. Nous analyserons les problèmes qui surviennent pour chacun des niveaux obtenus.

5.1.1 Parallélisation de la radiosité progressive

La méthode de radiosité progressive est basée sur le calcul progressif de l'illumination de l'ensemble des facettes de la scène. A chaque nouvelle étape d'illumination, une facette émettrice est choisie et se comporte comme une source vis-à-vis des autres facettes. Sa contribution à l'éclairement des autres facettes est déterminée par l'intermédiaire du calcul des facteurs de forme entre cette facette et le reste de l'environnement. Le principe de cet algorithme est schématisé ci-dessous.

```
Pour chaque phase de shooting
|
|     Pour chaque facette/sommet de la base de données
|     |
|     |     déterminer le facteur de forme avec la facette émettrice
|     |
|     FinPour
|
|     mettre à jour les radiosités
|
|     choisir une nouvelle facette émettrice
|
FinPour
```

Les différents niveaux de parallélisme de cette approche correspondent chacun à l'une des boucles de l'algorithme. Il est ainsi possible soit d'effectuer plusieurs phases d'illumination simultanément, soit de distribuer une seule phase d'illumination sur plusieurs processeurs. Nous introduirons, dans le cas des approches projectives, un troisième niveau, qui concerne le calcul simultané de plusieurs proxels.

Calcul parallèle de plusieurs émissions

Une phase de l'algorithme de radiosité progressive correspond à distribuer l'énergie lumineuse émise (ou réfléchie) par une facette sur l'ensemble des autres facettes. Cette distribution se traduit tout d'abord par la nécessité de calculer une colonne de facteur de forme. Dans la mesure où un facteur de forme représente une quantité purement géométrique, il n'y a pas de dépendance directe entre les calculs de plusieurs colonnes distinctes. Il est donc possible de calculer simultanément plusieurs colonnes de facteurs de forme depuis différentes facettes émettrices. Ce schéma correspond alors à une émission parallèle depuis plusieurs facettes distinctes (voir Figure 5.1).

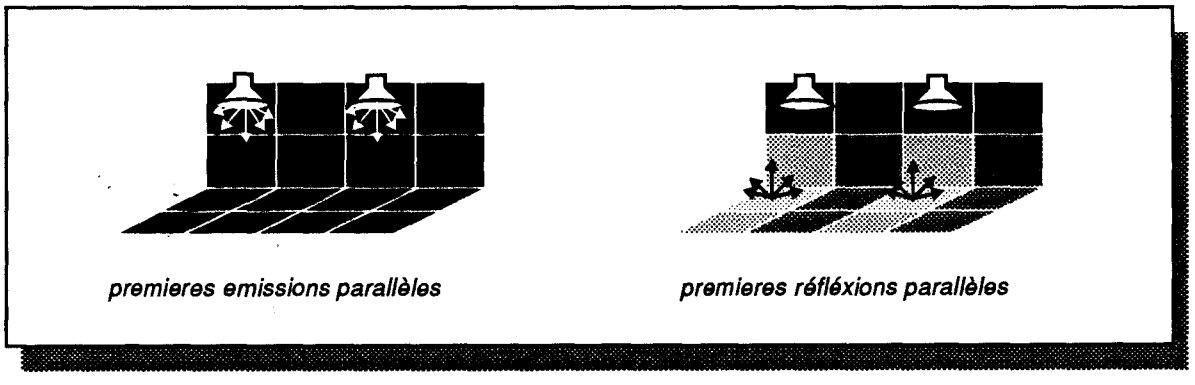


Figure 5.1 ***Exemples d'émissions parallèles***

Le problème de l'ordonnancement des émissions se pose néanmoins. En effet, dans l'algorithme séquentiel, chaque phase d'illumination dépend des résultats de la phase précédente. Dans la mesure où plusieurs phases d'illumination sont appliquées en parallèle, la sémantique de l'algorithme séquentiel se trouve modifiée. En effet, la mise à jour des radiosités, et donc le choix d'une nouvelle facette émettrice, ne se fera, a priori, plus dans le même ordre que l'ordre séquentiel.

Il convient cependant de noter que l'ordre d'évaluation des facettes émettrices n'influe pas sur la convergence finale, mais uniquement sur la vitesse de convergence. C'est la raison pour laquelle Cohen [Cohen88] a choisi d'émettre depuis la facette possédant la plus grande énergie latente, de manière à accélérer la vitesse de convergence. La modification de l'ordre de sélection des facettes émettrices n'est ennuyeuse que par l'augmentation potentielle du nombre d'itérations qu'elle implique, par rapport à celui de l'algorithme séquentiel. Ceci ne doit cependant pas être considéré comme un obstacle important, dans la mesure où le gain obtenu par une telle parallélisation compense le nombre d'itérations supplémentaires.

L'avantage de cette approche est qu'elle est relativement indépendante de l'algorithme utilisé pour le calcul des facteurs de forme. Seul le traitement des résultats diffère légèrement. Dans les cas des approches projectives, le résultat du calcul est une colonne de facteurs de forme entre une facette émettrice et l'ensemble des facettes de l'environnement. C'est une colonne de facteurs de forme entre la facette émettrice et l'ensemble des sommets de l'environnement, pour les approches utilisant le tracé de rayons. Dans le cas des algorithmes projectifs, chaque processeur prend en charge sa propre surface de projection, qui peut éventuellement être différente d'un processeur à l'autre (hémicube, plan unique, disque ...). De plus, le nombre de proxels utilisés par chaque processeur peut varier, en fonction, par exemple, de la quantité d'énergie à distribuer.

Des problèmes apparaissent cependant à différents endroits de l'algorithme. Le premier d'entre eux concerne la disponibilité de la base de données. Pour calculer une colonne de facteurs de forme, il est nécessaire de disposer de toutes les données décrivant la scène. Dans le cas des approches projectives, un processeur doit pouvoir avoir accès à l'ensemble des facettes de l'environnement, puisque chacune d'entre elles est potentiellement visible depuis la source. De même pour le tracé de rayons, où un processeur qui effectue une phase d'émission complète doit disposer de tous les sommets de la scène, car chacun d'entre eux peut recevoir de l'énergie depuis la facette émettrice. Ainsi, si chaque processeur n'a pas directement accès à l'ensemble de la base de données (base de données centralisée ou répartie entre les processeurs), de nombreuses communications seront nécessaires entre les différents processeurs, afin qu'ils puissent disposer des données dont ils ont besoin.

Le second problème concerne la mise à jour des radiosités. Après le calcul d'une colonne de facteurs de forme, celle-ci est utilisée pour calculer l'énergie reçue par chaque facette (ou par chaque sommet) depuis la facette émettrice. Dans la mesure où plusieurs processeurs peuvent terminer leur calcul d'une colonne de facteurs de forme simultanément, et donc vouloir mettre à jour, en même temps, la radiosité des facettes, il faut assurer la cohérence des valeurs mises à jour, que la base de données soit centralisée (accès à une ressource critique), dupliquée (mise à jour sur tous les sites) ou répartie (routage des valeurs à utiliser pour la mise à jour).

Il est à noter que ce type de parallélisation tirera pleinement profit d'un fonctionnement asynchrone des processeurs, dans la mesure où le volume de calcul est très différent d'une facette émettrice à l'autre.

Calcul distribué d'une émission

Une seule émission à la fois est effectuée, mais le calcul de la colonne de facteurs de forme correspondante est distribué sur différents processeurs. Ce type de parallélisation correspond au second niveau de boucle de l'algorithme de radiosité progressive (encadré sur l'algorithme ci-dessous).

```
Pour chaque phase de shooting
|
|   Pour chaque facette de la base de données
|   |
|   |     déterminer le facteur de forme avec la facette émettrice
|   |
|   FinPour
|
|   mettre à jour les radiosités
|
|   choisir une nouvelle facette émettrice
|
FinPour
```

Dans ce type de parallélisation, il est nécessaire de tenir compte de l'algorithme choisi pour le calcul des facteurs de forme, et de la façon de distribuer les calculs. En effet, un certain nombre de dépendances risquent de survenir selon la stratégie parallèle choisie.

Dans le cas des algorithmes projectifs, deux stratégies sont envisageables:

- Chaque processeur se charge d'une partie de la surface de projection, auquel cas aucune dépendance particulière n'apparait. L'ensemble des facettes de la base de données est projeté parallèlement sur chaque site, et les informations issues de chaque processeur sont directement exploitables pour la mise à jour des radiosités.
- Les processeurs collaborent aux projections sur la même surface, celle-ci étant dupliquée sur chaque processeur. Chacun d'entre eux traite alors une partie seulement de la base de données. Dans ce cas, les informations issues de chaque processeur ne sont pas directement utilisables, puisque chacune d'entre elles ne reflète qu'une information partielle sur la visibilité de chaque facette depuis la source. En effet, chaque proxel contient l'identité de la

facette qui y est visible. Cette facette est la facette la plus proche de la source, dans la direction du proxel, et pour la partie de la base de données traitée par le processeur. Chaque proxel étant dupliqué sur l'ensemble des processeurs, il est nécessaire de leur appliquer une seconde opération, visant à déterminer la facette réellement visible dans cette direction.

De la même manière, deux stratégies peuvent être adoptées pour les algorithmes utilisant le tracé de rayons:

- Chaque processeur se charge du calcul de visibilité pour une partie seulement des sommets de la base de données. Un processeur reçoit une partie des sommets décrivant la scène, et effectue le tracé de rayons à partir de ces sommets uniquement. Les informations de visibilité qu'il obtient sont directement utilisables pour la mise à jour des radiosités.

  Cette stratégie est similaire à celle de la parallélisation de l'algorithme du lancer de rayons par la technique dite du "flot de données", les rayons (issus des pixels de l'écran) étant assignés à différents processeurs. Ceux-ci sont alors chargés de calculer le point d'intersection le plus proche entre le rayon et les objets de la base de données. A noter, cependant, que dans le cas qui nous intéresse, le fait de trouver une intersection suffit à arrêter le calcul, le sommet dont est issu le rayon étant alors invisible depuis la source.

- Les processeurs collaborent pour le calcul de visibilité depuis les mêmes sommets. Chaque processeur effectue, pour l'ensemble des sommets, les calculs d'intersections sur la partie de la base de données qu'il possède. De la même manière que pour les approches projectives, il est nécessaire d'appliquer une nouvelle opération sur chacun des sommets de manière à s'assurer de sa visibilité depuis une source. En effet, puisque chaque processeur n'a appliqué qu'une partie seulement des calculs d'intersection, rien ne permet d'affirmer qu'un sommet déterminé comme visible sur un processeur, le sera aussi sur les autres.

  Cette stratégie est à rapprocher, cette fois, de la parallélisation du lancer de rayons par la technique du "flot de rayons", où un rayon passe successivement par chaque processeur. Ceux-ci disposent d'une partie de la base de données et recherchent les intersections du rayon avec leurs objets. Après être passé par tous les processeurs, le point d'intersection le plus proche (s'il existe) est obtenu.

Le problème de la disponibilité de la base de données se pose dans les différentes stratégies exposées ci-dessus. Dans le cas de l'utilisation d'une surface de projection fractionnée sur différents processeurs, chacun de ceux-ci a en effet besoin d'accéder à l'ensemble des facettes. Dans le cas opposé, par contre, (calcul conjoint de la même surface), les facettes sont implicitement distribuées sur les processeurs, et le problème de l'accès pour l'étape de projection ne se pose donc pas. Néanmoins, dans ce cas, un problème tout aussi important apparait, qui est de pouvoir fusionner les différentes informations de visibilité, réparties sur les différentes copies de la surface de projection. Ces problèmes apparaissent de façon similaire dans les algorithmes utilisant le tracé de rayons, où il sera nécessaire de faire transiter soit les rayons, soit les objets de la base de données.

A nouveau, le mode de fonctionnement optimum des différents processeurs sera ici asynchrone, puisque les différents calculs à effectuer sont de complexités différentes.

Calcul parallèle des proxels

Notons tout d'abord que cette possibilité n'existe qu'au sein des algorithmes de calcul des facteurs de forme basés sur une approche projective. Nous avons ajouté dans l'algorithme

général de radiosité progressive, une troisième boucle, qui correspond à la détermination de chaque proxel recouvert par une facette.

```
Pour chaque phase de shooting
|
|     Pour chaque facette de la base de données
|     |
|     |    Pour chaque proxel de la surface de projection
|     |    |
|     |    |     déterminer si la facette est visible dans le proxel
|     |    |
|     |    FinPour
|     |
|     FinPour
|
|     mettre à jour les facteurs de forme
|     mettre à jour les radiosités
|     choisir une nouvelle facette émettrice
|
FinPour
```

C'est ce niveau de parallélisme qui va être exploité par ces approches: les différents processeurs vont collaborer pour déterminer en quels proxels une facette est visible. Les processeurs sont donc alloués à des sous-ensembles disjoints de proxels, et ils travaillent tous sur la même facette en même temps. Dans le cas idéal, chaque proxel est géré par un processeur différent.

Cette méthode est à rapprocher des machines à parallélisme pixel, qui seront évoquées dans le chapitre 6, où chaque processeur est alloué à un pixel. Les processeurs reçoivent successivement les objets à afficher, et déterminent si l'objet courant est présent ou pas dans le pixel qu'il gère. Si c'est le cas, un tampon de profondeur est mis à jour, en fonction de la distance de l'objet au pixel. L'image finale est obtenue après passage de tous les objets.

Ce qui différencie principalement cette approche de la précédente (stratégie de distribution de la surface de projection entre les processeurs), est que les processeurs travaillent ici en mode synchrone, les opérations qu'ils ont à effectuer étant exactement les mêmes, et appliquées sur la même facette. Le grain de parallélisme est donc beaucoup plus fin, et génère une gestion plus délicate pour une utilisation optimum des processeurs.

Les problèmes principaux qui se posent dans ce genre d'approche ne sont pas tant ceux de l'accès à la base de données (les processeurs utilisés sont en général incapables de mémoriser eux mêmes les facettes, ce rôle étant dévolu à une machine hôte), que les problèmes soulevés par la récupération des facteurs de forme. Les informations de visibilité sont en effet distribuées sur de nombreux processeurs, et il est alors coûteux de récupérer ces informations pour pouvoir calculer les colonnes de facteurs de forme.

Une approche utilisant ce type de parallélisme sur une machine graphique spécialisée sera étudiée dans la suite de ce chapitre, tandis que nous développerons diverses approches visant le même niveau de parallélisation dans le chapitre 6.

5.1.2 Utilisation de la matrice des facteurs de forme complète

Dans le cas de cet algorithme initial de radiosité, la mise à jour des radiosités n'est effectuée qu'après le calcul de l'ensemble des facteurs de forme. Ceux-ci sont mémorisés sous forme matricielle, et l'obtention des valeurs de radiosité se fait par l'intermédiaire d'un algorithme

itératif d'inversion de matrice. Les deux étapes de calcul sont bien distinctes, comme le souligne l'algorithme ci-dessous.

```
Pour chaque facette de la base de données
|
|     Pour chaque facette de la base de données
|     |
|     |     Pour chaque proxel de la surface de projection
|     |     |
|     |     |     déterminer si la facette est visible dans le proxel
|     |     |
|     |     FinPour
|     |
|     FinPour
|
|     mettre à jour les facteurs de forme
|
FinPour

Tant que la convergence n'est pas atteinte
|
|     Pour chaque facette de la base de données
|     |
|     |     mettre à jour la radiosité
|     |
|     FinPour
|
FinTantque
```

Les niveaux de parallélisme utilisés pour la première phase de l'algorithme ne varient pas par rapport à ceux présentés dans le cadre de l'algorithme de radiosité progressive. La seule différence est que les problèmes concernant la mise à jour parallèle des radiosités sont déplacés après le calcul complet de la matrice des facteurs de forme.

La mise à jour des radiosités est effectuée sous forme itérative, chaque itération étant appliquée pour chaque facette. Ce processus peut donc être parallélisé, la mise à jour de la radiosité de chaque facette pouvant être, a priori, prise en charge par un processeur différent. Ceci correspond alors à la parallélisation de la dernière boucle "Pour" figurant dans l'algorithme ci-dessus. Néanmoins, cette mise à jour ne peut être appliquée qu'en fonction des valeurs de radiosité de toutes les autres facettes. Le problème de la disponibilité des informations énergétiques concernant chaque facette se pose donc à nouveau. Des communications seront ainsi nécessaires pour transmettre, à chaque processeur responsable de la mise à jour de la radiosité d'une facette, les valeurs de radiosité des autres facettes.

5.1.3 Classification des niveaux de parallélisme

Nous pouvons résumer les différents niveaux de parallélisme présents dans les algorithmes de radiosité, que ce soit par calcul de la matrice ou par émission, sous forme de 3 niveaux distincts:

- Niveau 1: plusieurs colonnes (ou lignes) de la matrice sont calculées en parallèle.
- Niveau 2: les processeurs coopèrent pour calculer une colonne (ou une ligne) de facteurs de forme.
- Niveau 3: les processeurs coopèrent pour calculer les proxels recouverts par une facette donnée. Ce niveau n'existe que dans les approches projectives.

Nous ne tenons pas compte, ici, du parallélisme potentiel de la phase de mise à jour des radiosités utilisée par l'approche nécessitant le calcul complet de tous les facteurs de forme.

Ces différents niveaux sont schématisés sur la figure ci-dessous (Figure 5.2).

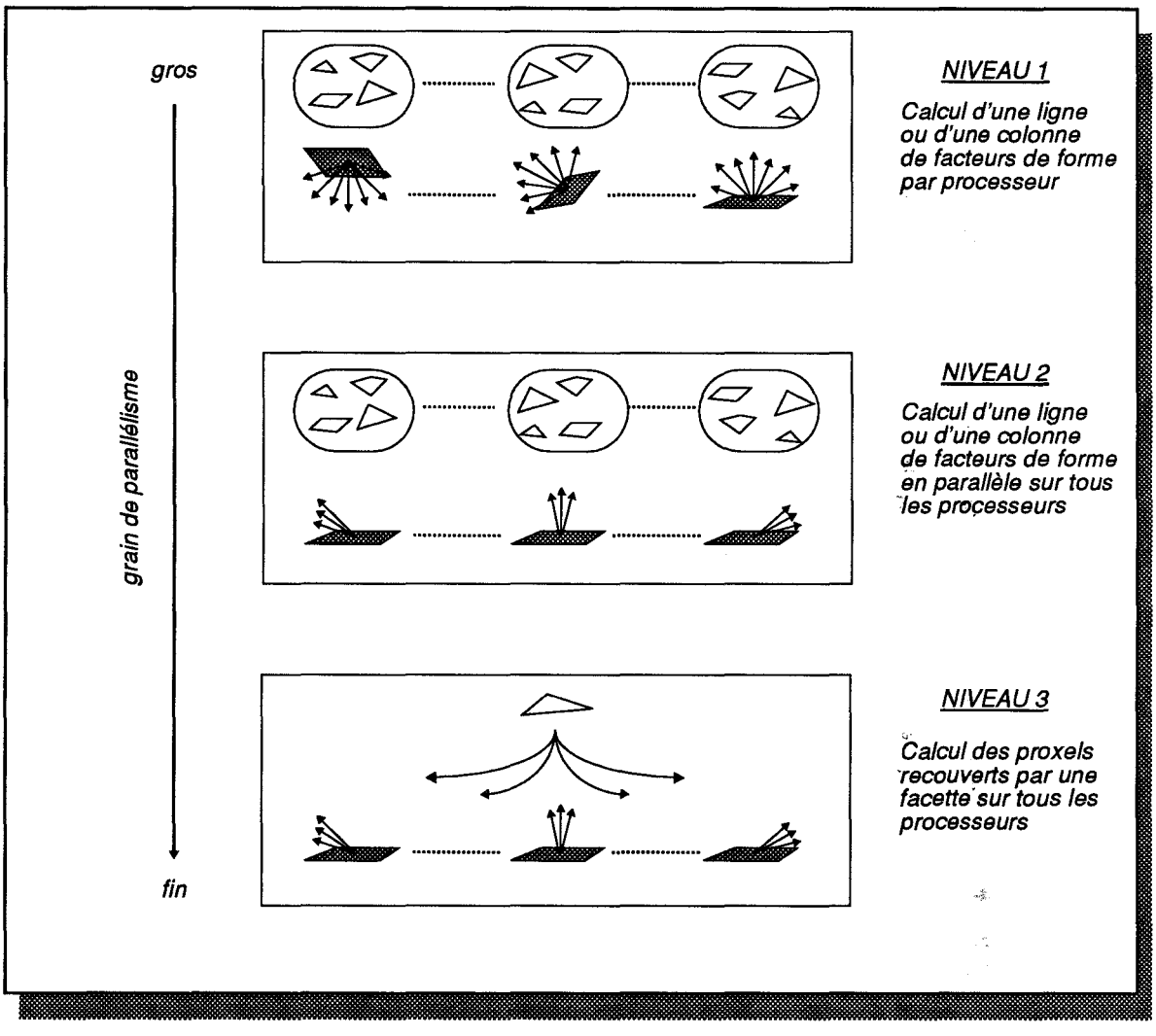


Figure 5.2 ***Niveaux de parallélisme pour la radiosité***

Le niveau 1 présente le plus gros grain de parallélisme utilisable, puisqu'il correspond en fait à la boucle la plus externe de l'algorithme. Le niveau 3 correspond au niveau présentant le grain le plus fin pour les algorithmes projectifs, puisque les actions effectuées à ce niveau sont les actions les plus élémentaires de l'algorithme qu'il soit possible de trouver.

Chacun de ces niveaux partage, à des degrés divers, les problèmes d'accès à la base de données que nous avons déjà évoqués. En prenant comme critère de classement l'accès à la base de données, les différentes approches parallèles ayant donné lieu à étude peuvent être rangées dans trois catégories:

a) base de données dupliquée sur chaque processeur;

b) base de données centralisée en un endroit seulement;

c) base de données répartie sur les différents processeurs.

La présentation des solutions parallèles effectivement implantées qui suit, utilise ces trois moyens de mémorisation de la base de données comme fil conducteur.

5.2 Approches parallèles de base: duplication des données

Nous avons évoqué, lors de la présentation des différentes sources de parallélisme de la radiosité, le problème de l'accès à la base de données. Un moyen simple pour résoudre ce problème consiste à dupliquer l'ensemble des facettes (ou des sommets) sur chacun des processeurs constituants la machine parallèle. Nous allons décrire, dans cette partie, un certain nombre d'implantations effectuées en suivant cette approche, et utilisant différents niveaux de parallélisme.

5.2.1 Calcul de la matrice des facteurs de forme

De manière à calculer l'ensemble des coefficients de la matrice des facteurs de forme, chaque processeur reçoit une partie des lignes à calculer, celles-ci étant distribuées de manière équitable entre les processeurs. Si la base de données est dupliquée sur chaque processeur, chacun d'entre eux peut travailler de manière totalement indépendante, jusqu'à ce qu'ils aient tous terminé le calcul des lignes qui leur sont allouées.

Une telle approche a été proposée par Price [Price 90], et, de manière un peu différente, par Purgathofer [Purga 90]. Dans les deux cas, la machine cible est un réseau de transputers T800, configuré en anneau.

Dans l'approche de Purgathofer, la base de données n'est pas, à proprement parlé, dupliquée sur chaque processeur, mais ne se trouve que sur un seul processeur, appelé "maître". Celui-ci envoie à travers l'anneau des paquets de facettes successifs. Ceux-ci sont récupérés au fur et à mesure par chaque processeur, qui les projette alors sur un hémicube (ou plusieurs selon la place disponible). Lorsque toutes les facettes ont transité à travers l'anneau, le calcul de (des) l'hémicube(s) présent(s) sur chaque processeur est terminé, et chacun d'entre eux en déduit une ligne de facteurs de forme. Celle-ci est mémorisée localement, l'hémicube est réinitialisé, et le maître recommence à envoyer la base de données par paquets successifs, de manière à permettre le calcul de nouvelles lignes. Ce fonctionnement est schématisé sur la Figure 5.3.

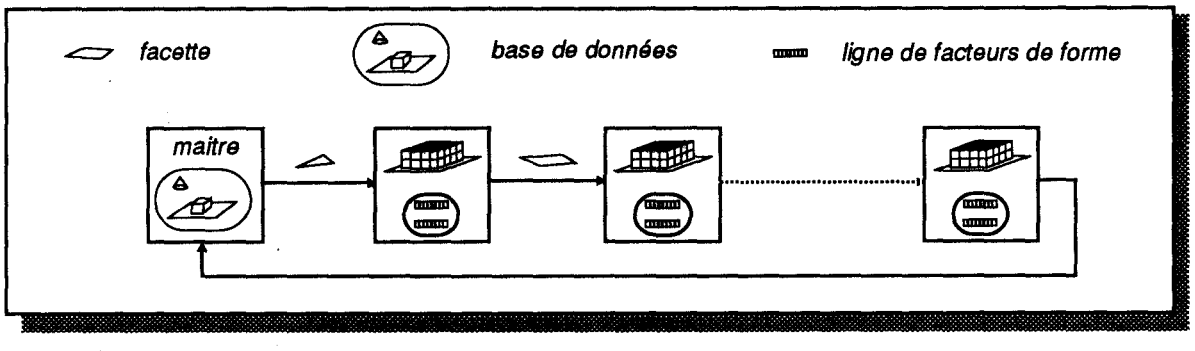


Figure 5.3 ***Principe de fonctionnement de l'implantation de Purgathofer***

Lorsque ces calculs sont terminés, les processeurs vont devoir collaborer pour utiliser les parties de la matrice dont ils disposent pour effectuer le calcul des radiosités. Celui-ci va être effectué en utilisant une version parallèle de la méthode de résolution itérative de Gauss-Siedel.

Chaque processeur dispose de quelques lignes de la matrice, qui correspondent chacune à l'une des équations du système (Eq. 1.4)(chapitre 1, page 17,) à résoudre. Chaque processeur est donc capable de calculer une partie des radiosités, mais a besoin, pour cela, des valeurs de radiosité des autres facettes. Une itération de l'algorithme de Gauss-Siedel va donc consister à faire transiter, le long de l'anneau, les valeurs de radiosité que possède chaque processeur. De cette manière, chacun d'entre eux va récupérer les radiosités de toutes les facettes, et va les utiliser pour mettre à jour celles qu'il gère. Après un tour complet, l'itération est terminée, et une nouvelle peut commencer en utilisant les valeurs qui viennent d'être calculées.

L'arrêt du processus est contrôlé par le maître, qui mémorise l'ensemble des radiosités qu'il voit passer pendant une itération. Lorsqu'une nouvelle itération commence, il compare les nouvelles

valeurs générées avec celles de l'itération précédente, et peut donc décider de l'arrêt du processus s'il juge que la précision est suffisante.

Price ne présente aucun résultat dans son article. Il note cependant que, du fait de l'absence de communications durant la phase de calcul de la matrice, l'efficacité de la parallélisation est maximale. Les communications nécessaires lors de la phase de mise à jour des radiosités génèrent une baisse d'efficacité, qu'il juge néanmoins négligeable devant le temps gagné durant la première phase.

Purgathofer fournit des résultats plus détaillés sur son implantation. Ceux-ci sont résumés sous forme de courbes sur la Figure 5.4. Ils ont été obtenus sur une scène constituée de 1463 facettes, et en utilisant un hémicube de résolution $100 \times 100 \times 50$.

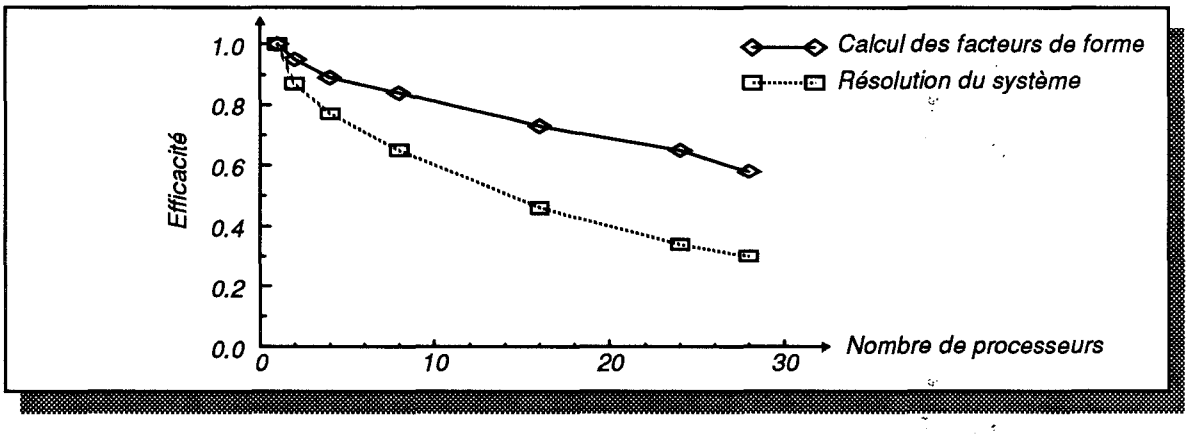


Figure 5.4 ***Evolution de l'efficacité des étapes de l'implantation de Purgathofer***

La courbe schématisant la phase de calcul des facteurs de forme présente une efficacité relativement intéressante pour un nombre moyen de processeurs. Ceci provient essentiellement du fait que la part des temps de communication sur le temps total de calcul de la matrice est faible comparée à la part du temps de calcul des hémicubes. Néanmoins, l'extension de cette solution à un nombre plus élevé de processeurs conduit nécessairement à une sous exploitation importante de chacun d'entre eux. En effet, dans cette implantation, chaque processeur peut calculer un nombre différent d'hémicubes, pour une même phase de transmission de la base de données. Ce nombre dépend de la place disponible sur chaque processeur, celle-ci variant en fonction des facteurs de forme mémorisés à chaque itération. Les temps de calcul peuvent donc être très différents d'un processeur à l'autre. Dans la mesure où un processeur ne mémorise pas les blocs de facettes qui lui parviennent, mais les utilise avant d'accepter un nouveau bloc, les processeurs les plus chargés ralentissent le passage des blocs de facettes vers les autres processeurs de l'anneau.

Comme l'a noté Price, la seconde phase de l'algorithme, qui concerne la mise à jour des radiosités, nécessite une grande quantité de communications. De plus, le temps nécessaire à la mise à jour des radiosités est beaucoup plus court que celui du calcul d'un hémicube. La proportion des temps de communication sur le temps total utilisé pour la mise à jour des radiosités devient donc importante, et génère une forte perte d'efficacité.

Il faut cependant rappeler que la solution consistant à calculer la matrice complète des facteurs de forme ne peut être utilisée, de manière pratique, que dans le cas de petites bases de données, le cas des bases importantes nécessitant alors une quantité trop importante de mémoire sur chaque processeur, ou un nombre très important de processeurs.

5.2.2 Calcul parallèle de plusieurs émissions

Si la base de données est dupliquée sur chaque processeur, le calcul des facteurs de forme depuis différentes facettes émettrices ne pose pas de problème, puisque tout processeur dispose de l'ensemble des informations qui lui sont nécessaires pour effectuer ce calcul. Différents

problèmes surviennent néanmoins lorsqu'un processeur termine le calcul du vecteur de facteurs de forme qui lui a été attribué:

- Mise à jour des radiosités: si l'intégralité des informations présentes au sein de la base de données (informations géométriques et énergétiques) est dupliquée sur chaque processeur, la mise à jour des radiosités de chaque facette par un processeur ne peut pas être uniquement faite de manière locale, sous peine d'incohérence. Les valeurs énergétiques calculées sur un processeur doivent nécessairement être transmises à tous les autres processeurs, afin que ceux-ci puissent prendre en compte les émissions qui ont été effectuées sur d'autres sites. Ceci va donc générer un surcoût important en communication, puisque chaque processeur se doit d'envoyer un vecteur de radiosité à l'ensemble des autres processeurs pour chaque phase d'émission qu'il effectuera.
- Choix d'une nouvelle facette émettrice: après la phase de mise à jour et de diffusion des radiosités, un processeur est à nouveau disponible pour calculer une nouvelle phase d'émission. Il choisit donc la facette qui possède la plus grande énergie latente dans sa mémoire, et effectue le calcul d'un vecteur de facteurs de forme depuis cette facette. Malheureusement, le fonctionnement totalement asynchrone des différents processeurs peut fort bien entraîner le choix de la même facette émettrice sur un autre processeur. Dans ce cas, non seulement des calculs seront effectués en double, mais ils génèreront de plus des erreurs dans la distribution des radiosités, puisque tout se passera comme si la facette avait émis la même énergie deux fois.

Pour résoudre de manière simple ces deux problèmes, l'ensemble des implantations effectuées selon ce modèle de duplication de la base de données utilisent un processeur de contrôle. La gestion des radiosités est centralisée sur un seul processeur (le contrôleur), ce qui permet de limiter considérablement les communications (un processeur n'a plus à diffuser les informations de radiosité à l'ensemble des autres processeurs, mais il ne les envoie que vers le contrôleur). La seconde conséquence de cette gestion centralisée est de permettre d'éviter les duplications de choix d'une facette émettrice, puisque celle-ci ne peut être sélectionnée que par le processeur de contrôle.

Le schéma général de fonctionnement de ces approches suit donc un modèle de *ferme de processeurs*, où chaque processeur applique la même tâche de calcul (une colonne de facteurs de forme) et renvoie ses résultats à un processeur de contrôle qui peut alors lui redistribuer du travail. Ce fonctionnement est schématisé sur la figure ci-dessous (Figure 5.5).

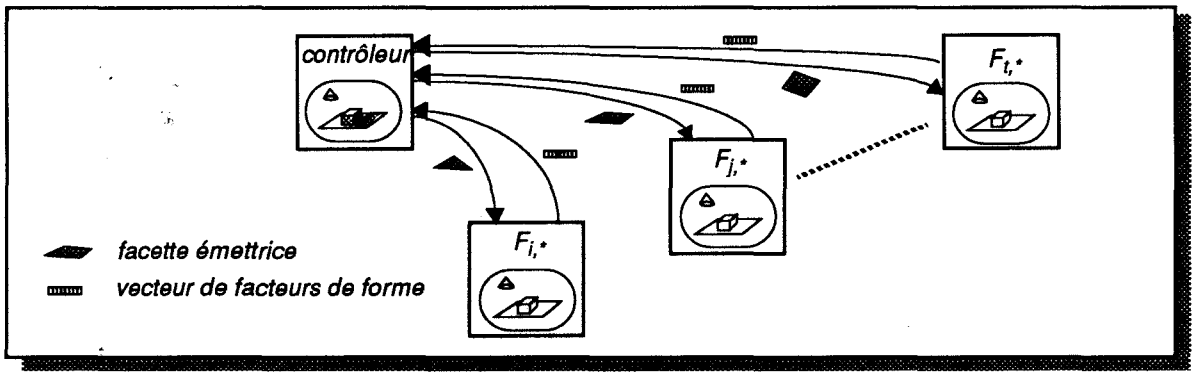


Figure 5.5 ***Principe de fonctionnement sous forme de ferme de processeurs***

Diverses implantations suivant ce schéma de fonctionnement ont été réalisées, la différence essentielle provenant des processeurs utilisés. La première catégorie d'implantations vise à l'utilisation des réseaux de stations de travail existantes [Chen 89], [SillA 89], [Recke 90]. Ceci se justifie par l'importance de la puissance de calcul et de la quantité mémoire ainsi offertes, de même que par les possibilités de communication inter-stations, via un réseau de type "Ethernet" par exemple (Figure 5.6). Il faut aussi noter que ce type de matériel est en général présent dans tous les laboratoires, alors que les machines parallèles sont moins répandues.

Nous avons développé ce type d'approche sur une machine à base de transputers (voir chapitre 4 pour une description détaillée de la machine et du transputer), permettant d'obtenir des puissances de calcul assez élevées, ainsi que des débits de communication très importants entre les processeurs. Le transputer disposant de 4 liens de communication, il est possible de construire différents réseaux, visant à limiter au maximum les distances de communication entre un processeur de la ferme et le contrôleur.

Une configuration arborescente a été implantée, le contrôleur représentant la racine de l'arbre, tandis que chaque feuille est représentée par un processeur de calcul. Chaque processeur est relié par l'un de ses liens à un processeur "père", et à trois processeurs "fils" par ses autres liens (Figure 5.6). Cette configuration a été choisie, car elle permet d'obtenir le plus court chemin entre une feuille de l'arbre et la racine. Une configuration sous forme de pipeline a auparavant été testée sur une machine à base de transputers (T-Rack [Illiev 87],[Knowl 87],[Boian 89]) par Leprêtre [Lepre 91] à Manchester, ainsi que sur le Multicluster II dont nous disposons. Les résultats obtenus, s'ils montrent un gain certain, soulignent l'inadéquation du pipeline en tant que réseau, du fait des pertes occasionnées sur les processeurs les plus éloignés du contrôleur.

