USTL
FLANDRES ARTOIS

LIFL
U.A. 369 du C.N.R.S.

LABORATOIRE D'INFORMATIQUE FONDAMENTALE DE LILLE

N° Ordre: 864

Année: 1992

THESE

présentée à

l'Université des Sciences et Techniques de LILLE FLANDRES ARTOIS

pour obtenir le titre de

DOCTEUR en INFORMATIQUE

par

Bruno VIDAL

VERS UN LANCER DE RAYONS DISCRET

Soutenance le 5 Février 1992 devant la commission d'examen

Membres du Jury:

Président:	B. PEROCHE
Rapporteur:	B. PEROCHE
Rapporteur:	J. FRANÇON
Directeur de thèse:	M. MERIAUX
Examinateur:	A. ATAMENIA
Examinateur:	M. LATTEUX
Examinateur:	E. LEPRETRE

UNIVERSITE DES SCIENCES
ET TECHNIQUES DE LILLE
FLANDRES ARTOIS

DOYENS HONORAIRES DE L'ANCIENNE FACULTE DES SCIENCES

M.H. LEFEBVRE, M. PARREAU.

PROFESSEURS HONORAIRES DES ANCIENNES FACULTES DE DROIT ET SCIENCES ECONOMIQUES, DES SCIENCES ET DES LETTRES

MM. ARNOULT, BONTE, BROCHARD, CHAPPELON, CHAUDRON, CORDONNIER, DECUYPER, DEHEUVELS, DEHORS, DION, FAUVEL, FLEURY, GERMAIN, GLACET, GONTIER, KOURGANOFF, LAMOTTE, LASSERRE, LELONG, LHOMME, LIEBAERT, MARTINOT-LAGARDE, MAZET, MICHEL, PEREZ, ROIG, ROSEAU, ROUELLE, SCHILTZ, SAVARD, ZAMANSKI, Mes BEAUJEU, LELONG.

PROFESSEUR EMERITE

M. A. LEBRUN

ANCIENS PRESIDENTS DE L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE

MM. M. PAREAU, J. LOMBARD, M. MIGEON, J. CORTOIS.

PRESIDENT DE L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE FLANDRES ARTOIS

M. A. DUBRULLE.

PROFESSEURS - CLASSE EXCEPTIONNELLE

M. CONSTANT Eugène	Electronique
M. FOURET René	Physique du solide
M. GABILLARD Robert	Electronique
M. MONTREUIL Jean	Biochimie
M. PARREAU Michel	Analyse
M. TRIDOT Gabriel	Chimie Appliquée

PROFESSEURS - 1ère CLASSE

M. BACCHUS Pierre	Astronomie
M. BIAYS Pierre	Géographie
M. BILLARD Jean	Physique du Solide
M. BOILLY Bénoni	Biologie
M. BONNELLE Jean-Pierre	Chimie-Physique
M. BOSCQ Denis	Probabilitéss
M. BOUGHON Pierre	Algèbre
M. BOURIQUET Robert	Biologie Végétale
M. BREZINSKI Claude	Analyse Numérique

M. BRIDOUX Michel	Chimie-Physique
M. CELET Paul	Géologie Générale
M. CHAMLEY Hervé	Géotechnique
M. COEURE Gérard	Analyse
M. CORDONNIER Vincent	Informatique
M. DAUCHET Max	Informatique
M. DEBOURSE Jean-Pierre	Gestion des Entreprises
M. DHAINAUT André	Biologie Animale
M. DOUKHAN Jean-Claude	Physique du Solide
M. DYMENT Arthur	Mécanique
M. ESCAIG Bertrand	Physique du Solide
M. FAURE Robert	Mécanique
M. FOCT Jacques	Métallurgie
M. FRONTIER Serge	Ecologie Numérique
M. GRANELLE Jean-Jacques	Sciences Economiques
M. GRUSON Laurent	Algèbre
M. GUILLAUME Jean	Microbiologie
M. HECTOR Joseph	Géométrie
M. LABLACHE-COMBIER Alain	Chimie Organique
M. LACOSTE Louis	Biologie Végétale
M. LAVEINE Jean-Pierre	Paléontologie
M. LEHMANN Daniel	Géométrie
Mme LENOBLE Jacqueline	Physique Atomique et Moléculaire
M. LEROY Jean-Marie	Spectrochimie
M. LHOMME Jean	Chimie Organique Biologique
M. LOMBARD Jacques	Sociologie
M. LOUCHEUX Claude	Chimie Physique
M. LUCQUIN Michel	Chimie Physique
M. MACKE Bruno	Physique Moléculaire et Rayonnements Atmosph.
M. MIGEON Michel	E.U.D.I.L.
M. PAQUET Jacques	Géologie Générale
M. PETIT Francis	Chimie Organique
M. POUZET Pierre	Modélisation - calcul Scientifique
M. PROUVOST Jean	Minéralogie
M. RACZY Ladislas	Electronique
M. SALMER Georges	Electronique
M. SCHAMPS Joel	Spectroscopie Moléculaire
M. SEGUIER Guy	Electrotechnique
M. SIMON Michel	Sociologie
Melle SPIK Geneviève	Biochimie
M. STANKIEWICZ François	Sciences Economiques
M. TILLIEU Jacques	Physique Théorique
M. TOULOTTE Jean-Marc	Automatique
M. VIDAL Pierre	Automatique
M. ZEYTOUNIAN Radyadour	Mécanique

PROFESSEURS - 2ème CLASSE

M. ALLAMANDO Etienne	Composants Electroniques
M. ANDRIES Jean-Claude	Biologie des organismes
M. ANTOINE Philippe	Analyse
M. BART André	Biologie animale
M. BASSERY Louis	Génie des Procédés et Réactions Chimiques

Mme BATTIAU Yvonne	Géographie
M. BEGUIN Paul	Mécanique
M. BELLET Jean	Physique Atomique et Moléculaire
M. BERTRAND Hugues	Sciences Economiques et Sociales
M. BERZIN Robert	Analyse
M.BKOUCHE Rudolphe	Algèbre
M. BODARD Marcel	Biologie Végétale
M. BOIS Pierre	Mécanique
M. BOISSIER Daniel	Génie Civil
M. BOIVIN Jean-Claude	Spectroscopie
M. BOUQUELET Stéphane	Biologie Appliquée aux enzymes
M. BOUQUIN Henri	Gestion
M. BRASSELET Jean-Paul	Géométrie et Topologie
M. BRUYELLE Pierre	Géographie
M. CAPURON Alfred	Biologie Animale
M. CATTEAU Jean-Pierre	Chimie Organique
M. CAYATTE Jean-Louis	Sciences Economiques
M. CHAPOTON Alain	Electronique
M. CHARET Pierre	Biochimie Structurale
M. CHIVE Maurice	Composants Electroniques Optiques
M. COMYN Gérard	Informatique Théorique
M. COQUERY Jean-Marie	Psychophysiologie
M. CORIAT Benjamin	Sciences Economiques et Sociales
Mme CORSIN Paule	Paléontologie
M. CORTOIS Jean	Physique Nucléaire et Corpusculaire
M. COUTURIER Daniel	Chimie Organique
M. CRAMPON Norbert	Tectolique Géodynamique
M. CROSNIER Yves	Electronique
M. CURGY Jean-Jacques	Biologie
Melle DACHARRY Monique	Géographie
M. DEBRABANT Pierre	Géologie Appliquée
M. DEGAUQUE Pierre	Electronique
M. DEJAEGER Roger	Electrochimie et Cinétique
M. DELAHAYE Jean-Paul	Informatique
M. DELORME Pierre	Physioliogie Animale
M. DELORME Robert	Sciences Economiques
M. DEMUNTER Paul	Sociologie
M. DENEL Jacques	Informatique
M. DE PARIS Jean Claude	Analyse
M. DEPREZ Gilbert	Physique du Solide - Cristallographie
M. DERIEUX Jean-Claude	Microbiologie
Melle DESSAUX Odile	Spectroscopie de la réactivité Chimique
M. DEVRAINNE Pierre	Chimie Minérale
Mme DHAINAUT Nicole	Biologie Animale
M. DHAMELINCOURT Paul	Chimie Physique
M. DORMARD Serge	Sciences Economiques
M. DUBOIS Henri	Spectroscopie Hertzienne
M. DUBRULLE Alain	Spectroscopie Hertzienne
M. DUBUS Jean-Paul	Spectrométrie des Solides
M. DUPONT Christophe	Vie de la firme (I.A.E.)
Mme EVRARD Micheline	Génie des procédés et réactions chimiques
M. FAKIR Sabah	Algèbre
M. FAUQUAMBERGUE Renaud	Composants électroniques

M. FONTAINE Hubert	Dynamique des cristaux
M. FOUQUART Yves	Optique atmosphérique
M. FOURNET Bernard	Biochimie Sturcturale
M. GAMBLIN André	Géographie urbaine, industrielle et démog.
M. GLORIEUX Pierre	Physique moléculaire et rayonnements Atmos.
M. GOBLOT Rémi	Algèbre
M. GOSSELIN Gabriel	Sociologie
M. GOUDMAND Pierre	Chimie Physique
M. GOURIEROUX Christian	Probabilités et Statistiques
M. GREGORY Pierre	I.A.E.
M. GREMY Jean-Paul	Sociologie
M. GREVET Patrice	Sciences Economiques
M. GRIMBLOT Jean	Chimie Organique
M. GUILBAULT Pierre	Physiologie animale
M. HENRY Jean-Pierre	Génie Mécanique
M. HERMAN Maurice	Physique spatiale
M. HOUDART René	Physique atomique
M. JACOB Gérard	Informatique
M. JACOB Pierre	Probabilités et Statistiques
M. Jean Raymond	Biologie des populations végétales
M. JOFFRE Patrick	Vie de la firme (I.A.E.)
M. JOURNEL Gérard	Spectroscopie hertzienne
M. KREMBEL Jean	Biochimie
M. LANGRAND Claude	Probabilités et statistiques
M. LATTEUX Michel	Informatique
Mme LECLERCQ Ginette	Catalyse
M. LEFEBVRE Jacques	Physique
M. LEFEBVRE Christian	Pétrologie
Melle LEGRAND Denise	Algèbre
Melle LEGRAND Solange	Algèbre
M. LEGRAND Pierre	Chimie
Mme LEHMANN Josiane	Analyse
M. LEMAIRE Jean	Spectroscopie hertzienne
M. LE MAROIS Henri	Vie de la firme (I.A.E.)
M. LEROY Yves	Composants électroniques
M. LESENNE Jacques	Systèmes électroniques
M. LHENAFF René	Géographie
M. LOCQUENEUX Robert	Physique théorique
M. LOSFELD Joseph	Informatique
M. LOUAGE Francis	Electronique
M. MAHIEU Jean-Marie	Optique-Physique atomique
M. MAIZIERES Christian	Automatique
M. MAURISSON Patrick	Sciences Economiques et Sociales
M. MESMACQUE Gérard	Génie Mécanique
M. MESSELYN Jean	Physique atomique et moléculaire
M. MONTEL Marc	Physique du solide
M. MORCELLET Michel	Chimie Organique
M. MORTREUX André	Chimie Organique
Mme MOUNIER Yvonne	Physiologie des structures contractiles
Mme MOUYART-TASSIN Annie Françoise	Informatique
M. NICOLE Jacques	Spectrochimie
M. NOTELET Françis	Systèmes électroniques
M. PARSY Fernand	Mécanique

M. PECQUE Marcel	Chimie organique
M. PERROT Pierre	Chimie appliquée
M. STEEN Jean-Pierre	Informatique

Remerciements

Je tiens à remercier ici:

- Monsieur Bernard PEROCHE, Professeur à l'Ecole des Mines de Saint-Etienne qui me fait l'honneur de présider ce jury,

- Messieurs B. PEROCHE déjà cité et Jean FRANÇON, Professeur à l'Université Louis Pasteur de Strasbourg d'avoir accepté d'examiner ce travail,

- Monsieur Michel MERIAUX, Professeur à l'USTL (EUDIL) pour m'avoir proposé ce sujet,

ainsi que:

- Messieurs Abdelghani ATAMENIA, Maître de conférences à l'USTL (IUT.A),
  Michel LATTEUX, Professeur à l'USTL,
  Eric LEPRETRE, Maître de conférences à l'USTL (CUEEP).