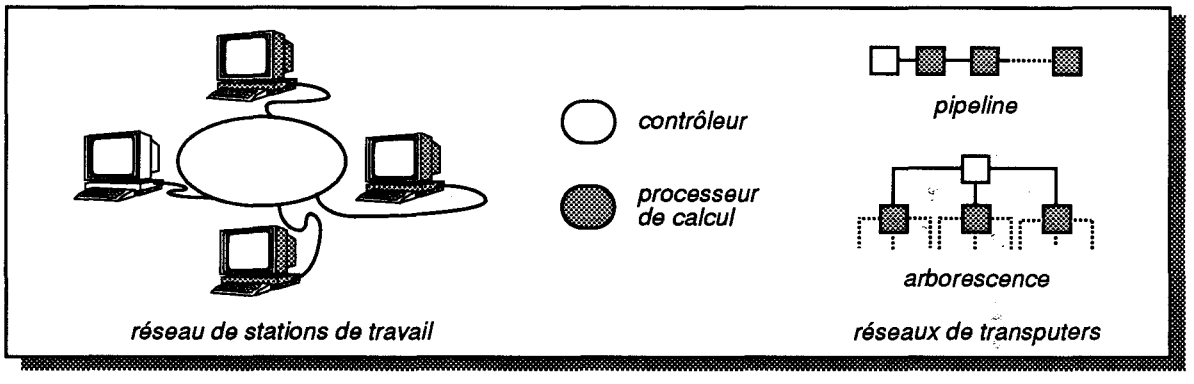


Figure 5.6 ***Les différentes configurations pour l'implantation de la ferme de processeurs***

L'utilisation d'une configuration arborescente permet de limiter considérablement ces pertes, en diminuant la longueur du chemin de chaque message (nouvelle facette émettrice, vecteur de facteurs de forme).

Nous présentons, sur la Figure 5.7, un certain nombre de résultats concernant ces implantations.

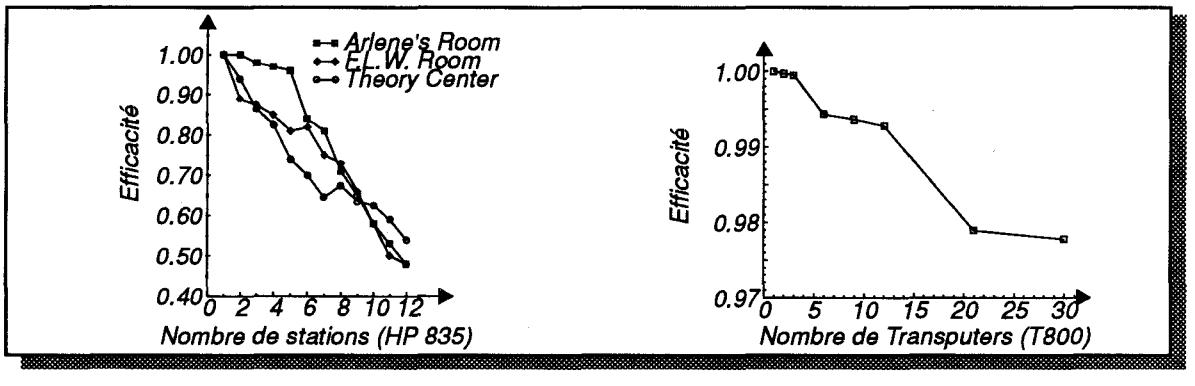


Figure 5.7 ***Comparaison de l'efficacité des implantations sur réseau de stations et sur transputers***

Les courbes de gauche présentent l'évolution de l'efficacité de l'implantation effectuée par Recker [Recke 90]sur un réseau de stations HP 835, pour différentes scènes. La courbe de droite correspond au même type de mesure, mais cette fois effectuée avec notre implantation sur réseau de transputers, le réseau étant configuré de manière arborescente. La définition de l'efficacité est celle définie au chapitre 4, c'est à dire :

$$E = \frac{T(1)}{N \times T(N)}$$

où $T(1)$ représente le temps d'exécution sur un processeur, N le nombre de processeurs utilisés, et $T(N)$ le temps d'exécution sur N processeurs.

Quelle que soit la scène utilisée, l'efficacité de l'implantation sur réseau de stations de travail décroît très rapidement. Plusieurs raisons se conjuguent pour expliquer une telle décroissance. Tout d'abord, le faible débit d'un réseau de type Ethernet (environ 10 Mbits/s) génère des temps de transmission relativement importants. D'autre part, la "ligne" de communication qui relie les différentes stations se comporte comme un bus partagé, ce qui implique nécessairement des conflits d'accès, d'autant plus nombreux que le nombre de stations augmente. Enfin, il est évident que dans une telle implantation, le contrôleur devient une ressource centrale pour tous les processeurs de la ferme. Dans la mesure où celui-ci ne peut servir qu'un seul processeur à la fois, c'est à dire récupérer son vecteur de facteur de forme puis lui envoyer une nouvelle facette émettrice, les processeurs qui tentent de lui renvoyer des facteurs de forme pendant qu'il est occupé, devront nécessairement patienter. Plus le nombre de stations augmente, et plus ce type de conflits se produit.

Dans le cas du réseau de transputers, l'efficacité reste par contre très élevée. Cela provient essentiellement du fait que les conflits de communication sont beaucoup moins fréquents, puisqu'il n'existe pas qu'une unique voie de communication entre le contrôleur et les processeurs. D'autre part, le débit de communication point à point entre deux processeurs est plus élevé que dans le cas du réseau Ethernet, puisqu'il est d'environ 20 Mbits/s. La courbe montre néanmoins très nettement les baisses d'efficacité survenant à l'ajout de processeurs dans le réseau: lorsque les processeurs rajoutés viennent compléter un niveau de l'arbre, l'efficacité résultante ne diminue que très faiblement, les conflits d'accès n'intervenant que sur le père du nouveau processeur. Par contre, dès que les processeurs ajoutés le sont à un nouveau niveau, la baisse est beaucoup plus nette, traduisant le fait que les processeurs du nouveau niveau ont un chemin plus long à parcourir jusqu'au contrôleur. De plus, les conflits de passage par les différents ascendants dans l'arborescence augmentent, puisque chacun d'entre eux possède alors un nombre de descendants plus élevé, et devient par conséquent un lieu de passage plus "fréquenté".

L'efficacité ne doit cependant pas rester le seul outil de mesure; il est aussi intéressant de comparer les temps de calcul résultant de la parallélisation sur les différentes machines. Nous avons réuni, sur les deux tableaux ci-dessous, les temps de calcul pour les deux implantations.

nombre de stations	*temps moyen (secondes)*
1	5
2	2.5
3	1.71
5	1.05
8	0.89
10	0.87
11	0.87
12	0.88

a)

nombre de processeurs	*temps moyen (secondes)*
1	58,74
2	29,37
3	19,57
6	9,74
9	6,48
12	4,85
21	2,73
30	1,91

b)

Table 5.1 ***Temps moyens de calcul pour le réseau de stations (a) d'après [Recke 90] et pour le réseau de transputers (b)***

Les temps présentés dans ces deux tableaux ont été obtenus pour la même résolution d'un hémicube ($180 \times 180 \times 90$), mais pour des bases de données de taille différente : la scène utilisée par Recker contient environ 5200 facettes ("Arlene's Room"), tandis que notre scène de test contient environ 9600 facettes (version simplifiée de la scène "Bureau1"). La comparaison, en terme de temps de calcul, doit donc être effectuée en divisant les temps obtenus sur transputers par 2 environ. Il apparait donc que l'utilisation d'un réseau de 30 transputers permet d'obtenir des temps de calcul moyen de même valeur que pour un réseau d'une dizaine de stations HP 835, sachant que l'augmentation du nombre de transputers doit permettre de diminuer encore les temps de calculs moyens, alors que ces temps recommencent à augmenter dès que le nombre

de stations devient plus important. Le même type de comportement est obtenu lorsque le nombre de facettes augmente sur chaque station [Recke 90].

Cette approche souffre cependant d'un handicap important: la taille mémoire nécessaire sur chaque processeur pour représenter à la fois l'hémicube et toute la base de données. Ceci apparait plus particulièrement sur les transputers, où la faible quantité mémoire disponible ne permet que la représentation de bases de données de faible complexité et l'utilisation d'hémicubes de faible résolution.

5.2.3 Calcul distribué d'une émission

Une seconde approche consiste à distribuer le calcul d'une seule émission sur les différents processeurs du réseau. Celle-ci a été implantée par Singh [Singh 92] sur un hypercube iPSC/2, en utilisant l'algorithme de l'hémicube. Le principe de l'algorithme repose sur la multiplication des volumes de projection (initialement, il existe 5 volumes différents, un par face de l'hémicube). Ceci est obtenu en découpant l'hémicube en un certain nombre de zones disjointes appelées *"hémiplan"*. Chacune de ces zones définit alors un angle solide différent (Figure 5.8)

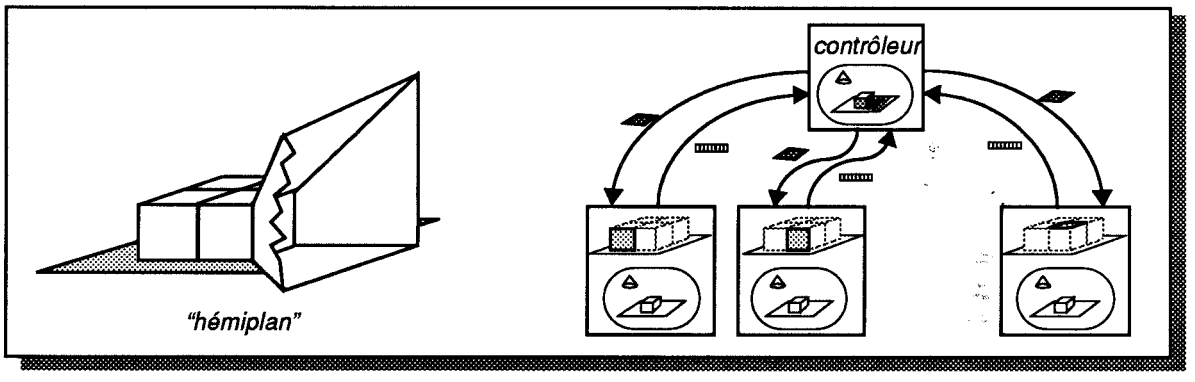


Figure 5.8 ***Principe de découpage et de distribution de l'hémicube***

Chacune de ces zones est distribuée à un processeur différent. Dans la mesure où chaque processeur dispose d'une copie complète de la base de données, les calculs de facteurs de forme peuvent être effectués de manière totalement indépendante. De la même manière que précédemment, les problèmes de mise à jour des radiosités et de choix d'une nouvelle facette émettrice sont résolus par l'intermédiaire d'un contrôle centralisé de ces deux tâches:

- les processeurs de la ferme sont chargés de calculer un vecteur de facteurs de forme pour la partie de l'hémicube qui leur est allouée, puis de renvoyer ce vecteur au contrôleur.
- le contrôleur récupère chaque vecteur, met à jour les radiosités et sélectionne puis envoie une nouvelle facette émettrice.

De manière à équilibrer la charge de chaque processeur (chaque hémiplan ne nécessite pas la même quantité de calcul), chaque processeur reçoit plusieurs hémiplans différents. Les résultats obtenus ne permettent cependant pas de juger correctement une telle approche, puisque seules deux mesures ont été fournies: en découpant l'hémicube en 12 parties distinctes, et en utilisant 4 processeurs, l'accélération obtenue par rapport à une approche séquentielle est d'environ 3.2, soit une efficacité d'environ 80%. De même, en utilisant 8 processeurs et le même nombre d'hémiplans, l'accélération est d'environ 5.5, soit une efficacité de l'ordre de 69%.

La perte d'efficacité de la seconde évaluation est due au déséquilibre de charge qui se produit entre les processeurs, puisque 4 d'entre eux vont avoir à calculer 2 hémiplans, alors que les 4 autres ne vont en avoir qu'un seul. La perte de 20% qui apparait dans le premier provient du même phénomène de déséquilibre de charge entre les processeurs, les temps de calcul pouvant être fort différents d'un hémiplan à l'autre. Dans ce cas, les processeurs ayant terminé leurs calculs doivent nécessairement attendre les autres processeurs, le fonctionnement de l'algorithme (un hémicube à la fois) imposant une resynchronisation des différents processeurs au début d'une phase d'émission.

L'obtention d'un équilibre de charge correct ne peut être obtenu qu'en découpant l'hémicube en un grand nombre d'hémiplans, et en les distribuant de manière uniforme sur les processeurs. Bien que cette approche permette de limiter le problème de la taille mémoire nécessaire à la représentation d'un hémicube, le fait de dupliquer la base de données ne permet toujours pas de l'utiliser pour des scènes de grande taille.

5.2.4 Conclusion

Ce type d'approche présente l'inconvénient important d'avoir à dupliquer entièrement la base de données sur chaque processeur. Si ceci peut, dans une certaine mesure, être acceptable sur une station de travail, la faible quantité mémoire généralement disponible sur un transputer, ou tout autre type de processeur élémentaire d'une machine parallèle, limite considérablement la complexité des scènes utilisables. D'autant plus qu'une grande partie de la mémoire doit être réservée à la représentation de l'hémicube (ou de tout autre algorithme projectif). Comme nous l'avons vu au chapitre 2, la place requise par un tel algorithme est loin d'être négligeable.

Il faut d'autre part noter, que l'utilisation du tracé de rayon à la place d'un algorithme projectif ne résout en rien le problème de la limitation de la taille mémoire; tout au plus permet-elle de le déplacer à un niveau supérieur de complexité des scènes utilisables.

Il nous apparait néanmoins intéressant de noter, au vu des résultats obtenus par notre implantation sur transputers, que ce type d'approche peut être très intéressant pour des scènes de complexité moyenne, les scènes de plus grande complexité devant alors faire appel à des solutions permettant soit de distribuer la scène sur différents processeurs, soit de ne la représenter qu'une seule fois au sein d'une mémoire partagée.

5.3 Utilisation d'une mémoire partagée

L'un des moyens qui peut être envisagé pour l'utilisation de bases de données importantes est d'utiliser des machines utilisant une mémoire unique, partagée par l'ensemble des processeurs. Dans ce cas, la scène est représentée une seule fois, les processeurs étant en concurrence pour l'accès aux informations dont ils ont besoin. Nous allons décrire, dans ce paragraphe, deux implantations différentes utilisant cette notion de mémoire partagée, la première dans le cadre d'une approche permettant le calcul simultané de plusieurs émissions; la seconde dans le cadre de l'utilisation du parallélisme proxel.

5.3.1 Calcul parallèle de plusieurs émissions

Le principal problème des machines utilisant une mémoire partagée est leur difficulté d'extension, du fait de l'augmentation des conflits d'accès à la mémoire lorsque le nombre de processeurs augmente. Pour contourner ce problème, il est possible de simuler une mémoire partagée sur une machine à mémoire distribuée, introduisant alors la notion de *Mémoire Virtuelle Partagée* (MVP). Nous allons tout d'abord décrire le fonctionnement d'un tel système, tel qu'il a été développé à Rennes, puis nous étudierons l'implantation d'un algorithme de radiosité progressive utilisant ce mécanisme.

La mémoire virtuelle partagée KOAN

Koan est le nom d'une MVP utilisée par Badouel [Badou 91] sur un hypercube iPSC/2. L'ensemble des espaces mémoire présents sur chaque processeur est perçu comme un seul et unique espace d'adressage virtuel, découpé en pages de même taille. Ces pages sont distribuées équitablement dans la mémoire locale de chaque processeur.

Lorsqu'un processus tournant sur un processeur désire une donnée qui ne se trouve pas dans l'une des pages locales, cette page est demandée au processeur qui la possède. Une partie de la mémoire locale de chaque processeur est utilisée en tant que cache et permet la mémorisation des pages demandées aux autres processeurs. Lorsque ce cache est plein, une nouvelle page arrivant d'un autre processeur sera rangée à la place de l'une des pages déjà présente dans le

cache, en suivant une stratégie permettant l'écrasement de la page la moins récemment utilisée. Notons que cette gestion des pages est rendu invisible au programmeur par l'implantation du mécanisme de gestion de la MVP au niveau du système d'exploitation, et le fait qu'elle utilise une unité particulière du 80386, la MMU (Memory Management Unit), ce qui permet une utilisation efficace des pages.

Utilisation dans le cadre d'un algorithme de radiosité

Le système de MVP KOAN a été précédemment utilisé dans le cadre de la parallélisation de l'algorithme du lancer de rayons, et a permis d'obtenir d'excellents résultats quant aux accélérations obtenues [Badou 90].

Dans cette implantation, les pixels sont distribués aux processeurs sous forme de blocs de pixels voisins, un processeur se chargeant alors du calcul complet d'un bloc. Dans la mesure où il existe une certaine cohérence entre les rayons issus de ces pixels (les rayons se dirigent dans la même direction et ont une forte chance d'avoir une intersection avec les mêmes objets), leur calcul par le même processeur permet de limiter les communications. En effet, lorsqu'un rayon est calculé par un processeur, il est possible que celui-ci ne possède pas, dans sa mémoire locale, l'ensemble des objets situés dans la direction du rayon avec lesquels il doit calculer une intersection. Placé dans le contexte d'une MVP, cela signifie qu'il y a un défaut de page. Cette page est alors récupérée par le processeur et placée dans sa mémoire cache. Lorsque les rayons voisins vont être traités à leur tour, il y a une forte probabilité pour que les pages qui leur sont nécessaires pour les calculs d'intersection soient déjà présentes dans le cache du processeur. Les communications nécessaires aux demandes de données sont alors fortement réduites, ce qui permet d'obtenir une accélération importante du processus de parallélisation.

Menard [Menar 92] propose d'utiliser ce concept de MVP pour implanter un algorithme de radiosité progressive basé sur le tracé de rayons (voir le chapitre 3, paragraphe 3.2). Chaque processeur prend en charge une facette émettrice différente et effectue le tracé de rayons à travers tous les proxels de l'hémisphère (Figure 5.9).

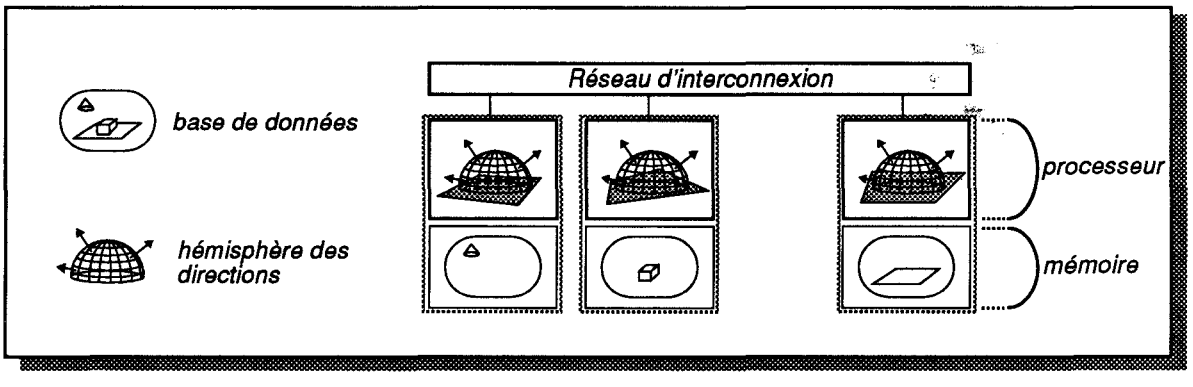


Figure 5.9 ***Distribution des tâches et de la base de données à travers le réseau***

Le calcul des facettes visibles au travers de chaque proxel nécessite d'importer les pages de la base de données qui ne sont pas présentes sur un processeur, de la même manière que pour un lancer de rayons traditionnel. Lorsque l'ensemble des proxels a été calculé, les processeurs effectuent une phase de calcul des radiosités pour chaque facette visible, puis doivent appliquer une opération de mise à jour de ces radiosités sur les différentes facettes. L'utilisation d'une MVP offre une vue globale des radiosités; chaque processeur peut donc effectuer lui même cette mise à jour, de même qu'il lui est possible de sélectionner la facette possédant la plus grande énergie. Il est cependant nécessaire que ces deux opérations se fassent en section critique, de manière à éviter que deux processeurs ne cherchent à modifier en même temps la radiosité d'une facette, ou à choisir la même facette émettrice.

L'auteur note cependant que l'utilisation de la notion de section critique risque de pénaliser très rapidement l'application, puisque plus le nombre de processeurs est important, plus le nombre de conflits pour l'accès à la ressource critique augmente. La conséquence directe de ce

phénomène bien connu est une augmentation du temps d'attente des processeurs pour obtenir l'accès à la ressource, et donc une perte notable d'efficacité.

L'utilisation de la MVP, pour cette phase de l'algorithme est donc écartée, et remplacée par la distribution des calculs de mise à jour des radiosités sur chacun des processeurs. Un processeur qui termine le calcul de sa colonne de facteurs de forme envoie à chaque processeur les facteurs de forme concernant les facettes qui se trouvent dans sa mémoire locale, accompagnés de la facette émettrice. A la réception de ces informations, les processeurs stoppent leur activité et mettent à jour les radiosités de leurs facettes. Ils sélectionnent de même la facette qui, localement, possède la plus grande énergie latente et la renvoient au processeur ayant demandé la mise à jour. Celui-ci, après avoir calculé la radiosité de ses propres facettes, collecte les facettes ayant la plus grande énergie latente qui lui parviennent, et choisit celle possédant la plus grande valeur.

Les résultats présentés pour différentes scènes de test montrent une efficacité importante, comprise entre 80 et 90% selon la scène, les pertes provenant essentiellement du temps de chargement des pages absentes, et de la mise à jour répartie des radiosités.

Il convient cependant de noter que si les performances semblent particulièrement bonnes, ceci est essentiellement dû au fait que la taille de la mémoire cache est suffisante, dans les cas présentés, pour mémoriser l'intégralité de la base de données. De ce fait, après seulement quelques applications par processeur, la base de données est toute entière disponible dans chaque processeur, évitant ainsi un nombre important de communications. En effet, le calcul des facteurs de forme, depuis une facette donnée, nécessite de disposer de toute la base de données qui se trouve dans le demi-espace des directions situé au dessus du plan de cette facette. Les pages contenant ces données devront donc être importées depuis les processeurs où elles se trouvent. Lors du calcul depuis une nouvelle facette émettrice, de nouvelles pages seront nécessaires et viendront compléter la partie de la scène déjà mémorisée dans le processeur. Et ainsi de suite, jusqu'à ce que l'intégralité de la scène ait été recopiée.

Ce phénomène provient du fait qu'il n'existe pas de cohérence entre les différentes facettes émettrices, au sens où ce terme est utilisé dans le cadre du lancer de rayons (directions de propagation voisines pour des rayons issus de pixels voisins). S'il semble envisageable de considérer que les facettes émettrices successivement choisies, dans une implantation séquentielle, sont voisines dans la scène (en considérant les différentes zones de la scène éclairées), cette "propriété" n'est plus valable dans un environnement parallèle, où plusieurs d'entre elles sont calculées simultanément. Les parties de l'environnement visibles depuis deux facettes émettrices reçues successivement par un processeur seront donc vraisemblablement très différentes. Si la taille du cache devient plus petite que la taille globale de la base de données, des défauts de page se produiront à chaque application d'une nouvelle facette émettrice, avec d'autant plus d'importance que la taille du cache sera faible. Dans les cas les plus défavorables, la plus grosse partie de la base de données devra être importée à chaque nouvelle application.

L'exploitation de la cohérence, propriété qui permet d'obtenir une efficacité maximale sur ce type d'architecture, ne peut être obtenue qu'à deux niveaux:

- envoyer des facettes émettrices voisines sur le même processeur, ce qui semble difficilement réalisable; en effet, cela nécessite la mise en oeuvre d'un mécanisme complexe de distribution des facettes émettrices, permettant de "réserver" certaines zones de la scène à un processeur particulier. Ceci est, d'une part, difficile à faire a priori, sans connaissances de la manière dont l'énergie va se répartir au fur et à mesure des émissions. D'autre part, il y a un risque de générer des déséquilibres de charge importants entre les processeurs, dans la mesure où certaines zones peuvent recevoir fréquemment de l'énergie, alors que d'autres ne recevront de l'énergie que lors d'un nombre restreint d'itérations.
- ne traiter qu'une seule facette émettrice à la fois sur l'ensemble des processeurs, en distribuant le calcul de proxels adjacents sur le même processeur (cohérence entre les rayons). Ceci offre de plus l'avantage d'exploiter la cohérence relative qui existe entre deux facettes émettrices successives, qui ont une grande chance d'être voisines. Le problème d'équilibrage de charge se pose néanmoins dans ce cas, puisqu'une telle approche nécessite

une resynchronisation des processeurs à la fin de chaque phase d'émission. A noter que ce problème de l'équilibrage des calculs entre les processeurs a de fortes ressemblances avec le problème du choix des pixels à distribuer dans le cas du lancer de rayons. La stratégie utilisée par Badouel dans ce cas devrait alors pouvoir répondre efficacement à ce problème.

5.3.2 Utilisation d'un parallélisme proxel

Le principe de ce type de parallélisme est d'affecter le calcul de couverture des proxels par une facette directement à différents processeurs. Bien que cette approche semble implicitement découler d'un parallélisme massif (voir chapitre 6), une implantation de ce type a été proposée sur une machine disposant d'un nombre restreint de processeurs, et équipée de circuits spécialisés dans l'affichage de polygones 3D. Le principe de l'implantation proposée par Baum [Baum 90] est d'utiliser le z_buffer câblé d'une station Silicon Graphics pour générer très rapidement un hémicube. Les informations de visibilité obtenues sont ensuite utilisées par les différents processeurs équipants la station pour effectuer la mise à jour des radiosités et le choix d'une nouvelle facette émettrice.

Dans la mesure où ce mode de parallélisation diffère sensiblement de ceux présentés jusqu'à présent, il nous semble important de détailler tout d'abord l'architecture des stations Silicon Graphics, afin de permettre une meilleure compréhension de l'algorithme proprement dit.

Architecture

Nous allons, en premier lieu, décrire l'organisation générale d'une station de travail Silicon Graphics, puis nous détaillerons précisément l'architecture et le mode de fonctionnement du module graphique, qui représente le coeur de la machine.

Schéma général

La machine utilisée par Baum est une Silicon Graphics 4D/280 GTX, composée d'une mémoire centrale, partagée par un à huit processeurs, d'un module d'entrées/sorties et d'une unité spécialisée dans les opérations d'affichage rapide de facettes. Ces diverses unités sont reliées par deux bus: le "Sync Bus", qui permet de synchroniser les différents processeurs, et le "MPlink Bus" qui permet d'effectuer des transferts de données entre les différentes unités. L'organisation générale de cette machine est schématisée sur la figure ci-dessous.

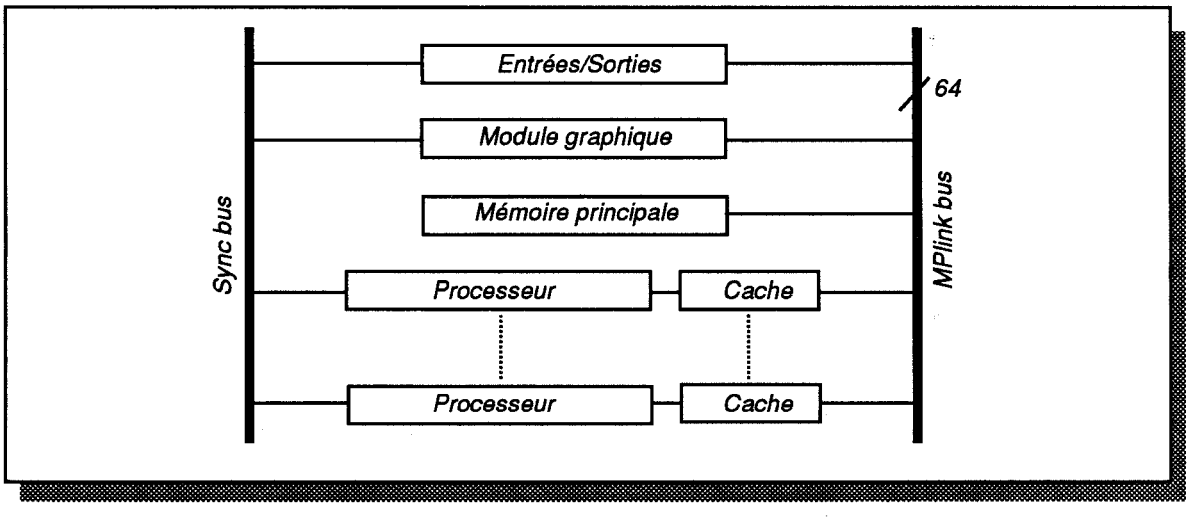


Figure 5.10 ***Organisation d'une station Silicon Graphics d'après [Akele 89]***

Chaque processeur est composé d'un unité centrale 32 bits de type RISC[1], d'une mémoire locale et de caches. Il dispose d'autre part d'une unité de calculs flottants.

1. Reduced Instruction Set Computer

Le module graphique

Ce module représente le coeur d'une station spécialisée dans l'affichage rapide de facettes. Son organisation, dans le cas de la Silicon Graphics GTX est schématisée sur la Figure 5.11. Il comprend 4 modules principaux, spécialisés dans les différentes étapes de calcul nécessaires à la transformation d'une facette en pixels.

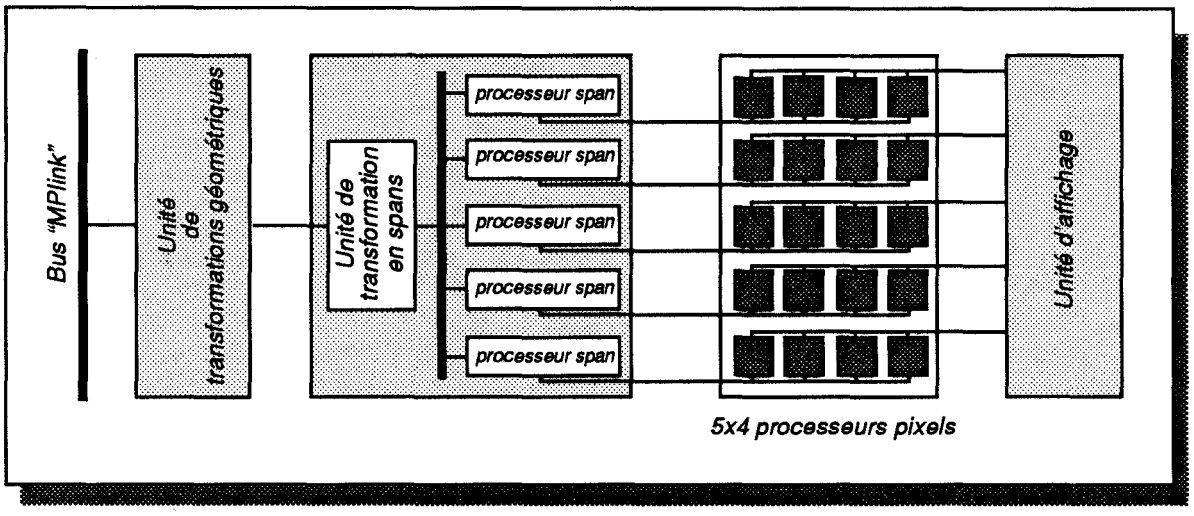


Figure 5.11 ***Schématisation du module graphique de la Silicon Graphics GTX d'après [Akele 88]***

Ces modules sont, respectivement:

- l'unité de transformations géométriques: celle-ci permet d'effectuer les transformations géométriques (changement de repère, fenêtrage, ...) préludant à l'affichage d' une facette qui lui parvient sur le *"MPlink"*. Elle est composée de 5 composants *"Geometry Engine"* [Clark 82] pipelinés, délivrant chacun une puissance de 20 MFlops. Cette unité fournit, en sortie, une facette en coordonnées écran, disposant de tous les attributs nécessaires à l'affichage (valeurs d'éclairage, de profondeur, de transparence, ...).
- l'unité de conversion des facettes en pixels: cette unité a pour objet de découper la facette qu'elle reçoit en segments verticaux (*spans*), correspondant chacun à une colonne écran, et de générer les pixels appartenant à ces différents segments. La première étape est effectuée par l'unité de transformation en spans, qui alimente ensuite 5 processeurs spans, gérant chacun $1/5^{ème}$ des colonnes de l'écran. Chacun d'entre eux calcule l'ensemble des pixels du span qu'il reçoit (valeurs de couleur, de profondeur, ...). Ces pixels sont ensuite envoyés vers l'unité de génération des pixels.
- l'unité de génération des pixels: le rôle de cette unité est de générer les pixels dans la mémoire de trame. Ceci est effectué au moyen de 20 processeurs spécialisés *"Image Engine"*, gérant chacun $1/20^{ème}$ de la mémoire de trame. Chacun de ces processeurs effectue l'élimination des parties cachées sur les pixels qu'il reçoit des processeurs spans, et est aussi capable de traiter les transparences, par application de combinaisons linéaires sur les couleurs présentes dans un pixel.
- l'unité d'affichage: elle permet un accès parallèle à la mémoire de trame, par l'intermédiaire de 5 processeurs *"Multimode Graphics"*, et effectue le transfert des pixels vers la sortie vidéo.

Le parallélisme exploité par cette unité est en premier lieu un parallélisme objet entre les différents étages du module graphique. Un parallélisme pixel relativement intéressant apparait dans l'unité de génération des pixels, puisque cette unité permet de générer simultanément 20 pixels dans la mémoire de trame.

Les performances annoncées pour cette machine sont d'environ 137000 triangles par seconde, pour des triangles de surface moyenne 50 pixels. Dans le cas de polygones de surface moyenne

100 pixels, les performances sont de l'ordre de 100000 polygones par seconde. Ces chiffres s'entendent avec ombrage de Gouraud et élimination des parties cachées.

L'algorithme

Le processus de calcul d'un hémicube est similaire au processus d'affichage d'une image sur un écran. Baum propose donc d'utiliser le module graphique des Silicon pour calculer les hémicubes, tandis que le reste des calculs sera dévolu aux différents processeurs disponibles. L'algorithme fonctionne alors selon un mode *Producteur/Consommateur*, les tâches de chacun d'entre eux étant:

- Producteur: celui-ci a pour tâche de calculer un hémicube, c'est à dire d'effectuer la projection de toutes les facettes sur les 5 faces utilisées. Il est constitué de deux parties: le module graphique, qui effectue l'étape de calcul proprement dite, et l'un des processeurs de la Silicon, qui est chargé de son pilotage (envoi des données).

  Pour effectuer le calcul d'un hémicube plutôt que l'affichage d'une image, les valeurs de couleur Rouge, Vert et Bleu sont remplacées par un entier 32 bits, qui représente l'identité de la facette, et le rendu est effectué en éclairage constant (*flat shading*), de manière à ne pas faire d'interpolation sur cette valeur. Après projection de l'ensemble des facettes, l'hémicube est transferé de la mémoire de trame vers la mémoire partagée de la station.

  Il faut cependant noter que la projection de la scène doit être effectuée 5 fois par le module graphique, une par face de l'hémicube.

- Consommateur: son rôle est d'effectuer toutes les tâches, autres que le calcul de l'hémicube, c'est à dire le calcul des vecteurs de facteurs de forme déduits d'un hémicube, la mise à jour des radiosités et le choix d'une nouvelle facette émettrice. Tous les processeurs disponibles sur la machine, hormis le processeur contrôlant le module graphique, sont utilisés en parallèle pour ces tâches.

  Pour le calcul des facteurs de forme, l'hémicube est découpé en un grand nombre de blocs de proxels disjoints. Chaque processeur, dynamiquement, reçoit un bloc, et le parcourt pour mettre à jour une copie locale du vecteur de facteurs de forme (la duplication du vecteur de facteurs de forme évite les problèmes de contention qui se produiraient, si ce vecteur n'était disponible qu'en un seul exemplaire dans la mémoire centrale de la machine). Après traitement d'un bloc, un processeur demande un nouveau bloc, ceci jusqu'à ce que tous les blocs aient été traités. Ce fonctionnement asynchrone correspond, d'une part, au mode de fonctionnement des différents processeurs de la Silicon, et d'autre part au fait que Baum autorise le calcul analytique des facteurs de forme en utilisant les informations de visibilité fournies par l'hémicube (voir chapitre 2, paragraphe X). Dans ce cas, il devient évident que les processeurs n'utilisent pas leurs blocs de proxels à la même vitesse, puisque celle-ci dépend de la proportion de calculs analytiques effectués dans chacun d'entre eux. Après traitement de tous les blocs de proxels, les vecteurs de facteurs de forme sont additionnés en mémoire centrale.

  Après calcul du vecteur de facteurs de forme, celui-ci est tronçonné en blocs de même taille, chaque bloc étant alloué à un processeur différent. Ceux-ci utilisent alors les informations de facteurs de forme pour mettre à jour les radiosités des facettes, et choisir la nouvelle facette émettrice.

Parallèlement au calcul des facteurs de forme, l'implantation de Baum autorise l'utilisation simultanée du module graphique pour l'affichage d'une image en fonction des radiosités déjà calculées. Dans ce cas, l'un des processeurs de la Silicon est réservé au processus de visualisation, l'accès aux valeurs de radiosité se faisant par l'intermédiaire de la mémoire partagée.

Résultats

Baum effectue une description précise des performances théoriques de son implantation en fonction de nombreux paramètres, tels que le nombre de facettes, le nombre de processeurs ou la résolution de l'hémicube. Dans la mesure où ces performances dépendent des performances matérielles de la machine, il introduit celles-ci dans son modèle théorique, puis les évalue de

manière pratique, sur différents jeux de tests. Il montre que les résultats calculés par le modèle théorique sont très proches des valeurs effectivement obtenues.

A titre d'exemple, son implantation effectue une passe d'émission par seconde, pour une base de données d'environ 8000 facettes et un hémicube de résolution 150×150. De plus, ces chiffres sont obtenus en autorisant l'utilisation du module graphique pour l'affichage d'une image, à une fréquence moyenne de 4 à 8 images par seconde.

Les performances maximales sont obtenues lorsque le calcul des facteurs de forme est fait uniquement en fonction des informations de visibilité recueillies en chaque proxel (pas de part analytique), et que le module graphique n'est utilisé que pour la génération de l'hémicube. Si l'évaluation des facteurs de forme nécessite une seconde phase analytique, Baum montre que les processeurs présents dans la Silicon ne peuvent plus consommer suffisamment rapidement les informations produites par le module graphique, et que celui-ci sera sous-exploité (dans ce cas, l'affichage d'images intermédiaires peut être autorisé). Ceci provient à la fois de l'augmentation de la complexité des calculs engendrés, mais aussi des conflits d'accès à la mémoire partagée (le calcul analytique des facteurs de forme nécessite en effet d'avoir accès à la description géométrique des facettes). Inversement, si le calcul des facteurs de forme ne nécessite que peu de calculs analytiques supplémentaires, l'utilisation du module graphique pour l'affichage d'images intermédiaires créera une sous-exploitation des processeurs chargés de la consommation des proxels.

Il est aussi important de tenir compte de la taille de la base de données et de la résolution de l'hémicube utilisé. Le premier paramètre influe directement sur les performances du module graphique, tandis que le second a une très forte influence sur les processeurs générant les facteurs de forme. Ainsi, si la résolution est trop importante, les processeurs ne peuvent effectuer le calcul des facteurs de forme et la mise à jour des radiosités suffisamment rapidement pour pouvoir traiter un nouvel hémicube généré par le module graphique. Inversement, un nombre trop élevé de facettes pour une résolution trop faible ne permettra pas d'alimenter les processeurs de la Silicon suffisamment rapidement.

5.3.3 Conclusion

Le concept de mémoire partagée, qui est utilisé dans les deux implantations décrites dans cette partie, permet d'éviter le problème de la duplication de la base de données qui est utilisée dans les approches décrites au paragraphe 5.2. A quantité de mémoire égale, il est donc possible d'utiliser des scènes de plus forte complexité. Ce concept facilite d'autre part considérablement la gestion des données, puisque celles-ci ne sont présentes qu'en un seul endroit, et le mécanisme d'accès n'est pas géré par le programmeur.

L'implantation proposée sur la mémoire virtuelle KOAN ne semble cependant pas être une solution satisfaisante dans sa forme actuelle, du fait de l'absence du mécanisme de cohérence sur laquelle elle devrait être basée, pour assurer une utilisation efficace de la notion de page. Il semble que ce mécanisme soit cependant présent dans l'algorithme de radiosité, et que son exploitation sur ce type d'architecture devrait offrir des performances intéressantes.

La solution développée sur la gamme de stations Silicon présente des performances élevées quant à la vitesse d'application d'une phase d'émission. Celles-ci sont obtenues par la conjugaison d'une solution matérielle dédiée pour la génération des informations de visibilité, et d'une solution logicielle parallèle pour le calcul des informations énergétiques. Néanmoins, l'obtention de la précision lors du calcul des facteurs de forme, nécessite le recours à une solution logicielle très coûteuse, qui est de plus pénalisée par l'utilisation d'une mémoire partagée. Des déséquilibres de charge importants peuvent d'autre part se produire entre les deux modules de l'application, en fonction de la taille de la base de données ou du nombre de directions d'échantillonnage. Notons, d'autre part, que cette solution n'est pas extensible, et que ses performances maximales dépendent principalement de celles du module graphique.

5.4 Utilisation d'une base de données répartie

La méthode de radiosité nécessite la manipulation de quantités très importantes de données, le nombre de facettes nécessaires à la modélisation de scènes de plus en plus complexes ne demandant qu'à croître. La parallélisation de cet algorithme conduit donc à répartir les données sur les différents processeurs d'une machine parallèle, la quantité de mémoire disponible par processeur ne pouvant être jugée suffisante pour mémoriser l'intégralité d'une scène. D'autre part, ces données ne sont pas indépendantes les unes des autres, puisque le principe de l'algorithme de radiosité est justement de prendre en compte les interactions de lumière qui se produisent entre elles.

Cette dépendance se traduit par la nécessité d'échanger des informations entre les différents processeurs utilisés dans l'application, de manière à permettre le calcul de ces interactions. Afin de limiter l'influence des communications sur le temps de calcul complet des radiosités, il est important de réduire au maximum le chemin que les informations empruntent, en utilisant une topologie adaptée à la fois à l'algorithme mis en oeuvre (méthode projective ou tracé de rayons) et au type d'informations échangées.

Nous allons détailler, dans cette partie, les différentes approches qui ont été étudiées, en fonction de la méthode de calcul des facteurs de forme utilisée. Pour chacune d'entre elles, nous définirons le type des messages échangés, et la configuration du réseau qui a été choisie.

5.4.1 Calcul de la matrice complète des facteurs de forme

Chalmers [Chalm 89], [Chalm 90] propose une implantation du calcul de l'ensemble des lignes de la matrice des facteurs de forme sur un réseau de transputers. La base de données est répartie équitablement sur l'ensemble des processeurs, et chaque processeur a le même nombre de lignes de la matrice à calculer. Enfin, le calcul des facteurs de forme est effectué par l'intermédiaire d'un hémicube.

Les informations à échanger sont donc des facettes qui se trouvent réparties sur les différents processeurs, l'ensemble des facettes devant, a priori, être importé par chaque processeur. Pour que le chemin à parcourir entre les différents processeurs soit le plus court possible, le réseau est configuré sous la forme d'un graphe appelé *AMP* ("A Minimum Path"), pour lequel Chalmers effectue une comparaison avec différentes autres configurations [Chalm 91]. Ce graphe correspond en fait à un arborescence dont toutes les feuilles sont connectées entre elles.

Deux scènes différentes ont été testées sur cet algorithme, contenant respectivement 112 et 448 facettes. Les résultats obtenus, pour des configurations variant de 1 à 8 processeurs, montrent une efficacité importante, de l'ordre de 90%, quel que soit le nombre de processeurs utilisés. Ceci provient essentiellement du nombre peu élevé de processeurs présents dans le réseau, la longueur du chemin entre les processeurs étant alors faible (1,5 en moyenne). L'augmentation de taille du réseau doit conduire à une baisse de performance. Cette approche est néanmoins à comparer favorablement à celle de Purgathofer, décrite au paragraphe 5.2.1, où la base de données complète transite successivement par chaque processeur. Dans ce cas, une notion de synchronisation intervient entre les processeurs, générant une perte d'efficacité plus importante que dans l'approche de Chalmers, où les processeurs sont totalement asynchrones.

Il faut cependant rappeler que ce type d'approche, où la matrice complète des facteurs de forme est calculée, est réservée aux petites base de données, ou aux réseaux de très grande taille, du fait de la quantité mémoire importante nécessaire à la représentation de la matrice des facteurs de forme.

5.4.2 Calcul parallèle de plusieurs émissions

Pour calculer la contribution lumineuse d'une facette émettrice sur l'ensemble des objets d'une scène, nous avons détaillé deux classes de méthodes fort différentes. Chacune d'entre elles a donné lieu à diverses propositions quant aux schémas de parallélisation qui lui sont applicables dans le cadre de la radiosité progressive, en considérant que la base de données est répartie entre les différents processeurs.

Approches projectives

Les approches que nous allons détailler dans ce paragraphe utilisent toutes le principe des méthodes projectives décrites au chapitre 2. Bien qu'elles aient été principalement implantées avec l'algorithme de l'hémicube, leur principe est bien évidemment applicable à tout autre algorithme du même type.

Dans le cas des algorithmes projectifs, deux types d'informations sont susceptibles d'être échangées entre les processeurs: les éléments de la base de données proprement dite, c'est à dire les facettes réparties sur le réseau ; et la surface de projection, qui représente l'ensemble des informations de visibilité nécessaires à une facette émettrice pour répartir son énergie. Ces deux approches sont décrites ci-dessous.

Déplacement de la base de données

Dans ce type d'approche, les calculs d'émission restent locaux à chaque processeur; dans la mesure où le calcul d'une émission nécessite de considérer la plupart des facettes de l'environnement, et que ces facettes sont réparties sur l'ensemble du réseau, des communications sont donc obligatoires entre les processeurs. Ces communications vont principalement prendre la forme de blocs de facettes échangés entre les divers processeurs, au gré de leurs besoins. Trois implantations suivant ce schéma général, deux d'entre elles utilisant un contrôle réparti, et la troisième un contrôle centralisé, sont détaillées ci-dessous.

Contrôle réparti

Chalmers implante cette solution sur un réseau de type AMP, chaque processeur disposant d'une partie de la base de données, et ayant pour tâche de calculer une phase d'émission complète [Chalm 91]. La principale caractéristique de son approche est d'utiliser un mode de fonctionnement totalement réparti, chaque processeur travaillant de manière totalement indépendante des autres. Ainsi, un processeur choisit une facette émettrice uniquement parmi celles qui sont présentes dans sa propre mémoire, y applique un hémicube et effectue l'étape de mise à jour des radiosités.

Du fait de la répartition des facettes, une certaine coopération entre les processeurs est néanmoins nécessaire lors des étapes de calcul des hémicubes et de mise à jour des radiosités.

- Le calcul d'un hémicube nécessite, a priori, la projection de l'intégralité de la scène. Chaque processeur doit donc, régulièrement, réclamer les facettes dont il ne dispose pas dans sa mémoire locale. De manière à éviter des temps d'attente inutiles pour le processeur qui envoie la requête, celle-ci est émise dès qu'un bloc arrive, afin d'anticiper le besoin qui se fera sentir lorsque le traitement du bloc courant est terminé. Si le nouveau bloc n'est pas arrivé, le processeur traite alors les facettes locales qui lui restent à projeter. Ce traitement est stoppé dès que le bloc demandé est arrivé.
- Lorsque le calcul de l'hémicube est terminé, le processeur calcule le vecteur de facteurs de forme correspondant. Notons que les auteurs autorisent l'utilisation de la méthode analytique décrite au paragraphe 2.6.2, page 49, lorsque le besoin de précision se fait sentir (violation de l'hypothèse de proximité). Dans ce cas, des communications supplémentaires sont nécessaires pour récupérer l'ensemble des facettes visibles au travers de chaque proxel.

  La mise à jour des radiosités, qui suit le calcul des facteurs de forme, pose aussi un problème, puisque les différentes facettes concernées par cette mise à jour sont réparties au travers du réseau. Le vecteur de facteurs de forme est alors diffusé à l'ensemble des processeurs, à charge pour ceux-ci de l'utiliser pour la mise à jour des radiosités de leurs facettes locales.

La notion de contrôle centralisé est cependant utilisée pour déterminer l'arrêt de l'algorithme. Lorsqu'un processeur sélectionne une nouvelle facette émettrice, la valeur de sa radiosité latente est comparée à une valeur de seuil fixée par l'utilisateur. Si cette valeur est plus petite que la valeur du seuil, le processeur informe le contrôleur du réseau (qui sert de liaison entre le réseau et le "monde" extérieur) qu'une convergence locale est atteinte. Lorsque le contrôleur est informé que tous les processeurs ont atteint un état de convergence locale, il vérifie qu'il

n'existe plus de messages transitant dans le réseau, et informe les processeurs de la fin du calcul.

Les principales informations échangées entre les processeurs, lors des différentes étapes de calcul, sont résumées sur la figure ci-dessous (Figure 5.12).

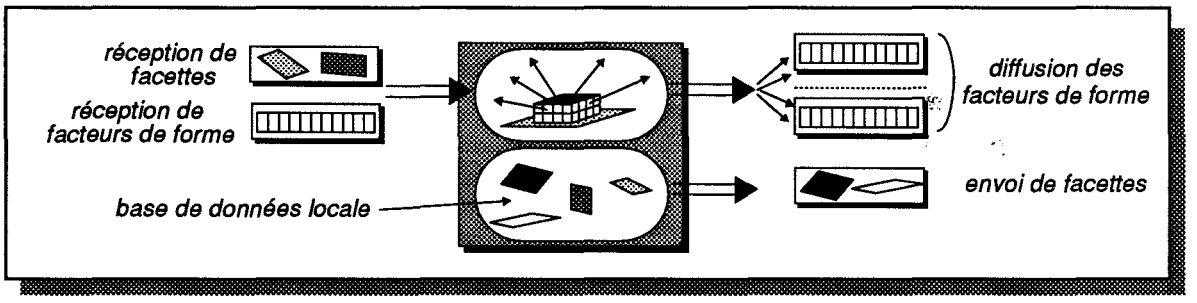


Figure 5.12 ***Principaux échanges de données entre les processeurs dans l'implantation de Chalmers***

Dans la mesure où le nombre de messages échangés est très important, Chalmers étudie les différentes configurations de réseaux permettant de réduire la longueur du chemin à parcourir entre deux processeurs quelconques. Il compare les résultats obtenus sur un anneau, un tore et une AMP. Les résultats obtenus quant au tore et à l'AMP apparaissent dans la Figure 5.13.

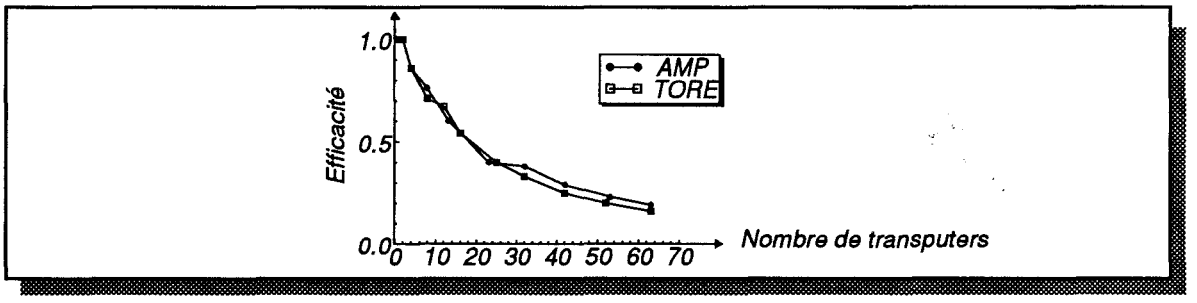


Figure 5.13 ***Evolution de l'efficacité en fonction du nombre de processeurs, pour deux configurations différentes du réseau (d'après [Chalm 91])***

Ces courbes traduisent l'efficacité de l'implantation sur les deux types de réseau, en fonction du nombre de processeurs utilisé. Ces résultats ont été obtenus sur une base de données constituée de 448 facettes. Des résultats similaires apparaissent dans le cas d'une base de données plus importante (2268 facettes).

S'ils montrent un léger avantage pour le réseau AMP, qui possède une longueur moyenne entre deux processeurs plus courte que dans le cas du tore, ils soulignent très nettement que le coût de communication d'une telle approche devient excessif dès lors que le nombre de processeurs augmente. L'étude des résultats présentés par Chalmers montre qu'une accélération maximale pour ces deux solutions se situe aux environs de 30 processeurs, avec une accélération d'un facteur 12 environ. Les performances obtenues avec une configuration en anneau n'ont pas été représentées, car elles sont beaucoup plus faibles que dans les deux autres configurations. Ceci provient du fait que, sur ce type de configuration, il n'existe qu'un seul chemin de communication que tous les messages doivent nécessairement emprunter. L'auteur note ainsi que l'anneau est totalement saturé au delà de 40 processeurs.

La très forte diminution de l'efficacité de cette approche quand le nombre de processeurs augmente, provient essentiellement de l'énorme quantité de messages qui doivent être échangés entre les processeurs. A chaque nouvelle application d'un hémicube, l'intégralité de la base de données transite par un grand nombre de processeurs, réduisant d'autant leur disponibilité pour les tâches de calcul. Il faut d'autre part noter que le parallélisme utilisé induit le calcul simultané de N hémicubes, où N représente le nombre total de processeurs disponibles. Dans la mesure où aucun mécanisme de tampon n'est envisagé sur les processeurs traversés par un bloc de facettes (ce bloc pourrait être utilisé par le processeur traversé, sans

qu'il ait à en faire la demande), le calcul simultané de N hémicubes multiplie le transit de la base de données par un facteur N.

Rappelons aussi que l'utilisation du calcul analytique des facteurs de forme nécessite des communications supplémentaires importantes, et que d'autre part il est nécessaire de diffuser chaque vecteur de facteurs de forme à l'ensemble des processeurs (un vecteur de facteurs de forme comporte autant d'éléments que de facettes utilisées dans la base de données).

Notons enfin deux effets secondaires de l'augmentation du nombre de processeurs: l'augmentation de la distance moyenne à parcourir par un message, et la diminution du nombre de facettes mémorisées localement. Le premier point génère des temps d'attente plus importants pour le retour des facettes demandées. Comme ce temps est mis à profit pour traiter les facettes locales, celles-ci sont "consommées" beaucoup plus rapidement lorsque le nombre de processeurs augmente. Quand toutes ses facettes locales ont été traitées, le processeur devient inactif. Le deuxième point induit une multiplication des requêtes.

Il faut aussi noter que la gestion totalement répartie du choix des facettes émettrices (un processeur choisit la nouvelle facette émettrice uniquement parmi ses propres facettes), entraîne un ralentissement de la convergence, puisqu'une facette sélectionnée n'est plus forcément celle qui possède la plus grande radiosité latente de la base de données. De même, ce mécanisme génère un déséquilibre de charge entre les processeurs quand certains d'entre eux possèdent peu de facettes susceptibles d'être sélectionnées en tant que facette émettrice par rapport aux autres processeurs. Le temps de calcul global dépend alors de la vitesse du processeur le plus "lent".

Une seconde approche, basée sur ce principe de déplacement de la base de données a été proposée par Thomin [Thomi 93], sur un réseau configuré en anneau. Chaque processeur a en charge le calcul d'un hémicube différent, et les données nécessaires à ces calculs transitent successivement par chacun des processeurs, le long de l'anneau. Cette approche diffère donc de celle proposée par Chalmers sur un anneau, de par le fait que les processeurs fonctionnent en mode synchrone, et communiquent tous dans le même sens. Un cycle de l'algorithme se décompose alors en 4 phases distinctes:

1. Sélection d'une facette émettrice par processeur;
2. Circulation des facettes au travers de l'anneau pour le calcul des hémicubes;
3. Calcul des facteurs de forme à partir des hémicubes générés lors de l'étape précédente;
4. Circulation des facettes pour la mise à jour des radiosités, en fonction des facteurs de forme précédemment calculés.

Nous avons vu que l'élection totalement répartie des facettes émettrices posait un certain nombre de problèmes ayant une influence directe sur le nombre d'itérations nécessaires pour atteindre la convergence. Thomin propose donc une méthode différente de choix des facettes émettrices, permettant d'assurer que chaque processeur effectue un calcul sur l'une des P facettes possédant la plus grande énergie latente, où P représente le nombre de processeurs disponibles. Lors de la phase de sélection d'une facette émettrice, chaque processeur sélectionne les P facettes de sa base de données qui possèdent la plus grande énergie latente. Une copie de cette liste est transmise au processeur suivant dans l'anneau. Chaque processeur dispose dès lors de deux listes de taille P. Il fusionne ces deux listes, les trie et ne garde que les P facettes les plus lumineuses. Puis il passe à nouveau une copie de cette liste à son successeur. Après P étapes de ce type, chaque processeur dispose de la même liste, les facettes présentes dans cette liste étant, de plus, celles qui possèdent la plus forte énergie latente. Chaque processeur utilise alors la facette de la liste qui correspond à sa position dans l'anneau.

Les autres étapes restent classiques, et concernent tout d'abord le transit de chaque base de données locale par l'ensemble des processeurs, de manière à compléter les informations de visibilité de chaque hémicube. Le calcul des facteurs de forme est alors effectué localement à chaque processeur, et est suivi d'une nouvelle phase de transit de toutes les facettes le long de

l'anneau, de manière à ce que leur radiosité puisse être mise à jour pour chacune des P facettes émettrices. Le fonctionnement de cette approche est schématisé sur la Figure 5.14.

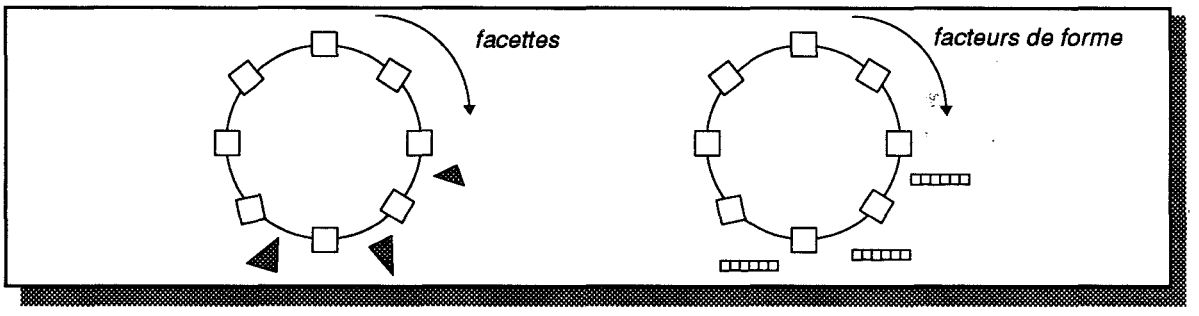


Figure 5.14 ***Schémas de communications sur l'anneau***

A noter que, de la même manière que l'approche de Chalmers, un contrôleur est inséré dans l'anneau, afin de pouvoir assurer une liaison avec l'extérieur et détecter la fin du calcul.

L'avantage principal par rapport à l'approche précédente est de permettre un coût constant en terme de nombre d'étapes de communications effectuées, égal au nombre de processeurs, tant pour l'étape de calcul des hémicubes que pour l'étape de mise à jour des radiosités. En effet, pour la première étape, comme pour la seconde, P étapes de communication de longueur unitaire (processeurs voisins) sont utilisées. Dans l'implantation de Chalmers, en supposant que chaque processeur reçoit en même temps une nouvelle base de données, P étapes sont aussi nécessaires, mais de longueur plus importante. En effet, dans cette implantation, les processeurs font une demande explicite des facettes dont ils ont besoin. La longueur du chemin dépend donc de la taille du réseau. Il en est de même pour la diffusion des facteurs de forme.

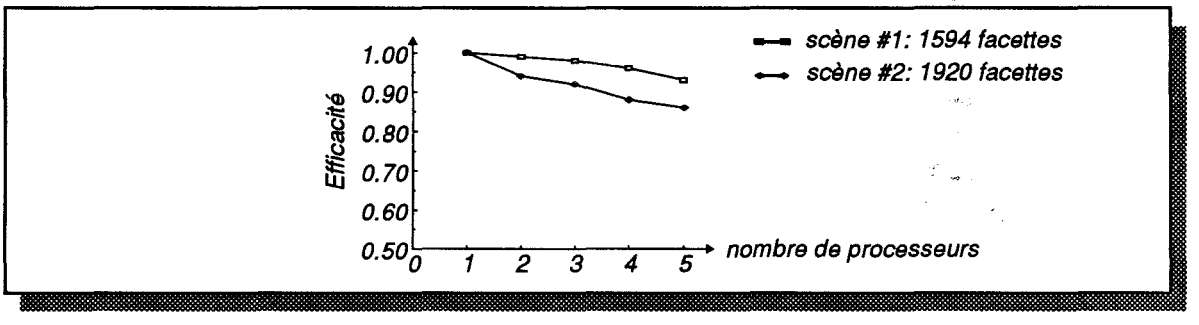


Figure 5.15 ***Evolution de l'efficacité en fonction du nombre de processeurs***

Nous avons représenté sur la Figure 5.15, l'évolution de l'efficacité de cette implantation pour deux scènes différentes. Bien que celle-ci reste élevée, une décroissance très nette apparait, d'autant plus forte que le nombre de facettes augmente. Ceci provient essentiellement du mode de fonctionnement de cette approche. En effet, bien que les processeurs effectuent leurs calculs d'élimination des parties cachées de manière indépendante, le mode de transmission des données implique une resynchronisation des processeurs à la fin du traitement d'un bloc de facettes. De ce fait, l'ensemble du réseau fonctionne à la vitesse du processeur le plus lent. Ce phénomène ne pourra vraisemblablement que croître avec le nombre de processeurs ou l'augmentation du nombre de facettes.

Il est d'autre part intéressant de noter que l'efficacité obtenue est supérieure à celle de l'implantation de Chalmers, pour un nombre de facettes supérieur. Ceci s'explique tout d'abord par la limitation de la longueur des chemins de communication, mais surtout par le fait que seule une opération d'élimination des parties cachées est utilisée pour le calcul des facteurs de forme. Si une seconde passe analytique est appliquée, comme c'est le cas dans l'implantation de Chalmers, il faut alors effectuer un nouveau passage des facettes par tous les processeurs. De plus, la proportion des calculs analytiques étant fort variable d'une facette émettrice à l'autre, il est fort probable que le mode de fonctionnement synchrone imposé par l'anneau devienne très pénalisant.

Utilisation d'un contrôle centralisé

Feda [Feda 91] implante une solution similaire à celle qui vient d'être présentée, au mode de contrôle prêt. L'un des processeurs du réseau joue en effet le rôle de contrôleur pour la phase de choix des facettes émettrices. Lorsqu'un processeur a terminé le calcul d'un hémicube, et l'envoi des messages de mise à jour des radiosités, il choisit une nouvelle facette émettrice parmi ses facettes locales. Plutôt que de commencer immédiatement le calcul d'un nouvel hémicube sur cette facette, il en communique la description au contrôleur, qui gère une table des facettes émettrices à calculer. Lorsqu'il reçoit une facette émettrice, le contrôleur la range dans sa table. Il recherche ensuite la facette qui possède la plus grande énergie latente dans la table, et envoie son identité au processeur demandeur. Celui-ci en demande alors la description au processeur qui la détient, puis y applique un hémicube. Le processeur qui a envoyé la facette émettrice en choisit alors une nouvelle dans sa base de données, puis la renvoie au contrôleur de manière à combler la place laissée vide dans sa table. Ce fonctionnement est schématisé sur la Figure 5.16.

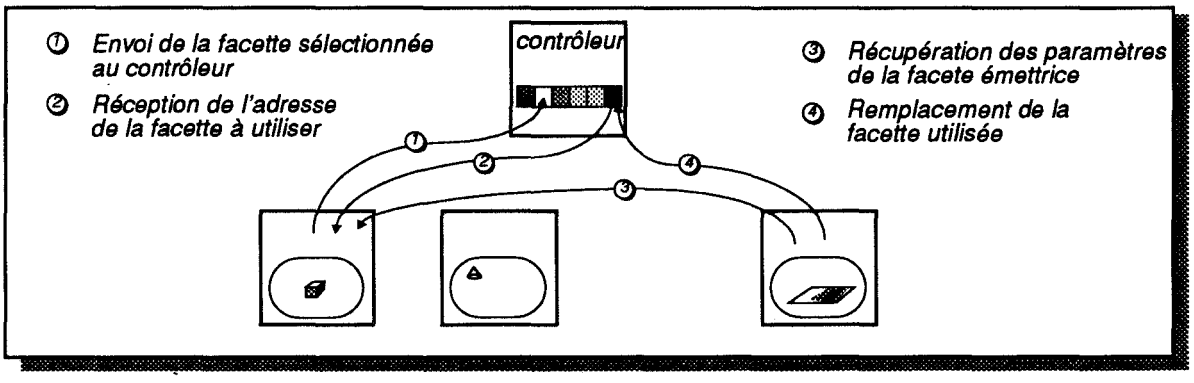


Figure 5.16 ***Principe de la gestion du choix d'une facette émettrice***

Cette gestion permet d'éviter le problème de l'implantation précédente concernant le déséquilibre de charge, puisque les processeurs effectuent une "concertation" sur le choix des facettes émettrices à calculer et participent ainsi tous aux calculs, même s'ils ne disposent plus de facettes ayant à émettre dans leur propre ensemble de facettes. Elle permet de plus d'accélérer la convergence, toujours par rapport à l'approche précédente, puisque la facette choisie dans la liste est toujours celle qui possède la plus grande énergie latente à un instant donné.

Feda étudie un certain nombre de paramètres susceptibles de faire varier les temps de calculs obtenus, tels que la taille optimale des blocs de facettes à échanger (des messages de petite taille nécessitent, entre autres, des communications plus fréquentes et génèrent donc des temps de gestion des protocoles plus importants), ou telle que la possibilité de calculer simultanément plusieurs hémicubes sur le même processeur (lorsqu'il y a assez de mémoire disponible) de manière à amortir les importations de blocs de facettes. La figure ci-dessous présente l'efficacité obtenue en utilisant de 1 à 8 processeurs, sur deux scènes de complexités différentes. Les accélérations maximales correspondantes sont d'environ 7.8 pour la plus petite des deux scènes, et 5.3 pour la plus grande, dans le cas de l'utilisation de huit transputers.

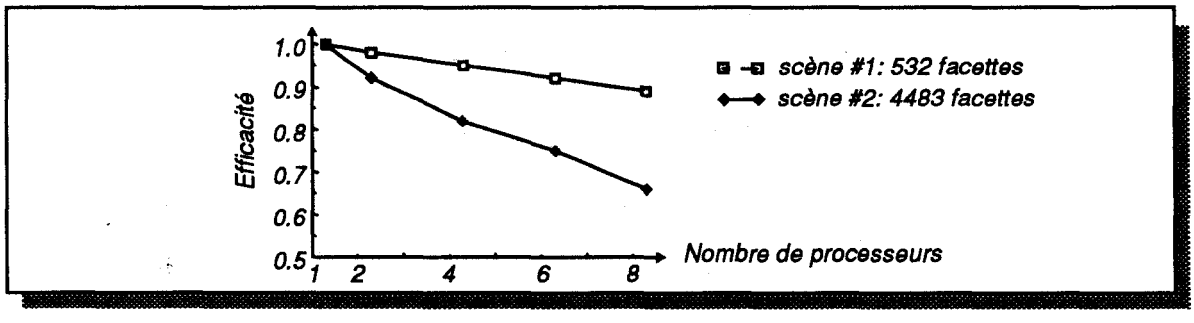


Figure 5.17 ***Evolution de l'efficacité en fonction du nombre de processeurs ([Feda 91])***

Les performances obtenues, au vu de ces deux courbes, restent relativement importantes, même lorsque la taille de la base de données augmente. Du fait de l'augmentation des communications, plus la taille des scènes devient importante et plus l'efficacité décroît. Il faut cependant noter que le nombre de processeurs reste faible, ce qui permet d'obtenir une distance moyenne entre les processeurs assez courte, de l'ordre de 1.46 sur la configuration AMP utilisée. L'augmentation du nombre de processeurs, comme dans l'implantation de Chalmers, devrait rapidement conduire à une diminution importante des performances, du fait du surcoût de communication induit pour la récupération des facettes externes.

Bien qu'une comparaison directe avec l'approche précédemment développée sur un réseau AMP soit délicate (scènes différentes), il semble intéressant de noter que l'efficacité obtenue par Chalmers pour une scène utilisant 448 facettes décroît beaucoup plus rapidement dans l'intervalle variant de 1 à 8 processeurs que l'efficacité obtenue par Feda sur la scène contenant 532 facettes (efficacités respectives de 0.75 et 0.92 pour 8 processeurs). Il y a deux raisons principales à cela. Tout d'abord, Chalmers utilise la notion de calcul analytique des facteurs de forme, notion qui n'est pas implantée dans l'algorithme de Feda. Celui-ci fait donc l'économie des communications supplémentaires qui apparaissent avec l'utilisation de cette notion. Enfin, Chalmers n'utilise pas de contrôle centralisé pour le choix d'une nouvelle facette émettrice. Des déséquilibres de charge se produisent donc, dès lors qu'un processeur ne dispose plus, dans sa base de données locale, d'une facette possédant une radiosité latente suffisante pour être sélectionnée.

Le principal problème des deux approches utilisant un réseau AMP que nous venons de décrire est de nécessiter le déplacement de toute la base de données pour chaque hémicube calculé. Dans la mesure où celle-ci est équitablement distribuée entre les processeurs, plus la taille du réseau augmente, et plus le chemin moyen à parcourir par une facette devient important. La solution proposée par Thomin est intéressante pour un nombre faible de processeurs, mais son fonctionnement synchrone pénalise lourdement une configuration plus importante.

Nous terminerons donc ce paragraphe consacré aux implantations utilisant le déplacement des facettes entre processeurs pour effectuer l'étape de projection, par l'analyse d'une proposition émise par Feda qui nous semble intéressante. Il semble raisonnable de considérer que, pour un grand nombre de scènes, la partie de la base de données mémorisée en chaque processeur n'occupe pas toute la mémoire effectivement disponible (sans oublier l'espace nécessaire à la représentation de l'hémicube et du tampon d'entrée pour les facettes importées). Ceci signifie qu'il est alors possible, dans chaque processeur, de dupliquer une partie de la base de données, dans le but de réduire les communications.

Cette idée soulève cependant un certain nombre de problèmes annexes. Tout d'abord, cette mémorisation supplémentaire ne pourra être réellement intéressante que si les facettes qui sont mémorisées "en plus" sur un processeur ne sont pas des facettes qui se trouvent sur un processeur voisin (au sens large du terme), mais sur l'un des processeurs les plus éloignés possible. La vérification de cette "propriété" pour chaque processeur permettra alors une réduction maximale des distances moyennes à parcourir par les blocs de facette, un processeur disposant alors directement des informations les plus longues à obtenir. Il sera alors possible aux voisins proches de récupérer les facettes sur ce processeur, plutôt que sur un processeur distant.

Il faut cependant prendre la précaution de distinguer, au sein de chaque processeur, les facettes qui ne représentent qu'une copie permettant de limiter les distances de communication, des facettes originales, correspondant à la répartition équitable entre les processeurs. En effet, lors de la phase de mise à jour des radiosités, seules les facettes originales doivent voir leur radiosité mise à jour, de manière à éviter la non consistance des données et les calculs multiples de la même facette.

Déplacement de la surface de projection

Le deuxième type d'informations pouvant être échangées dans le cas des algorithmes projectifs est la surface de projection elle même, c'est à dire l'ensemble des proxels qui la constituent. Dans ce cas, pour pouvoir être complétée, elle doit nécessairement passer par tous les processeurs sur lesquels la base de données est distribuée. La configuration du réseau la plus

naturelle pour ce genre d'application est alors un pipeline, dont les différentes tâches sont assez simples à déterminer. Chaque processeur du pipeline est chargé de récupérer la surface de projection de son voisin de gauche, d'y projeter les facettes qu'il possède, puis de passer cette surface de projection à son voisin de droite. Un processeur spécialisé, placé en sortie du pipeline, récupère la surface de projection complétée et l'utilise pour calculer le vecteur de facteurs de forme. La phase de mise à jour des radiosités est effectuée, et une nouvelle surface de projection peut être envoyée dans le pipeline pour y être calculée. De manière à augmenter les performances, P surfaces de projection se succèdent en permanence dans le réseau, où P représente le nombre de processeurs disponibles. Ce principe général de fonctionnement est schématisé sur la Figure 5.18 dans le cas de l'hémicube.

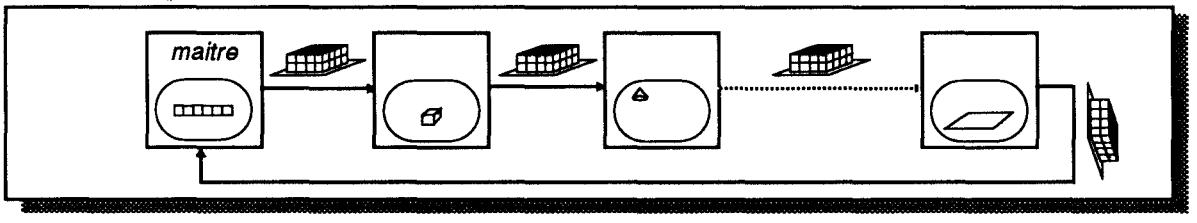


Figure 5.18 ***Le pipeline d'hémicubes***

Une étude plus complète de cette approche peut être trouvée dans [Vilap 92], tandis que son implantation effective a été développée dans [Thomi 93]. La base de données géométriques est répartie sur les différents processeurs de l'anneau, tandis que la totalité de la base de données (informations énergétiques et géométriques) est représentée sur le processeur maître. Ceci permet de centraliser la mise à jour des radiosités, ainsi que le choix de la nouvelle facette émettrice. De plus, la présence des informations géométriques permet de générer la surface de projection qui doit être appliquée sur la facette émettrice (la génération d'une nouvelle surface de projection nécessite de calculer un point d'application, une direction de "visée" et une matrice de changement de repère à appliquer sur les facettes à projeter).

Les résultats obtenus sur un réseau de 5 transputers dans [Thomi 93] montrent une bonne efficacité, très comparable à celles présentées sur la Figure 5.15, page 105. Néanmoins, pour les mêmes raisons, à savoir la nécessité de synchroniser les différents processeurs du pipeline pour traiter une nouvelle surface de projection, l'efficacité de ce type de solution décroît rapidement avec l'augmentation du nombre de processeurs.

Critère de choix

Le choix entre les deux stratégies de transfert de données doit être effectué sur la même architecture, et ne doit tenir compte, a priori, que des quantités d'information qui sont échangées entre les processeurs. En effet, le temps de calcul d'une étape de projection sur l'ensemble des processeurs est identique, que les éléments transférés soient des facettes ou des hémicubes, puisque l'opération à appliquer est exactement la même, et porte sur les mêmes données.

Dans ce cas, le choix se résume à limiter au maximum les temps de communication entre les processeurs, c'est à dire à choisir les informations à transférer qui occupent le moins de place possible.

En prenant pour hypothèse que chaque facette de la scène est un triangle, la quantité minimale de mémoire nécessaire à sa représentation géométrique est de 52 octets (3 sommets constitués chacun de 3 coordonnées réelles; un identificateur de facette codé sur 4 octets, et une normale codée sur 3 coordonnées réelles).