Je remercie également mes collègues ainsi que tout le personnel du laboratoire d'informatique pour l'ambiance chaleureuse qu'ils entretiennent.

TABLE DES MATIERES

CHAPITRE 1

Introduction

TABLE DES MATIERES

CHAPITRE 1 : Introduction

La technique du lancer de rayons permet d'obtenir des images de synthèse réalistes, en prenant en compte les effets de reflets et transparence des objets. En contrepartie, le principal inconvénient de cette méthode est le temps nécessité par le calcul d'une image qui devient vite rédhibitoire pour la création de séquences animées. Le but de ce travail est d'étudier une architecture de machine dédiée au lancer de rayons. Nous allons exposer brièvement la méthode du lancer de rayons classique puis nous énumérerons les différents procédés permettant d'accélérer le calcul des images. C'est dans ce contexte que s'inscrit notre machine RC^2 dont la principale caractéristique est d'utiliser un espace discret pour représenter les objets et suivre les rayons lumineux. Nous aborderons ensuite les problèmes liés à la discrétisation de la scène. Nous introduirons tout d'abord des notions de topologie en espace discret, puis nous étudierons différentes méthodes permettant de tracer des segments dans l'espace en 3 dimensions. Cet algorithme est très important car il est à la base de tout algorithme rencontré en géométrie discrète. Nous montrerons ensuite comment discrétiser une scène découpée en facettes polygonales planes puis nous détaillerons l'implémentation de cette méthode sur notre architecture RC^2. Nous étudierons ensuite différentes stratégies pour calculer l'éclairage en chaque point de la scène et nous proposerons une nouvelle méthode qui consiste à émettre des cônes de lumière à partir de chacune des sources lumineuses de façon à modéliser d'une manière plus fidèle le comportement physique de la lumière. Nous exposerons ensuite l'implémentation de cet algorithme sur notre architecture cible et nous donnerons les résultats de simulations effectuées. Enfin dans une dernière partie, nous ferons une étude des différents problèmes introduits par la discrétisation à la fois des objets et des rayons, et plus particulièrement celui du calcul d'intersection rayon/objet puis nous proposerons certaines techniques pour y remédier.

1.1 La méthode du lancer de rayons

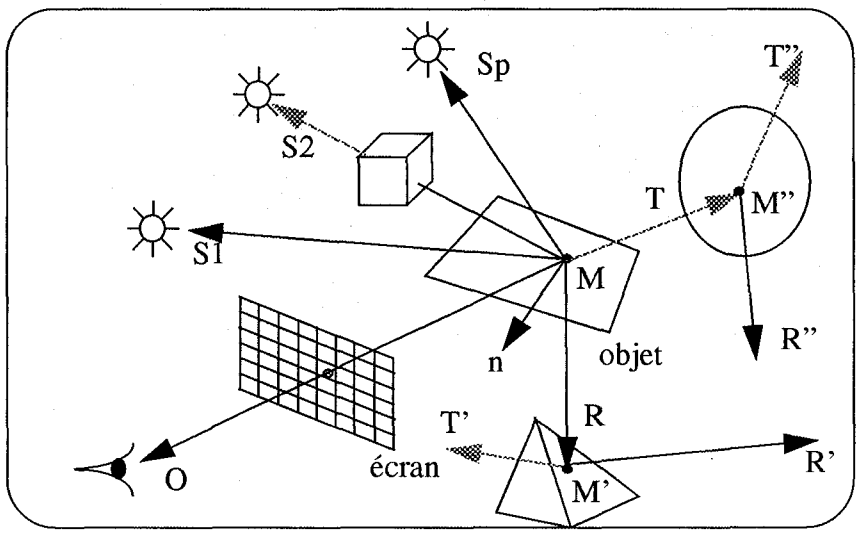


Figure 1

Le lancer de rayons est basé sur la simulation de l'optique géométrique. Il utilise le principe du retour inverse de la lumière. Un rayon est envoyé depuis l'oeil vers chaque pixel de l'écran puis, pour chaque rayon, est ensuite calculée l' intersection de celui-ci avec les objets constituant la base de données. Après avoir déterminé le point d'intersection le plus proche de l'observateur, l'algorithme en calcule l'éclairage. En ce point, sont envoyés un rayon vers chaque source lumineuse $\mathbf{S_i}$ (rayon d'ombre) et des rayons secondaires (un réfléchi **R** et un réfracté **T**). Le processus est réitéré pour les rayons secondaires qui coupent d'autres objets de la scène.
L'intensité de chaque point est déterminée par le modèle de Phong et Blinn.
Elle est d'abord composée d'un terme ambiant qui est le produit de l'intensité ambiante (I_a), identique en tout point de la scène, par un coefficient dépendant de l'objet (K_a):

$$\text{Terme ambiant} = K_a \cdot I_a$$

Pour chaque source lumineuse, un rayon est lancé à partir du point considéré vers celle-ci. Si le rayon rencontre un objet, le point est dans l'ombre portée de cet objet, sinon il est éclairé par la source, et à l'éclairage ambiant précédemment calculé sont ajoutées une composante diffuse Dj et une composante spéculaire Sj définies de la façon suivante:

Composante diffuse: $D_j = K_d \cdot (\vec{n} \cdot \vec{L}_j) \cdot E_j$

Composante spéculaire: $S_j = K_s \cdot \omega(\alpha) \cdot (\vec{r}_j \cdot \vec{V})^m \cdot E_j$

$\vec{n}$: vecteur unitaire normal à la surface

$\vec{L}_j$: vecteur unitaire dans la direction de la source S_j

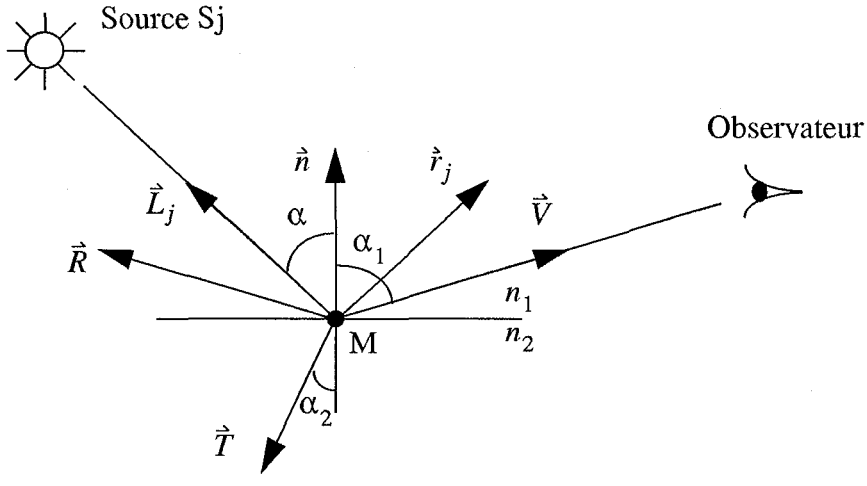
α : angle d'incidence de la lumière par rapport à $\vec{n}$

$\vec{V}$: Vecteur unitaire dans la direction de l'observateur

$\vec{r}_j$: vecteur unitaire symétrique de $\vec{L}_j$ par rapport à $\vec{n}$

E_j : Intensité de la source lumineuse S_j

ω : fonction de réflectance



La somme des intensités lumineuses reçues au point M est effectuée, puis
- Si la surface est réfléchissante, un rayon R réfléchi est lancé dans la direction symétrique de V par rapport à n. A l'éclairement de l'objet, est ajouté l'éclairement provenant de la direction de réflexion I_R. Cette intensité correspond à l'intensité du point M' coupé par R pondérée par un coefficient d'atténuation qui peut éventuellement dépendre de la distance.
- Si la surface est transparente, un rayon est lancé dans la direction de réfraction en utilisant la loi de Snell:

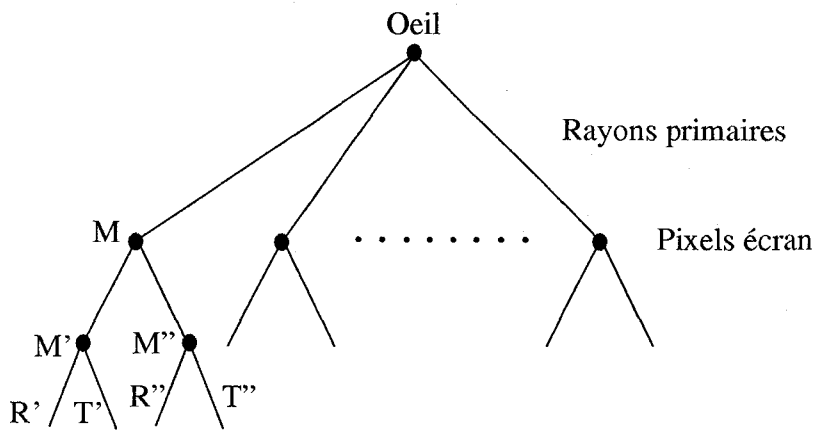
$$n_1 \cdot \sin\alpha_1 = n_2 \cdot \sin\alpha_2$$

n_1 et n_2 étant les indices respectifs des milieux séparés par la surface. A l'éclairement de l'objet, est ajouté l'éclairement provenant de la direction de transmission I_T. Cette intensité correspond à l'intensité du point M" de l'objet coupé par T pondérée par un coefficient d'atténuation.