Un proxel contient une information de distance (un réel), et une information permettant de déterminer la facette visible à cet endroit (un identificateur de facette codé sur 4 octets), soit un total de 8 octets (les facteurs de forme élémentaires sont transmis indépendamment de la surface de projection, puisqu'ils ne servent qu'à la mise à jour des radiosités).

Si R est la résolution de l'hémicube, le nombre total de proxels, N_p,est donné par:

$$N_p = 3 \times R^2$$

soit une taille totale T_{hc} de $T_{hc} = 24 \times R^2$ octets.

Il est alors possible d'obtenir le nombre de facettes N_f qui représente la même taille mémoire qu'un hémicube de résolution R. Celui-ci est donné par:

$$N_f \approx \frac{R^2}{2}$$

A titre d'exemple, transférer un hémicube de résolution 100 correspond au transfert de 5000 facettes; le transfert d'un hémicube de résolution 200 à celui de 20000 facettes. Notons que le nombre de facettes obtenu ne correspond pas au nombre total de facettes de la base de données, mais au nombre de facettes présentes sur chaque processeur. En effet, les facettes transitant à un instant donné entre deux processeurs ne représentent pas l'intégralité de la base de données, mais uniquement la partie de la scène qui a été implantée sur un processeur.

Utilisation du tracé de rayons

Nous avons présenté, au chapitre 3, différentes méthodes utilisant le principe du tracé de rayons pour calculer l'énergie reçue par chaque facette. L'une d'entre elles [Menar 92] a été implantée dans le cadre d'une architecture basée sur le concept de mémoire virtuelle partagée, et a été étudiée au paragraphe 5.3.1. Nous allons détailler ici deux autres implantations, utilisant elles aussi le principe du tracé de rayons, mais utilisée dans un contexte de mémoire répartie.

Ces deux implantations possèdent comme point commun d'utiliser la notion de "voxelisation" de l'espace de la scène, qui permet d'optimiser les calculs d'intersection par le suivi incrémental des rayons. Les voxels générés sont répartis sur l'ensemble des processeurs. Les différences essentielles concernent tout d'abord la manière de répartir les voxels sur les processeurs, ainsi que la façon de représenter les données. Les configurations des réseaux utilisés sont elles aussi très différentes, et sont liées essentiellement à la façon dont les données sont distribuées sur les processeurs. Enfin les algorithmes de calcul des échanges lumineux proprement dit, sont eux mêmes différents. Chacun de ces points est abordé lors de la description des algorithmes.

Configuration linéaire des processeurs

Dans la première approche décrite, Guitton [Guitt 91] propose de répartir les voxels utilisés pour le découpage de l'espace de la scène en plusieurs "tranches" d'épaisseurs éventuellement différentes, et d'allouer chacune de ces tranches à un processeur différent. La configuration résultante pour les processeurs est alors une configuration linéaire, permettant les échanges de données dans les deux sens (voir Figure 5.19).

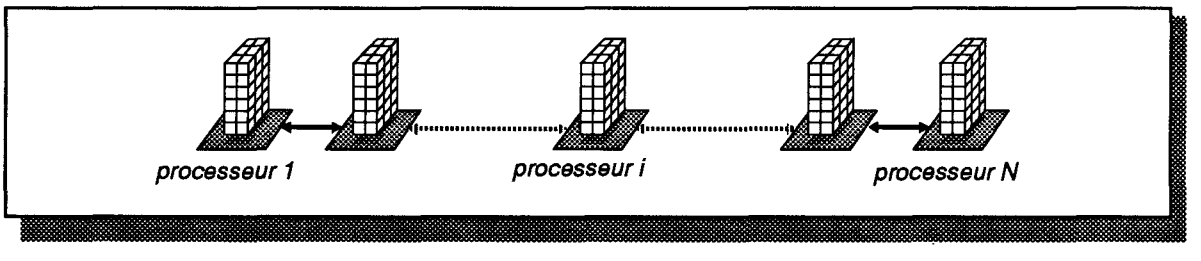


Figure 5.19 ***Répartition des voxels par tranches***

Les données sont représentées, à l'intérieur de chaque tranche, sous forme de *"tuiles"* (tile), permettant de regrouper à la fois les informations géométriques et énergétiques, sans que celles ci soient forcément en même nombre. Lorsqu'une tuile appartient à plusieurs tranches différentes, elle est dupliquée dans chaque tranche, mais une seule d'entre elle (la tranche propriétaire) est autorisée à effectuer des opérations de mise à jour des radiosités.

La structure de tuile provient de la constatation que l'échantillonnage des caractéristiques géométriques et énergétiques d'un objet n'a pas forcément à être appliqué aux mêmes endroits ni en même nombre. Par exemple ([Guitt 91]), un mur peut être représenté géométriquement

par une seule facette, alors que de nombreuses facettes sont nécessaires pour capter les variations de radiosité sur l'ensemble de sa surface. La notion de tuile permet alors de concilier les besoins respectifs de chacune des deux informations, en générant un découpage différent pour chacune, tout en conservant des points d'accès communs aux deux découpages.

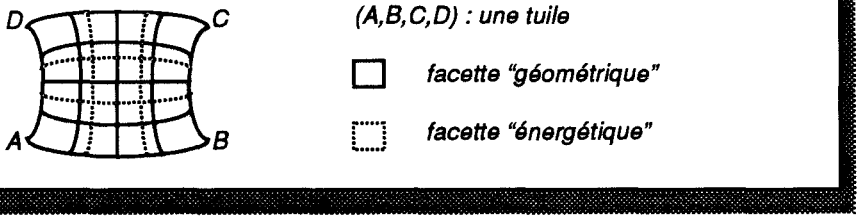


Figure 5.20 ***Exemple de recouvrements des informations géométriques et énergétiques au sein d'une tuile***

L'utilisation d'une représentation paramétrique des coordonnées au sein d'une tuile permet de retrouver les facettes géométriques et énergétiques auxquelles appartient un point qui se trouve à l'intérieur de la tuile (par exemple le point d'intersection de la tuile avec un rayon).

Deux algorithmes différents de mise à jour des radiosités sont envisagés sur cette architecture:

- Méthode stochastique: on y considère la propagation de nombreux rayons directement issus d'une tuile émettrice. En fonction de l'énergie à émettre, l'hémisphère des directions d'émission est échantillonné par un nombre N de rayons, possédant chacun une même quantité d'énergie.
- Méthode déterministe: chaque tuile de l'environnement envoie un rayon vers la tuile émettrice, dont le but est de déterminer si cette tuile est visible depuis la source.

Ces deux méthodes sont schématisée sur la figure ci-dessous (Figure 5.21).

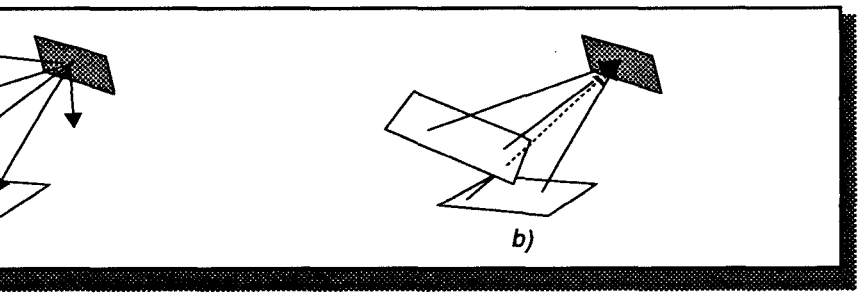


Figure 5.21 ***Calcul de l'éclairage par les méthodes stochastique (a) et déterministe (b)***

La parallélisation de ces méthodes sur la configuration linéaire déjà décrite suit alors sensiblement le même principe. Pour chaque processeur, la tuile la plus énergétique est recherchée.

Dans le cas de la première méthode, chaque processeur détermine, de manière stochastique, un ensemble de rayons issus de sa tuile émettrice, puis effectue le suivi incrémental de ces rayons jusqu'à ce qu'ils rencontrent une autre tuile (auquel cas la contribution lumineuse du rayon est calculée et ajoutée à la tuile réceptrice), ou qu'ils sortent de la tranche gérée par le processeur, auquel cas le rayon est transmis au processeur voisin, qui le prend alors en charge.

Une première phase de transmission des tuiles émettrices sélectionnées est nécessaire dans la méthode déterministe, afin que chaque tuile puisse savoir dans quelles directions lancer ses rayons d'ombrage. Après cette phase d'initialisation, chaque tuile envoie ses rayons d'ombrage au travers du réseau, un rayon étant traité par un processeur jusqu'à ce qu'il atteigne, soit la tuile émettrice locale (auquel cas la valeur énergétique reçue par la tuile origine est calculée et lui est renvoyée), soit une tuile occultante (une valeur énergétique nulle est renvoyée à la tuile origine), soit les limites de la tranche gérée par le processeur (auquel cas le rayon est transmis au processeur voisin).

L'arrêt du processus d'éclairement est assuré par l'intermédiaire d'un processus spécialisé, appelé *superviseur*. Celui-ci compte les rayons encore actifs dans le réseau (les différents processeurs l'informent du départ et de l'arrivée des rayons qu'ils gèrent), de même que les processeurs ne possédant plus de facettes disposant d'une radiosité latente suffisante pour être sélectionnées. Lorsque tous les rayons émis sont arrivés à leur destination, et qu'il n'existe plus aucune tuile susceptible d'être sélectionnée pour émettre, le superviseur informe les processeurs de la fin du calcul.

Ces approches n'ont pas été, à notre connaissance, implantées, et il semble difficile de les évaluer a priori. Elles possèdent l'avantage de distribuer équitablement la base de données entre les différents processeurs, ainsi que d'utiliser une structure d'accélération voxel qui permet d'obtenir un gain important des calculs d'intersection. Néanmoins, un découpage uniforme de la base de données en tranche de même taille risque d'entraîner un déséquilibre de charge selon que les processeurs possèdent un grand nombre de tiles ou pas, ou selon qu'ils gèrent des tiles fortement lumineux, ou au contraire peu éclairés. Pour résoudre ce problème, les auteurs proposent d'évaluer le coût de traitement de chaque voxel, en fonction du nombre de tiles qui y sont présents et de leur énergie (approximation du nombre de rayons qui vont être lancés depuis ce voxel ou nombre d'intersections qu'un rayon traversant le voxel devra calculer). Ce coût est additionné pour chaque tranche de largeur 1 voxel, et la répartition est effectuée en fonction du coût de traitement estimé de chaque tranche.

Utilisation d'une grille de processeurs

Une seconde approche utilisant le tracé de rayons a été proposée sur la machine VOXAR [Jessel 92], [Pauli 92]. Cette machine est composée d'un réseau de 36 transputers, configurés sous forme d'un hypertore de dimensions $4 \times 3 \times 3$.

L'espace de la scène est voxélisé, et les voxels voisins sont regroupés par blocs de même taille, appelés *métavoxels* (Figure 5.22.a). Ces métavoxels sont ensuite distribués sur les processeurs de manière cyclique, des blocs voisins se trouvant répartis sur des processeurs voisins.

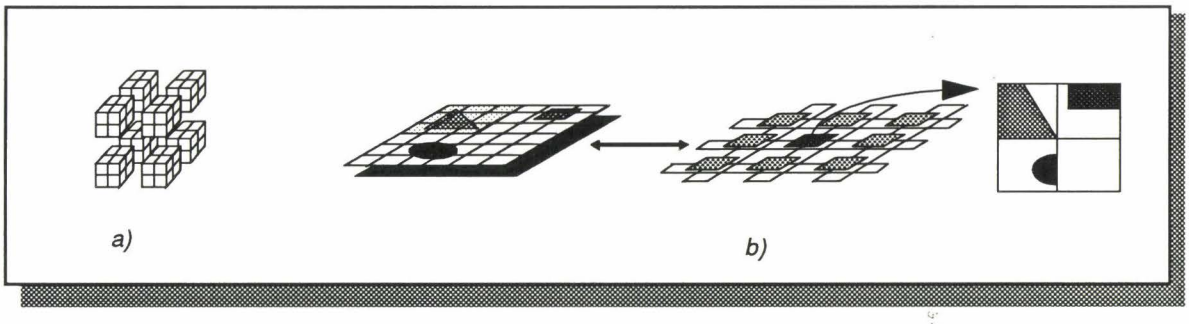


Figure 5.22 ***Découpage de l'espace en métavoxels (a) et exemple de distribution cyclique 2D (b)***

La Figure 5.22.b, montre un exemple d'une distribution cyclique des métavoxels en 2 dimensions, sur une grille de 9 processeurs.

Chaque voxel contient la liste des objets qui y figurent, représentés sous forme de *mailles* triangulaires, chaque sommet de la maille étant constitué d'un *atome*. Un atome constitue un point d'échantillonnage sur un objet, et contient toutes les informations relatives à l'objet, telles que sa couleur, sa radiosité ou encore sa texture. Lorsqu'une maille est présente dans plusieurs métavoxels, elle est dupliquée dans chacun d'entre eux. Comme dans l'approche précédente, seul un métavoxel (un processeur) est autorisé à modifier les valeurs énergétiques de la maille. Celui-ci est déterminé comme étant le métavoxel qui possède la plus petite coordonnée des atomes de la maille.

L'algorithme de calcul de radiosité est similaire à l'algorithme proposé par Wallace et détaillé au chapitre 3. Chaque itération consiste alors en 3 phases successives:

- sélection des sources: chaque processeur détermine la maille qui possède la plus grande énergie latente parmi les mailles qu'il gère.

- diffusion des sources: les facettes émettrices retenues dans chaque processeur sont transmises à l'ensemble des processeurs, de manière à ce que chacun puisse appliquer la troisième phase.
- acquisition de l'énergie: lorsque les sources sont connues, chaque processeur lance des rayons depuis chacun de ses atomes vers les sources. Le suivi des rayons est alors effectué de manière incrémentale de voxel en voxel. Lorsqu'un rayon sort d'un métavoxel géré par un processeur, il est transmis au processeur voisin pour poursuivre le suivi du rayon. Lorsqu'un rayon trouve une intersection avec un objet, ou qu'il parvient sur la source, un message est envoyé à l'atome source, contenant la valeur de l'énergie à rajouter: 0 s'il y a eu occultation, la contribution de la source sur l'atome sinon.

Le processus de calcul se termine lorsqu'il n'y a plus aucune maille possédant une énergie latente suffisante pour être sélectionnée, c'est à dire que chaque processeur, à la fin d'une phase de sélection, transmet une maille "vide" aux autres processeurs.

Cette implantation est essentiellement basée sur un parallélisme par messages, les différents processeurs s'échangeant un grand nombre de rayons. Cependant, une répartition des données par voxels conduirait très vite à une saturation des communications entre processeurs, dans la mesure où ceux-ci n'auraient à traiter qu'un seul voxel à la fois sur le trajet du rayon. C'est la raison pour laquelle les auteurs ont introduit la notion de métavoxel. Dans ce cas, un rayon traversant un processeur est suivi par celui-ci tout au long de son parcours à travers un métavoxel. Il convient néanmoins de ne pas utiliser des métavoxels de trop grande taille, qui conduisent à un déséquilibre de charge important, tant au niveau de la répartition des objets sur les processeurs, que sur la répartition des charges de calculs. En effet, si les métavoxels sont de trop grande taille, certains d'entre eux contiendront un grand nombre de mailles (si les métavoxels se trouvent à l'emplacement d'un objet), tandis que d'autres n'en contiendront que très peu (métavoxels contenant du "vide"). De ce fait, le suivi d'un rayon à travers les métavoxels contenant de nombreuses mailles sera beaucoup plus coûteux qu'à travers les métavoxels n'en contenant que peu. D'autre part, la constitution de méta voxels trop grands peu conduire à une baisse de performance de certains processerus, si les métavoxels qu'ils gèrent ne sont traversés que par un nombre faible de rayons. Ces phénomènes sont mis en évidence dans [Pauli 92], où différentes mesures d'activité moyenne des processeurs sont effectuées en fonction de deux tailles de métavoxels: $8 \times 8 \times 8$ et $16 \times 16 \times 16$. Dans le premier cas, l'activité moyenne est d'environ 73%, alors que dans le second, elle n'est que de 33%.

Aucune méthode n'est cependant proposée pour le calcul de la taille des métavoxels à utiliser, en fonction de la scène à traiter. D'autre part, il est difficile de juger de l'efficacité d'une telle approche, puisque son architecture est statique, et que peu de mesures de temps sont disponibles. Il semble cependant que l'utilisation d'un parallélisme par message ait un coût assez élevé.

5.5 Le problème de la subdivision adaptative

Le principe de la subdivision adaptative est basé sur la recherche d'une variation trop forte de radiosité à l'intérieur d'une même facette. Lorsqu'une telle variation est détectée, par exemple par comparaison des valeurs de radiosité de deux sommets voisins, la facette doit être subdivisée en *éléments* plus petits, un nouveau calcul de radiosité devant lui être appliqués.

Les travaux de [Baum 91] ont montré que ce découpage devait vérifier un certain nombre de propriétés, basées sur la notion de voisinage entre facettes (voir chapitre 1), qui permettent d'éviter l'apparition de discontinuités le long d'une arête commune à deux facettes lors de l'étape d'affichage. Si ces propriétés sont assez facilement applicables dans une implantation séquentielle, leur gestion dans un environnement parallèle est beaucoup plus délicate.

Nous avons vu qu'il existait différentes approches envisageables pour la représentation de la base de données: soit par duplication sur chaque processeur, soit par partage via une mémoire commune, soit enfin par répartition entre les différents processeurs. Chacune de ces approches possède un certain nombre d'avantages pour l'implantation d'un algorithme de subdivision adaptative, mais soulève aussi de nombreux problèmes. Nous allons passer en revue chacun de

ces points pour les différentes approches, en ne considérant que l'algorithme de radiosité progressive.

Duplication de la base de données

Rappelons que, quel que soit le mode de représentation de la base de données, la connaissance du voisinage d'une facette est indispensable, d'une part pour décider de sa subdivision , et d'autre part, pour effectuer une subdivision répondant aux critères de Baum.

Cette approche répond bien à ce problème, puisque l'intégralité de la base de données est présente sur chaque processeur. De ce fait, chacun d'entre eux peut appliquer localement les opérations nécessaires à la subdivision adaptative. Il faut néanmoins tenir compte de deux paramètres essentiels, avant de pouvoir envisager une telle implantation:

- la subdivision adaptative est déclenchée sur des critères de variation énergétique au sein d'une facette. De ce fait, pour que chaque processeur puisse déterminer les endroits où la subdivision adaptative est nécessaire, il lui faut avoir accès aux radiosités des facettes.
- la base de données mémorisée sur chaque processeur doit être la même, afin que les *éléments* générés par l'un d'entre eux puissent être utilisés aussi par les autres, et que cela ne génère pas d'incohérence dans la gestion de la base de données.

Nous avons vu que le premier point peut être résolu par la centralisation des informations énergétiques sur un processeur spécialisé, le contrôleur, ce qui permet une gestion aisée de la mise à jour des radiosités et du choix des facettes émettrices.

Le second point, quant à lui, pose des problèmes très difficiles à résoudre. En effet, la contrainte de conserver une base de données identique sur chaque processeur (y compris sur le contrôleur) génère tout d'abord un surplus important de communications , destinées à diffuser les informations concernant les subdivisions effectuées sur un processeur à l'ensemble du réseau. D'autre part, les processeurs fonctionnent de manière asynchrone. Dans ce cas, il est nécessaire de fixer l'attitude d'un processeur qui n'a pas terminé ses calculs de facteurs de forme, et qui reçoit des informations de subdivision adaptative. Deux possibilités se présentent alors:

- il annule les calculs en cours, de manière à prendre en compte les nouvelles informations qui lui parviennent. Cela conduit alors à une séquentialisation de l'algorithme tant que subsistent des possibilités de subdivision sur l'un des processeurs, puisque seuls les calculs effectués sur le processeur le plus rapide sont utilisés. Il faut d'autre part tenir compte de l'asynchronisme qui existe entre les processeurs, qui peut être une source de diffusion multiple d'ordres de subdivision.
- il continue ses calculs sur la base de données dont il dispose, et ne prendra en compte les informations de subdivisions externes qu'à la fin de son calcul. Dans ce cas, une perte de précision se produit, du fait de l'absence de calcul sur une facettisation plus précise de la scène, de même qu'il existe un risque de redondance des calculs, si les subdivisions générées par le calcul local sont les mêmes que celles envoyées par un autre processeur.

Il reste que le principal défaut de la duplication de la base de données sur chaque processeur est le coût mémoire. Même s'il existe une place suffisante pour la base de données initiale, l'utilisation de la subdivision adaptative risque de très rapidement remplir la mémoire et de ne pas permettre la poursuite de l'algorithme de subdivision.

Répartition de la base de données

Dans l'ensemble des implantations utilisant une répartition des facettes sur les différents processeurs du réseau, celles-ci sont réparties aléatoirement (ou modulo le nombre de processeurs), de manière à obtenir un certain équilibre de charge sur le réseau. De ce fait, des facettes voisines dans la scène se trouvent sur des processeurs distants. Il devient alors particulièrement difficile de gérer les informations de voisinage permettant d'obtenir une

subdivision cohérente entre des facettes voisines. Prenons, par exemple, le cas de la Figure 5.23 ci-dessous.

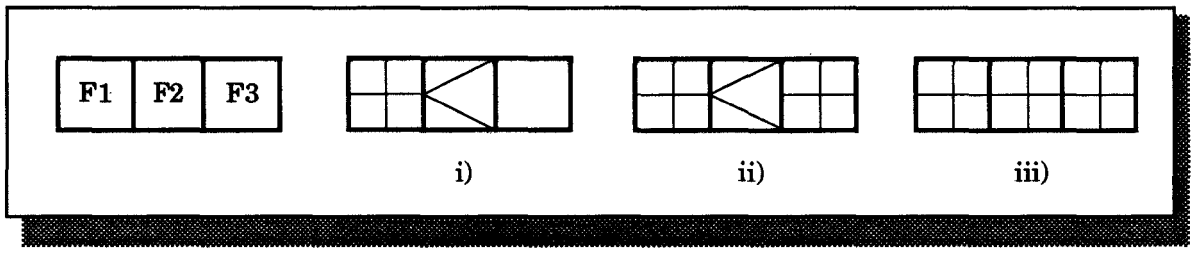


Figure 5.23 ***Exemple des problèmes de voisinage pour la subdivision adaptative***

Les trois facettes voisines F1, F2 et F3 sont sur des processeurs différents. Le premier processeur applique une opération de subdivision sur F1, la découpant en 4 éléments. Pour éviter la création d'un point de discontinuité le long d'une arête subdivisée, il est nécessaire d'adapter le découpage de la facette voisine (cf Chapitre 1, paragraphe 1.3.3). Une demande d'ancrage doit donc être envoyée à chacune des facettes voisines de F1, par exemple F2 (i). La facette F2 peut à nouveau recevoir un ordre d'ancrage depuis une autre de ses facettes voisines F3 (ii), l'obligeant à se désancrer, à se subdiviser, et à avertir ses facettes voisines de son état (iii) (réclamer un ancrage).

Outre ces problèmes de gestion du voisinage, il convient de rappeler que la décision de subdiviser une facette est prise en comparant les valeurs de radiosité aux sommets. Celles-ci, dans le cadre d'une approche projective, ne peuvent être obtenue que par l'intermédiaire d'une moyenne des valeurs de radiosité des facettes auxquelles le sommet appartient. Or, ces facettes se trouvent sur des processeurs différents, et la mise à jour de leur radiosité est effectuée de manière asynchrone. Il est donc fort probable que les informations de radiosité utilisées pour vérifier le besoin de subdivision, ne se trouvent faussées par la prise en compte, à un moment donné, de valeurs de radiosité présentant un nombre de mises à jour différents, et donc des valeurs totalement incohérentes. Ce qui peut avoir comme conséquence de générer un grand nombre de subdivisions inutiles.

Il semble donc plus indiqué d'utiliser une approche où les facettes voisines se trouvent sur le même processeur. Dans ce cas, la distribution des facettes doit se faire objet par objet, un objet (élémentaires: polygone, sphère, ...) devant alors nécessairement avoir toutes ses facettes sur le même processeur. Le processeur propriétaire peut alors effectuer toutes les subdivisions dont il a besoin, sans tenir compte (temporairement) d'éventuelles mises à jour des radiosités qui lui parviennent de l'extérieur. Cela pose néanmoins le problème de l'équilibrage des charges entre les processeurs, tant au niveau mémoire (certain processeurs se verront attribuer de grandes surfaces, possédant énormément de facettes, telles que les murs d'une pièce, ou des surfaces qui génèreront beaucoup de subdivisions), qu'au niveau calcul. Il semble néanmoins difficile, compte tenu des différents problèmes que nous avons évoqués ci-dessus, d'utiliser une autre démarche.

Utilisation d'une mémoire partagée

Dans ce contexte, où les processeurs partagent un accès à une mémoire unique, nous avons étudié deux approches différentes, qui permettent le calcul parallèle d'une seule ou de plusieurs émissions. Nous traitons ici du problème de la subdivision adaptative parallèle, indépendamment de la machine utilisée, la seule restriction étant l'utilisation d'une mémoire partagée.

Dans le cas où les processeurs coopèrent pour le calcul d'une émission, il est alors tout à fait possible d'implanter un algorithme parallèle de subdivision adaptative. En effet, après la phase de mise à jour des radiosités, chaque processeur peut se charger de la subdivision d'une partie de la base de données, sous réserve que ces parties soient bien distinctes d'un processeur à l'autre. Cette condition introduit alors le même type de solution que celle vue au paragraphe précédent, c'est à dire qu'il faut que chaque processeur s'occupe de la subdivision des facettes

d'un objet différent, de manière à assurer l'absence de conflits. Avec pour conséquences, le même type de problèmes d'équilibre de charge. Cette solution permet cependant une gestion simple de la cohérence de la base de données.

Les machines à mémoire partagée disposent d'un avantage non négligeable sur les machines à mémoire répartie, pour les approches à calcul d'émissions multiples, qui est d'assurer facilement la cohérence de la base de données. En effet, un processeur qui termine la mise à jour des radiosités peut effectuer la subdivision adaptative en section critique sur l'ensemble des facettes, puis libérer celles-ci pour un nouveau processeur. Le problème est alors, comme précédemment, d'autoriser une perte de précision pour les processeurs qui ne disposaient pas des nouveaux éléments.

Conclusion

Nous avons vu, au travers de ce paragraphe, que la subdivision adaptative pose de gros problèmes de parallélisation, du fait de la modification structurelle qu'elle effectue sur la base de données. Il nous semble donc que seules deux solutions peuvent permettre l'implantation de cet algorithme: soit une solution utilisant une mémoire répartie et une distribution des facettes par objets; soit une solution basée sur une mémoire distribuée et appliquant un algorithme de calcul parallèle d'une émission. Ces deux solutions offrent la structuration nécessaire de la base de données, tenant compte des informations de voisinage entre facettes, qui permet d'obtenir le maximum d'efficacité quant au processus de subdivision lui même.

Notons enfin qu'à notre connaissance, une seule approche parallèle de la radiosité a, à ce jour, utilisé la notion de subdivision adaptative. Celle-ci a été développée dans le cadre de la machine VOXAR [Pauli 92], et autorise la subdivision locale d'une facette (d'une maille) au sein d'un processeur. Elle ne tient cependant pas compte des problèmes de voisinage entre facettes, créant ainsi des discontinuités le long des arêtes.

5.6 Classification des approches MIMD étudiées

Les différentes approches utilisant un parallélisme de type MIMD que nous avons étudié dans ce chapitre ont été regroupées dans la Table 5.2 ci-dessous. Elles ont été classées en fonction des différentes critères que nous avons détaillé au cours du chapitre.

algorithme de radiosité	*représentation de la scène*	*mémoire*	*type de parallélisme*	*niveau de parallélisme*	*références*
calcul de la matrice complète	*dupliquée*		*par le contrôle*	*1*	*Price*
	transite		*flux*	*1*	*Purgathofer*
	distribuée		*par le contrôle*	*1*	*Chalmers*
radiosité progressive	*dupliquée*	*répartie*	*par le contrôle*	*1*	*Recker,Sillion Chen,Renaud Leprêtre*
				2	*Singh*
	répartie			*1*	*Chalmers,Feda*
			par les données	*1 et 2*	*Guitton,Jessel*
				1et 2	*Thomin,Vilaplana*
	transite		*flux*	*1*	*Thomin*
	unique	*partagée*	*par le contrôle*	*2 et 3*	*Baum*
				1	*Menard*

Table 5.2 ***Classifications des approches à parallélisme MIMD pour la radiosité***

On peut y distinguer le type d'algorithme de radiosité utilisé, ansi que la façon dont la base de données est représentée. De même, nous avons fait figurer le type d'architecture sur lequel l'implantation a été faite (mémoire répartie ou partagée). La notion de type de parallélisme correspond à celles que nous avons définies dans le chapitre 4. Les niveaux de parallélisme correspondent, quant à eux, aux niveaux examinés au paragraphe 5.1.3.

5.7 Conclusion

L'utilisation du parallélisme de type MIMD dans le cadre de la radiosité semble mieux adapté au parallélisme à gros grain. En l'occurrence, les résultats obtenus par calcul parallèle de plusieurs émissions sont, en général, plus intéressants que ceux obtenus par association des processeurs pour le calcul d'une même émission.

Le calcul simultané de différentes émissions peut en effet bénéficier de l'asynchronisme offert par le fonctionnement MIMD, puisque les quantités de calcul différent d'une émission à l'autre. A l'inverse, la collaboration des processeurs pour le calcul d'une unique émission nécessite une resynchronisation avant tout nouveau calcul dans le cas des méthodes projectives, ou une surcharge de travail du processeur possèdant la facette émettrice, dans le cas des approches utilisant le lancer de rayon que nous avons décrites.

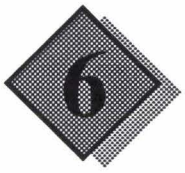
Le niveau le plus fin de parallélisme exploitable dans les méthodes projectives n'offre que peu d'intérêt dans un environnement MIMD. Au contraire, il tirera pleinement profit de l'utilisation de circuits spécialisés ou de machines parallèles à fonctionnement SIMD.

Nous avons d'autre part pu constater que la répartition de la base de données entre les processeurs génère rapidement des coûts de communications prohibitifs. Ceux-ci découlent directement du fait qu'il existe une trés forte dépendance entre les données pour l'algorithme de radiosité, puisque la distribution d'énergie diffuse est effectuée dans un demi-espace complet à chaque itération.

Il nous semble néanmoins envisageable de tenir compte d'un certain nombre de cohérences que nous avons évoquées, en vue de limiter les communications et augmenter les performances. Ces notion de cohérence sont proches de celles utilisées en lancer de rayons, mais leur utilisation devra vraisemblablement suivre un schéma différent, où les processeurs collaborent pour le calcul d'une émission. Dans la mesure où ce type d'approche génère un déséquilibre de charge, il convient cependant de se demander si celui-ci pourra être contrebalancé efficacement par le gain obtenu.

Enfin, il faut noter que le problème de la subdivision adaptative reste ouvert, en ce sens qu'aucune solution parallèle n'a à ce jour réellement exploité cet algorithme. Cela tient sans doute au fait qu'il est nécessaire de maintenir des informations de voisinage entre facette, afin de ne pas générer de discontinuités lors de l'affichage. La localisation des facettes d'un objet donné sur un même processeur doit pouvoir résoudre ce problème, mais risque de générer des problèmes annexes de déséquilibre de charge. Il nous semble cependant, qu'excepté l'utilisation d'une mémoire partagée, ce type de répartition est le seul qui permett de considérer la subdivision adaptative sur des calculateurs à mémoire distribuée.

CHAPITRE



Etude d'approches à parallélisme massif

Le développement de machines permettant d'atteindre un parallélisme massif a été l'un des grands axes de recherche au début des années 80. Grâce aux progrès réalisés en matière d'intégration, ces machines en sont arrivées au stade commercial et offrent des performances très élevées pour un coût relativement réduit. Les équipes travaillant dans des domaines réclamant une quantité de calculs très importante (physique, mathématiques, météorologie ...) ont été très tôt attirés par la puissance de ces machines. La complexité sans cesse croissante des algorithmes de synthèse d'images a conduit les équipes graphiques à se tourner elles aussi vers ces machines. Des études ont ainsi été menées sur l'implantation de l'algorithme du lancer de rayons, et, plus récemment, de l'algorithme de radiosité sur ce type de machines.

Nous nous proposons d'étudier les différents shémas de parallélisation massive de la radiosité, en fonction des algorithmes de calcul des facteurs de forme utilisant les méthodes projectives. Nous détaillons différentes solutions envisagées et les problèmes qu'elles posent de par leur contexte de fonctionnement de type SIMD. Nous étudions, d'autre part, le problème des communications entre processeurs qui dépend fortement de l'architecture cible et des possibilités de communication qu'elle offre.

Dans ce chapître, nous présentons, dans un premier temps, les schémas de parallélisme envisageables dans un contexte massivement parallèle. Nous présentons ensuite une implantation sur la Maspar d'une approche basée sur un parallélisme appliqué au niveau du proxel. Après avoir évalué une approche objet, nous proposons une approche hybride, permettant de concilier les avantages respectifs de chacune des deux approches précédentes.

6.1 Schémas de parallélisme

Notre objectif est ici d'étudier différents shémas de parallélisation massive pour les algorithmes de calcul des facteurs de forme, utilisant les approches projectives décrites au chapitre 2. Dans ces approches, la phase de projection, nécessaire au calcul des facteurs de forme, se rapproche du problème de l'affichage de facettes sur un écran. Un certain nombre de solutions à ce problème ont été proposées, dans le cadre d'architectures massivement parallèles, spécialisées dans le rendu.

Le pipeline de rendu y est décomposé en deux étapes principales: une étape de calculs géométriques, qui prépare les paramètres de projection, suivie d'une étape de conversion des objets en pixels, pour l'affichage proprement dit. Ce découpage se retrouve dans la plupart des architectures étudiées: une machine hôte effectue l'ensemble des calculs géométriques, tandis qu'une seconde machine est dédiée à la phase de conversion. Si le hôte reste une machine d'usage général, la puissance nécessaire lors de la seconde phase, rend la définition d'architectures massivement parallèles spécialisées souhaitable. Nous décrivons ci-dessous les principales caractéristiques et réalisations utilisées dans ce type de machine : l'approche pixel et l'approche objet.

6.1.1 Machine pixel

Dans cette approche, le parallélisme massif est atteint en associant un processeur à chaque pixel de l'écran. Chaque processeur a alors pour tâche de déterminer si un objet, fourni par la machine hôte, couvre le pixel qu'il gère. Si c'est le cas, il effectue les opérations d'élimination des parties cachées (tampon de profondeur) et, si l'objet est visible, applique diverses fonctions de rendu (éclairage par interpolation, anti-aliassage, texturage ...). Après projection de l'ensemble des objets, chaque processeur connait l'objet visible en son pixel, et dispose des informations nécessaires à l'affichage de l'image finale.

Divers mécanismes de distribution des objets aux processeurs ont été étudiés par Leprêtre [Lepre 89]. La classification qu'il propose regroupe trois mécanismes différents (Figure 6.1) :

- La diffusion : les caractéristiques de l'objet à afficher sont transmises simultanément à l'ensemble des processeurs, qui travaillent alors tous sur le même objet. Ce mécanisme est bien adapté au mode de fonctionnement SIMD, mais ne fournit qu'un rendement assez médiocre, dans la mesure où le nombre de pixels effectivement couverts par la projection est faible comparé au nombre total de pixels.

  Ce mode de distribution est utilisé pour les machines Pixel-Planes 4 [Fuchs 85] et Pixel-Planes 5 [Fuchs 89]. La diffusion est assurée par l'intermédiaire d'arbres d'additionneurs, permettant d'évaluer des équations linéaires de la forme $V(x, y) = Ax + By + C$ pour Pixel-Planes 4, et quadratiques pour Pixel-Planes 5, (x, y) représentant les coordonnées d'un pixel. Les objets sont définis à l'aide d'expressions de ce type, une facette triangulaire étant par exemple représentée par sept expressions linéaires: trois pour les côtés, une pour la profondeur, et trois pour les valeurs d'éclairage.

- Le multipipeline : les processeurs sont regroupés en lignes, et reçoivent des informations depuis leur voisin de gauche pour les propager vers leur voisin de droite. Chaque colonne de processeurs traite donc simultanément le même objet, les objets pouvant être différents d'une colonne à l'autre. Les calculs de visibilité (évaluation de l'équation des droites du contour), de profondeur et d'éclairage sont effectués incrémentalement le long de chaque pipeline. Le rendement de ce mécanisme reste cependant identique à celui du mécanisme précédent. Il pourrait être augmenté en déterminant la bande horizontale dans laquelle se trouve chaque objet, et en transmettant plusieurs objets dont les bandes ne se recouvrent pas. Une telle approche nécessite cependant un prétraitement coûteux sur la machine hôte pour déterminer les différents objets à envoyer.

  Le projet RC [Atamenia 89], basé sur ce principe, utilise un réseau multipipeline alimenté par un pipeline vertical de découpage. Les objets (facettes) transitent de manière incrémentale dans le pipeline vertical, un objet effectivement présent sur une ligne étant alors envoyé dans le pipeline horizontal correspondant. Ses valeurs de profondeur et de couleur sont calculées incrémentalement de processeur en processeur, l'élimination des parties cachées étant effectuée dans les processeurs où l'objet est visible.

- La propagation : l'accès au réseau s'effectue par l'intermédiaire d'un processeur (le germe), placé au centre du réseau, qui se charge de diffuser les caractéristiques de l'objet qu'il a reçu du hôte, vers ses voisins immédiats. Ceux-ci à leur tour propagent l'information vers leurs propres voisins. Aucune machine utilisant ce principe n'a, à ce jour, été étudiée.