Nous obtenons donc, pour le calcul de l'éclairement en M, la formule suivante:

$$I_M = K_a \cdot I_a + K_d \cdot \sum_{j=1}^{N_{src}} ((\vec{n} \cdot \vec{L}_j) \cdot E_j) + K_s \cdot \sum_{j=1}^{N_{src}} (\omega(\alpha) \cdot (\vec{r}_j \cdot \vec{V})^m \cdot E_j) + I_R + I_T$$

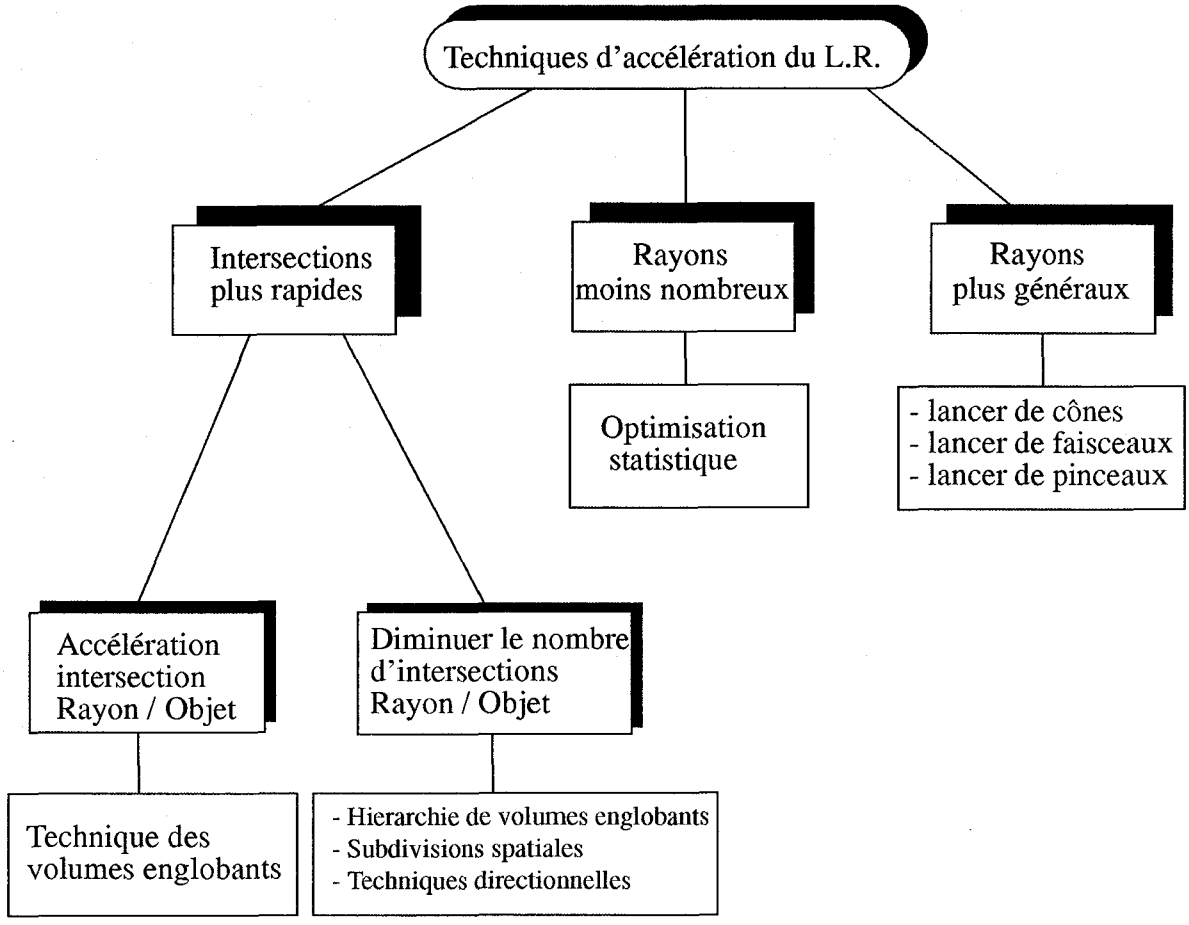
Les rayons primaires, réfléchis et réfractés peuvent être structurés en une arborescence dans laquelle le niveau supérieur correspond aux rayons issus de l'oeil de l'observateur, et pour chaque noeud, deux sous branches sont créées, l'une correspondant au rayon réfléchi en ce point, l'autre au rayon réfracté. La récursion se termine lorsqu'un rayon ne coupe aucun objet et donc sort de la scène ou bien lorsqu'une branche de l'arbre a atteint un niveau maximum fixé, ou bien également lorsque l'intensité lumineuse devient inférieure à un seuil déterminé à l'avance.



1.2 Techniques d'accélération du lancer de rayons

Un des inconvénients majeurs du lancer de rayons est la quantité importante de calculs nécessaires, ce qui prend énormément de temps. Se sont alors développées différents axes de recherche visant à accélérer la méthode du lancer de rayons classique.
L'algorithme du lancer de rayon est pratiquement basé sur une seule opération: le calcul d'intersection entre un rayon en 3D et un objet géométrique primitif (polygone, sphère, cylindre, surface paramétrique...). C'est ce calcul d'intersection qui consomme la majeure partie du temps d'exécution de l'algorithme. Selon les estimations effectuées par Whitted, 95% du temps est ainsi dépensé.

1•2•1 Classification des méthodes d'optimisation



Il existe trois types de stratégies différentes pour accélérer le L.R. [GLA 89].
Il est possible soit de réduire le coût moyen de l'intersection entre un rayon et l'environnement, soit de réduire le nombre total de rayons qui coupent l'environnement, soit de traiter ensemble plusieurs rayons individuels en les regroupant dans un concept plus général que celui de rayon.

1•2•2 Hiérarchies de volumes englobants

Cette méthode consiste à trouver un volume englobant chaque objet et permettant un calcul d'intersection plus simple que le calcul d'intersection avec l'objet lui-même. L'intersection réelle avec l'objet sera calculée uniquement si le rayon coupe le volume englobant. Ceci permet d'accélérer le cas des rayons qui ne coupent pas l'objet. Selon Whitted, les volumes englobants les plus simples sont des sphères.
Cependant, la complexité de l'algorithme reste en O(N), N étant le nombre d'objets. La méthode ne fait que diminuer la constante multiplicative.
Afin d'obtenir une complexité de l'ordre de Log(N), on crée une structure hiérarchique de volumes englobants. La méthode consiste à rassembler plusieurs volumes englobants à l'intérieur d'un volume englobant plus grand, et à construire ainsi une arborescence dans laquelle les feuilles correspondent aux objets eux-mêmes et la racine au volume contenant la scène entière. Si un rayon ne coupe pas un volume donné, il est alors inutile de tester l'intersection avec les volumes ou objets contenus à l'intérieur de celui-ci.

Rubin & Whitted ont employé des boîtes parallélépipédiques. Le calcul de l'intersection avec les volumes englobants est plus aisé car il est inutile de connaître le point d'intersection exacte, mais seulement le fait que le rayon coupe ou non ce volume.
A chaque fois qu'une intersection avec un objet est trouvée, la distance correspondante est mémorisée, ce qui permet d'éviter le calcul d'intersection avec d'autres volumes englobants si la distance minimale à tout objet de ce volume englobant est supérieur à la distance mémorisée. Weghorst & al. ont créé une méthode heuristique permettant l'optimisation des volumes englobants. Elle réalise un compromis entre la finesse du volume entourant l'objet et la rapidité du calcul d'intersection rayon / volume, car un volume trop grand entraîne le calcul d'intersection avec de nombreux rayons qui ne couperont aucun objet contenu dans le volume.
Il est également possible d'utiliser plusieurs volumes englobant pour entourer un objet et en faire la réunion ou l'intersection.

Pour construire la hiérarchie précédente, il faut décider quel groupe d'objets rassembler et quel type de volume englobant utiliser. Il est bien sûr inenvisageable de tester toutes les hiérarchies possibles, car leur nombre augmente exponentiellement avec le nombre d'objets de la scène. La stratégie adoptée consiste à regrouper entre eux les objets proches les uns des autres de façon à limiter la taille du volume englobant.
Goldsmith & Salmon ont développé une technique de construction d'arborescence de volumes englobants dans laquelle chaque objet est ajouté à celle-ci à l'endroit où le coût estimé est le plus faible. Le coût de fabrication de l'arbre étant faible par rapport à celui du L.R. lui-même, il est possible d'examiner plusieurs solutions et d'en retenir la meilleure.

Certaines méthodes tentent d'approcher l'enveloppe convexe d'un objet. Chaque rayon coupe un volume convexe au plus deux fois et donc le test d'intersection est plus simple. Kay & Kajiya utilisent des familles de plans parallèles pour entourer l'objet (slabs). L'intersection des régions définies par ces plans forme le volume englobant. 3 "slabs" différents suffisent en 3D mais plus le nombre est important meilleure est l'approximation de l'enveloppe convexe de l'objet. Pour déterminer l'intersection avec un tel volume, est calculée pour chaque "slab" l'intersection du rayon avec chacun des 2 plans le constituant.

1•2•3 Subdivision spatiale 3D

Cette technique consiste à découper l'espace en sous-volumes appelés voxels. Un voxel correspond à un cuboïde aligné sur les axes. Une étape de prétraitement construit une partition de l'espace en voxels. Chaque voxel comprend la liste des objets se trouvant entièrement ou partiellement à l'intérieur de celui-ci. L'intersection d'un rayon n'est testée qu'avec les objets contenus dans le voxel traversé par le rayon. Les voxels sont examinés dans l'ordre dans lequel le rayon les atteint. Dès qu'une intersection avec un objet a été déterminée, il est inutile de tester les voxels suivants, nous sommes sûrs d'avoir trouvé l'intersection la plus proche de l'origine du rayon.

1•2•3•1 subdivision spatiale non uniforme

L'espace est discrétisé en régions de taille différentes. La subdivision est plus fine dans les régions de l'espace où la densité d'objets est la plus importante. Cette partition peut être structurée en octree. L'octree est une structure hiérarchique qui contient en racine l'espace total et dans laquelle des volumes parallélépipédiques sont subdivisés récursivement en 8 octants jusqu'à ce que chaque voxel-feuille vérifie un critère de

simplicité fixé. A chaque voxel est affectée une liste des objets dont la surface coupe le voxel.

L'algorithme de suivi de rayon est le suivant:

- Déterminer le voxel dans lequel est le rayon
- Calculer l'intersection avec tous les objets contenus dans celui-ci
- Si pas d'intersection, déterminer le point d'entrée dans le voxel suivant.
- Réitérer le processus jusqu'à ce que le rayon sorte de l'environnement ou coupe un objet.

1•2•3•2 Subdivision spatiale uniforme

Fujimoto & al. découpent l'espace en voxels de taille identique qu'ils gèrent dans une structure appelée SEADS (Spatially Enumerated Auxiliary Data Structure) [FUJ 86].
L'avantage de cette subdivision est son indépendance par rapport à l'environnement. Le calcul des voxels traversés par un rayon est plus simple et peut être effectué de manière incrémentale. Pour cela a été élaboré un algorithme appelé 3DDDA (3 Dimensional Digital Difference Analyzer) analogue à celui pour la discrétisation de ligne 3D, à la seule différence que ce dernier calcule la liste des voxels les plus proches de la ligne réelle alors que l'algorithme 3DDDA donne la liste des voxels effectivement coupés par le rayon. L'inconvénient de cette méthode est d'une part que la traversée des régions vides par les rayons est plus lente et d'autre part, cette technique requiert la mémorisation d'une importante quantité de données (liste des objets par voxels).