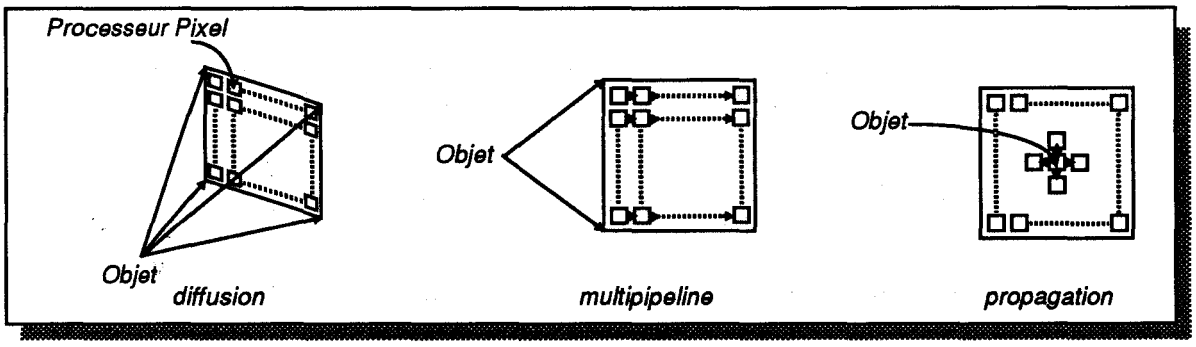


Figure 6.1 ***Mécanismes de distribution des objets aux processeurs***

Ces différents mécanismes de distribution présentent tous un rendement identique. Les différences interviennent surtout au niveau de la réalisation du schéma de communication, la diffusion d'un objet nécessitant l'accès à tous les processeurs, alors que le nombre de points d'entrée est forcément réduit dans les deux autres mécanisme.

6.1.2 Machine objet

La seconde façon d'atteindre le parallélisme massif est d'associer un processeur à chaque objet. Chaque processeur a alors pour tâche de déterminer les pixels recouverts par son objet. Le but étant de générer une image, il est nécessaire de combiner les pixels obtenus par les différents processeurs, de manière à effectuer l'élimination des parties cachées. Ce mécanisme de combinaison des pixels est en général appelé *Décideur.*

Différentes approches ont été proposées pour la réalisation du décideur, parmi lesquelles le pipeline et l'arborescence (Figure 6.2) :

- Dans le cas de l'arborescence, les processeurs traitent simultanément le même pixel. Si l'objet est présent dans le pixel, ils génèrent une valeur de profondeur et une valeur d'éclairage. Ces valeurs remontent ensuite dans l'arborescence, une comparaison de profondeur étant effectuée en chaque noeud de l'arbre. La valeur la plus proche est transmise à l'étage suivant de l'arbre.
- La seconde approche utilise une configuration pipeline des processeurs objets. Chaque processeur reçoit un pixel de son voisin de gauche, détermine si son objet est visible dans ce pixel, effectue éventuellement le test de profondeur et envoie le pixel vers son voisin de droite. L'élimination des parties cachées se fait donc le long du pipeline, le pixel correct étant obtenu aprés passage par tous les processeurs.

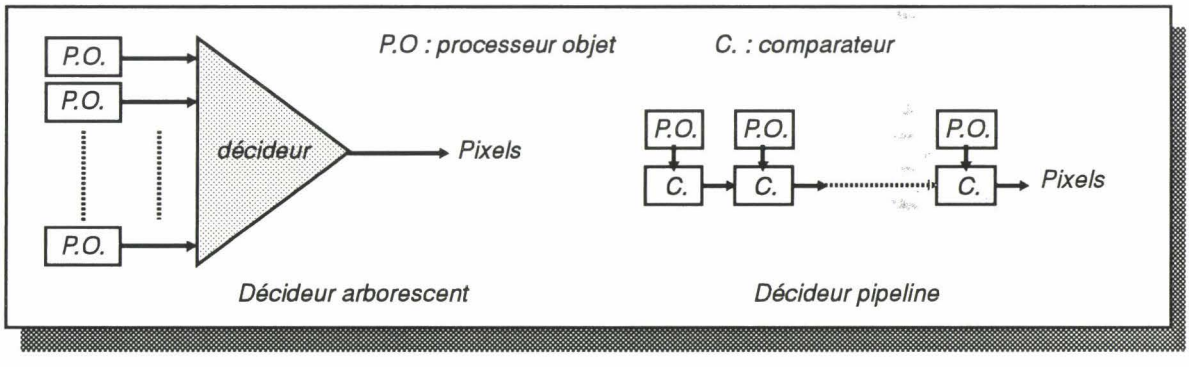


Figure 6.2 ***Shéma des architectures utilisables pour réaliser le décideur***

L'utilisation d'un décideur arborescent a été étudié dans le cadre de la machine IMOGENE [Chail 91], mais des problèmes de connectiques ont orienté les études vers l'utilisation d'une structure pipeline. Le décideur pipeline a aussi été adopté dans deux autres machines objets, GSP/NVS [Deeri 88]et PROOF [Schne 88]. Une description détaillée de ces machines peut être trouvée dans [Chail 92].

Le rendement de ces architectures est identique à celui des machines pixel, puisque la présence de chaque objet est calculée en chaque pixel. Elles offrent cependant l'avantage d'être extensible, contrairement aux approches pixels.

6.2 Une implantation utilisant un parallélisme proxel

L'implantation que nous allons décrire est basée sur le principe du parallélisme pixel. Chaque PE est chargé de déterminer la facette visible en un (ou plusieurs) proxels, les proxels étant distribués sur l'ensemble du réseau. Certaines différences existent cependant avec les architectures matérielles présentées au paragraphe 6.1.1, page 118.

Tout d'abord, les calculs classiquement dévolus à la machine hôte sont distribués sur le réseau de processeurs. Ceci se justifie par le fait que les PEs de la MP-1 sont d'un usage plus général que les processeurs des architectures pixel. Autant donc profiter de la puissance de calcul qu'ils offrent pour d'autres opérations que la simple conversion des objets en pixels.

D'autre part, afin de profiter de l'importante quantité de mémoire disponible sur l'ensemble du réseau (16384 PEs disposant chacun de 16 Ko de RAM, soit environ 268 Mo de RAM), les facettes de la base de données sont distribuées sur les PEs. Cette distribution permet la représentation de base de données plus importantes et évite le surcoût de communication qui apparaitrait s'il était nécessaire de distribuer les facettes à projeter depuis le hôte pour chaque nouvelle facette émettrice.

Enfin, une étape supplémentaire, inhérente à la distribution des facettes, est nécessaire pour mettre à jour les radiosités des facettes en fonction des proxels où elles sont visibles, ce qui nécessite des communications supplémentaires entre les PEs.

Nous abordons, dans un premier temps, le problème du choix de la méthode de distribution des facettes (diffusion, multipipeline, propagation) pour la phase de conversion des facettes en proxels. Les modes de répartition des données (facettes et proxels) sur le réseau de processeurs sont ensuite détaillés, en tenant compte du mode de distribution précédemment choisi. Après avoir précisé les différentes étapes de l'algorithme, le problème de la mise à jour des radiosités est analysé et diverses méthodes, utilisant les spécificités de la MP-1 concernant les communications inter-PEs, sont étudiées. Enfin, nous présentons quelques résultats concernant les différents algorithmes projectifs implantés, et nous effectuons une comparaison de notre approche avec des travaux précédents.

6.2.1 Méthode de distribution des facettes à projeter

Nous avons évoqué, au paragraphe 6.1.1, trois modes de distribution pour les approches pixels. Leur rendement étant identique, le choix est guidé par l'architecture de la machine cible.

Dans le cas de la Maspar, c'est la diffusion qui semble la plus appropriée, puisque le dispositif de diffusion fait partie de l'architecture de la machine. Les autres shémas restent cependant utilisables mais posent des problèmes plus délicats à traiter.

Un rendement supérieur à celui de la diffusion avec un shéma multipipeline, ne peut être obtenu qu'en traitant simultanément plusieurs facettes. Ceci nécessite un tri des facettes à projeter, de manière à déterminer celles qui peuvent être traitées en même temps. Ce tri est une opération coûteuse et complexe en séquentiel, et semble difficile à effectuer lorsque les facettes sont réparties sur un très grand nombre de processeurs.

La solution par propagation, quant à elle, semble peu efficace, dans la mesure où elle génère un surcoût important en étapes de communication. En effet, les communications ne peuvent s'effectuer que dans une seule direction à un instant donné. Propager les paramètres de projection sur une topologie à quatre voisins nécessite donc 4 cycles de communication, et 8 cycles avec une topologie à 8 voisins. D'autre part, il est nécessaire d'utiliser un schéma de propagation particulier, de préférence câblé, afin de ne pas effectuer de rétropropagation [Lepre 89].

6.2.2 Répartition des facettes sur le réseau

La représentation de la base de données sur la machine hôte uniquement n'est justifiée que si c'est elle qui effectue le travail de préparation des données avant projection. Dans la mesure où l'ensemble de l'algorithme est exécuté sur les PEs, les facettes de la base de données sont réparties sur le réseau de processeurs. Ceci permet en outre de disposer d'une quantité mémoire supérieure à celle disponible sur le hôte, et de limiter les communications entre celui-ci et le réseau.

Cette répartition est effectuée à raison d'une facette par PE, modulo le nombre de PEs. Les facettes sont distribuées dans l'ordre où elles ont été modélisées, ce qui permet une répartition à peu près homogène de la base de données (des facettes voisines, qui ont toutes les chances

d'avoir un comportement identiques vis à vis de la facette émettrice, ne sont pas sur le même processeur). Ceci permet de plus de conserver un moyen aisé de récupération des résultats aprés calcul, par rapport à un mode de distribution aléatoire.

Les informations mémorisées, pour une facette, comprennent à la fois les aspects géométriques (sommets, normale, ...) et énergétiques(radiosités, radiosités latentes, réflectances), puisque les étapes de projection et de calcul des radiosités sont toutes deux effectuées sur la grille de PEs.

6.2.3 Répartition des proxels

Afin de faciliter les explications, nous allons traiter le problème de la répartition des proxels, indépendamment de la surface de projection utilisée. Des précisions seront données lors de la description des différentes implantations effectivement réalisées. Nous considèrerons donc, dans ce qui suit, que nous nous limitons à une partie carrée d'un plan parallèle au plan de la facette d'application. Cette partie est découpée uniformément en proxels carrés.

Il existe une analogie certaine entre le découpage du carré de projection, et la répartition des PEs sous forme de grille. Idéalement, chaque PE devrait se voir allouer un proxel différent (des PEs voisins obtenant des proxels voisins). Dans la mesure où il n'y a généralement pas autant de PEs que de proxels à représenter, il est nécessaire de distribuer plusieurs proxels par PE. Deux types de répartitions peuvent être considérées, par *bloc* ou *cyclique*:

- La répartition par blocs consiste à affecter, à chaque processeur, une région de la surface de projection formée par une grille de proxels voisins (voir Figure 6.3 a)
- Dans le cas de la répartition cyclique, la surface de projection est découpée en zones contigües et non recouvrantes, de même taille et de même forme que la grille de processeurs. Chaque processeur a alors en charge un proxel de chaque zone, l'emplacement du proxel dans la zone correspondant à l'emplacement du processeur dans la grille (Figure 6.3 b).

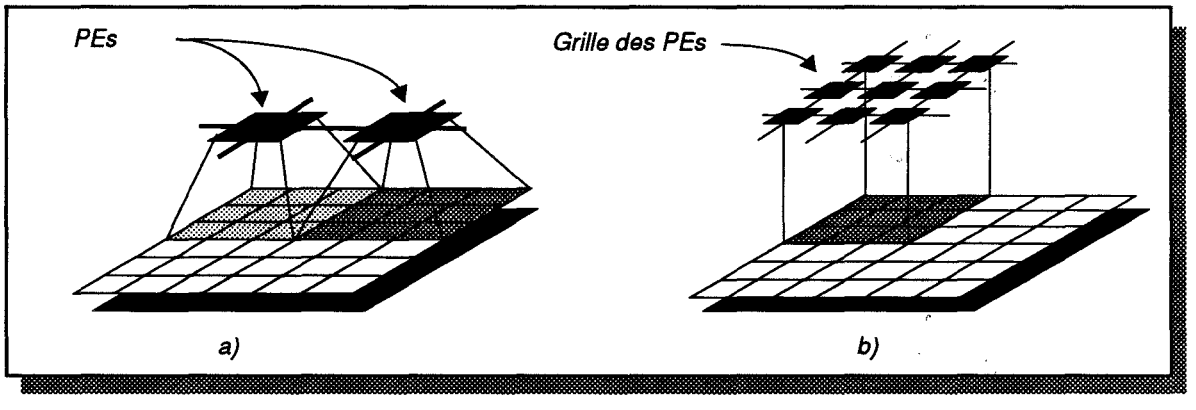


Figure 6.3 ***Répartition des proxels par blocs a) et par cycle b)***

La répartition cyclique est la mieux adaptée au mode de distribution séquentiel des facettes à projeter. En effet, une facette projetée ne couvre, en général, que très peu de proxels, au regard du nombre total de proxels. Cela a pour conséquence qu'avec une répartition par blocs, le nombre de blocs dans lesquels une partie de la facette est visible est très faible. De ce fait, si chaque PE traite une zone différente, l'ensemble du calcul de visibilité ne repose que sur un nombre réduit de processeurs.

A l'inverse, en utilisant une répartition cyclique, le traitement d'une zone ne repose pas sur un processeur unique, mais sur l'ensemble du réseau, chaque processeur travaillant sur un proxel différent de la zone. D'autre part, comme le nombre de zones recouvertes par une facette projetée est faible, un algorithme simple de détection des zones recouvertes accélèrera considérablement le processus, en limitant le calcul de visibilité à ces seules zones.

6.2.4 Utilisation des rectangles englobants

Le calcul du rectangle englobant une projection permet de détecter simplement les zones potentiellement recouvertes par celle-ci.

La projection d'une facette sur l'hémicube ou sur un plan de projection unique fournit un contour projeté convexe, pour lequel il est très simple de déterminer le rectangle englobant. Dans le cas du disque de projection, nous avons exposé une méthode de calcul du rectangle englobant d'un contour formé d'arcs elliptiques (paragraphe 2.2.3, page 16).

Pour chacune des méthodes citées ci-dessus, il est donc possible d'exploiter les avantages de la répartition cyclique des proxels.

Notons enfin que le principe de limitation des zones traitées aux zones recouvertes par le rectangle englobant est actuellement utilisé dans le contexte des architectures pixels sous le terme de *Réalisation Partielle* [Karpf 93]. Le raisonnement qui a été suivi est identique au nôtre: utiliser un processeur par pixel génère un rendement très faible; il est donc intéressant de diminuer le nombre de processeurs, et de les faire travailler successivement sur des zones différentes, sachant que seules les facettes visibles dans la zone seront traitées. Différentes études ont été menées par Molnar [Molna 92]pour la machine Pixel-Planes 5, qui ont abouti à la réalisation d'une grille de processeurs pixels ne comprenant que 128x128 processeurs.

6.2.5 Etapes des algorithmes

Nous allons nous attacher, dans cette partie, à passer en revue chacune des étapes de notre algorithme. Cet algorithme est identique pour les différentes implantations que nous avons effectuées. Seuls changent les paramètres de projection à utiliser, et certains détails de représentation des plans de projection. L'algorithme, sous sa forme générale, est schématisé ci-dessous, en pseudo-langage.

```
Pour chaque phase d'illumination
|
|     Appliquer les transformations géométriques
|     Calculer les paramètres de projection
|
|     Pour chaque facette à projeter
|     |
|     |     Diffusion des paramètres de projection
|     |     Projection de la facette
|     |
|     Fin pour
|
|     Calcul des facteurs de forme
|     Mise à jour des radiosités
|     Choix d'une nouvelle facette émettrice
|
Fin pour
```

Chaque ligne en italique représente une partie de programme exécutée en parallèle sur les PEs, tandis que les autres lignes sont exécutées sur l'ACU (en séquentiel). Hormis l'étape de diffusion des paramètres de projection, la totalité de l'algorithme est exécutée sur la grille de processeurs. Seules les boucles de contrôle sont exécutées sur l'ACU. Ceci nous permet de profiter au maximum de la puissance disponible.

Les différentes étapes de l'algorithme sont :

1. **Les transformations géométriques** :
   Celles-ci regroupent plusieurs opérations différentes déja utilisées dans l'algorithme séquentiel, dont le changement de repère des facettes, l'élimination des facettes trivialement invisibles[1] depuis la source, et le fenêtrage par les plans de contour du plan de projection.

L'étape d'élimination des facettes trivialement invisibles est un peu particulière dans le contexte parallèle, puisqu'elle a pour but de construire sur chaque processeur la liste des facettes qui seront projetées. Cette liste est construite à partir d'une table d'index mise à jour au fur et à mesure du traitement des différentes facettes, et réinitialisée à chaque nouvelle facette émettrice. De cette manière, seules les facettes potentiellement visibles seront prises en compte lors des étapes suivantes.
Ces opérations doivent être effectuées pour chaque facette et répondent bien à l'impératif de parallélisme de données imposé par la Maspar. Elles sont en effet bien définies et leur exécution ne présente que très peu d'aspects conditionnels. La charge est donc identique pour chaque PE, exceptées les petites différences dues à l'utilisation de 2 type de facettes (triangulaires et rectangulaires).

2. **Calcul des paramètres de projection** :
Plusieurs paramètres de projection doivent être calculés pour chaque facette à projeter. Ceux-ci dépendent de la surface de projection utilisée, mais restent sensiblement du même type: zones recouvertes par le rectangle englobant de la facette projetée, équation des droites ou des arcs elliptiques formant le contour projeté, valeurs initiales pour la zone de départ et valeur des incréments permettant le passage d'une zone à l'autre, etc... Toutes ces valeurs doivent être calculées pour chaque facette, et leur mode de calcul est rigoureusement identique pour toutes les facettes. Cette étape est donc bien adaptée au mode de fonctionnement SIMD, et la charge est répartie de manière équitable entre les différents PEs.

3. **Diffusion des paramètres de projection** :
Nous avons précisé que l'algorithme ne traite qu'une seule facette à la fois, les PEs déterminant simultanément si celle-ci est présente ou pas dans leurs proxels. Les différentes facettes étant réparties sur les PEs, il est nécessaire de diffuser les informations relatives à la projection d'une facette donnée à l'ensemble des PEs.
L'utilisation des réseaux *Xnet* où *global router* est ici mal adaptée, car elle nécessite trop d'étapes de communications. Une analyse des méthodes de diffusion d'une information à l'ensemble des PEs effectuée dans [XX] montre que le moyen le plus rapide est de passer par l'ACU. Les paramètres de projection sont donc récupérés par l'ACU, puis diffusés à l'ensemble des PEs du réseau (Figure 6.4). Les PEs peuvent alors appliquer l'étape de projection, un nouveau PE étant ensuite sélectionné pour diffuser ses paramètres.

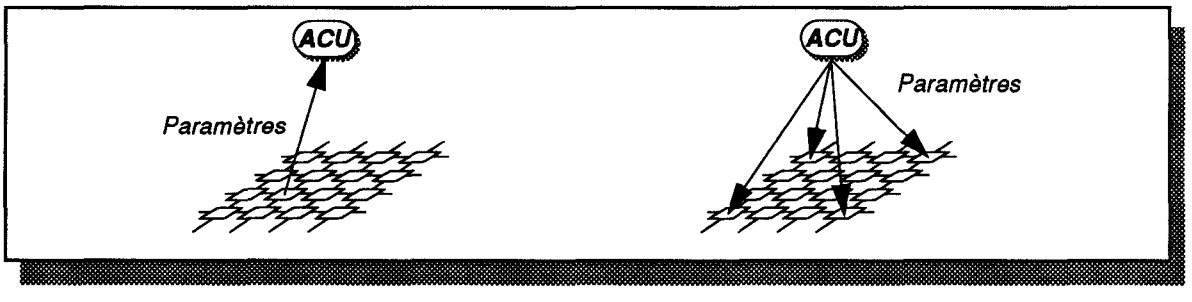


Figure 6.4 ***Récupération et diffusion des paramètres de projection***

4. **Projection d'une facette** :
Pour cette étape, les PEs disposent des différents paramètres permettant d'effectuer la projection d'une facette. Pour chaque zone couverte par le rectangle englobant la facette à projeter, les équations de droite (ou celles des arcs d'ellipse) sont évaluées en chaque proxel. Si chacune d'entre elles fournit un résultat de même signe, le proxel est considéré comme étant à l'intérieur du contour projeté, et une opération de comparaison de profondeur est effectuée avec la facette précédemment mémorisée.

1. Une facette est considérée comme trivialement invisible depuis une facette émettrice si elle se trouve sous le plan support de celle-ci, ou si elle lui "tourne le dos" (sa normale est dirigée dans la même direction que celle de la facette émettrice).

A noter que le passage d'une zone à l'autre est effectuée de manière incrémentale, tant selon l'axe des abscisses, que selon l'axe des ordonnées.

5. **Calcul des facteurs de forme - Mise à jour des radiosités** :
Après projection de l'ensemble des facettes, les informations de visibilité sont utilisées pour calculer le facteur de forme entre chaque facette et la facette émettrice, de manière à mettre à jour leur radiosité. Si cette étape est triviale dans une application séquentielle, elle devient particulièrement délicate dans un contexte parallèle, où l'ensemble des proxels constituants le tampon de profondeur, ainsi que les facettes de la base de données, sont distribués sur l'ensemble des processeurs. Nous abordons ce problème de manière plus détaillée, dans le paragraphe 6.2.6 "Mise à jour des radiosités" ci-dessous.

6. **Choix d'une nouvelle facette émettrice** :
Comme dans l'algorithme séquentiel, la nouvelle facette émettrice est choisie en fonction de sa quantité d'énergie latente. Le problème provient à nouveau ici de la distribution des données sur les différents processeurs du réseau.
L'environnement de développement de la Maspar nous fournit un certain nombre d'opérations de réduction[1], applicables avec différents opérateurs arithmétiques et différents formats de données. Parmi ceux-ci, les opérateurs *Maximum* et *Minimum*, utilisables avec les fonctions *reduceMaxf* et *reduceMinf*, qui fournissent respectivement la plus grande et la plus petite valeur d'une variable réelle multiple.
Dans un premier temps, nous recherchons la facette possédant la plus grand énergie latente localement à chaque PE. Lorsque celle-ci a été determinée, l'opérateur *reduceMaxf* lui est appliqué, ce qui permet de déterminer la nouvelle facette émettrice. Ses paramètres sont alors récupérés sur l'ACU, qui se charge de déterminer la nouvelle matrice de changement de repère et lance le calcul d'une nouvelle phase d'émission.

6.2.6 Mise à jour des radiosités

Après la phase de projection, les informations contenues dans chaque proxel (facteur de forme élémentaire, identité de la facette visible) sont utilisées pour calculer les facteurs de forme, puis les radiosités. Rappelons cependant que les facettes sont distribuées au travers du réseau, de même que les proxels. L'association d'un proxel et de la facette qu'il contient nécessite de nombreuses communications. Nous allons étudier, ci-dessous, les solutions qui se présentent, en fonction des différents outils de communication disponibles.

Utilisation du réseau Xnet

Ce réseau permet d'effectuer des communications entre chaque PE et l'un de ses huit voisins directs. Le fonctionnement du réseau reste cependant de type SIMD, tous les processeurs devant communiquer simultanément dans la même direction. Du fait de la répartition des proxels et des facettes, deux stratégies de communication sont envisageables : soit les proxels se déplacent vers la facette, soit c'est la facette qui recherche les proxels où elle est visible.

Déplacement des proxels

Dans cette stratégie, le facteur de forme élémentaire associé à chaque proxel est transmis à la facette qui y est visible, après la phase d'élimination des parties cachées. En suivant le schéma de répartition des facettes que nous avons déjà précisé, chaque processeur traitant un proxel est capable de déterminer l'emplacement du processeur où se trouve la facette visible à travers son proxel.

Pour atteindre un processeur en utilisant le réseau Xnet, chaque proxel se dirige d'abord horizontalement vers la colonne où se trouve ce processeur. Cette opération est effectuée simultanément pour N proxels, où N représente le nombre de processeurs utilisés. Lorsque ceci

1. On appelle *réduction* le mécanisme qui permet, à partir d'une valeur prise sur chaque PE actif, d'appliquer une opération arithmétique sur ces valeurs et de fournir un résultat scalaire. Par exemple, faire la somme globale d'une valeur V présente sur chaque PE. Cette opération de réduction s'effectue en un temps logarithmique, plutôt qu'en un temps linéaire.

est fait, une deuxième étape de communication consiste à déplacer tous les proxels verticalement vers le processeur destinataire (Figure 6.5). De cette manière, les N proxels atteignent bien la facette à laquelle ils appartiennent.

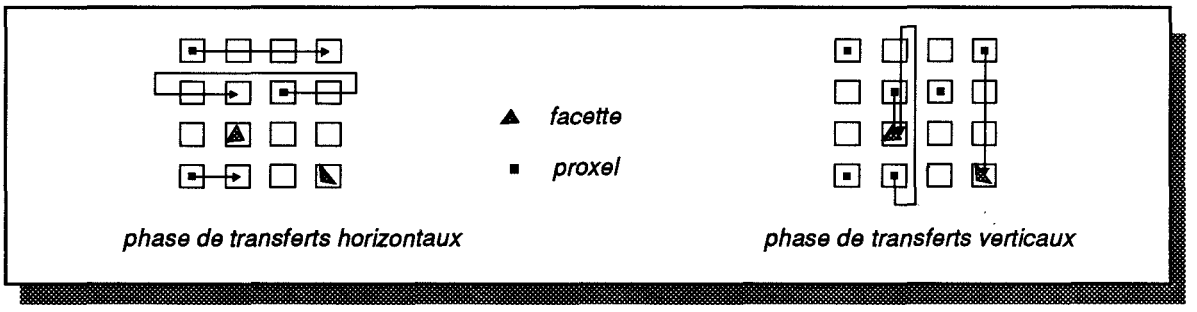


Figure 6.5 ***Phases de déplacement horizontaux et verticaux des proxels***

De manière à simplifier l'algorithme, les communications horizontales ne sont effectuées que dans un seul sens (vers l'est), de même que les communications verticales (vers le sud). Ceci nécessite évidemment de faire parcourir un plus long chemin à certains proxels, mais permet de diminuer la complexité de l'algorithme tout en conservant exactement le même nombre de cycles[1] de communication que dans le cas de communications dans des directions différentes. En effet, positionner un proxel dans la bonne colonne nécessite ici, au maximum, N cycles de communication vers l'est. Si l'on autorise les communications vers l'est et vers l'ouest, chacune des directions nécessite, au maximum, $N/2$ cycles, soit un même total de N cycles, sachant que le fonctionnement SIMD de la Maspar n'autorise qu'une seule direction de communication à la fois. Le même raisonnement est applicable verticalement, et diagonalement sur une topologie à huit voisins.

Le nombre total de cycles de communications nécessaires à la mise à jour de l'ensemble des facteurs de forme peut alors être évalué. Si l'on note P le nombre total de proxels utilisés par l'algorithme projectif, et N la taille du réseau (nombre de processeurs situés sur un côté de la grille de PEs, en supposant une grille carrée), alors N_{tc} , le nombre total de cycles nécessaires au transfert des proxels vers les facettes, est:

$$2\frac{P}{N} \leq N_{tc} \leq P + \frac{P}{N} \qquad \textbf{(Eq. 6.1)}$$

Il est facile de déterminer le nombre de cycles horizontaux, puisque celui-ci est égal à N. Dans le cas vertical, par contre, il est beaucoup plus difficile à évaluer. En effet, lors de la propagation horizontale, un processeur particulier peut avoir à mémoriser les N proxels de sa ligne (ceci signifie que chacun des proxels de la ligne appartient à une facette gérée par un processeur d'une même colonne). Dans la mesure où la translation verticale ne peut être effectuée que pour un seul proxel à la fois par processeur, N étapes de propagation peuvent être nécessaires pour "écouler" ces proxels. Le coût d'une étape de propagation verticale étant le même que pour la propagation horizontale, soit N cycles, le nombre maximum de cycles verticaux sera donc donné par N^2. Dans la mesure où ces deux étapes de propagation doivent être effectuées autant de fois qu'il y a de proxels par processeur, soit P/N^2, le nombre minimum de cycles nécessaires est:

$$N_{min} = (N+N)\frac{P}{N^2} = 2\frac{P}{N}$$

tandis que le nombre maximum est donné par :

$$N_{max} = (N+N^2)\frac{P}{N^2} = P + \frac{P}{N}$$

L'écart entre les deux bornes est considérable, puisqu'il correspond à peu près au nombre total de proxels utilisés pour le découpage du plan de projection. Le nombre de cycles le plus élevé

1. Nous appelerons "cycle de communication", pour le réseau Xnet, la période de temps nécessaire pour effectuer une communication entre deux PEs voisins. Pour le réseau Global Router, celui-ci correspond à la période de temps nécessaire pour la réalisation d'une communication sans attente (cf problèmes de conflits) entre deux PEs distants.

correspond aux cas où un grand nombre de proxels adjacents sur une ligne tentent d'accéder à la même colonne. Le nombre de cycles le plus faible se produit, quant à lui, lorsque tous les proxels de chaque ligne se dirigent vers une colonne différente. Malheureusement, dans la mesure où il existe une très forte dépendance sur l'identité de la facette projetée sur des proxels voisins, le plus mauvais cas se produit avec une fréquence très élevée, entrainant des temps de communication prohibitifs.

Le pseudo-code de cet algorithme est décrit ci-dessous.

```
Pour chaque proxel local
|
|     propager_proxel_vers_la_droite_jusque_bonne_colonne
|
|     Pour tous les proxels arrivés dans la bonne colonne
|     |
|     |     propager_proxel_vers_le_bas_jusque_bonne_ligne
|     |
|     |     mettre_à_jour_le_facteur de forme
|     |
|     FinPour
|
FinPour
```

Déplacement des facettes

Ce shéma de mise à jour est l'inverse du précédent : une facette dont le facteur de forme est recherché, est transmise à chacun des processeurs qui possède au moins un proxel où cette facette est visible, à charge à chacun de ces processeurs d'ajouter la contribution de son (ses) proxel(s) (Figure 6.6).

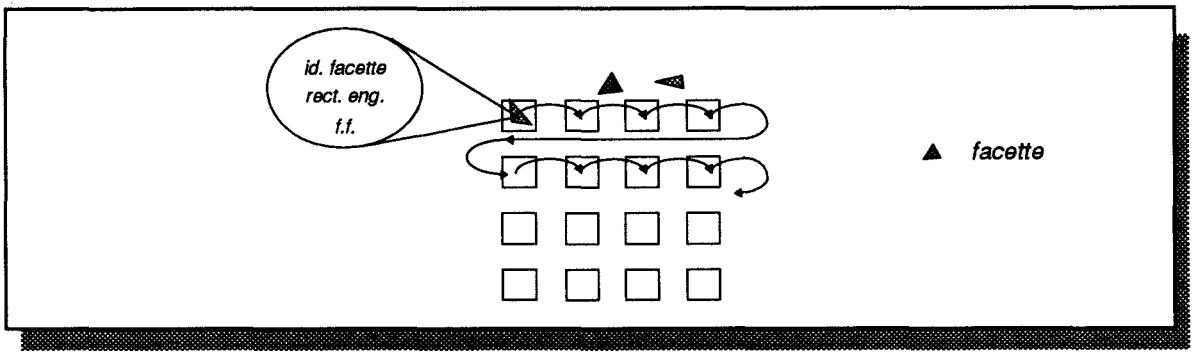


Figure 6.6 ***Déplacement successif des facettes par tous les processeurs***

Ce shéma se heurte néanmoins à un obstacle important : une facette n'a aucune connaissance de l'emplacement des processeurs où sont mémorisés les proxels qu'elle recouvre. Il lui est donc nécessaire de passer par chaque processeur. Ceci induit un surcoût important, mais offre l'avantage d'un fonctionnement de type SIMD, tant du point de vue de l'algorithme que de celui des communications. L'algorithme de mise à jour est écrit ci-contre sous forme d'un pseudo-langage.

```
Pour chaque facette locale
|
|     ff = 0.0; id = numéro_de_la_facette;
|
|     Pour y =1 jusqu'à hauteur_de_la_grille_de_PEs
|
|     |     Pour x = 1 jusqu'à largeur_de_la_grille_de_PEs
|     |
|     |     |     recherche_et_maj(id, proxels, ff);
|     |     |     transmettre_vers_la_droite(id, ff);
|     |     |
|     |     FinPour
|     |
|     |     transmettre_vers_le_bas(id, ff);
|     |
|     FinPour
|
|     maj_radiosite(id, ff);
|
FinPour
```

Chaque processeur sélectionne l'une de ses facettes locales, et prépare le contenu du message qui va transiter dans le réseau (facteur de forme initialisé à 0, identité de la facette). Ce message est transmis à tous les PEs, colonne par colonne, puis ligne par ligne, cette double boucle assurant, d'une part, un unique passage par chaque processeur, et d'autre part le retour d'une facette à son point de départ. Lorsqu'un processeur reçoit l'un de ces messages, il recherche si la facette est visible dans l'un de ses proxels. Si c'est le cas, il modifie le facteur de forme de la facette. Après vérification de chacun des proxels, le message, éventuellement modifié, est transmis au processeur voisin.

L'un des inconvénients de cette approche réside dans la nécessité de tester l'ensemble des proxels, alors que la facette n'est potentiellement visible que dans un nombre restreint d'entre eux. Comme nous l'avons déjà dit, il est impossible de déterminer a priori les proxels recouverts. Par contre, il est tout à fait possible de limiter le domaine de recherche au sein des proxels d'un PE en utilisant la notion de rectangle englobant. Si les limites des zones recouvertes par le rectangle englobant sont incluses dans le message, chaque processeur n'aura à tester qu'une petite partie de ses proxels.

Notons cependant que, pour une phase de mise à jour donnée, le gain obtenu sera limité par la facette recouvrant le plus grand nombre de zones (ayant le plus grand rectangle englobant), puisque N facettes sont traitées simultanément et que le fonctionnement est de type SIMD.

Il est important de pouvoir déterminer le nombre de cycles de communication utilisés par cette approche, de manière à obtenir un critère théorique de comparaison avec l'approche précédente. Le nombre de cycles de communication par passe de calcul est assez simple à déterminer, puisqu'il est de N^2, où N^2 représente le nombre de processeurs (une passe de calcul correspond au calcul des facteurs de forme de N facettes, une par PE). Le nombre total de passes, quant à lui, est égal au plus grand nombre de facettes présentes sur un PE. Celui-ci est égal à la partie entière par excès du nombre total de facettes de la base de données (F), divisé par le nombre de processeurs. Nous obtenons donc:

$$N_{tc} = N^2 \left\lceil \frac{F}{N^2} \right\rceil \approx F \qquad \textbf{(Eq. 6.2)}$$

Le nombre de cycles de communication est donc approximativement égal au nombre de facettes de la base de données. Cette approche semble donc intéressante dans le cas de petites bases de données, mais devient très rapidement fortement pénalisante lorsque le nombre de facettes augmente.

Comparaison

Nous avons comparé les temps de mise à jour des radiosités pour ces deux approches, en utilisant deux bases de données distinctes ("bureau1" 8000 facettes, et "bureau2" 17000 facettes). La Figure 6.7 ci-dessous montre l'évolution des temps moyens de mise à jour en fonction de la résolution utilisée pour le découpage du plan de projection.

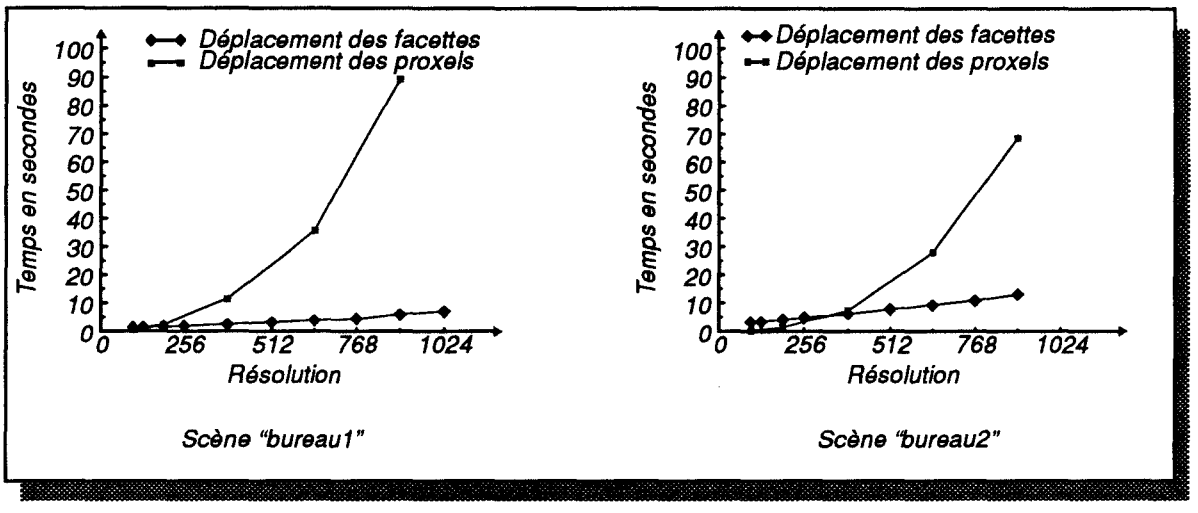


Figure 6.7 ***Evolution du temps de mise à jour des radiosités pour les solutions utilisant le réseau Xnet***

Ces mesures ont été effectuées sur une machine dotée de 1024 processeurs. L'algorithme utilisé est celui du disque de projection, et la mise à jour des radiosités par déplacement des facettes utilise le principe des rectangles englobants pour limiter l'espace de recherche des proxels au sein de chaque processeur.

Comme il fallait s'y attendre, l'évolution du temps de mise à jour est linéaire dans le cas du déplacement des facettes, puisque le nombre de proxels recouverts augmente linéairement en fonction du nombre total de proxels. De même, ce temps croît en fonction du nombre de facettes comme le montre l'équation (Eq. 6.2).

Dans le cas du déplacement des proxels, l'évolution des courbes s'effectue en fonction du carré de la résolution adoptée (linéairement en fonction du nombre de proxels utilisés). Une différence notable apparait cependant entre les deux bases de données. De manière "inatendue", il semble que plus le nombre de facettes d'une scène est élevé, et plus le temps de propagation des proxels diminue.

En fait, ceci s'explique relativement facilement : dans toutes les scènes que nous avons modélisées, la taille réelle des facettes (le pas de facettisation) est la même. Par contre, la taille réelle des scènes est très différente (voir Annexe A). Ceci a pour conséquence que, pour une facette émettrice quelconque, la taille relative des facettes vues depuis cette facette est plus grande dans le cas d'une scène de petite dimension (les facettes sont plus proches les unes des autres), que dans le cas d'une scène de grande dimension. De ce fait, le nombre de proxels recouverts diminue, de même que le nombre de proxels qui s'accumulent dans la même colonne lors de la propagation horizontale. Le nombre total de cycles diminue donc en conséquence.

Dans les deux cas, les temps de mise à jour restent cependant très élevés, ce qui risque de pénaliser lourdement l'application. Nous allons donc étudier la possibilité d'utiliser le second réseau de communication disponible sur la Maspar.

Utilisation du global router

Le réseau *global router* permet d'effectuer des communications directes entre PEs quelconques du réseau. Les deux stratégies citées ci-dessus (déplacement des proxels ou des facettes) restent envisageables pour la mise à jour des facteurs de forme. Il convient cependant de mettre

l'accent sur l'un des inconvénients majeurs de ce mode de communication: les conflits d'accés au réseau et aux PEs.

Nous avons vu que les PEs sont regroupés par groupes de 16, et qu'ils partagent un unique accès en entrée et en sortie au *global router*. De ce fait, si plusieurs PEs d'un même groupe cherche à communiquer vers l'extérieur, ou si plusieurs communications sont demandées vers différents PEs d'un même groupe, ces communications ne peuvent être effectuées que l'une après l'autre.

De même, si plusieurs PEs cherchent à communiquer avec le même destinataire, les communications se font séquentiellement, l'ordre étant indéterminé. Ces deux types de conflits surviennent, quelque soit la stratégie adoptée.

Nous allons analyser les deux stratégies précédentes dans le contexte de l'utilisation du *global router*, et voir celle qui permet de limiter au maximum les communications.

Déplacement des facettes

L'emplacement des proxels recouverts par la projection est inconnu. Il faut donc, pour chaque facette, établir une communication entre le PE qui la gère, et l'ensemble des PEs du réseau.

On peut choisir d'effectuer simultanément une communication depuis chaque PE vers un autre PE. En s'arrangeant pour que chaque communication mette en rapport des paires de groupes disjoints le nombre total de cycles de communications est donné par :

$$C = 32\,(N-1)\,N_{fp}$$

où N représente le nombre de processeurs et N_{fp} le nombre de facettes par processeur. En effet, il faut 16 cycles minimum pour envoyer l'identité de N facettes différentes vers N processeurs différents (du fait du regroupement des processeurs en groupes de 16). Après recherche de la facette dans ses proxels, chaque processeur doit renvoyer le facteur de forme qu'il a calculé localement vers le processeur propriétaire de la facette. Il faut donc à nouveau 16 cycles de communication. Dans la mesure où une facette doit passer par tous les processeurs, il faut $N-1$ étapes de ce type. Le tout doit évidemment être multiplié par le nombre de facettes mémorisées dans chaque processeur.

A noter qu'il est possible, avec cette approche, de limiter le domaine de recherche aux proxels appartenant aux zones recouvertes par le rectangle englobant des facettes, sous réserve que celui-ci soit transmis avec l'identité de la facette.

Néanmoins, elle dépend directement du nombre de facettes de la base de données, et génère plus de travail que nécessaire, puisque de nombreuses recherches ont lieu sur des proxels où les facettes ne sont pas visibles (soit parce que la facette est occultée, soit parce qu'elle n'est pas visible dans les proxels d'un processeur).

Déplacement des proxels

L'avantage de cette approche est de permettre l'association directe des proxels où une facette est visible avec le processeur où elle est mémorisée la facette. Elle ne nécessite donc pas de recherche au sein des processeurs; il suffit en pratique d'additionner la contribution du proxel (facteur de forme élémentaire) au facteur de forme de la facette.

Elle génère cependant des conflits de communication entre PEs émetteurs (PEs émetteurs d'un même groupe), mais surtout au niveau des PEs récepteurs. En effet, une facette projetée, si elle est visible, recouvre en général plusieurs proxels. Ces proxels sont voisins sur le plan de projection, et le sont aussi sur les processeurs. Lors des communications, il y a donc une forte

probabilité pour que de nombreux conflits se produisent, puisque plusieurs proxels tenterons de se rendre sur le même PE (Figure 6.8).

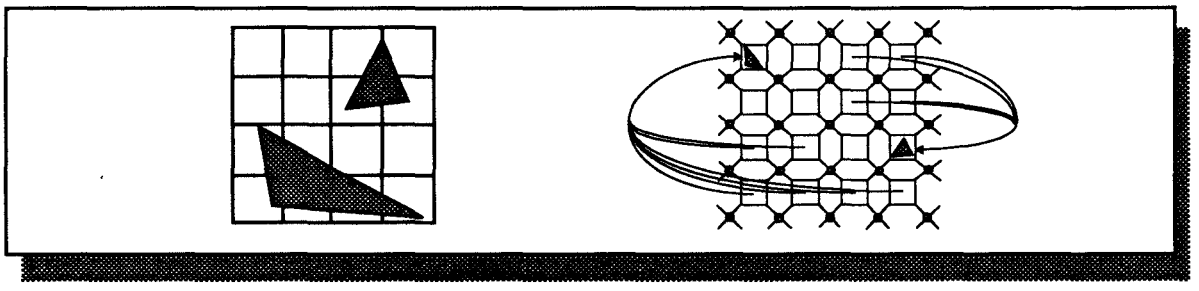


Figure 6.8 ***Shématisation des conflits en utilisant le global router***

Les temps de communication d'une telle approche sont évidemment très difficile à prévoir, puisqu'ils dépendent essentiellement du nombre de proxels recouverts par chaque facette. Il est cependant possible d'évaluer le nombre minimum de cycles de communication nécessaire dans le meilleur des cas.

Puisque chaque groupe de processeurs est composé de 16 PEs, il faut au minimum 16 étapes de routage pour transférer N proxels, avec N représentant le nombre total de PEs dans le réseau. Ces 16 étapes ne peuvent être obtenues que si, lors d'une étape, tous les processeurs qui communiquent envoient leur proxel vers des processeurs de groupes distincts. En d'autres termes, il ne faut pas que des processeurs tentent de communiquer avec le même groupe. En supposant que chaque processeur gère N_p proxels, le nombre minimum de cycles de routage pour que chaque proxel atteigne sa facette est donné par:

$$C_{min} = 16N_p$$

Cette valeur idéale est cependant bien loin d'être obtenue, comme le montre la courbe de la Figure 6.9, les valeurs de l'axe des ordonnées étant représentées de manière logarithmique.

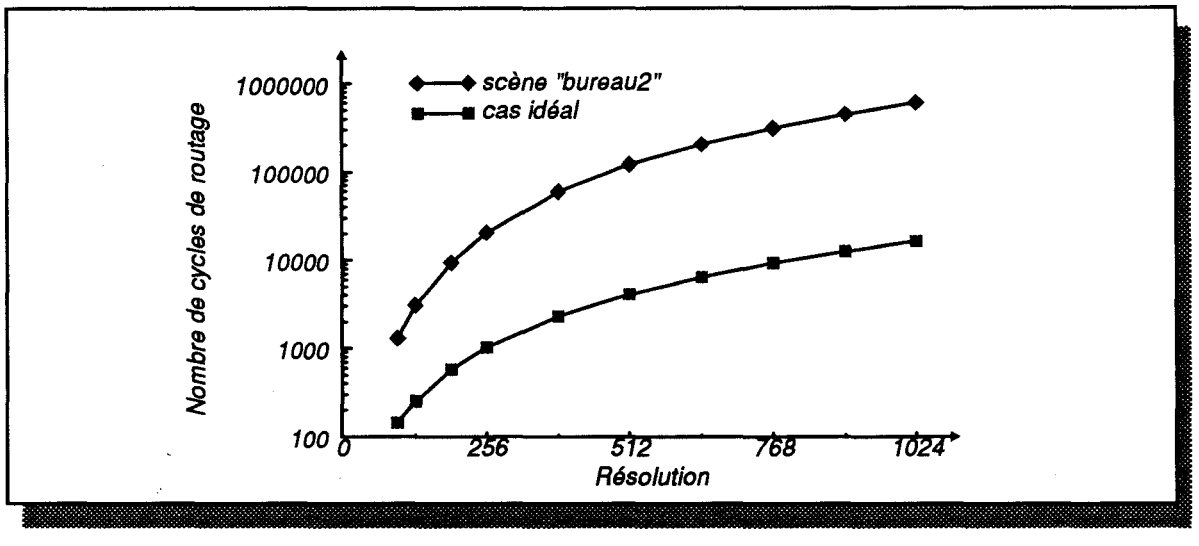


Figure 6.9 ***Comparaison de l'évolution du nombre de cycles de routage entre le cas idéal et une scène test.***

Le nombre moyen de communications ayant lieu simultanément peut être chiffré pour ces courbes, ce qui permet d'évaluer le facteur de gain qui pourrait être obtenu. Dans le cas d'un réseau 1024x1024, divisé en 64 groupes de 16 PEs, 64 communications peuvent être effectuées simultanément. Dans le cas d'une résolution 512 et 1024 du carré englobant le disque de projection, seules 2,16 et 1,75 communications, en moyenne, sont effectivement réalisées à chaque étape, ce qui représente un gain potentiel d'environ 30 et 36 respectivement.

Ce gain ne peut être obtenu qu'en se rapprochant au plus près du nombre maximal M_c de communications en un cycle, celui-ci étant égal au nombre de groupes de PEs présents dans la

machine (64 pour une configuration disposant de 1024 processeurs, 1024 pour une configuration de 16K PEs). Dans notre cas, pour pouvoir émettre simultanément M_c proxels, il faut, tout d'abord, que les facettes auxquelles ils sont destinés soient toutes différentes. Il est de plus nécessaire que les facettes de destination soient toutes réparties sur des processeurs situés dans des groupes différents.

Il est, a priori, impossible de déterminer simplement si les N facettes destinataires sont toutes différentes. Cela nécessiterait de pouvoir les comparer toutes ensemble, et d'effectuer, éventuellement, des modifications du choix des proxels émetteurs. Ces opérations ont un coût trop important pour en espérer un gain.

De même, s'assurer que tous les groupes où se trouvent les PEs de destination soient bien différents est une opération complexe, puisqu'elle nécessiterait, elle aussi, une comparaison des groupes d'arrivée.

Optimisation des communications

En fonction des remarques faites ci-dessus, nous allons tenter d'optimiser les communications, tout d'abord dans le cas de l'utilisation du Global Router, puis nous appliquerons le même type de raisonnement au réseau Xnet.

Pour éviter autant que faire se peut que les PEs ne tentent de mettre à jour le facteur de forme de la même facette, il est possible de tirer parti de la remarque suivante: la répartition cyclique des proxels sur le réseau a pour conséquence que les proxels recouverts par une facette dans une même zone vont se trouver sur des processeurs voisins (Figure 6.10).

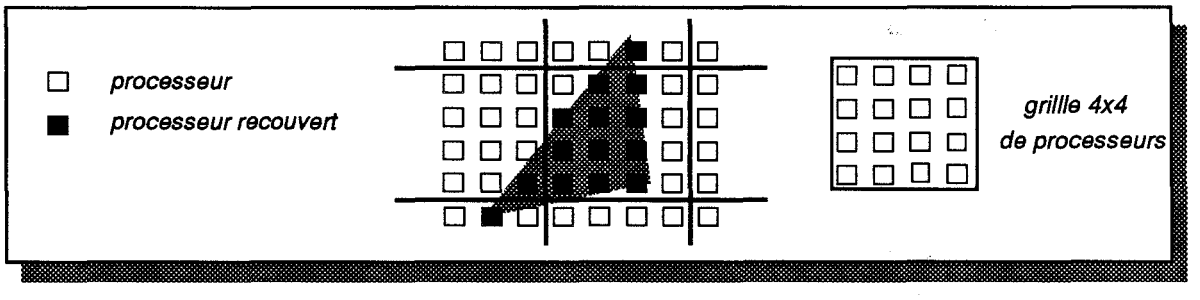


Figure 6.10 ***Exemple de répartition des proxels recouverts par une facette sur le réseau de PEs***

De ce fait, si la transmission des proxels est effectuée zone par zone, chaque processeur considérant un proxel de la même zone, les conflits sont inévitables, et seront d'autant plus nombreux que le nombre de proxels recouverts sera important. Ceci se produit avec plus d'intensité lorsque la résolution augmente, puisque le nombre de proxels recouverts par les facettes visibles va augmenter.

Pour limiter les conflits, il est donc intéressant que chaque processeur envoie un proxel situé dans une zone différente, car la probabilité que la facette qui y est visible soit différente est alors plus élevée. Il faut cependant que le choix de cette zone soit suffisamment simple pour ne pas perdre le gain obtenu. Dans la solution que nous avons développé, le choix d'une zone est effectué en suivant la règle suivante:

$$I_{pzone} = (ixproc + iyproc \times L_{zone} + z) \; mod \; (NbZones)$$

où

- I_{zone} est le numéro de la zone où se trouve le proxel à utiliser pour un processeur,
- $ixproc$ et $iyproc$ correspondent à l'abscisse et à l'ordonnée d'un processeur dans la grille de PEs (ces valeurs sont "connues" par chaque PE),
- L_{zone} est le nombre de PEs sur un côté de la grille (largeur de la grille),
- $NbZones$ est le nombre total de zones de proxels de taille $L_{pe} \times L_{pe}$,
- z représente une valeur variant successivement de 1 à NbZones,
- mod est l'opérateur "modulo", qui fournit le reste d'une division entière.

Ceci correspond en fait, pour chaque PE d'une ligne de la grille, à utiliser la zone adjacente à la zone utilisée par son voisin de gauche. De même pour les processeurs situés sur une même colonne. Le choix de la zone est appliqué modulo le nombre total de zones, et autant de fois qu'il y a de proxels par processeur. La Figure 6.11 schématise ce fonctionnement pour un réseau constitué de 2 processeurs et 9 zones d'application. Dans cette figure, les proxels Pi sont envoyés simultanément via le *global router* à l'étape i.

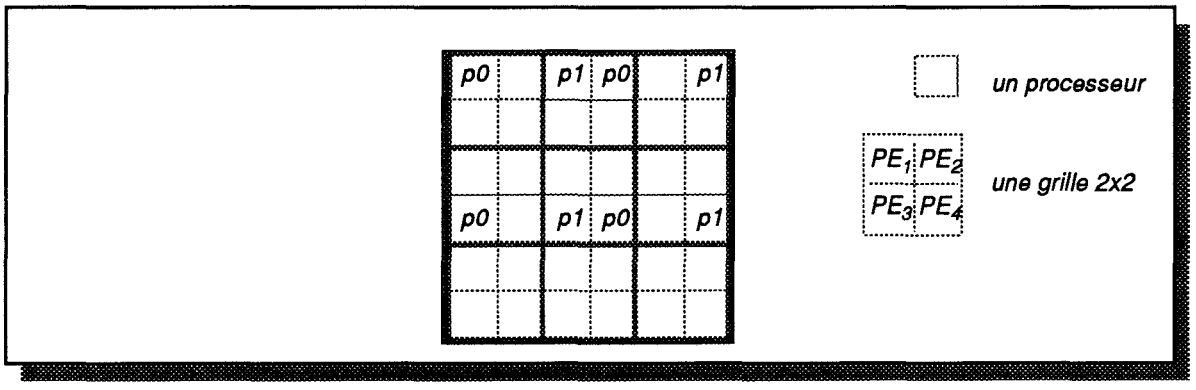


Figure 6.11 ***Choix des proxels dans différentes zones***

Bien évidemment, des conflits peuvent toujours se produire, si une facette recouvre en grande partie plusieurs zones. Néanmoins, cette stratégie très simple permet de réduire considérablement leur nombre, une facette n'étant généralement présente que dans un nombre limité de zones. La figure suivante (Figure 6.9) représente le nombre de cycles de routage nécessaires à la mise à jour des radiosités en utilisant ou pas cette stratégie "*modulo*".

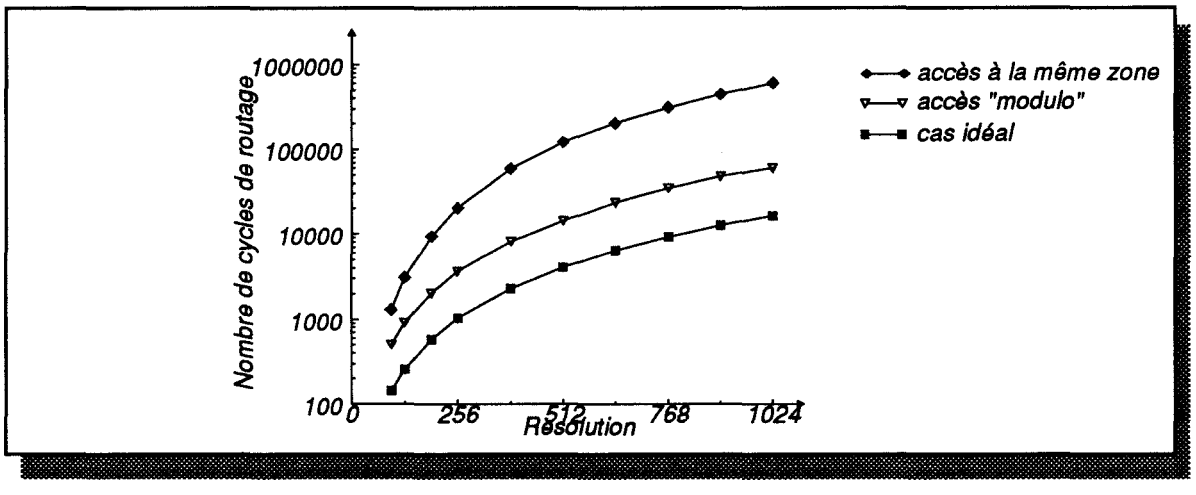


Figure 6.12 ***Comparaison de l'évolution du nombre de cycles de routage en utilisant la stratégie de routage "modulo".***

La comparaison des deux courbes montre très nettement le gain obtenu : il est d'environ 20 pour une résolution de 256, et de 34 pour une résolution de 1024. Ces mesures ont été effectuées sur la base de données "bureau2" pour l'algorithme utilisant le disque de projection.

La dernier point à considérer pour limiter les conflits est celui concernant les conflits d'accès au même PE destinataire. Ils se produisent quand au moins deux PEs tentent d'envoyer simultanément un proxel vers deux facettes différentes, mais mémorisées sur le même processeur. Il est a priori difficile dans ce cas de pouvoir proposer une méthode réduisant ces conflits, car aucune propriété particulière ne semble pouvoir être utilisée. En effet, il n'existe aucun lien entre la position des facettes projetées sur la surface de projection et leur gestion sur les différents processeurs de la machine.

Nous pouvons généraliser la stratégie "modulo" au déplacement des proxels à l'aide du réseau Xnet. En effet, les conflits qui se produisent possèdent la même source : si tous les proxels de la même zone sont considérés en même temps, la probabilité est grande pour qu'une majorité d'entre eux se dirigent vers le même processeur. De nombreux proxels s'accumulent alors sur la même colonne, et augmentent d'autant le nombre de cycles nécessaire à leur écoulement. Nous avons représenté, sur la figure ci-dessous(Figure 6.13), les temps de mise à jour par déplacement des proxels, avec cette stratégie.

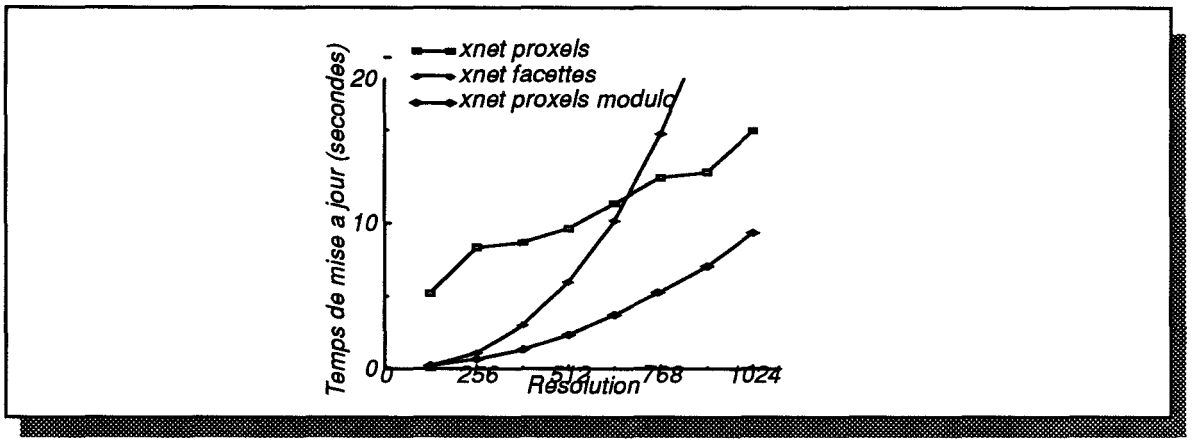


Figure 6.13 ***Temps moyen de mise à jour des radiosité par déplacement des proxels.***

Elle se compare favorablement aux temps de mise à jour sans la stratégie "modulo" et avec la mise à jour par déplacement des facettes. Notons que la différence est faible pour des résolutions peu élevées. Ceci provient du fait que, dans ce cas, les facettes projetées occupent une surface plus petite en nombre de proxels. L'accumulation des proxels sur la même colonne est donc moins forte. Ces chiffres ont été obtenus sur la base de données "Bureau2", en utilisant une Maspar dotée de 16K PEs.

Utilisation de l'ACU

Une dernière solution consiste à traiter une seule facette à la fois, en la diffusant à l'ensemble des processeurs. Ceux-ci vont alors rechercher simultanément la même facette. L'utilisation du global router est ici peut intéressante, puisqu'elle nécessite N-1 cycles de communication pour la diffusion de l'identité de la facette. Il est beaucoup plus intéressant d'utiliser l'ACU comme moyen de diffusion, comme nous l'avons déjà vu lors de l'étape de diffusion des paramètres de projection.

Le fonctionnement est alors le suivan t: l'ACU recupère l'identité d'une facette, la diffuse aux PEs. Ceux-ci recherchent si la facette est visible dans leurs proxels, et mettent à jour un facteur de forme partiel local. Lorsque tous les proxels ont été inspectés, une opération de réduction de type addition (*reduceAddf*) est effectuée, le résultat étant alors fournit au processeur possédant la facette utilisée.

Le temps de mise à jour est alors quasiment indépendant de la résolution utilisée, et croît linéairement en fonction du nombe de facettes.

Comparaison

Nous présentons, sur la Figure 6.14 une comparaison des temps de calcul entre les différentes solutions que nous avons étudiées jusqu'à présent.

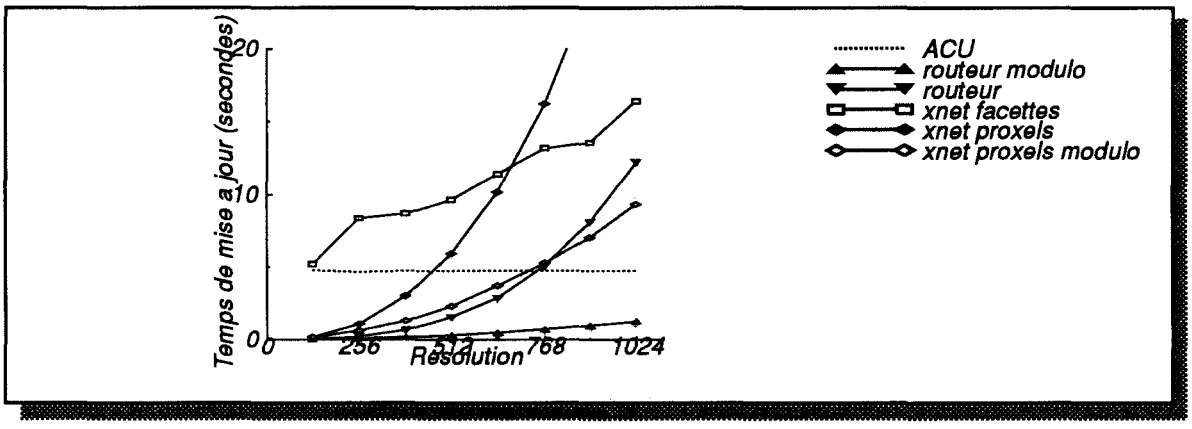


Figure 6.14 ***Comparaison des temps de mise à jour des radiosités (scène "Bureau2")***

Ces courbes ont été obtenues pour la scéne "Bureau2", pour une Maspar équipée de 16K PEs. Il apparait que la solution qui offre les temps de calculs les moins élevés correspond à l'utilisation du Global Router, associé à la stratégie "modulo".Elle est de loin plus intéressante que les autres solutions, quelles qu'elles soit. Ces résultats seront comparés dans le chapitre 7 aux courbes de temps de mise à jour des radiosités qui ont été obtenue sur les autres scènes,

6.2.7 Quelques résultats

Nous avons implantés quatre algorithmes projectifs différents : l'hémicube, les plans de projection utilisant soit le découpage de Sillion, soit un découpage régulier, et le disque. Pour chacun d'entre eux, différentes mesures ont été prises, pour plusieurs bases de données. Ces mesures concernent le temps moyen d'une phase de projection, le rendement obtenu, le gain obtenu par l'utilisation de la notion de réalisation partielle (rectangles englobants) et le coût des différentes étapes.

Nous présentons ici quelques uns de ces résultats, permettant de se faire une bonne idée des performances de l'approche massivement parallèle proxel. Des résultats plus détaillés seront présentés au chapître 7.

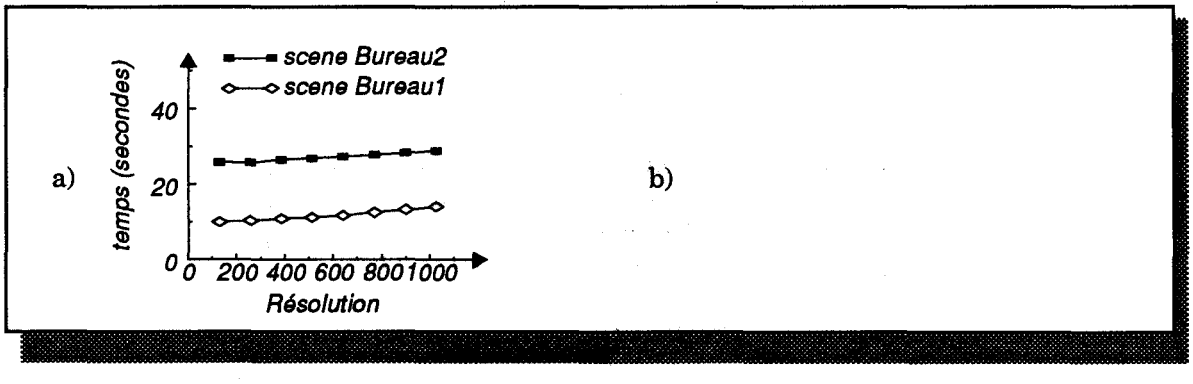


Figure 6.15 ***Temps de calcul (a) et rendement (b) pour le disque de projection.***

Ces résultats ont été obtenus pour les scènes "Bureau1" et "Bureau2", sur une Maspar équipée de 16K PEs. L'algorithme utilisé est le disque de projection. Les résultats obtenus sont à comparer favorablement à ceux de l'approche séquntielle, bien que le gain (environ 4 en résoltuion maximale) soit relativement faible. Ceci provient du faible rendement obtenu par cette approche (Figure 6.15.b). A noter cependant la faible croissance des temps, du fait de l'utilisation de la notion de rectangles englobants.

6.2.8 Conclusion

Cette approche, basée sur un parallélisme proxel, permet d'obtenir un certain gain par rapport à une implantation séquentielle. Elle souffre néanmoins d'un très mauvais rendement, puisque le nombre de proxels effectivement présents à l'intérieur du contour d'une facette projeté est faible par rapport au nombre total de proxels calculés simultanément. L'utilisation des rectangles englobants une projection permet de limiter considérablement le nombre de zones à considérer, mais ne permet cependant d'utiliser efficacement l'ensemble des processeurs.

6.3 Evaluation d'une approche objet

L'approche objet, telle qu'elle a été décrite au paragraphe 6.1.2, est basée sur deux mécanismes principaux: un processeur par objet, qui détermine en tout pixel, si un objet est présent ou pas; un décideur, qui effectue l'élimination des parties cachées afin d'obtenir l'objet finalement présent dans un pixel.

Nous allons nous attacher, dans ce paragraphe, à évaluer l'opportunité de l'implantation d'une approche objet sur la Maspar, en étudiant en particulier le coût de calcul d'un proxel et les possibilités d'implantation d'un mécanisme de décision efficace.

6.3.1 Calcul d'un proxel

Le fonctionnement d'un algorithme basé sur une approche objet est identique, que le décideur soit arborescent ou pipeline. Le temps total d'exécution est donné par:

$$T_{exe} = N_p(T_c + T_{pc})$$

où

- T_{exe} représente le temps total d'exécution de la phase de projection
- N_p est le nombre de proxels à évaluer
- T_c est le temps de calcul nécessaire à l'évaluation de la présence de l'objet dans le proxel
- T_{pc} représente le temps utilisé pour la détermination des parties cachées (mécanisme de décision)

Le calcul permettant de déterminer si une facette est visible en un proxel consiste à évaluer, dans le cas de facettes planes convexes, chacune des équations de droite délimitant le contour projeté, et à vérifier qu'elles donnent toutes le même signe. Nous avons évalué le temps de traitement d'un proxel T_c, indépendamment de la méthode de décision utilisée, en supposant que les facettes à projeter possèdent toutes le même nombre de côtés. En l'occurence, nous avons choisi, dans ces mesures, d'utiliser des facettes triangulaires, chaque processeur ne disposant que d'une seule facette. L'ensemble des calculs a été effectué incrémentalement sur des données entières, puis des données réelles.

Les résultats obtenus nous fournissent un temps T_c égal à environ 9.10^{-5} secondes et 13.10^{-5} secondes respectivement, pour l'arithmétique entière et flottante. Le traitement de 10^6 proxels (résolution 1000x1000 d'un plan unique) nécessiterait donc au minimum 90 secondes, ce temps étant multiplié par le nombre de facettes gérées par chaque processeur s'il y a plus de facettes que de processeurs.

Bien que ces résultats nous permettent de conclure à l'inefficacité d'une telle approche sur la MP-1, il nous semble intéressant d'étudier le problème du décideur sur une architecture massivement parallèle telle que la Maspar. En effet, si l'accroissement des performances des PEs pourrait laisser envisager une implantation future basée sur un parallélisme objet, son efficacité reposerait alors essentiellement sur son mécanisme de décision.

6.3.2 Décideur arborescent

La configuration d'un décideur sous forme d'arbre permet d'obtenir la facette la plus proche visible au travers d'un pixel en un temps logarithmique, plutôt qu'en un temps linéaire.

Différentes fonctions de réduction, disponibles dans le langage MPL, s'apparentent à un décideur arborescent. Il est ainsi possible, à l'aide de la fonction *reduceMinf*, de simuler un décideur arborescent calculant la plus petite valeur *Vmin* parmi *N* valeurs réelles *V*.

Ceci ne permet cependant pas d'assurer l'efficacité du mécanisme. En effet, ces différentes fonctions de réduction ne sont pas cablées au sein de la machine, mais implantées dans le micro-code. Si leur temps d'exécution peut être considéré comme négligeable pour des utilisations occasionnelles, celui-ci devient très vite important dans le cas qui nous intéresse, puisque l'appel à la fonction *reduceMinf* devra être fait pour chaque proxel généré.

A titre d'indication, nous avons fait figurer dans le tableau ci-dessous, le temps d'exécution de certaines fonctions de réduction, pour une configuration disposant de 1024 processeurs. Les fonctions utilisées concernent un ensemble de valeurs réelles, et sont, respectivement *reduceMinf* (le minimum), *reduceMaxf* (le maximum), *reduceAddf* (la somme) et *reduceMulf* (le produit).

Fonction	*reduceMinf*	*reduceMaxf*	*reduceAddf*	*reduceMulf*
Temps (s)	*12,59*	*11,05*	*27,11*	*35,14*

Table 6.1 ***Aperçu des temps d'exécution de quelques fonctions de réduction***

Ces temps ont été obtenus pour 100000 itérations (100000 proxels), ce qui correspond approximativement à une résolution de 316x316 proxels. Ces temps croissant linéairement, l'exécution du mécanisme de réduction pour une surface découpée en 1000x1000 proxels nécessite environ 2 minutes de calcul à lui seul. Un tel mécanisme ne peut être, a priori, utilisé que s'il existe un câblage sous-jacent offrant une puissance suffisante.

6.3.3 Décideur pipeline

Dans le cas d'un décideur pipeline, les proxels passent successivement par chaque proceseur. Un processeur recevant un proxel détermine si l'objet qu'il gère y est visible et, si c'est le cas, effectue une comparaison de profondeur avec l'objet qui s'y trouve déjà. Lorsque l'objet le plus proche a été rangé dans le proxel, celui-ci est transmis au processeur suivant. L'élimination des parties cachées est donc effectuée par comparaisons successives le long du pipeline.

Deux paramètres sont alors à mesurer pour estimer le coût du décideur pipeline: la comparaison entre la valeur calculée localement et la valeur présente dans le proxel fournit par le processeur de gauche; la transmission du proxel vers le voisin de droite.

La comparaison (test de profondeur et échange des valeurs) ne doit être effectuée que si l'objet est effectivement présent dans le proxel qui parvient au processeur. Cependant, il faut rappeler que nous nous trouvons dans un mode de fonctionnement SIMD. Par conséquent, chaque processeur effectue simultanément le même code de comparaison. Au vu du nombre élevé de processeurs utilisés, il semble fortement probable, qu'à chaque nouveau cycle de traitement d'un proxel dans le pipeline, au moins un processeur (si ce n'est plus) effectuera la comparaison. Le temps total T_{tc} utilisé pour les comparaisons pendant le traitement de l'ensemble des proxels est donc défini comme:

$$T_{tc} = T_c(N_{prox} + N_{proc})$$

où

- T_t représente le temps d'une seule comparaison
- N_{prox} est le nombre total de proxels
- N_{proc} est le nombre de processeurs du pipeline

Il faut en effet ajouter, au nombre de proxels traités, le temps d'amorçage du pipeline, qui est égal au produit T_cN_{proc}. Les simulations numériques, tenant compte du test et de l'échange des

valeurs d'identité et de profondeur, nous permettent d'évaluer le temps de comparaison T_c à $1{,}8.10^{-5}$ secondes.

Aprés avoir effectué la comparaison, le proxel doit être transmis vers le processeur suivant dans le pipeline. Le temps total de transmission T_{tt} vérifie la même équation que celle définie pour le temps total de comparaison T_{tc}:

$$T_{tt} = T_t(N_{prox} + N_{proc})$$

où

- T_t représente le temps d'une seule transmission
- N_{prox} est le nombre total de proxels
- N_{proc} est le nombre de processeurs du pipeline

Un problème se pose néanmoins lors de cette phase de transmission, car nous avons supposé jusqu'à présent que les processeurs étaient disposés en pipeline. Or, les processeurs de la Maspar sont disposés selon une grille à deux dimensions. La simulation d'un pipeline nécessitera donc deux étapes de communications:

1. Une première étape consiste à envoyer le proxel local à chaque processeur vers le processeur de droite, sauf pour les processeurs situés en fin de ligne.
2. Lors de la seconde étape, ces processeurs envoient leur proxel vers le processeur situé au sud-est par rapport à leur position. La connexion torique des processeurs de la grille assure que le proxel arrivera sur le premier processeur de la ligne du dessous (nous négligeons, pour les mesures, le cas du dernier processeur de la grille).

La simulation de ces communications, chaque proxel contenant une valeur réelle (profondeur) et une valeur entière (identité de la facette), permet d'estimer la valeur de T_t à environ 5.10^{-5} secondes.

Le coût total du décideur pipeline pour un proxel, prenant en compte à la fois le temps de comparaison et le temps de transmission, se monte ainsi à $6{,}8.10^{-5}$ secondes. L'exécution du mécanisme de réduction pipeline sur un réseau de 1024 processeurs, chacun d'entre eux ne gérant qu'une seule facette, représente donc un coût d'environ 68 secondes pour traiter 10^6 proxels, le temps d'amorçage du pipeline étant négligeable devant le nombre de proxels à traiter.