L'inconvénient des techniques de subdivision spatiales est le calcul répété de l'intersection d'un rayon avec le même objet si celui-ci s'étend sur plusieurs voxels. Arnaldi et Al. ont mis au point un système de boîtes aux lettres sur les objets indiquant, pour un rayon donné, si le calcul d'intersection a déjà été effectué.

1•2•4 Techniques directionnelles

Cette technique prend en compte la direction des rayons mais nécessite une très grande capacité de stockage. L'espace est découpé en pyramides de base sur l'une des faces du cube contenant la scène et de sommet un point particulier de l'espace. Le découpage peut être régulier ou non.
Le Light-Buffer proposé par Haines & Greenberg est une technique qui accélère le calcul des ombres, pour des sources ponctuelles. Cet algorithme exploite le découpage en pyramides pour déterminer si un point est à l'ombre. La recherche d'un éventuel objet occulteur est restreinte à un petit ensemble d'objets en tenant compte de la direction allant de la source au point considéré. Pour chaque source lumineuse est créé un cube différent, découpé en pyramides ayant pour sommet cette source. Pour chaque direction, est établie une liste d'objets correspondants triée par profondeur croissante à partir de la position de la source. Pour déterminer si un point appartenant à une surface est à l'ombre pour une source donnée, l'orientation de la surface par rapport à la source lumineuse est d'abord examinée. Si elle est dirigée dans la direction opposée, le point est alors à l'ombre, sinon la liste des objets susceptibles de masquer la source est recherchée. L'intersection avec ces objets est recherchée par profondeur croissante à partir de la source jusqu'à ce qu' un objet soit détecté entre la source et le point considéré, qui masque alors la source, ou que soit atteint un objet dont la profondeur est supérieure à celle du point en question, auquel cas le point est éclairé.

1•2•5 Rayons moins nombreux

Cette technique permet de réduire le nombre de rayons dont il faut calculer l'intersection avec l'environnement. Ceci prend en compte à la fois les rayons primaires, réfléchis, réfractés et les rayons d'ombre.

1•2•5•1 Contrôle adaptatif de la profondeur de l'arbre

Au lieu d'arrêter la réémission de rayons réfléchis et réfractés lorsqu'est rencontrée une surface non réfléchissante ou opaque, ou bien lorsqu'est atteinte une profondeur déterminée dans l'arbre des rayons, le critère utilisé dans la méthode introduite par Hall et Al. est le suivant. Si la contribution d'un pixel est telle que, même si on continuait la récursion, elle serait de toutes manières inférieure à un seuil fixé, le calcul des rayons est alors arrêté. Ceci élimine le calcul inutile d'un grand nombre de rayons.

1•2•5•2 Suréchantillonage

Sur certaines partie de l'image, l'intensité des pixels est relativement constante et donc il suffit de ne calculer qu'un petit nombre d'échantillons et d'interpoler les valeurs intermédiaires. Le résultat obtenu reste visuellement correct. Ceci permet d'éliminer un grand nombre de rayons primaires ainsi que les sous-arbres associés Une méthode d'échantillonnage aléatoire a été proposée par Cook et Al. Ces techniques permettent en résumé d'exploiter la cohérence image.

1•2•6 Rayons plus généraux

La méthode classique de L.R. présente l'avantage de proposer une représentation simple des rayons, supposée infiniment fins, et un calcul d'intersection efficace. Ce qui confère à cette méthode une certaine généralité puisqu'elle permet de traiter l'interaction de rayons avec un grand nombre d'objets différents. Une autre approche a été envisagée, visant à améliorer la technique précédente. L'idée retenue par cette nouvelle méthode est de définir un concept plus général dont le rayon serait un cas particulier. Sont alors apparus les cônes, faisceaux et pinceaux qui regroupent un ensemble de rayons et permettent de tracer simultanément plusieurs rayons et d'exploiter ainsi la cohérence de la scène. Cependant ceci entraîne certaines contraintes sur l'environnement:

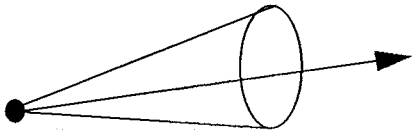
- Limitation des types d'objets primitifs utilisables
- Calcul d'intersection non exacte

Les avantages à en tirer en retour sont:

- Une exécution plus rapide
- Un anti-aliassage
- Des effets optiques supplémentaires.

1•2•6•1 Le tracé de cônes

J. Amanatides a proposé une généralisation des rayons par des cônes circulaires. Un cône est donc défini par son sommet, une ligne directrice et un angle solide [AMA 84].



L'algorithme de tracé de cônes consiste alors à déterminer la portion du cône arrêtée par un objet, et pour chaque réflexion ou réfraction sur un objet, calculer les nouveaux cônes réémis. Une caractéristique intéressante est la possibilité de recréer l'effet de pénombre inexistant dans la méthode classique. Cependant la difficulté du calcul d'intersection cône/objet impose que la scène ne soit constituée que des objets de base suivants: sphères, plans et polygones.

1•2•6•2 Le tracé de faisceaux

Cette méthode a été introduite par Heckbert et Hanrahan [HEC 84]. Pour eux, un faisceau est un cône à section polygonale quelconque qui regroupe l'ensemble des rayons passant à l'intérieur du volume ainsi défini. Si on se limite aux objets construits à l'aide de facettes polygonales planes, ceci confère des caractéristiques particulières aux faisceaux: un faisceau reste un faisceau après intersection avec un objet ou réflexion sur un objet. La réfraction quant à elle, ne préserve pas la nature des faisceaux, à cause de la non linéarité. De façon analogue à l'algorithme du L.R., une arborescence de faisceaux réfléchis et transmis est alors construite mais celle-ci est plus difficile à élaborer que dans le cas des rayons. L'algorithme de tracé de faisceaux permet d'éliminer le traitement des objets éloignés et qui sont occultés par des objets plus proches.
Dadoun et Kirkpatrick ont permis d'accélérer la méthode en introduisant une représentation hiérarchique de la scène et en combinant l'utilisation de volumes englobants avec une subdivision spatiale pour obtenir un arbre binaire d'enveloppes convexes.

1•2•6•3 Le tracé de pinceaux

Cette notion émane de Shinya et Al. Un pinceau correspond à un ensemble de rayons situés autour d'un rayon directeur. Ces deux chercheurs ont étudié l'interaction de tels pinceaux avec l'environnement et ont montré que les transformations correspondantes sont presque linéaires et peuvent donc être représentées de façon matricielle. La propagation de pinceaux est alors obtenue par composition de matrices. L'approximation effectuée n'est toutefois valable que pour des surfaces lisses ne présentant pas de discontinuité ni d'arrêtes franches. Pour les calculs des points ne vérifiant pas cette propriété, il faut utiliser un L.R. classique.

1•2•7 Parallélisation du lancer de rayons

Un des pôles de recherche s'est orienté sur les techniques de parallélisation de l'algorithme de L.R.
Une première classe utilise des machines parallèles existantes pour implémenter cet algorithme. Une version vectorielle de l'algorithme de L.R. a été implémentée sur Cyber 205 par Plunket en 1985. Cette technique prend une liste de rayons et détermine les intersections avec tous les objets de la scène. Cet algorithme est plus complexe que la version scalaire car il traite simultanément plusieurs pixels et nécessite un espace mémoire important. Bouatouch a étudié l'implantation de l'algorithme de L.R. sur un hypercube.
Une seconde classe est basée sur le développement de nouvelles architectures de machines spécialisées dans le L.R. intégrant certaines des techniques d'accélération présentées dans les paragraphes précédents. C'est ainsi que sont nés les réseaux de processeurs. Ceux-ci sont composés de cellules élémentaires (processeurs) interconnectées entre elles selon une topologie caractérisant le réseau. Un des exemples en est la machine Cristal-TPX développée par Brusq en 1986 au CCETT de Rennes,

dans laquelle un groupe de pixels est associé à chaque processeur. La machine Cube 3D de Kaufman utilise une décomposition volumique de la scène en voxels [KAU 88]. Cette machine a été conçue pour pouvoir accéder en parallèle à une rangée de voxels se projetant sur un même pixel écran. Elle permet de déterminer le voxel visible le plus proche dans la rangée en un temps logarithmique. Le changement de la position de l'observateur est obtenu par rotation de la base de données dans la mémoire.
Il faut également noter le projet VOXAR développé à l'université Paul Sabatier de Toulouse par l'équipe de R. Caubet. Il se compose d'un réseau tridimensionnel de processeurs appelé "Hypertore" réalisé autour de transputers Inmos. Cette machine utilise une subdivision spatiale régulière de la scène en voxels ainsi qu'un suivi analytique de rayons [CAU 88] [CAU 89].
Notre projet RC^2 s'inscrit dans cet axe de recherche. C'est une architecture massivement parallèle dédiée à la synthèse d'images et qui utilise la technique du L.R. Sa principale caractéristique est d'utiliser un réseau physique 2D de processeurs pour gérer l'espace virtuel 3D contenant la scène [AML 90]. Elle entre dans le cadre des machines à partitionnement spatial de l'image. Comme dans VOXAR et Cube 3D, nous définissons une subdivision régulière de l'espace 3D dans laquelle les objets sont discrétisés. Chaque objet de la scène est décomposé en cubes élémentaires appelés **voxels** dans lesquels sont stockés les caractéristiques de l'objet. C'est le détail le plus fin qui puisse être mémorisé. Un voxel possède deux états: Vide/Occupé. Afin de limiter le nombre de voxels occupés, nous ne mémorisons que les voxels constituant la surface extérieure des objets. Les rayons sont eux-aussi discrétisés et nous utiliserons donc un algorithme de suivi de rayons incrémental pour le calcul des ombres et l'éclairement des objets.

1.3 Description de l'architecture RC^2

Elle se compose d'un pipeline à 4 étages [ATA 89]:

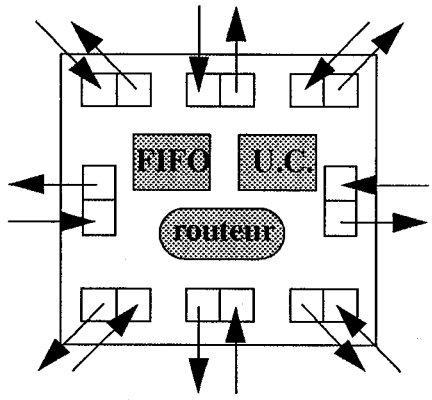
1: Sous-système "Géométrie"
2: Etage "Précalcul"
3: Distribution des objets
4: Réseau cellulaire

Le sous-système "Géométrie" (1) se charge de la modélisation de la scène. Il fournit des objets dans l'espace écran et doit être suffisamment rapide pour alimenter les étages suivants. L'étage "Précalcul" (2) découpe les polygones formant les objets en trapèzes à côtés parallèles horizontaux. La distribution des objets (3) aux cellules du réseau se fait par le bord gauche du réseau. Il y a propagation d'une onde dans le réseau RC^2 (4) qui convertit les objets en éléments de volumes (Voxels) stockés à l'intérieur de l'espace mémoire virtuel 3D associé au réseau physique 2D.