6.3.4 Conclusion sur l'approche objet

Les chiffres obtenus ci-dessus montrent l'importance du coût de l'implantation d'une approche objet sur la MP-1. Ce coût est résulte à la fois du calcul de présence proprement dit, mais aussi du mécanisme de décision qui fait intervenir un nombre important de communications.

Ces coûts sont principalement liés au nombre de proxels à calculer et, dans une moindre mesure, au nombre de facettes de la base de données. Contrairement à l'implantation d'une approche proxel, telle que nous l'avons étudiée dans la paragraphe 6.2, une approche objet nécessite d'évaluer la présence d'un objet en chaque proxel. En effet, le mode de fonctionnement SIMD de l'approche ne permet pas de bénéficier de la notion de rectangle englobant, dans la mesure où le nombre élevé de facettes traitées simultanément nécessite le calcul d'un proxel au moins en un processeur. Lorsque le nombre de facettes présentes en chaque processeur augmente, l'utilisation des rectangles englobants pourrait être envisagée, mais il n'est pas du tout certain que le surcoût induit pour leur gestion puisse être contrebalancé par le gain obtenu.

En tout état de cause, l'implantation d'une approche objet sur une machine massivement parallèle semble difficilement envisageable, de par le faible rendement qu'elle procure. Elle nécessiterait des processeurs beaucoup plus puissant, et surtout, un mécanisme cablé de décision actuellement absent de la Maspar ou de beaucoup d'autres machines massivement parallèles.

6.4 Une solution utilisant une approche hybride

Le principal défaut des approches basées sur un parallélisme proxels ou objets concerne le très faible rendement qu'elles fournissent. Ceci est dû au fait que les calculs effectués ne sont pas restreints aux proxels effectivement recouverts, mais appliqués à l'ensemble des proxels utilisés sur la surface de projection. Bien que ces approches possèdent un mode de fonctionnement de type SIMD, et donc exploitable sur la MP-1, elles ne permettent pas d'atteindre des performances suffisantes. De plus, augmenter le nombre de PEs ne suffit pas à augmenter les performances dans des proportions très importantes.

La seule façon d'augmenter notablement les performances, hormis l'augmentation de puissance des processeurs eux-mêmes, est d'exploiter au maximum l'ensemble des processeurs de manière utile. En d'autres termes, il s'agit de trouver un algorithme de type SIMD, applicable à l'ensemble des processeurs, et tel qu'aucun processeur n'ait à effectuer des opérations sur des proxels qui ne présentent pas d'intérêt. Nous allons présenter ci-dessous un algorithme dôté de telles propriétés. Il permet d'exploiter, selon les étapes, soit un parallélisme objet, soit un parallélisme de type proxel. Nous allons détailler ci-dessous les différentes étapes de l'algorithme, puis nous analysrons un certain nombre de résultats obtenus.

6.4.1 Présentation de l'algorithme

Le principe de base de cette nouvelle approche est de ne traiter que les proxels utiles, c'est à dire uniquement les proxels effectivement recouverts par la projection d'une facette. Deux phases sont alors à distinguer dans l'algorithme:

- la phase de conversion des facettes en proxels: chaque processeur traite une facette différente et détermine l'ensemble des proxels recouverts par cette facette.
- la phase d'élimination des parties cachées: chaque processeur traite le sous-ensemble des proxels précédemment générés qui appartient à la partie de la surface de projection qu'il gére, et lui applique une opération d'élimination des parties cachées.

Pour chacune de ces étapes, les traitements appliqués sur un proxel seront tous utiles, en ce sens que le proxel calculé sera nécessairement couvert par la facette utilisée par un processeur (contrairement à l'approche proxel), et que l'opération de comparaison de profondeur ne sera appliquée que dans le processeur où le proxel est visible (contrairement à l'approche objet). Ces deux étapes sont détaillées ci-dessous.

Génération des proxels

Chaque processeur traite une facette différente. Le traitement consiste à générer la liste des intersections entre les lignes horizontales d'une grille de proxels et le contour de la facette à projeter (*spans*). Cette opération est schématisée sur la Figure 6.16.

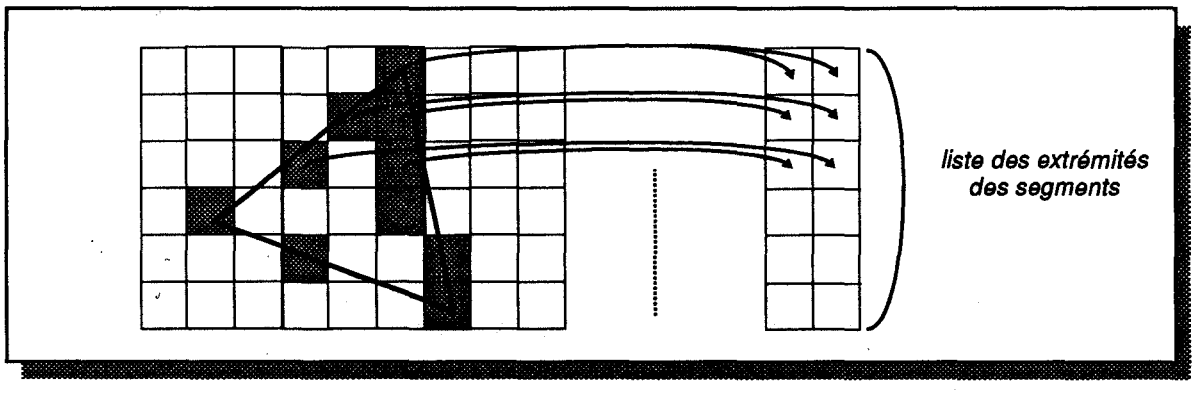


Figure 6.16 ***Génération de la liste des intersections entre une facette et les lignes de proxels***

Cette opération est appliquée simultanément sur *N* facettes, *N* étant le nombre de processeurs disponibles dans le réseau. Elle est ensuite à nouveau appliquée sur les autres facettes

mémorisées sur chaque PE (dans le cas où il y a plus de facettes que de PEs). Les informations mémorisées dans la liste des extrémités comprennent, pour chaque élément: un indice de ligne, deux indices de colonne (début et fin du segment) et deux valeurs de profondeur (une pour chacune des deux extrémités). De manière à éviter une perte de temps, cette liste n'est pas triée, les nouveaux segments s'ajoutant à la suite de ceux précédemment calculés.

Le choix de la génération des segments intersection, plutôt que des proxels eux-même, est motivé par plusieurs constatations:

- Chaque facette recouvre un nombre relativement important de proxels. Bien que ce nombre soit difficile à évaluer a priori, il est en moyenne plus élevé que le nombre de lignes coupant le contour de la facette (il est au minimum égal au nombre de lignes). D'un point de vue mémoire, il est donc plus intéressant de ne mémoriser que les extrémités d'un segment plutôt que l'ensemble de ses proxels, sachant qu'une grande partie de la mémoire de chaque PE doit être réservée pour la représentation de la base de données et du plan de projection.
- La génération de segments est une opération qui permet un meilleur équilibre de charge entre les différents processeurs que la génération de tous les proxels. En effet, obtenir les extrémités d'un segment revient à calculer l'intersection d'une ligne de balayage avec deux côtés de la facette projetée. Cette opération peut donc être effectuée efficacement en mode SIMD, puisqu'elle est la même pour tous les PEs. Générer l'ensemble des proxels d'un segment, par contre, introduit un déséquilibre de charge entre les processeurs, dans la mesure où les segments n'ont pas tous la même longueur. En mode SIMD, l'ensemble des processeurs devra donc fonctionner au rythme du plus lent pour chaque segment (le problème similaire des différences de hauteur entre facettes sera traité un peu plus loin dans cette partie).
- L'élimination des parties cachées, qui suit l'opération de conversion des objets en proxels, nécessite de transmettre les proxels générés vers les processeurs qui sont chargés de la comparaison de profondeur pour les parties de la surface de projection qu'ils mémorisent. Dans la mesure où le nombre total de proxels obtenus est supérieur au nombre total de segments calculés, il est plus intéressant de transmettre des segments plutôt que des proxels.

De la même manière que pour le problème de la longueur différente des lignes de proxels se pose celui des hauteurs différentes des facettes projetées. En effet, le nombre de lignes où une facette est présente varie fortement d'une facette à l'autre. Le mode de fonctionnement SIMD des processeurs implique donc que l'ensemble du réseau va travailler au rythme du plus lent, c'est à dire de celui qui calcule les segments recouverts par la facette la plus haute. Nous avons donc implanté un mécanisme de changement de contexte, inspiré d'un algorithme du même type, développé pour la machine Sympathix [Letel 93]. Son principe est assez simple: à chaque fois que le traitement d'une facette est terminé sur un processeur, c'est à dire que sa dernière ligne a été traitée, une nouvelle facette est immédiatement "poussée" à sa place, de manière à ce qu'elle puisse être utilisée dès le traitement général d'une nouvelle ligne. Pour éviter une perte de temps due au calcul des nouveaux paramètres à utiliser, tous les paramètres propres à chaque facette sont calculés et mémorisés avant le déclenchement de l'étape de conversion des facettes en segments. De cette manière, le changement de contexte se résume à une simple modification (affectation) des paramètres qui étaient utilisés.

Dans la mesure où chaque processeur applique simultanément la même opération sur une facette différente, cet algorithme posséde les caractéristiques d'une approche à parallélisme objet. Ce qui le différencie, jusqu'à présent, des approches objet précédemment présentées, est que, au lieu de rechercher la présence d'un objet en chaque proxel, l'ensemble des proxels réellement recouverts est généré (par l'intermédiaire des listes de segments). Ceci permet une optimisation considérable du processus de conversion, dans la mesure où le nombre de proxels réellement recouverts est très faible, comparé au nombre total de proxels résultant du découpage de la surface de projection. Il reste cependant à résoudre le problème de l'élimination des parties cachées, afin d'obtenir la facette visible en chaque proxel.

Elimination des parties cachées

L'implantation des structures arborescentes ou pipelines pour le décideur des approches objet, est difficilement envisageable sur la Maspar. D'une part, les coûts de communication induits par de tels mécanismes sont prohibitifs (cf paragraphe 6.3). D'autre part, la génération de proxels différents sur les PEs ne permet plus leur utilisation, puisque ces deux mécanismes impliquent une régularité dans l'ordre de traitement des proxels.

Une approche à parallélisme proxel semble, par contre, beaucoup plus indiquée, compte tenu du type d'informations générées par la première étape de l'algorithme. En effet, après la phase de conversion, chaque processeur dispose d'une liste de segments qu'il faut utiliser pour générer l'ensemble des proxels. Cette structure correspond fortement à un shéma de diffusion de type multipipeline, où la génération des proxels recouverts se fait le long des lignes de la grille de projection. Dans la mesure où il est souhaitable de ne traiter en chaque processeur que les proxels réellement utiles, ce schéma de diffusion devra être adapté à la longueur de chaque segment généré.

Il faut cependant souligner le fait que les listes de segments sont réparties de manière hétérogène sur les processeurs. Comme chaque processeur gère une partie seulement des proxels, il est nécessaire d'effectuer le transfert de chaque segment vers le(s) processeur(s) qui gére(nt) la partie de la surface de projection à laquelle ils appartiennent.

Avant d'aborder ce point, il est important de savoir de quelle manière les proxels sont distribués sur la grille de PEs. Deux modes de répartition ont été présentés au paragraphe 6.2.3 : la répartition par blocs et la répartition cyclique. L'utilisation d'une répartition par blocs génère nécessairement des déséquilibres de charge lors du traitement des segments, puisque le nombre de segments présents dans une zone (allouée à un processeur) varie fortement d'une zone à l'autre, en fonction de la complexité locale de "l'image" projetée. Une répartition par cycle, par contre, permet d'équilibrer le nombre de segments traités par chaque processeur, puisque le "poids" du traitement d'un segment repose sur tous les processeurs qui gérent l'un des proxels du segment.

L'algorithme implanté s'effectue en deux phases:

- Une phase de distribution des segments, qui permet d'envoyer chaque segment vers le processeur qui gère son premier proxel.
- Suivie de la phase d'élimination des parties cachées proprement dite, où chaque segment est propagé vers le processeur de droite, jusqu'à ce que tous les proxels du segment aient été traités.

Ces différentes étapes vont toutes deux utiliser les communications de manière intensives. La première pour envoyer les segments vers les PEs de départ; la seconde pour effectuer la propagation des segments selon le mode multipipeline. Chacune de ces étapes est détaillée ci-dessous.

Distribution des segments

Le problème de la distribution des segments est similaire au traitement effectué dans le cas de la récupération des proxels, lors de la phase de mise à jour des radiosités (paragraphe 6.2.6). Nous avons évalué les mêmes types de communication que dans cette phase.

La distribution des segments utilisant le global router utilise le même principe : chaque PE selectionne l'un de ses segments, puis détermine le processeur qui gère le premier proxel du segment. Le segment est alors envoyé au PE. Cette phase de distribution est effectué jusqu'à ce que tous les segments aient atteints leur processeur de départ. La Figure 6.17.a représente le temps moyen de distribution des segments en fonction de la résolution du plan

de projection, et la Figure 6.17.b représente la répartition moyenne des segments sur les processeurs.

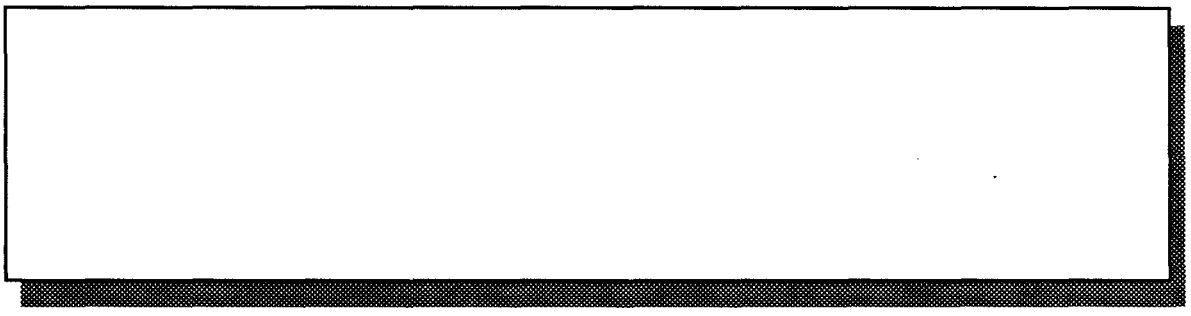


Figure 6.17 ***Temps moyen de distribution des segments (a) et répartition moyenne des segments sur les processeurs (b).***

La seconde approche utilise le réseau Xnet. Chaque processeur sélectionne un segment et le propage vers le bas jusqu'à ce qu'il ait atteint la bonne ligne. Pour atteindre le processeur de départ, le segment doit ensuite se propager horizontalement, jusqu'à atteindre la bonne colonne. La figure X montre l'évolution du temps moyen de propagation vertical, tandis que la figure X.b représente la même évolution pour les propagation horizontale.

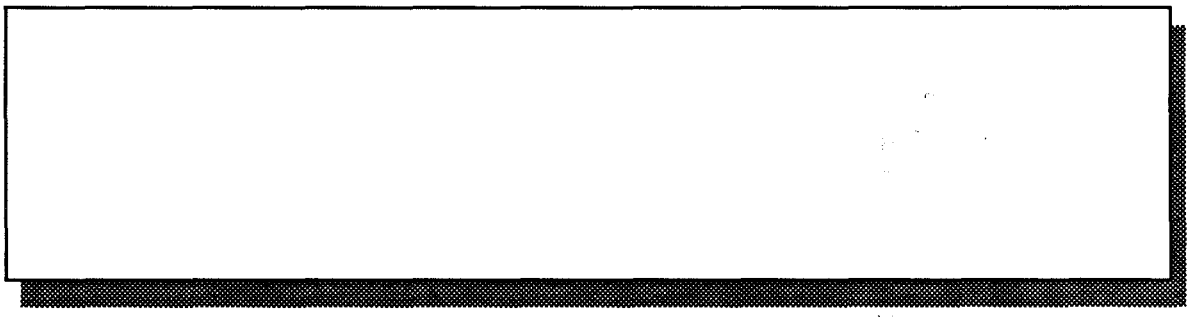


Figure 6.18 ***temps moyen de propagation vertical (a) et horizontal (b) en utilisant le réseau Xnet.***

Propagation des segments

Lorsque la phase de distribution des segments aux différents processeurs est terminée, chacun d'entre eux dispose de la liste complète des segments qui commencent dans l'un de ses proxels. Commence alors la phase de propagation de ces segments, permettant de compléter l'élimination des parties cachées.

Chaque processeur active l'un de ses segments et prépare les différentes valeurs qu'il sera nécessaire de propager (entre autres : longueur restante, profondeur du proxel courant, incrément de profondeur,...). Il effectue les opérations d'élimination des parties cachées sur le proxel courant pour son segment, met à jour les différentes valeurs utilisées, puis le transmet à son voisin de droite, en même temps qu'il reçoit un nouveau segment depuis son voisin de gauche. Lorsque l'un des segments reçus est terminé (longueur nulle), il active l'un des ses segments locaux. Notons qu'il n'y a aucun problème de débordement lorsqu'un segment arrive sur la dernière colonne de la grille de processeurs, puisque les connexions physiques sont rebouclées vers les processeurs de la première colonne.

L'algorithme de propagation des segments est écrit ci-dessous en pseudo-langage

```
Activer_un_segment_local

Faire
|
|     Si segment_terminé
|     |
|     |     activer_un_segment_local
|     |
|     fsi
|
|     comparaison de profondeur
|
|     mettre_à_jour_les_incréments_du_segment
|
|     transmettre_et_recevoir_un_segment
|
Jusqu'à ce qu'il reste des segments à propager
```

Les temps de propagations ont été mesurés pour différentes résolutions d'un plan de projection, et apparaissent sur la figure ci-dessous (Figure 6.19).

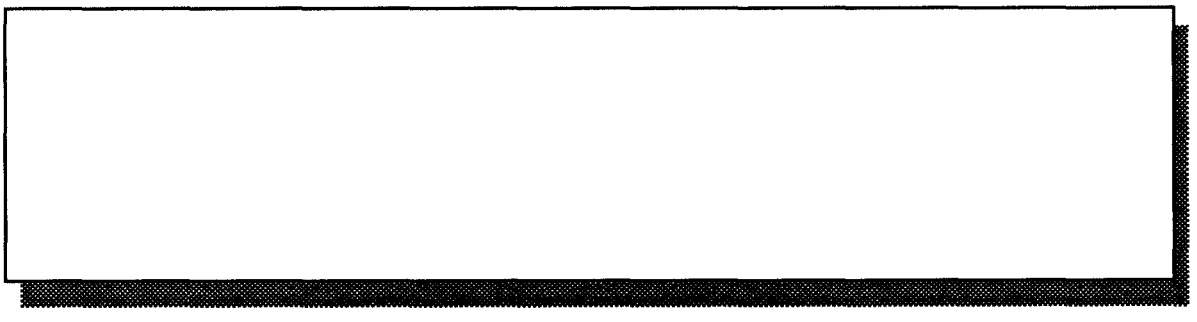


Figure 6.19 ***Evolution des temps moyens de propagation pour différentes résolutions***

Résultats

L'avantage de cette approche est de limiter considérablement le nombre de processeurs effectuants des opérations inutiles. Chaque processeur calcule tout d'abord les segments des facettes dont il dispose, puis transmet ces segments aux processeurs où ils commencent. Les segments ne sont ensuite traités que sur leur longueur exacte, et ne passent donc que par des processeurs appliquant des opérations utiles. Les déperditions n'interviennent alors que pendant les communications. La Figure 6.20représente l'évolution de la proportion du temps de communication, pour la scène "Bureau2", selon la résolution.

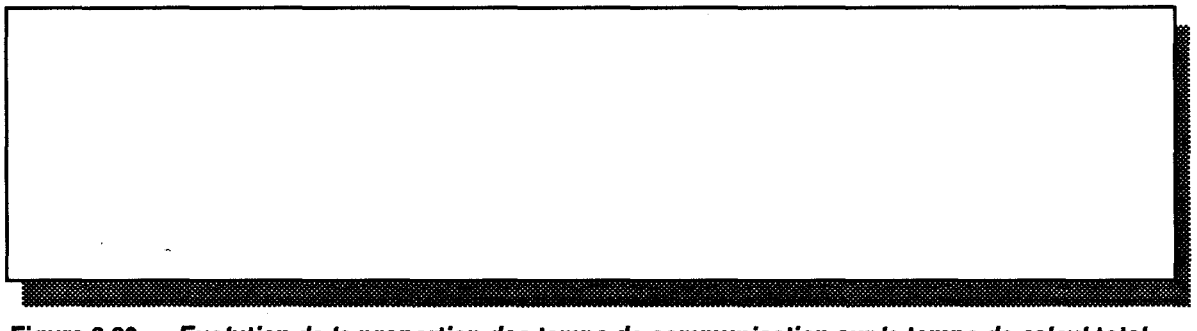


Figure 6.20 ***Evolution de la proportion des temps de communication sur le temps de calcul total.***

Cet algorithme est relativement proche d'un algorithme proposé par Varshney [Varsh 92]. La principale différence provient de la façon de distribuer les proxels sur la grille de processeurs. Varshney utilise une distribution par bloc, ce qui génère un fort déséquilibre de charge entre les processeurs. En effet, les facettes à projeter ne se répartissent pas uniformément le plan de projection qu'il utilise. De ce fait, des processeurs se voient attribuer un grand nombre de facettes, alors que d'autres n'en obtiennent que très peu.

Notre approche assure implicitement l'équilibre de la charge, puisque les chaque facette se répart sur un grand nombre de processeurs. Le tableau ci-dessous résume les temps moyens de calcul obtenus avec notre approche en fonction de la résolution utilisée.

Table 6.2 ***Temps de calculs total pour les différentes base de données***

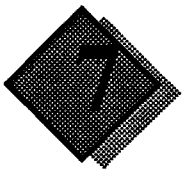
6.4.2 Conclusion

Nous avons étudié différentes approches permettant d'implanter les méthodes projectives sur une machine massivement parallèle utilisant un fonctionnement SIMD. Les solutions classiquement utilisées pour l'implantation physique du parallélisme pixel ou du parallélisme objet sont mal adaptée à ce genre d'application. En effet, si un faible rendement peut être admis dans le cas de circuits massivement parallèles spécialisés, en regard des perfomances atteintes, celui-ci ne peut être toléré dans le cas d'une machine d'usage général.

L'étude des différentes opérations à appliquer lors de la phase de projection nous a permis de concevoir une approche hybride, utilisant les avantages conjugués du parallélisme objet et du parallélisme pixel. D'autre part, l'étude des moyens de communications disponibles sur la Maspar nous ont permis d'étudier les schémas de communication les plus efficaces pour chacune des opérations à appliquer. Il reste cependant que ces communications restent coûteuses sur ce type de machine.

Notre approche n'utilise qu'un plan de projection unique, découpé uniformément en proxels de même taille. Nous avons montré, au chapitre 2, que ce type de support de projection génère un suréchantillonnage important sur une grande partie du carré de projection. L'extension de l'algorithme à l'hémicube ou au plan de projection de Sillion ne pose cependant pas de problèmes. Nous travaillons sur son extension au disque de projection, qui permet de diminuer à la fois le nombre de projection à appliquer et le nombre d'échantillons à utiliser.

CHAPITRE



Résultats

Nous présentons, dans ce chapitre, un certain nombre de résultats obtenus pour les différentes implantations que nous avons effectué sur la Maspar. Ces résultats regroupent différentes mesures appliquées sur l'approche à parallélisme proxel, puis sur l'approche hybride que nous avons proposé.

7.1 Approche à parallèlisme massif proxel

Dans cette approche, chaque facette est traitée séquentiellement, et l'ensemble des processeurs calculent si cette facette est présente ou pas dans les proxels qu'ils gèrent.

7.1.1 Evolution des temps de calcul

Différents algorithmes d'échantillonnage ont été implantés, parmis lesquels l'hémicube, le plan unique utilisant un découpage uniforme et le découpage proposé par Sillion, et enfin le disque. L'évolution des temps de calcul pour ces différents algorithmes est représenté sur la Figure 7.1.

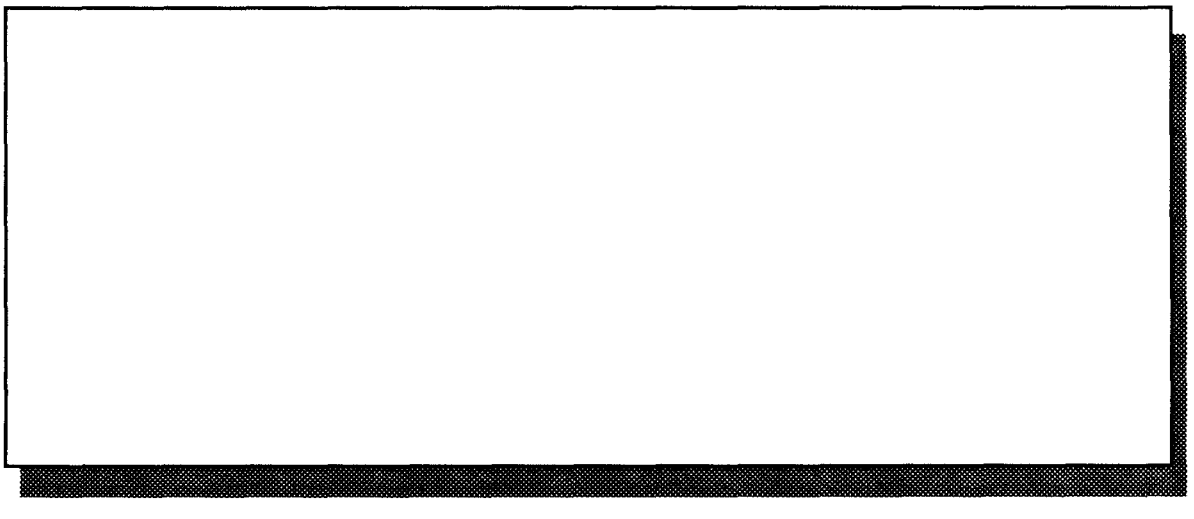


Figure 7.1 ***Evolution des temps de calcul moyen pour les différentes scènes.***

La phase de mise à jour des radiosités a été effectuée, dans tous les cas, à l'aide du réseau de routage Global Router, en utilisant la stratégie "modulo" que nous avons décrite au chapitre précédent.

7.1.2 Utilisation des rectangles englobants

Les rectangles englobant chacune des projections sont utilisés de manière à limiter le nombre de zones du plan de projection à considérer lors des phases de calcul d'élimination des parties

cachées. Nous avons représenté, sur la Figure 7.2, le nombre moyen de zones considérées pour chaque scène.

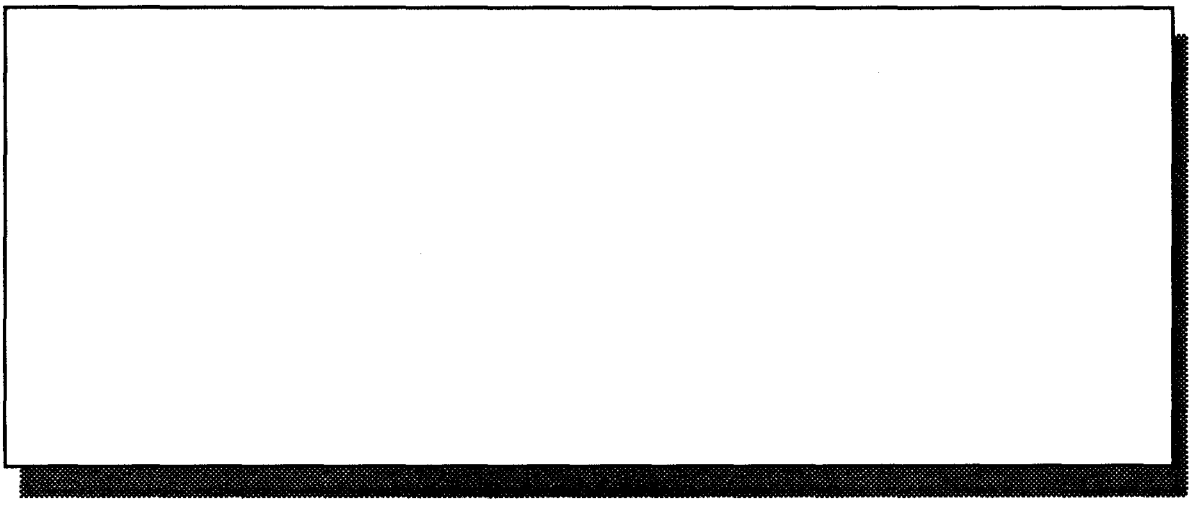


Figure 7.2 *Evolution du nombre de zones considérées en utilisant les rectangle englobants*

7.1.3 répartition des temps calcul

7.1.4 Mise à jour des radiosités

Différents schémas de communication ont été évalué pour la mise à jour des radiosités. Une comparaison, en termes de temps de calcul, est proposée sur la Figure 7.3. La machine utilisée dispose de 16K processeurs, ce qui permet 1024 connexions simutanée par l'intermédiaire du global router.

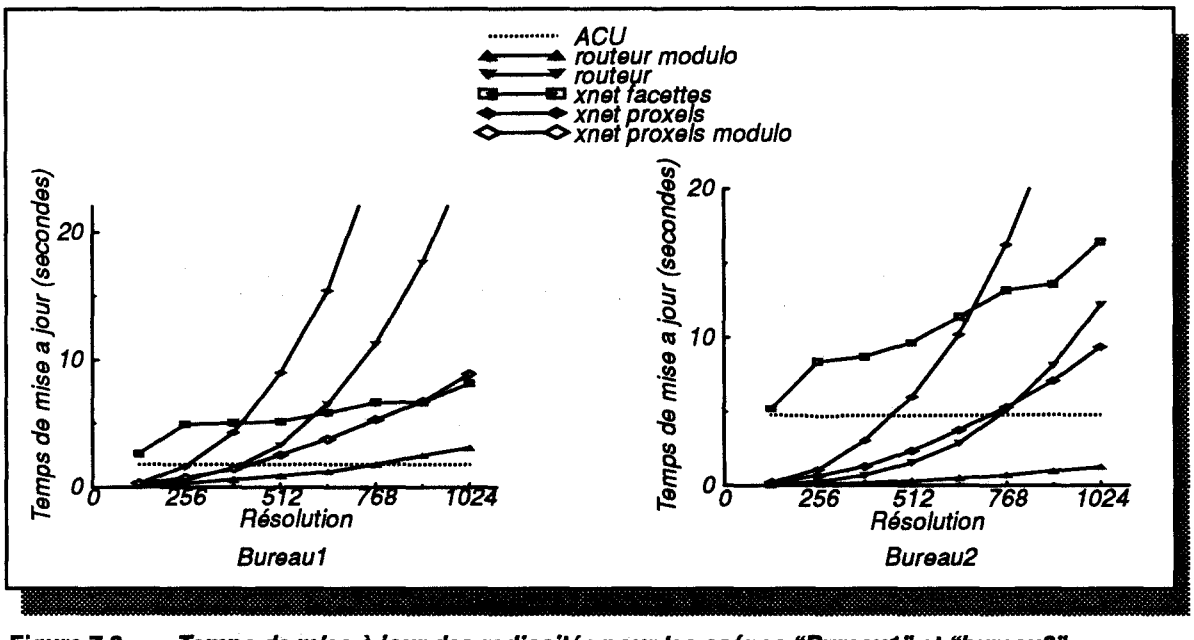


Figure 7.3 *Temps de mise à jour des radiosités pour les scénes "Bureau1" et "bureau2"*

L'évolution des différentes courbes montre l'intérêt très net qu'il y a à utiliser le global router, associé à notre stratégie modulo. Il apparait que l'utilisation de cette stratégie pour les scènes "Bureau1" et "Bureau2" présente une différence intéressante : en effet, la scène "bureau2", bien

que plus complexe, necessite moins de temps pour la mise à jour des radiosités, que l'on utilise la stratégie modulo ou pas. En fait ceci s'explique par le fait que la base de données "Bureau1" est plus petite (en terme de dimensions absolues) que la scène "Bureau2". Les distances entre les facettes sont alors en moyenne plus grandes dans cette seconde scène, et la surface projetée des facettes plus petite. De ce fait, le nombre total de conflits provenant du phénomène de voisinage décrit au chapitre précédent, diminue et permet d'obtenir des performances plus inportantes quand au débit de mise à jour.

Notons que l'utilisation de l'ACU peut être intéressante dans le cas de base de données de peitites taille, puisque le temps de mise à jour est alors directement proportionnel au nombre de facettes de la scène.

7.1.5 Rendement des différents algorithmes

Les rendements qui sont présentés ci-dessous ont été calculés en déterminant le nombre de proxels situés à l'intérieur du contour projeté, divisé par le nombre total de proxels concerné par la phase de calcul.

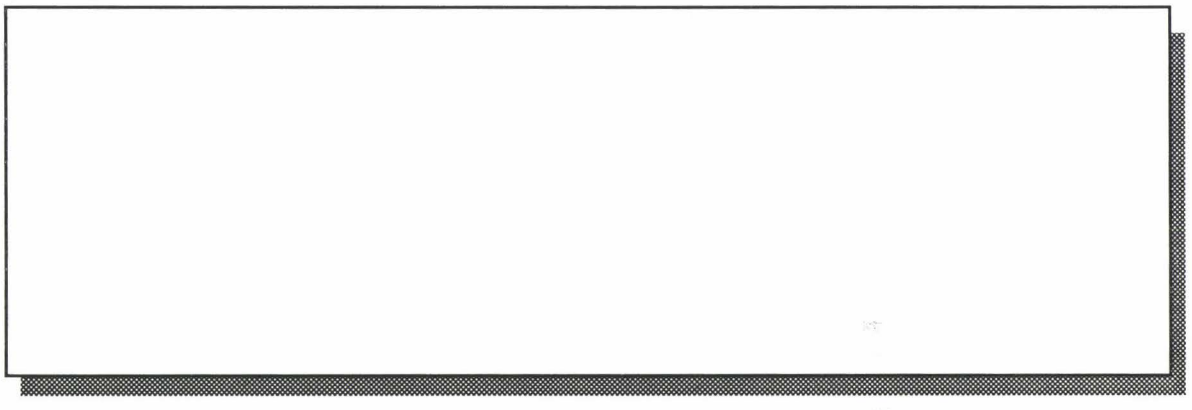


Figure 7.4 *Evolution du rendement en fonction de la résolution.*

7.2 Approche hybride

Nous présentons dans cette partie les résultats obtenus avec notre approche hybride. Celle-ci n'a, pour le moment, été utilisée que sur une surface de projection représentée par un plan de projection unique, découpée uniformément en proxels de même taille. Les résultats présentés reprennent, en partie, ceux déjà obtenus dans le chapitre 6, mais en les généralisant à l'ensemble des scènes de test (cf Annexe A).

7.2.1 Temps de calcul

La Figure 7.5 présente l'évolution des temps moyens de calcul pour chacune des scènes de test, en fonction de la résolution utilisée pour le plan de projection.

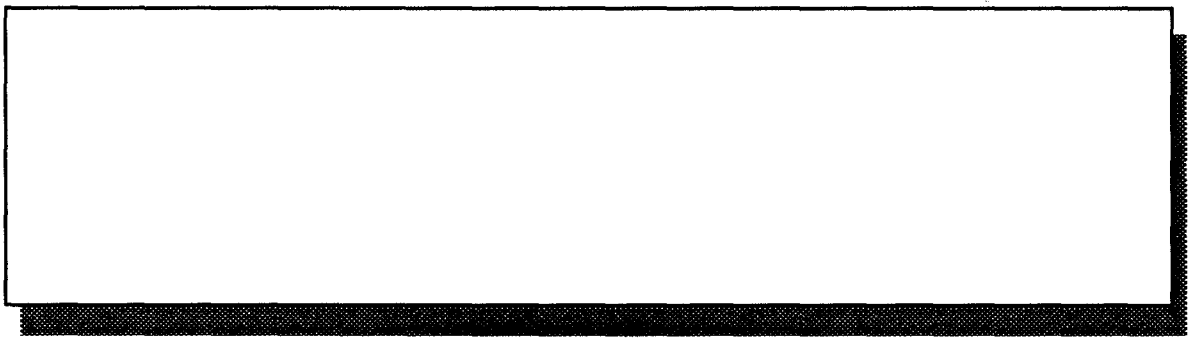


Figure 7.5 ***Temps de calcul en fonction de la résolution***

7.2.2 Répartition des temps de calcul

La répartition des temps de calcul entre les différentes étapes de l'algorithme sont présentées sur la figure ci-dessous (Figure 7.6).

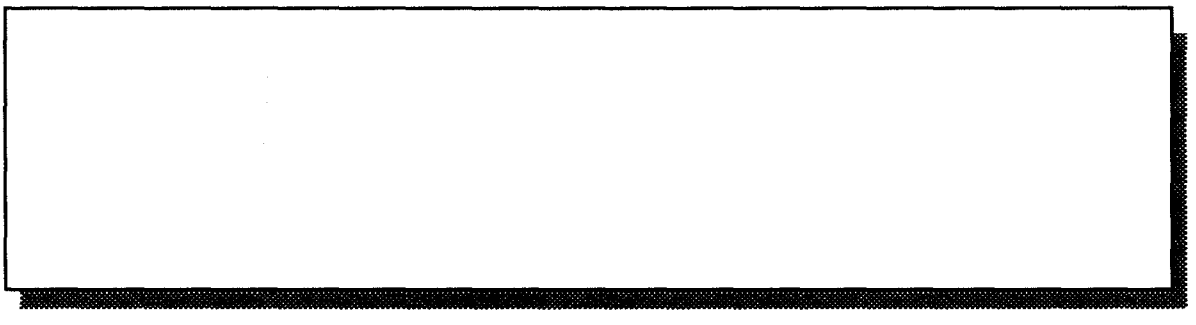


Figure 7.6 ***Répartition moyenne des temps de calcul pour les différentes scènes.***

Ces temps sont présentés pour différentes résolutions du plan, et pour chacune des scènes de test.