Le réseau cellulaire proprement dit correspond au module principal de l'architecture. C'est une machine à fonctionnement MIMD et à communication par messages. Il est composé de processeurs disposés en n lignes et n colonnes [ATA 89]. Chaque cellule élémentaire (x,y) du réseau gère une partie de la scène. Cette partie correspond à l'ensemble des voxels ayant mêmes coordonnées X,Y et de coordonnée Z quelconque. Chaque cellule est reliée à ses 8 voisines Est, NE, Nord, NO, Ouest, SO, Sud, SE. La connectivité 8 est réalisée physiquement, ou bien peut être obtenue par routage de messages sur un réseau de connectivité 4 voisins. Une cellule est capable d'effectuer un calcul et de transmettre le résultat à ses voisines.

1•3•1 Composition de la cellule de base

Chaque cellule est composée d'une unité de calcul et de mémoire afin de stocker le contenu des voxels de la scène gérés par elle. Elle comprend également un routeur qui permet d'acheminer les messages qui lui sont envoyés, de boîtes aux lettres et de files d'attente associées pour communiquer avec les voisines. Pour cela nous avons un buffer de sortie pour chaque direction d'émission ainsi qu'un buffer d'entrée pour chaque direction de réception. La cellule passe dans l'état **bloqué** si l'un de ses buffers de sortie est plein. On ajoute une file d'attente (FIFO) pour tamponner les messages reçus et écouler plus rapidement le flux en évitant les conflits en amont de la cellule.

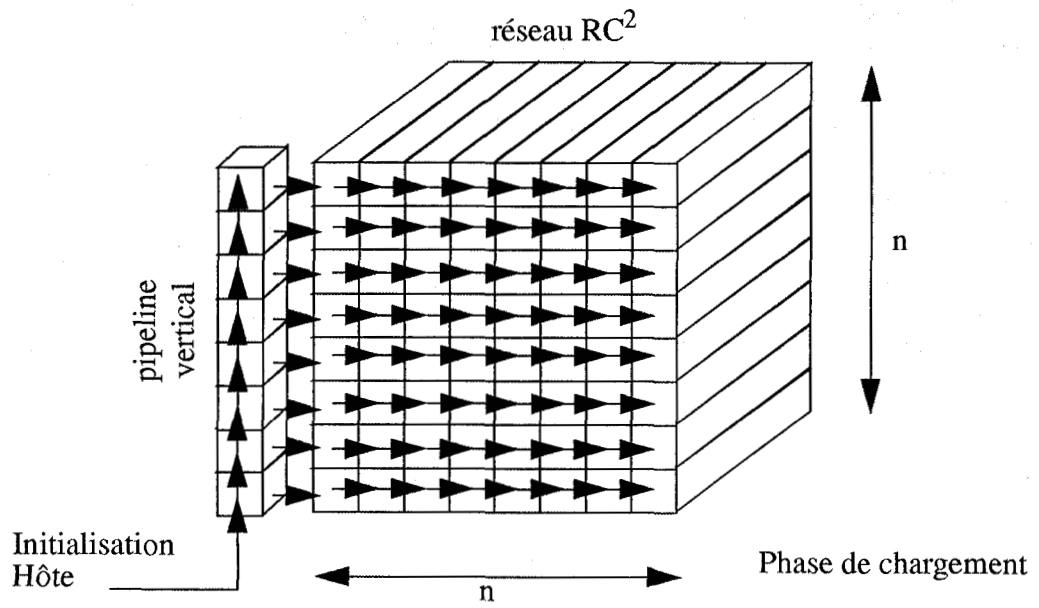


1•3•2 Fonctionnement du réseau

Le réseau fonctionne en deux étapes. La première correspond au chargement de la scène, dans laquelle la base de données fournie par le modeleur est décomposée en voxels qui sont stockés dans les cellules correspondantes. La seconde effectue un lancer de rayon cellulaire. A la fin de l'étape de chargement, nous pouvons afficher une première image de la scène, qui a été calculée de façon incrémentale au cours de cette phase (Ombrage de Gouraud ou Phong). La seconde phase accroit la qualité de l'image en exécutant un L.R. distribué.

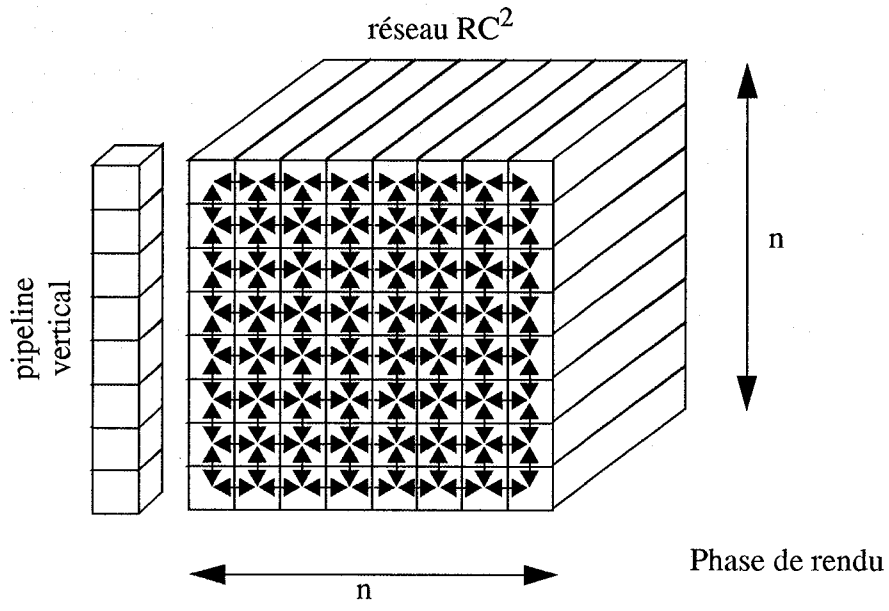
1^{o}) Chargement de la scène

Ce traitement découpe la scène et donne à chaque cellule les objets discrétisés en voxels qu'elle doit gérer. Les objets sont découpés en trapèzes dans l'étage de précalcul puis chaque trapèze est ensuite injecté dans le pipeline de distribution et découpé en segments horizontaux. Ces segments sont enfin transformés en voxels dans le réseau RC^2 fonctionnant en mode multipipeline. Chaque cellule reçoit les caractéristiques de l'objet. Un pipeline vertical de cellules est connecté à la première colonne du réseau. Il permet, dans la phase de chargement, le découpage de la scène et envoie à chaque rangée du réseau les données d'initialisation qui lui correspondent (mode multipipeline).



2°) Exécution du L.R.

La seconde phase correspond à l'étape de rendu dans laquelle sera calculé l'éclairage de chaque voxel. L'image sera affichée au fur et à mesure de sa construction. Chaque cellule calcule le rayon primaire intersectant le pixel géré par celle-ci ainsi que le sous-arbre correspondant de l'arbre des rayons qui est ainsi distribué sur le réseau. Chaque branche est évaluée en parallèle. Un algorithme de suivi de rayon incrémental est utilisé pour gérer le cheminement des rayons de cellule en cellule. Une cellule R doit donc router le rayon reçu et s'il y a intersection, elle doit renvoyer la valeur de l'intensité lumineuse de l'objet coupé par le rayon reçu à la cellule émettrice E. Le processus est récursif, c'est à dire que la cellule R peut éventuellement réémettre deux rayons secondaires et attendre les valeurs des intensités retournées par ces rayons avant de retourner sa propre valeur à la cellule E. Pour savoir s'il y a ou non intersection entre un rayon et un objet, il suffit de comparer la profondeur courante du rayon avec celle des voxels occupés dans la cellule coupée par ce rayon. Le calcul d'intersection rayon/objet est ainsi grandement simplifié.



Dans l'étude qui suit, un grand nombre de résultats s'appliquent à une architecture 3D abstraite composée d'un cube à connexions locales. Nous préciserons, lorsque cela sera nécessaire, les résultats spécifiques à notre machine RC^2.

CHAPITRE 2

Tracés de segments

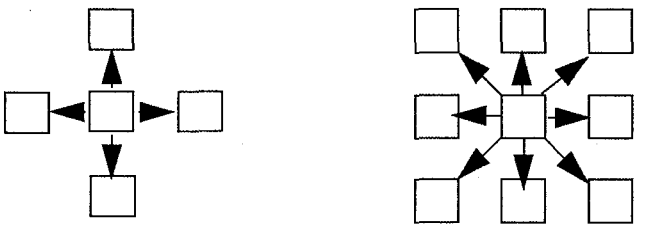
TABLE DES MATIERES

CHAPITRE 2 : Tracés de segments

2.1 Notions de topologie

Nous allons introduire les notions de connectivité d'une surface et d'un objet 1D quelconque (un segment par exemple), puis nous verrons les problèmes qui peuvent se poser pour que l'intersection entre un rayon et une surface dans le cas discret soit cohérente avec l'intersection dans le cas continu.
La connectivité d'un réseau correspond au nombre de cellules voisines d'une cellule donnée. C'est en fait le nombre de liens partant d'une cellule donnée vers les autres cellules.

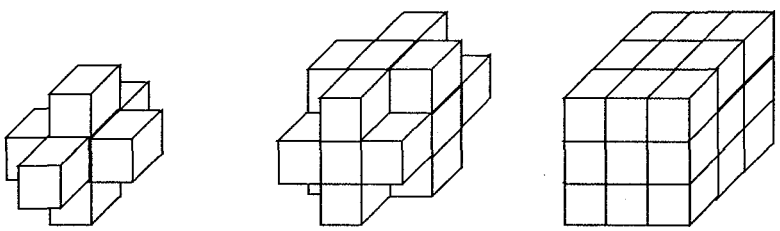
Exemples de connectivité:



Réseau 2D
Connectivité 4

Réseau 2D
Connectivité 8

Figure 2.1 : Connectivité d'un réseau 2D



Réseau 3D
Connectivité 6

Réseau 3D
Connectivité 18

Réseau 3D
Connectivité 26

Figure 2.2 : Connectivité d'un réseau 3D

La connectivité d'un objet discrétisé correspondra au nombre de voisins possibles d'un voxel donné de cet objet [ROS 89].

<u>Définitions en 2D:</u>

Deux pixels P(x ,y) et P'(x',y') seront dits adjacents si

$$\begin{cases} x'=x+dx \\ y'=y+dy \end{cases} \quad \text{avec} \quad \begin{cases} dx \in \{-1,0,1\} \\ dy \in \{-1,0,1\} \end{cases} \quad \text{et} \quad (dx,dy)\neq(0,0)$$

Un pixel possède donc 8 pixels adjacents.

Un segment discrétisé sera de connectivité 4 (un pixel possède 4 côtés)
si tout couple (P,P') de pixels adjacents appartenant au segment discrétisé est tel que P et P' sont reliés par un côté commun ou bien il existe un pixel P" adjacent à la fois à P et P' appartenant au segment discrétisé.

Un segment discrétisé sera de connectivité 8 (4 côtés + 4 coins)
s'il existe un couple de pixels adjacents (P,P') appartenant au segment discrétisé tel que P et P' soient reliés par un sommet commun et il n'existe aucun autre pixel adjacent à P et P' et appartenant au segment discrétisé.

Définitions 3D:

Deux voxels V(x , y) et V'(x',y') seront dits adjacents si

$$\begin{cases} x'=x+dx \\ y'=y+dy \\ z'=z+dz \end{cases} \quad \text{avec} \quad \begin{cases} dx \in \{-1,0,1\} \\ dy \in \{-1,0,1\} \\ dz \in \{-1,0,1\} \end{cases} \quad \text{et} \quad (dx,dy,dz) \neq (0,0,0)$$

Un voxel possède donc 26 voxels adjacents.

Un segment (resp. une surface) discrétisé(e) sera de connectivité 6 (un cube possède 6 faces) si pour tout couple (V,V') de voxels adjacents appartenant au segment (à la surface) discrétisé(e), il existe une suite $V_0=V$, V_1 , V_2 , ... $V_n = V'$ de voxels adjacents à V et V' appartenant au segment (resp.à la surface) discrétisé(e), dans laquelle chaque voxel V_i est relié au suivant V_{i+1} par une face commune.

Un segment (resp. une surface) discrétisé(e) sera de connectivité 18 (6 faces + 12 arrêtes) si pour tout couple (V,V') de voxels adjacents appartenant au segment (à la surface) discrétisé(e), il existe une suite $V_0=V$, V_1 , V_2 , ... $V_n = V'$ de voxels adjacents à V et V' appartenant au segment (resp.à la surface) discrétisé(e), dans laquelle chaque voxel V_i est relié au suivant V_{i+1} par une face ou une arrête commune.

Un segment (une surface) discrétisé(e) sera de connectivité 26 (6 faces,12 arrêtes,8 sommets) si tout couple de voxels adjacents (V,V') appartenant au segment (à la surface) discrétisé(e) est tel que V et V' sont reliés par une face, une arrête ou un sommet commun.

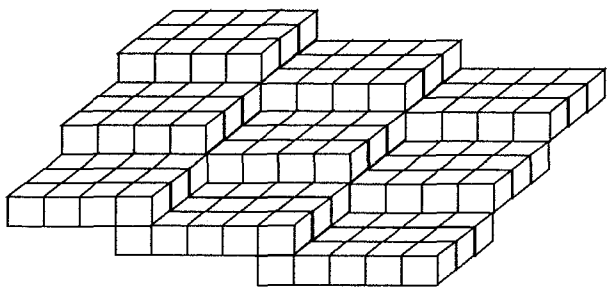


Figure 2.3 : Surface en connectivité 6

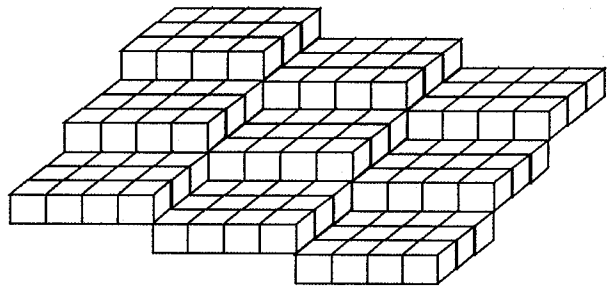


Figure 2.4 : Surface en connectivité 18

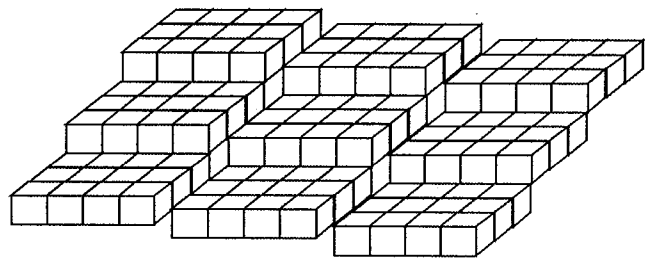
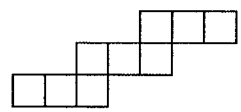
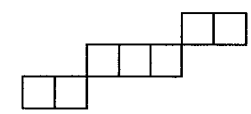


Figure 2.5 : Surface en connectivité 26

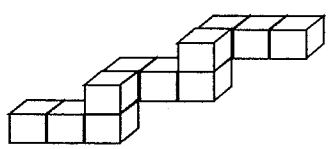


segment 2D
connectivité 4

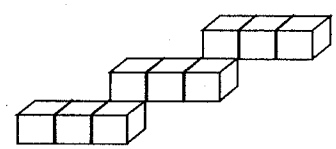


segment 2D
connectivité 8

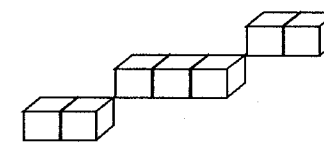
Figure 2.6 : Connectivité segments 2D



segment 3D
connectivité 6



segment 3D
connectivité 18



segment 3D
connectivité 26

Figure 2.7 : Connectivité segments 3D

Dans l'étape de lancer de rayons, des rayons discrétisés vont alors rencontrer des surfaces elles aussi discrétisées. Selon la connectivité des rayons ainsi que celles des surfaces, l'intersection pourra, dans certains cas, ne pas être détectée et un rayon pourra alors tout simplement passer au travers de la surface normalement coupée (effet passe muraille). Ce problème s'étend à l'intersection entre un objet 1D quelconque avec un autre objet 1D ou 2D.

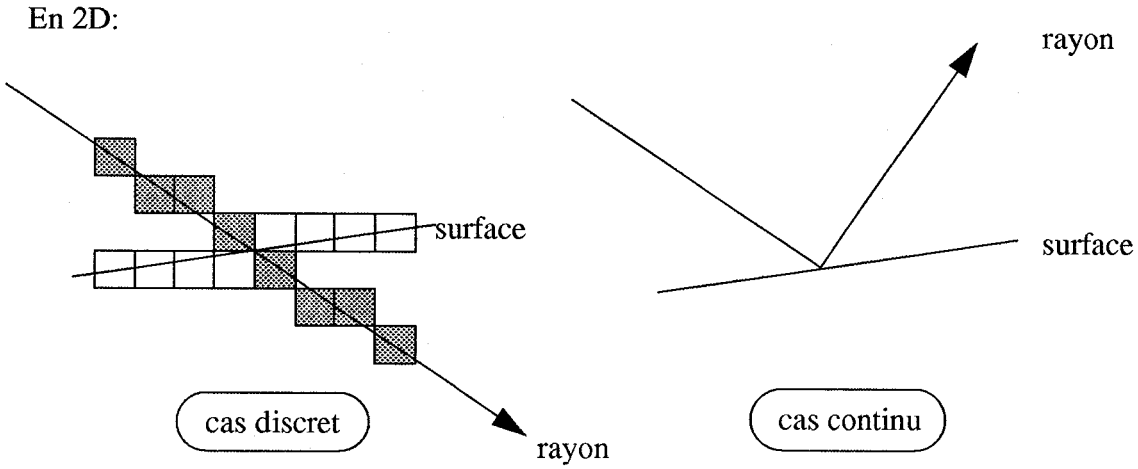


Figure 2.8 : Effet "passe-muraille" en 2D

Connectivité rayons	Connectivité surface 4	8
4	NON	NON
8	NON	OUI

Effet passe muraille possible

Figure 2.9 : Cas où se produit l'effet passe-muraille 2D

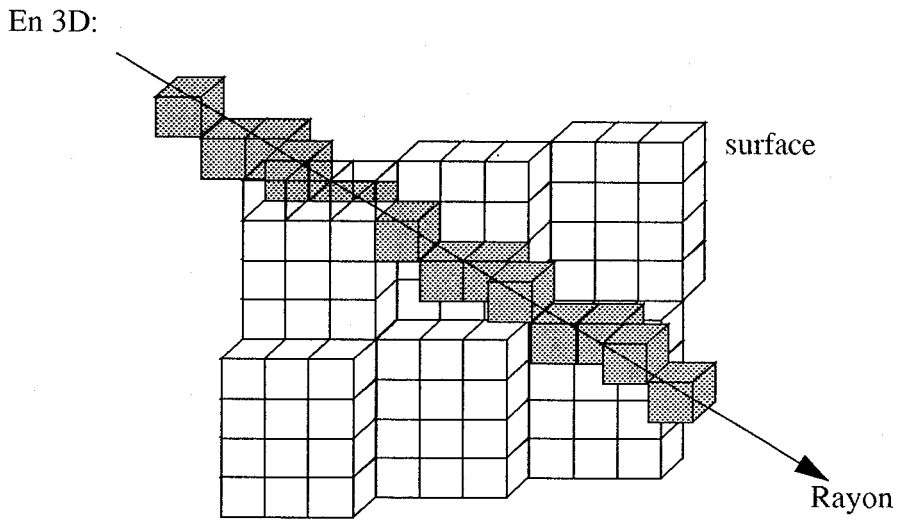


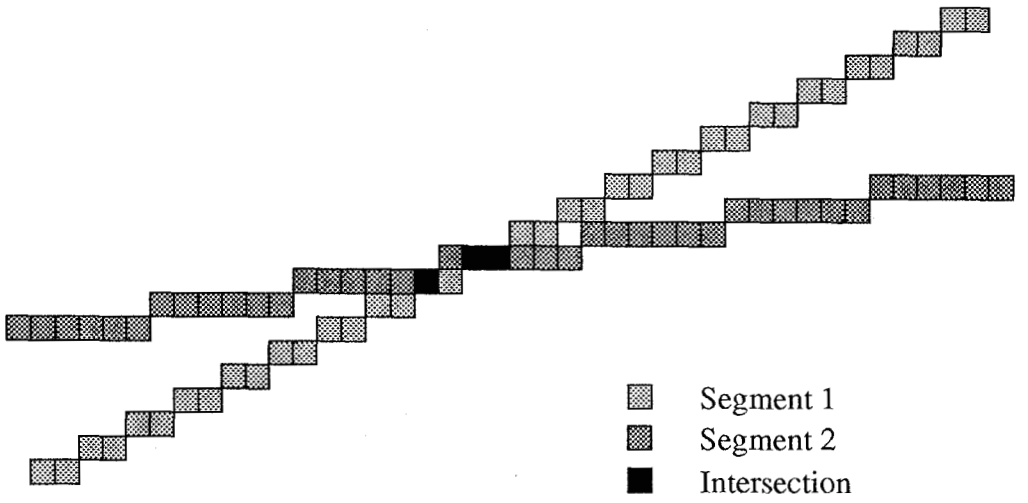
Figure 2.10 : Effet "passe-muraille" 3D

Connectivité rayons \ Connectivité surface	6	18	26
6	NON	NON	NON
18	NON	OUI	OUI
26	NON	OUI	OUI

Figure 2.11 : Cas où se produit l'effet passe-muraille 3D

En 2D, pour éviter l'effet passe-muraille, il faut que les surfaces ou les rayons soient de connectivité 4.
En 3D, pour éviter ce phénomène, il faut que les surfaces ou les rayons soient de connectivité 6.