7.2.3 Temps de distribution des segments

Nous avons représenté ci-dessous les temps de distribution des segments vers le processeur contenant leur premier proxel, en utilisant les réseaux Xnet et Global Router.

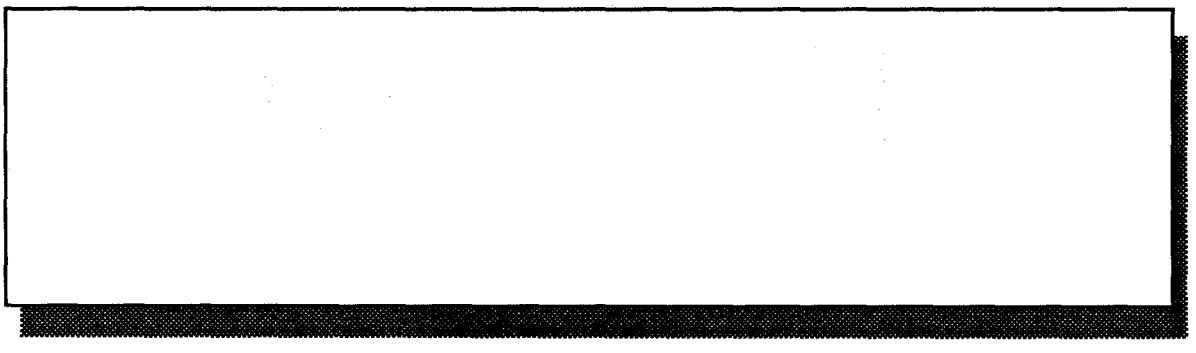


Figure 7.7 ***Temps moyens de distribution des segments en utilisant le réseau Xnet (a) ou le global router (b)***

7.2.4 Répartition des segments sur les processeurs.

L'évaluation de la charge des processeurs peut être obtenue, pour la seconde phase du calcul, en visualisant le nombre de segments commencant sur chaque processeur. La Figure 7.8 représente le nombre moyen de segments par processeur, en fonction de la résolution du plan de projection.

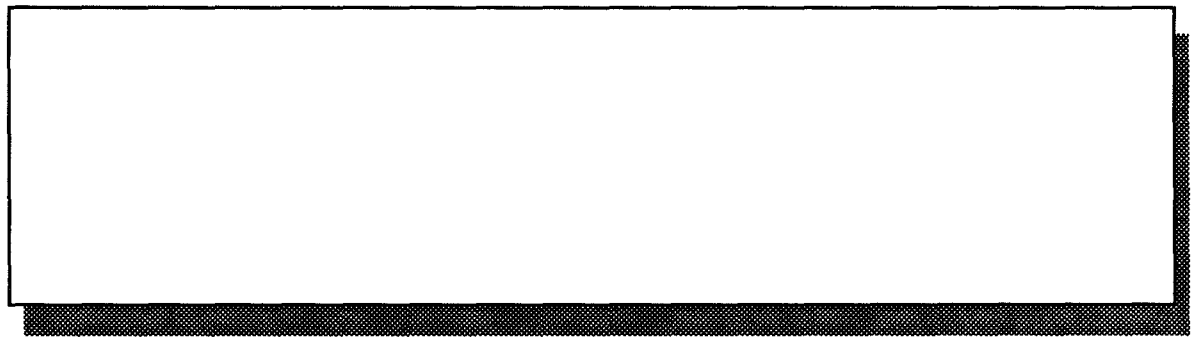
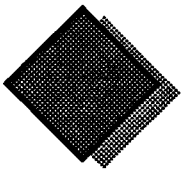


Figure 7.8 ***Nombre moyen de segments par processeur.***

Conclusion



L'algorithme de radiosité permet d'obtenir des images très réalistes, en prenant en compte l'illumination globale d'une scène dans le cas de surfaces purement diffuses.Il nécessite le calcul de quantités géométriques appelées facteurs de forme, qui permet de quantifier la quantité d'énergie lumineuse échangée entre deux surfaces.

De nombreuses méthodes d'échantillonnage ont été proposées pour approximer les facteurs de forme, en prenant en compte les occultations entre objets. Elles peuvent être divisées en deux catégories, selon que leur principe de fonctionnement est basé sur des projections ou sur le lancer de rayons.

Nous avons présenté ces différentes méthodes en détail, et proposé un critère d'équivalence pour les méthodes projectives. Celui-ci nous a alors permis de comparer les différents algorithmes utilisés par ces méthodes selon différents critères. Un algorithme projectif, application directe de l'équivalent de Nusselt, a été introduit, et permet d'obtenir un gain important au niveau des coûts mémoire, tout en conservant une précision équivalente aux autres algorithmes. Le gain obtenu, en termes de coût de calcul, n'est pas aussi important que l'on pouvait l'espérer, du fait de la méthode de remplissage trop simple utilisée à ce jour. Des travaux sont en cours pour définir une méthode de plus haut niveau, permettant, entre autres, de ne pas considérer les proxels extérieurs à la projection lors de la phase d'élimination des parties cachées.

Les algorithmes utilisant le principe du lancer de rayons ont été présentés, de même que les nombreuses méthodes d'accélération habituellement utilisées. Du fait du nombre important de rayons à lancer dans le cas où une grande précision est demandée, les temps de calcul peuvent devenir très importants. La diminution du nombre de rayons génère des phénomènes de sous-échantillonnage, qui perturbent considérablement la précision du calcul des échanges lumineux. Nous avons proposé une nouvelle méthode d'échantillonnage, basée sur le tracé de faisceau, permettant de remplacer un grand nombre de rayons. Cette méthode est cependant coûteuse, dans l'état actuel de nos travaux. Néanmoins, elle peut bénéficier d'une grande partie des techniques d'accélération présentées dans le cas du lancer de rayons. Les résultats actuels suggèrent d'utiliser le lancer de faisceaux lorsqu'une grande précision est demandée, ce qui se produit dans le cas des source primaires, et de lancer un nombre restreint de rayons lors des autres phases.

Ces différents algorithmes présentent tous un coût de calcul élevé, qui ne peut être diminué que par l'augmentation de la puissance des processeurs, ou l'utilisation d'un grand nombre de processeurs travaillant en parallèle. C'est dans cette seconde voie que s'est engagée la seconde partie de notre travail.

Nous avons tout d'abord présenté les différents niveaux de parallélisme qui peuvent être extraits de l'algorithme de radiosité progressive, en tenant compte de la méthode de calcul des facteurs de forme utilisée. Trois niveaux principaux se sont dégagés de cette étude, correspondant chacun à une boucle de l'algorithme. Les différentes approches envisageables pour la parallélisation, dans un contexte MIMD, ont alors été présentées, en détaillant nos propres travaux et les nombreuses implantations ayant déjà été effectuées sur ce sujet. Ces approches ont été développées selon trois catégories distinctes, dépendantes du mode de

représentation de la base de données: duplication sur tous les processeurs, répartition équitable ou représentation unique. Les deux premières catégories correspondent aux machines à mémoire distribuée, tandis que la dernière correspond aux architectures à mémoire partagée.

Bien que les résultats obtenus dans le cas de la duplication de la base de données présentent une efficacité importante, cette approche n'est pas envisageable lorsque la base de données croît en complexité. Les approches utilisant une répartition globale de la scène offrent une solution à ce problème, en permettant le traitement de scènes beaucoup plus importantes. Cependant, du fait du modèle d'illumination utilisé en radiosité (éclairages et réflexions purement diffus), il apparait une très forte dépendance entre les différents objets d'une scène. Un grand nombre de communications apparaissent alors lorsque ceux-ci sont répartis sur différents processeurs, générant une forte diminution de l'efficacité. Il nous semble cependant envisageable d'exploiter des propriétés de cohérence à différents niveaux des algorithmes projectifs, permettant de réduire considérablement ces coûts. Ces propriétés de cohérence sont présentes à la fois au niveau du calcul de deux émissions successives depuis des points voisins, et au niveau de l'émission elle-même, entre des directions d'émission voisines.

Nous avons d'autre part analysé un problème délicat à traiter dans le cadre d'un environnement parallèle utilisant une mémoire répartie. Ce problème concerne la subdivision adaptative des objets, aux endroits où ils présentent une forte variation d'intensité lumineuse. La résolution de ce problème nécessite la connaissance du voisinage des différents points de calcul de la radiosité. Il ne peut pas être résolu dans le cas d'une distribution quelconque de la base de données; il est nécessaire de représenter les points de calcul voisins qui appartiennent au même objet sur le même processeur, de manière à pouvoir appliquer efficacement les subdivisions locales.

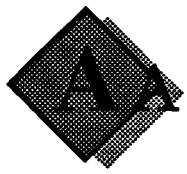
La dernière partie de notre travail a concerné l'étude de la parallélisation des méthodes projectives sur un calculateur massivement parallèle, à mémoire distribuée et fonctionnant sous un mode SIMD. Nos travaux se sont appuyés, dans un premier temps, sur l'étude des architectures massivement parallèles dédiées à l'affichage d'objets simples, pour lesquelles il est possible de distinguer deux classes d'approches : les approches pixels et les approches objets. Ces machines présentent en effet un fort taux de parallélisme, et de nombreuses similitudes avec le problème qui nous intéresse, à savoir l'élimination des parties cachées sur une grille d'échantillons.

Les deux solutions utilisées par ces architectures ont été envisagées dans le cadre d'une implantation sur la Maspar. L'approche objet a été écartée, d'une part du fait du très faible rendement qu'elle autorise, et d'autre part de la difficulté d'implanter un mécanisme de décision efficace pour l'élimination des parties cachées. Nous avons implanté une solution basée sur un parallélisme proxel, qui permet un rendement plus élevé en utilisant la notion de rectangles englobants. Les résultats obtenus, s'ils montrent une certaine accélération par rapport à une approche séquentielle, restent cependant modestes, du fait du rendement encore faible de cette approche, et des problèmes de communication qui apparaissent dans l'environnement massivement parallèle.

Nous avons ensuite introduit une méthode hybride, qui utilise les deux types de parallélisme précédents, en exploitant leurs avantages respectifs. Cette méthode est bien adaptée au fonctionnement de type SIMD, et présente un rendement beaucoup plus élevé que dans les approches précédentes. Elle nécessite cependant encore de nombreuses communications, qui la pénalisent quelque peu et ne lui permettent pas d'offrir le maximum de ses performances.

L'utilisation des méthodes utilisant le lancer de rayons pour le calcul des facteurs de forme n'a pas été envisagée dans le cadre du travail concernant la parallélisation massive. Ce sujet fait cependant l'objet de nos préoccupations actuelles, non pas uniquement dans le cadre strict de la radiosité, mais dans le but d'intégrer un modèle de réflexion étendu dans une approche massivement parallèle.

ANNEXE

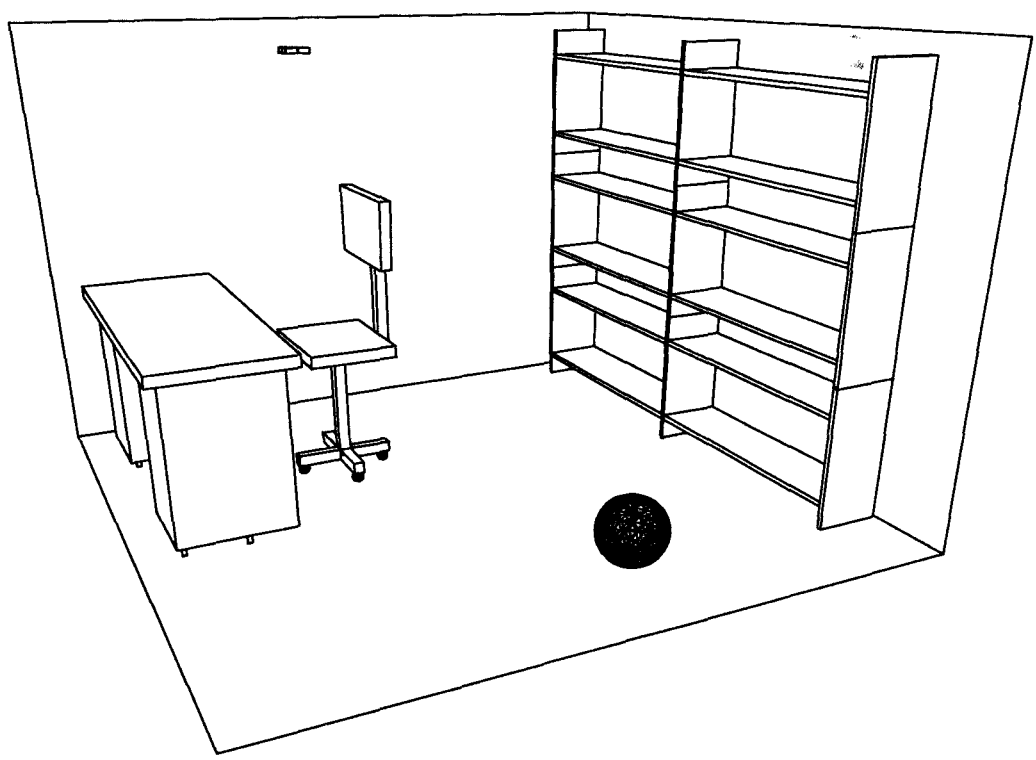


Bases de données utilisées

Nous décrivons ici succintement les différentes scènes que nous avons utilisées pour nos simulations, en précisant en particulier leur taille.

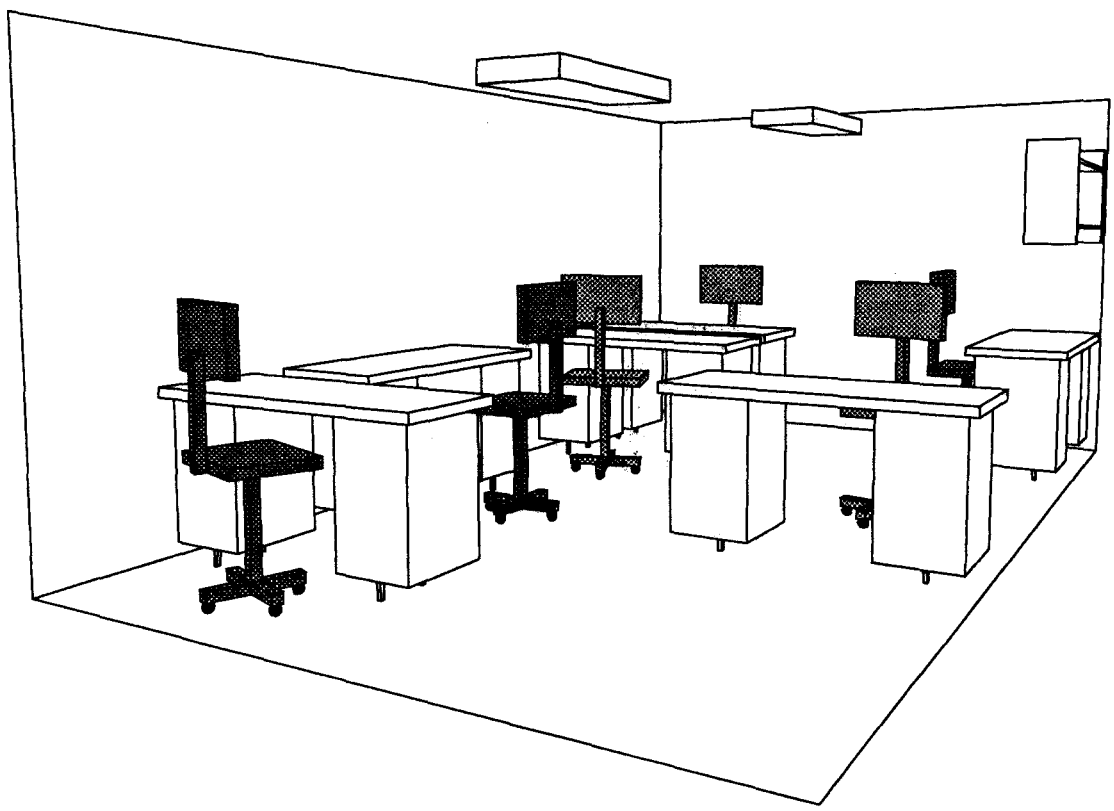
A.1 Scène "Bureau1"

La scène "bureau1", schématisée ci-dessous, est la plus petite des trois scènes que nous avons utilisées. Elle comprend 7994 facettes et 10340 sommets. Ces quantités sont regroupées en 268 surfaces polygonales et une surface sphèrique, modélisant 63 objets différents. Elle ne comprend qu'une seule source primaire.



A.2 Scène "Bureau 2"

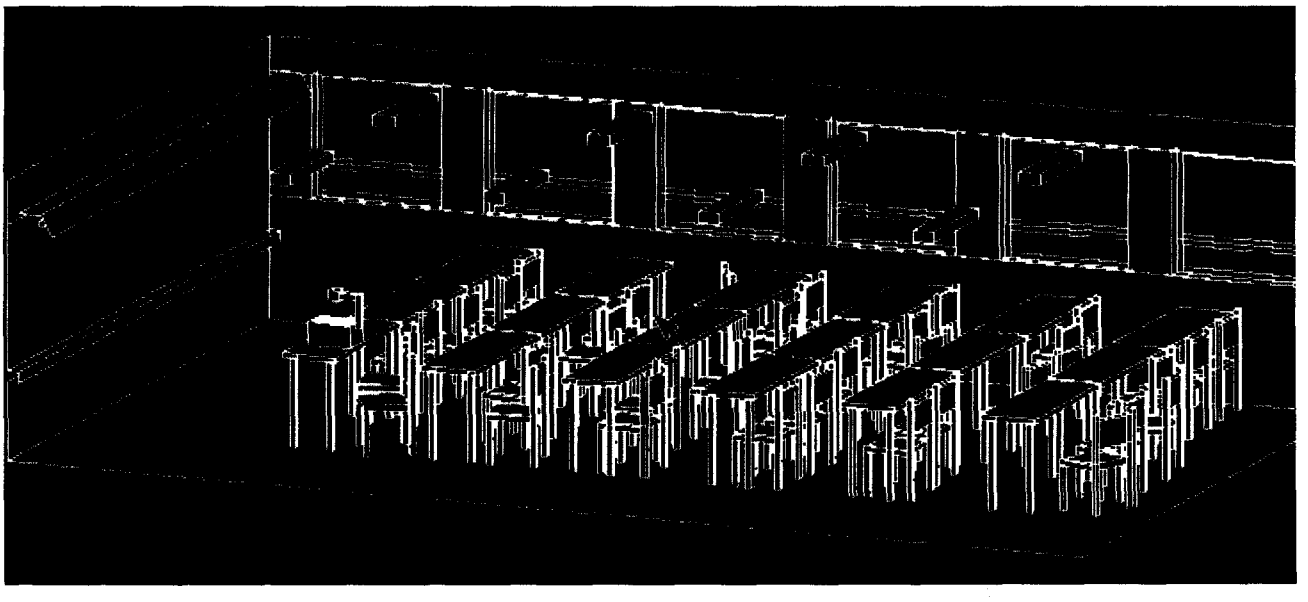
Cette seconde scène comprend deux grandes sources primaires, et 152 objets. Ces objets sont constitués de 606 surfaces polygonales et sphèriques, elles-mêmes découpées en 17279 facettes. Elle comprend d'autre part 22827 sommets.



Pour des raisons de visibilité, les "chaises-à-roulettes" présentes dans la scène ont été volontairement grisées.

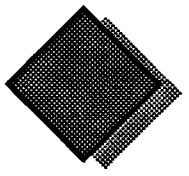
A.3 Scène "Classe"

Cette troisème scène est beaucoup plus complexe que les deux précédentes.



Elle comprend, globalement, 26 sources d'éclairage primaire, réparties entre les fenêtres, les néons (tableau, plafond) et le rétroprojecteur. Elle contient 581 objets, découpés en 2197 surfaces polygonales. Ces surfaces sont elles-mêmes découpées en 38532 facettes et 52742 sommets.

Bibliographie



[Airey 89] J. Airey, M. Ouh-young
"Two adaptative techniques let progressive refinement outperform the traditionnal radiosity algorithm"
University of North Caroline at Chapel Hill, Technical Report TR 89-020 August 1989

[Airey 90] J.R. Airey, J.H. Rohlf, F.P. Brooks
"Towards image realism with inteactive update rates in complex virtual buildings environments"
Computer Graphics Vol 24 No 2, pp 41-50, March 1990

[Akele 88] K. Akeley, T. Jermoluk
"High-performance polygon rendering"
Computer Graphics, Vol 22 No 4, pp 239-246, August 1988

[Akele 89] K. Akeley
"The Silicon Graphics 4D/240 GTX Superworkstation"
IEEE Computer Graphics and Applications, Vol 9 No4 pp 71-83, July 1989

[Amana 84] J. Amanatides
"Ray tracing with cones"
Computer Graphics Vol 18 No 3, SIGGRAPH'84, pp 129-135, 1984

[Appel 68] A. Appel
"Some techniques for shading machine renderings of solids"
AFIPS Spring Joint Computer Conference, pp 37-45, 1968

[Arvo 87] J. ARvo, D. Kirk
"Fast ray tracing by ray classification"
Computer Graphics Vol 21 No 4, SIGGRAPH'87, pp 55-64, 1987

[Arvo 91] J. Arvo et al
"Graphics Gems"
Academic Press, Boston, 1991

[Badou 90] D. Badouel
"Schémas d'exécution pour les machines parallèles à mémoire distribuée. Une étude de cas: le lancer de rayons"
Thèse de doctorat, Université de Rennes I, Octobre 1990

[Badou 91] D. Badouel, K. Bouatouch, Z. Lahjomri, T. Priol
"KOAN: a versatile tool for parallelizing realistic rendering algorithms"
Rapport de Recherche 1505, INRIA, 1991

[Baum 89] D.R. Baum, H.E. Rushmeier , J.M. Winget
"Improving radiosity solutions through the use of analytically determined form-factors"
Computer Graphics Vol 23 No 3 pp 325-334 July 1989

[Baum 90] D.R. Baum, J.M. Winget
" Real time radiosity through parallel processing and hardware acceleration"
Computer Graphics Vol 24 No 2 pp 67-75 March 90

[Baum 91] D.R. Baum, S. Mann, K.P. Smith, J.M. Winget
"Making radiosity usable: Automatic preprocessing and meshing techniques for the generation of accurate radiosity solutions"
Computer Graphics Vol. 25 No 4, pp 51-60, July 1991

[Bian 92] B. Bian, N. Wittels, D.S. Fussell
"Non-uniform patch luminance for global illumination"
Graphics Interface'92, pp 310-318, May 1992

[Blank 90] T. Blank
"The Maspar MP-1 architecture"
35th IEEE Computer Society International Cnference, San Francisco, 1990

[Boian 89] L.B. Boianov
"Higher speed communications in a multitransputer system"
Manchester University Thesis June 1989

[Campb 90] A.T. Campbell, D.S. Fussel
"Adaptative mesh generation for gloabl diffuse illumination"
Computer Graphics Vol 24 No 4, pp 155-163, August 1990

[Catmu 75] Catmull.E
Computer Display of Curved Surfaces
Proc. of IEEE Conf. Comput. Graphics Pattern Recognition Data Struct., May 1975, p. 11

[Chail 91] C. Chaillou
"Etude d'un processeur de visualisation d'images de synthèse en temps réel exploitant un parallélisme massif objet: le projet I.M.O.G.E.N.E."
Thèse de doctorat, Université de Lille, Janvier 91

[Chail 92] C. Chaillou
"Architectures des systèmes pour la synthèse d'images"
DUNOD informatique, Paris, Mai 1992

[Chalm 89] A.G. Chalmers, D.J. Paddon
"Implementing a radiosity method using a parallel adaptative system"
1rst International Conference on Applications of Transputers, 1989

[Chalm 91] A.G. Chalmers, D.J. Paddon
"Parallel processing of progressive refinement radiosity methods"
proceedings of 2nd Eurographics Workshop on Rendering, Barcelona May 1991

[Chalm 90] A.G. Chalmers, D.J. Paddon
"Parallel Radiosity Methods"
Transputer Research and Applications 4, 1990, IOS Press, pp 183-193

[Chen 89] S.E. Chen
"A progressive radiosity method and its implementation in a distributed processing environment"
Cornell University Master Science Thesis, New York, Januar 1989

[Cheun 89] H.C. Cheung
"A high performance graphics system for transputer arrays"
Manchester University Thesis, Januar 1989

[Clark 82] J.H. Clark
"The geometry engine: a VLSI geometry system for graphics"
Computer Graphics, Vol 16 No 3, July 1982

[Cohen 85] M.F. Cohen, D.P. Greenberg
"The hemicube : a radiosity solution for complex environments"
SIGGRAPH'85 Vol 19 No 3 pp 31-40, July 1985

[Cohen 86] M.F. Cohen, D.P. Greenberg, D.S. Immel, P.J. Brock
"An efficient radiosity approach for realistic image synthesis"
IEEE Computer Graphic and Applications pp 26-35, March 1986

[Cohen 88] M.F. Cohen, S.E. Chen, J.R. Wallace, D.P. Greenberg
"A progressive refinement approach to fast radiosity image generation"
SIGGRAPH'88 Vol 22 No 4 pp 75-84, August 1988

[Cook 84] R.L. Cook, T. Porter, L. Carpenter
"Distibuted ray tracing"
Computer Graphics Vol 18 (SIGGRAPH'84), pp 137-147, Juillet 1984

[Dec 92] Digital Equipment Corporation
"DECmpp Architecture Specification"
DEC, Maynard, 1992

[Degra 93] S. Degrande
"Définition d'une machine cellulaire pour la mise en oeuvre d'un lancer de rayons massivement parallèle"
Thèse de doctorat, Université de Lille, Novembre 1993

[Deeri 88] M. deering, S. Winner, B. Schediwy
"The triangle processor and normal vector shader : a VLSI system for high performance graphics:
ACM Computer Graphics, Vol 22 No 4, pp 21-30, Août 1988

[Farell 76] R. Farell
"Determination of configuration factors of irregular shapes"
Journal of Heat Transfer Vol 98 No2 pp 311-313 May 1976

[Feda 91] M. Feda, W. Purgathofer
"Progressive refinement radiosity on a transputer network"
proceedings of 2nd Eurographics Workshop on Rendering, Barcelona May 1991

[Flynn 72] M.J. Flynn
"Some computer organizations and their effectivness"
IEEE transaction on computer, Vol 21 No 9, pp 948-960, Septembre 1972

[Fuchs 89] Fuchs H., Poulton J., et al
"Pixel-planes 5: A Heterogeneous Multiprocessor Graphics System Using Processor-Enhanced Memories"
SIGGRAPH'89 Vol 23 No 3 pp79-88, July 1989

[Fujim 86] A. Fujimoto, T. Tanaka, K. Iwata
"ARTS: Accelerated Ray Tracing System"
IEEE Computer Graphics and Applications Vol 6 No 4, pp 16-26, 1986

[Ghaza 88] D. Ghazanfarpour
"An alias-free adaptative subdivision beam tracing"
Rapport Interne R91/02, Université Louis Pasteur, Strasbourg, Juin 91

[Ghori 93] H. Ghoris
"Utilisation du tracé de cônes pour le calcul des échanges lumineux en radiosité"
Mémoire de DEA, LIFL, Juillet 1993

[Gebha 61] B. Gebhart
"Heat transfer"
Mc Graw Hill, New York 1961

[Glass 84] A.S. Glassner
"Space subdivision for ray tracing"
IEEE Computer Graphics and Applications Vol 4 N0 10, pp 15-22, 1984

[Goldf 89] J. Goldfeather
"Progressive radiosity using hemispheres"
University of North Carolina at Chapel Hill, Technical report TR 89-002 Februar 1989

[Golds 71] R.A. Goldstein, R. Nagel
"3-D visual simulation"
Simulation, pp 25-31, Janvier 1971

[Goral 84] C.M. Goral, K.E. Torrance, D.P. Greenberg, B. Battaile
"Modeling the interaction of light between diffuse surfaces"
SIGGRAPH'84 Vol 18 No3 pp 213-222, July 1984

[Goura 71] H. Gouraud
"Continuous shading of curved surfaces"
IEEE transaction on computers, Vol c-20 No 6 pp 623-629, Juin 71

[Guitt 91] P. Guitton, J. Roman, C. Schlick
"Two parallel approaches for a progressive radiosity"
2nd Eurographics Workshop on Rendering, Barcelona 1991

[Haine 91] E. Haines, J. Wallace
"Shaft culling for efficient ray-traced radiosity"
2nd Eurographics Workshop on rendering, May 1991

[Heckb 84] P. Heckbert, P. Hanrahan
"Beam tracing polygonal objects"
Computer Graphics, Vol 18 No 3, pp 119-127, Julliet 1984

[Heckb 92] P.S. Heckbert
"Discontinuity meshing for radiosity"
Third Eurographics Workshop on Rendering, pp 203-216, Bristol, Mai 1992

[Helios 89]
The helios Operating System
University Press, Cambridge 1989

[Hodgm 74] G.W. Hodgman, I.E. Sutherland
"Reentrant polygon clipping"
Communications ACM Vol 17 No1 1974

[Illiev 87] M.S. Illiev, A.E. Knowles
"Runtime program debugger for the M.U. T-Rack"
ParSiFal Internal Document PSF/MU/WP5/87/9

[Jessel 92] J.P. Jessel, R. Caubet
"The implementation of an extended radiosity on the VOXAR machine"
Transputers'92, pp 86-102, Arc-et-Senans, 20-22 May 1992

[Kajyi 86] J.T. Kajiya
"The rendering equation"
Computer Graphics Vol 20 No 4, pp 143-150, August 86

[Kapla 85] M.R. Kaplan
"Space tracing: a constant time ray tracer"
State of the art in image synthesis, SIGGRAPH'85 course notes, 1985

[Kay 79] D.S. Kay, D.P. Greenberg
"Transparency for computer synthesized images"
Computer Garphics Vol 13 (SIGGRAPH'79), pp 158-164, 1979

[Knowl 87] A.E. Knowles
"ParSiFal : A parallel simulation facility based on transputers"
ParSiFal Internal Document PSF/MU/WP1/87/3

[Langu 91] E. Languénou, K. Bouatouch, P. Tellier
"Une nouvelle approche réaliste de simulation d'éclairage dans un environnement diffus"
Rapport de recherche No 1553, INRIA, Rennes, 1991

[Lefev 92] V. Lefevere, C. Chaillou. M. Meriaux
"Low cost hardware for real time Phong shading"
Graphics Interface'92 Workshop on local illumination, 1992

[Lepre 89] E. Leprêtre
"Algorithmique parallèle et architectures spécialisées pour la synthèse d'images"
Thèse de doctorat, Université de Lille, Juin 89

[Lepre 91] E. Lepretre, C. Renaud, M. Meriaux
"La radiosité sur transputers"
La Lettre du Transputer No 12, pp 49-61 December 1991

[Letel 93] L. Letellier, J. Rebillat, H. essafi, J.L. Basille, D. Juvin
"Un accelérateur graphique SIMD à changement de contexte"
5èmes rencontres sur le parallélisme, pp-91-94, Brest Mai 1993

[Lisch 92] D. Lischinski, F. Tampieri, D.P. Greenberg
"Discontinuity meshong for accurate radiosity"
IEEE Computer Graphics and Applications, Vol 12, No 3, pp 25-39, Novembre 1992

[Menar 92] D. Menard, K. Bouatouch, T. Priol
"Un algorithme de radiosité parallèle utilisant une mémoire virtuelle partagée"
Actes de GROPLAN'92, pp 177-185, Nantes Novembre 1992

[Meyer 90] U. Meyer
"Hemicube ray-tracing: a method for generating soft shadow"
Eurographics'90, Elsevier Science Publisher, pp 365-376, 1990

[Molna 92] S.E. Molnar, J.G. Eyles, J.W. Poulton
"Pixel-Flow: high speed rendering using image composition"
Computer Graphics Vol 26 No 3 (SIGGRAPH'92), pp 231-240, Juillet 1992

[Nicko 90] J.R. Nickolls
"The design of the Maspar MP-1: a cost effectuve massively parallel computer"
32th IEEE Computer Society International Conference, San Francisco, 1990

[Nishi 85] T. Nishita, E. Nakamae
"Continuous tone representation of three-dimensional objects takong account of shadows and interreflection"
Computer Graphics Vol 19 No3 (SIGGRAPH'85), pp 23-30, Juillet 1985

[Nusse 28] W. Nusselt
"Graphische bestimmung des winkelverhaltnisses bei der warmestrahlung"
VDI Z, 72:673, 1928

[Pauli 92] M. Paulin, C. Metge, J.P. Jessel, Y. Duthen, R. Caubet
"Etude du comportement sur une architecture parallèle et un découpage 3D d'une radiosité progressive"
Actes de GROPLAN'92, pp 187-194, Nantes Novembre 1992

[Phong 75] B.T. Phong
"Illumination for computer generated pictures"
Communications ACM Vol 18 No 6 June 75

[Picot 92] K. P. Picott
"Extensions of the Linear and Area Lighting Models"
IEEE Computer Graphics & Applications, Mars 1992, pp 31-38

[Price 90] M. Price, G. Truman
"Radiosity in parallel"
Proceedings of the 1rst conference on applications of transputers, pp 40-47 IOS Press

[Purga 90] W. Purgathofer, M. Zeiller
"Fast Radiosity by Parallelization"
Eurographics Workshop on photosimulation, realism and physics in Computer Graphics, pp 173-183, Rennes June 1990

[Recke 90] R.J. Recker, D.W. George, D.P. Greenberg
"Acceleration techniques for progressive refinement radiosity"
Computer Graphics Vol 24 No 2 pp 59-66 March 90

[Rubin 80] S. Rubin, T Whitted
"A three dimensional representation for fast rendering"
Computer Graphics Vol 14 No 3 , SIGGRAPH'80, pp 110-116, 1980

[Sanso 91] J.P. Sansonnet, C. Germain-Renaud
"Les ordinateurs massivement parallèles"
Editions Armand Colin, Paris, Mars 1991

[Schne 88] B.O. Schneider
"A processor for an object-oriented rendering system"
Computer graphics forum, No 7, pp 301-310, 1988

[Shers 93] Shersson
"?????????????"
Séminaire LIFL

[SillA 89] F. Sillion
"Simulation de l'éclairage pour la synthèse d'images : réalisme et interactivité"
PhD thesis , Paris-Sud University, June 89

[SillB 89] F. Sillion, C. Puech
"A general two-pass method integrating specular and diffuse reflection"
SIGGRAPH'89 Vol 23 no 3 pp 335-344, August 89

[Singh 92] D.B. Singh, S.G. Abraham, F.H. Westervelt
"Computing radiosity solution on a high performance workstation LAN"
1rst High performance distributed computing, pp 248-257, N.Y. Septembre 1992

[Spenc 90] S.N. Spencer,
"The hemisphere radiosity method: a tale of two algorithm"
Eurographics Workshop on Photosimulation, Realism and Physics in Computer Graphics, pp 127-135, June 1990

[Telli 91] P. Tellier, E. Maisel, K. Bouatouch, E. Languénou
"Using coherence to accelerate radiosity"
Publication Interne No 616, IRISA, Rennes, 1991

[Thomi 93] P. Thomin
"Algorithmes paralléles pour la synthése d'images par radiosité sur calculateur a mémoire distribuée"
PhD Thesis, Valenciennes University, February 93

[Varsh 92] A. Varshney, J.F. Prins
"An environment-projection approach to radiosity for mesh-connected computers"
3rd Eurographics Workshop on Rendering, pp 271-281, May 92

[Vilap 92] J. Vilaplana
"Parallel radiosity solutions based on partial results messages"
3rd Eurographics Workshop on Rendering, pp 259-270, May 92

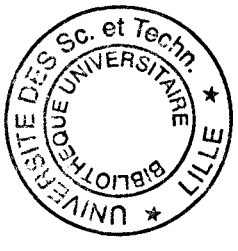
[Walla 89] J.R. Wallace, K.A. Elmquist, E.A Haines
"A ray tracing algorithm for progressive radiosity"
Computer Graphics, Vol 23 no 3 pp 315-324 July 1989

[Walto 87] G.N. Walton
"Algorithms for calculating radiation view factors between pane convex polygons with obstructions"
Fundamental and applications of radiation heat transfer, HTD Vol 72, pp 45-52, 1987

[Ward 88] G.J. Ward, F.M. Rubinstein, R.D. Clear
"A ray tracing solution for diffuse interreflection"
Computer Graphics Vol 22 No 4 (SIGGRAPH'88), pp 85-92, Août 1988

[Warno 69] J.E. Warnock
"A hidden surface algorithm for computer generated halftone pictures"
Research Report 4-15, University of UTAH, June 69

[Whitt 80] T. Whitted
"An improved illumination model for shaded display"
Communications ACM Vol 23 No 6 pp 343-349, 1980


UNIVERSITE DES Sc. et Techn.
BIBLIOTHEQUE UNIVERSITAIRE
LILLE