En espace continu, l'intersection de deux droites non parallèles se réduit à un point.
Dans un espace discret, cette propriété n'est plus vraie et l'intersection de deux droites discrètes non parallèles peut comporter plusieurs pixels. Deux segments discrets peuvent se couper plusieurs fois:



2.2 Les différentes méthodes de tracé 3D

Les algorithmes permettant de tracer un segment de droite entre deux points de l'espace 3D discret (Voxels) sont à la base de tous les autres algorithmes. Ils se déduisent des algorithmes de tracé de segments dans le plan (Algorithmes de Bresenham [BRE 65], DDA [FIE 85], dichotomique [PEL 85], Loceff [LOC 80]). Nous les classons en deux catégories: les méthodes "par projection" et les méthodes "par calcul direct".
Dans les méthodes "par projection" sont calculées indépendamment deux projections du segment 3D sur deux plans de base, l'algorithme restitue le segment 3D à partir de ces

deux projections. Le problème réside alors dans le choix des deux projections à prendre en compte. En effet, de façon à pouvoir reconstruire correctement le segment 3D, il faut choisir parmi les trois projections possibles du segment 3D sur chacun des plans de base XOY, XOZ, YOZ, celles qui possèdent le plus de points, car l'une d'entre elles peut ne pas donner un segment de type Bresenham et conduirait à des erreurs.
Dans l'exemple suivant, le segment final contient 12 voxels. Pour construire ce segment il faut donc utiliser les projections $proj_1$ sur le plan XOZ et $proj_2$ sur le plan XOY. En effet, la projection $proj_3$ sur YOZ ne contient que 8 pixels et, combinée à l'une des deux autres projections $proj_1$ ou $proj_2$, ne permet pas de retrouver un segment 3D unique.

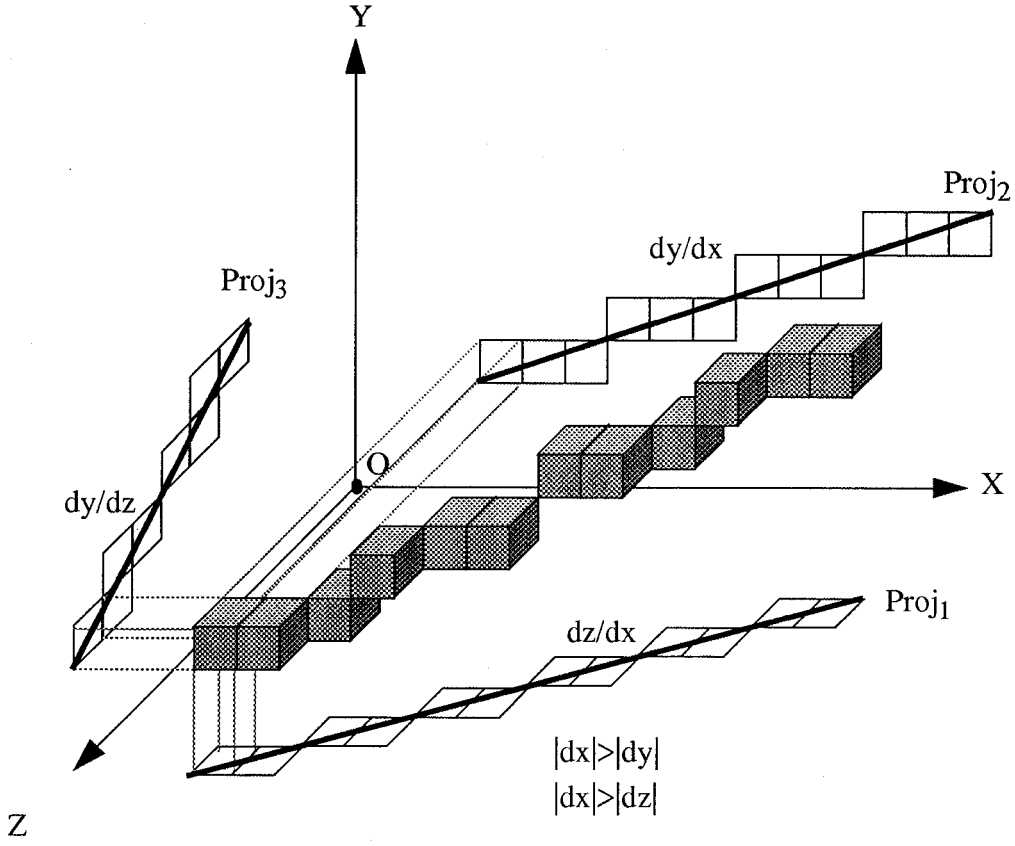


Figure 2.12 : Projections d'un segment 3D

Le segment étant défini par la différence dx, dy, dz entre les coordonnées de ses extrémités, la technique consiste à prendre comme direction principale du segment celle parmi OX, OY, OZ qui correspond à la valeur maximale des incréments |dx|, |dy|, |dz|. Nous utiliserons donc les projections sur les deux plans contenant cette direction.
A cette méthode un peu complexe, nous préférerons, lorsque cela est possible, la seconde méthode dite "par calcul direct" obtenue par extension des algorithmes 2D en 3D. Nous calculons directement les voxels du segment par des algorithmes dont la structure est calquée sur celle des algorithmes 2D correspondants.

2.3 Les algorithmes 3D

2•3•1 Algorithme de Bresenham étendu 3D

Cet algorithme calcule trois erreurs selon les trois directions X, Y, Z par rapport au segment réel et les compare à un seuil de référence qui dépend du segment à tracer. Le tracé commence à partir de l'une des extrémités du segment puis, pour chacune des directions X, Y, Z, si l'erreur est supérieure au seuil, l'algorithme incrémente la

coordonnée correspondante et détermine ainsi le voxel suivant à afficher. Cette étape est réitérée jusqu'à ce que l'autre extrémité soit atteinte. L'algorithme de Bresenham présente l'avantage de travailler sur des nombres entiers, ce qui lui confère une certaine rapidité. Sa complexité est en O(N), N étant le nombre de points à afficher.

```
BRESENHAM
   DiffX=|XB-XA|
   DiffY=|YB-YA|
   DiffZ=|ZB-ZA|
   StepX=sgn(XB-XA)
   StepY=sgn(YB-YA)
   StepZ=sgn(ZB-ZA)
   seuil=max(DiffX,DiffY,DiffZ)
   X=XA; Y=YA; Z=ZA
   ErreurX=seuil/2
   ErreurY=seuil/2
   ErreurZ=seuil/2
   Afficher(X,Y,Z)
   pour i de 1 à seuil
     ErreurX=ErreurX+DiffX
     ErreurY=ErreurY+DiffY
     ErreurZ=ErreurZ+DiffZ
     Si ErreurX ≥ seuil
       alors X=X+StepX
             ErreurX=ErreurX-seuil
     Fsi
     Si ErreurY ≥ seuil
       alors Y=Y+StepY
             ErreurY=ErreurY-seuil
     Fsi
     Si ErreurZ ≥ seuil
       alors Z=Z+StepZ
             ErreurZ=ErreurZ-seuil
     Fsi
     Afficher(X,Y,Z)
   Fpour
FIN
```

2•3•2 Algorithme DDA étendu 3D

Cet algorithme détermine d'abord le nombre de points à afficher et calcule trois incréments selon les trois directions X, Y, Z. Le segment est tracé en partant de l'une de ses extrémités puis l'algorithme calcule la position du voxel suivant en incrémentant chacune des coordonnées de la valeur correspondante. Cette étape est répétée tant que le nombre de points à afficher n'est pas atteint. Les calculs sont effectués sur des nombres fractionnaires puis arrondis à l'entier le plus proche pour l'affichage. La complexité de cet algorithme est identique à celle de l'algorithme de Bresenham, soit O(N).

```
DEBUT
   diffX=XB-XA
   diffY=YB-YA
   diffZ=ZB-ZA
   AdifX=Abs(diffX)
   AdifY=Abs(diffY)
   AdifZ=Abs(diffZ)
   longueur=max(Adifx,AdifY,AdifZ)
   Xincr=diffX/longueur
   Yincr=diffY/longueur
   Zincr=diffZ/longueur
   X=XA; Y=YA; Z=ZA
   pour compteur de 0 à longueur
     Afficher (Arrondi(X), Arrondi(Y), Arrondi(Z))
     X=X+Xincr
     Y=Y+Yincr
     Z=Z+Zincr
   Fpour
FIN
```

2•3•3 Algorithme dichotomique 3D

Cet algorithme détermine le segment en découpant celui-ci en sous-segments de façon récursive. A la première étape, le milieu M du segment [A,B] à tracer est calculé, puis la même procédure est appliquée aux deux sous-segments [A,M] et [M,B] qui se divisent à leur tour en quatre autres sous-segments. Le processus s'arrête lorsque les deux extrémités du sous-segment passé à la procédure sont identiques. Le calcul du milieu est effectué en arithmétique fractionnaire puis arrondi à l'entier le plus proche.
Les segments obtenus par cet algorithme diffèrent donc un peu de ceux obtenus par les deux méthodes précédentes.

```
DEBUT
   Tracer3(XA,YA,ZA,XB,YB,ZB)
FIN

PROCEDURE Tracer3(X1,Y1,Z1,X2,Y2,Z2)
   Si ((X1 <> X2) ou (Y1 <> Y2) ou (Z1 <> Z2))
     alors Xm = Arrondi ((X1+X2)/2)
           Ym = Arrondi ((Y1+Y2)/2)
           Zm = Arrondi ((Z1+Z2)/2)
           Afficher(Xm,Ym,Zm)
           Tracer3(X1,Y1,Z1,Xm,Ym,Zm)
           Tracer3(Xm,Ym,Zm,X2,Y2,Z2)
   Fsi
FIN
```

La complexité est en $O(\log_2 N)$ si nous disposons de N processeurs pour tracer le segment, et en O(N) si nous travaillons avec un seul processeur.

2•3•4 Algorithme de Loceff

Cet algorithme fonctionne en 2D et fractionne le segment en paliers de droite horizontaux [LOC 80], ce qui permet de déterminer en une seule fois un ensemble de points. Il est basé sur le fait qu'un segment 2D discrétisé ne contient que deux types de paliers ayant respectivement pour longueur N ou N-1, N dépendant du segment à tracer [REV 88].
L'extension de cet algorithme pour des segments 3D est peu aisée car, comme le montre l'exemple suivant, la propriété précédente n'est plus vérifiée pour des segments 3D et la longueur des paliers peut varier de plus d'une unité.

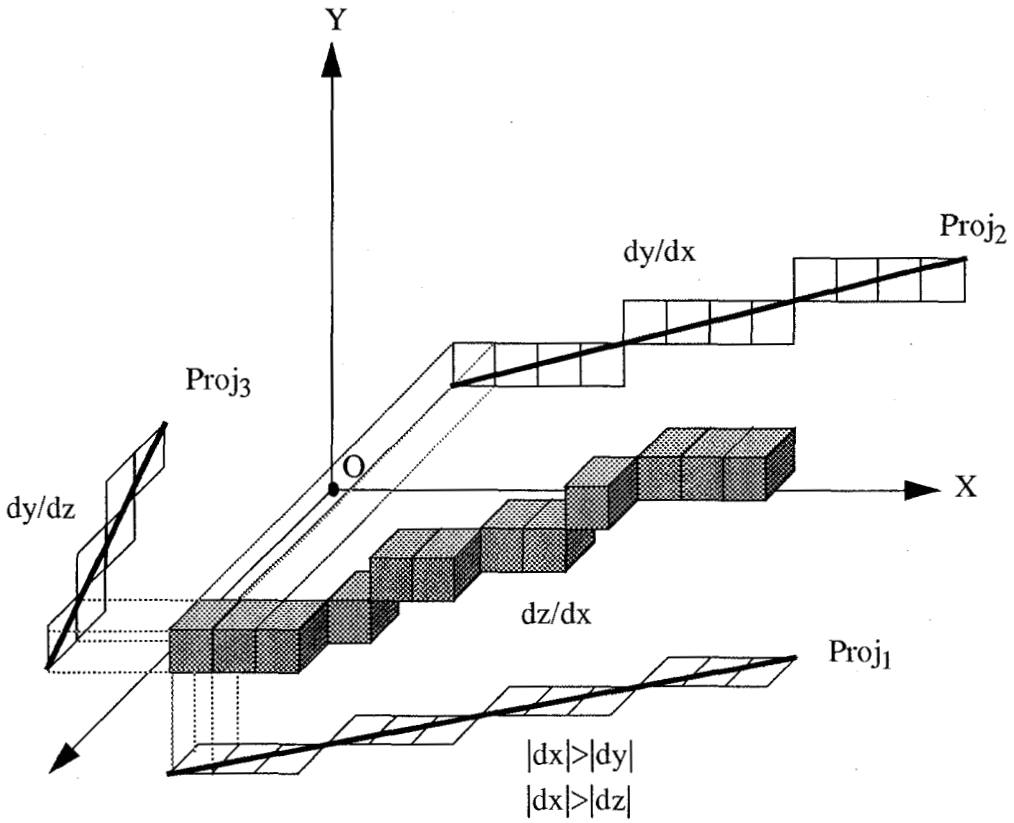
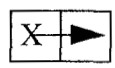


Figure 2.13 : Paliers de droites 3D

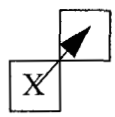
Nous pouvons cependant utiliser cet algorithme dans la méthode "par projection" pour calculer les deux projections du segment 3D.
Dans ce qui suit, l'algorithme de Loceff est présenté pour le premier octant dans un espace 2D.

Il existe deux mouvements de base M_1 et M_2 qui affichent le point courant et déplace celui-ci d'une position à droite ou en diagonale. [DUR 83].

M_1 : Afficher (X,Y)
X=X+1



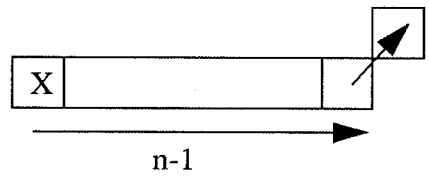
M_2 : Afficher (X,Y)
X=X+1; Y=Y+1



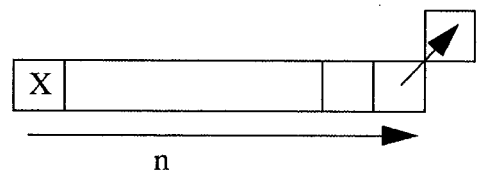
A partir de ces deux mouvements élémentaires, sont construits les mouvements

composés M'_1 et M'_2 suivants qui affichent le point courant tout en déplaçant celui-ci d'un certain nombre de positions.

M'_1: $(N-1)*M_1$; M_2



M'_2: $N*M_1$; M_2



Les différents paliers de droites sont construits par l'algorithme suivant:

```
LOCEFF
    N=ENT(DX/DY)
    A'=DX-N*DY
    B'=(N+1)*DY-DX
    IQ=-DY/2
    FIN=DY
    pour i de 1 à FIN
      Si IQ<0
        alors M'1; IQ=IQ+A'
        sinon M'2; IQ=IQ-B'
      Fsi
FIN
```

Le nombre d'itérations pour tracer un segment par cette méthode dépend de la longueur des paliers de droites et donc de la pente du segment. Un segment de pente 0 est tracé en une seule itération tandis qu'un segment de pente 1 est tracé en N_p itérations, N_p étant le nombre de points à afficher.

2•3•5 Algorithme d'Amanatides et Woo

Les quatre algorithmes précédents construisent des segments ayant une connectivité à 26 voisins. Ils donnent l'ensemble des voxels "les plus proches" du segment continu.
L'algorithme d'Amanatides & Woo construit, quand à lui, des segments ayant une connectivité à 6 voisins [AMA 87]. Il utilise l'équation paramétrique de la droite support et donne l'ensemble des voxels par lesquels passe le segment continu en effectuant un calcul d'intersection avec les bords des différents voxels rencontrés. On détermine à chaque fois le "prochain" bord coupé (c'est à dire le pas auquel on traversera X, Y ou Z). Le passage d'un voxel au suivant s'effectue toujours par une face commune, ce qui selon les notions de topologie confère de "bonnes" propriétés aux segments construits de cette sorte.

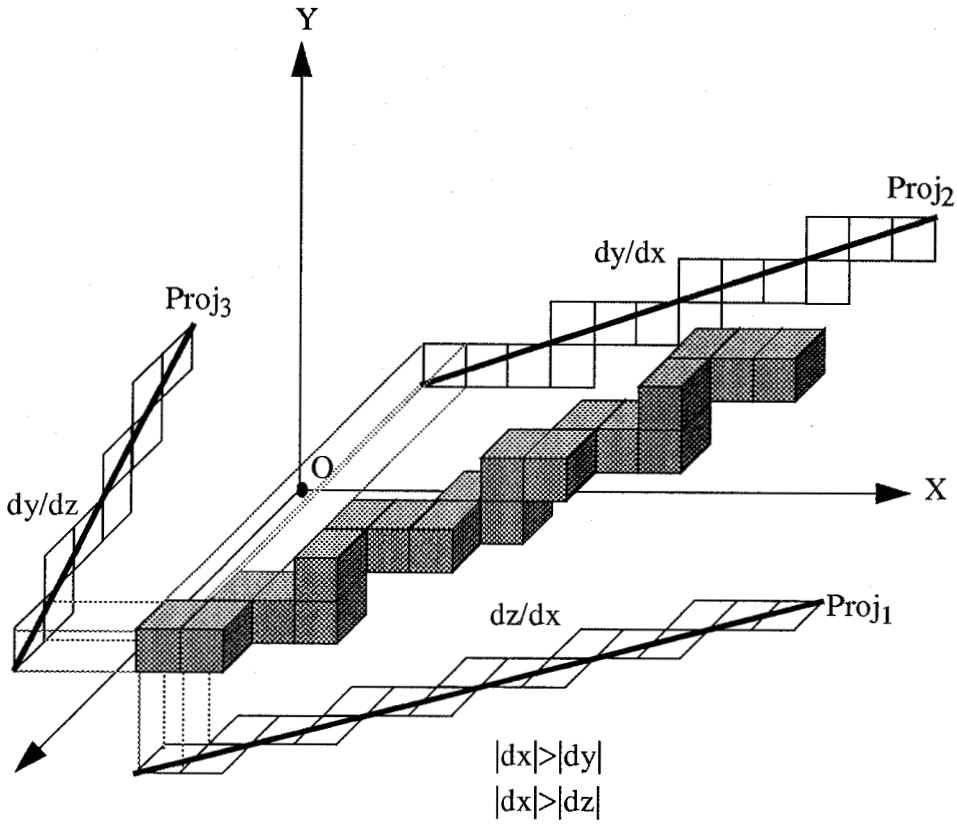


Figure 2.14 : Tracé de segments AMANATIDES

```
AMANATIDES
   StepX=sgn(X_B-X_A)
   StepY=sgn(Y_B-Y_A)
   StepZ=sgn(Z_B-Z_A)
   Si StepX=0
     alors TmaxX=1
     sinon TdeltaX=1/|X_B-X_A|
           TmaxX=TdeltaX/2
   Fsi
   Si StepY=0
     alors TmaxY=1
     sinon TdeltaY=1/|Y_B-Y_A|
           TmaxY=TdeltaY/2
   Fsi
   Si StepZ=0
     alors TmaxZ=1
     sinon TdeltaZ=1/|Z_B-Z_A|
           TmaxZ=TdeltaZ/2
   Fsi
   X=X_A; Y=Y_A; Z=Z_A
   Afficher(X,Y,Z)
   Tant que ((TmaxX ≤1) ou (TmaxY ≤1) ou (TmaxZ ≤1))
     Si TmaxX<TmaxY
       alors Si TmaxX<TmaxZ
               alors X=X+StepX
                     TmaxX=TmaxX+TdeltaX
               sinon Z=Z+StepZ
                     TmaxZ=TmaxZ+TdeltaZ
             Fsi
       sinon Si TmaxY<TmaxZ
               alors Y=Y+StepY
                     TmaxY=TmaxY+TdeltaY
               sinon Z=Z+StepZ
                     TmaxZ=TmaxZ+TdeltaZ
             Fsi
     Fsi
     Afficher(X,Y,Z)
   FinTQ
FIN
```

La complexité de cet algorithme est en O(N), N représentant le nombre de voxels traversés par le segment réel.

2.4 Tracé de cercles et segments sur un réseau neuronal booléen

Nous avons également étudié l'implantation d'algorithmes de tracé de cercles et de segments sur des réseaux neuronaux [MER 89]. Chaque pixel correspond à une cellule et effectue une partie du calcul. Le processus est donc distribué sur les pixels. Les réseaux

neuronaux diffèrent des autres réseaux cellulaires par le fait qu'ils possèdent une connectivité plus élevée. Une cellule peut être reliée à toute autre cellule du réseau. La sortie d'une cellule est une fonction logique de ses entrées. En fait chaque cellule réalisera une fonction "OU". Une cellule Ci sera donc activée si l'une au moins des cellules dont elle dépend l'est également. La génération des pixels sera monotone, c'est à dire qu'une fois qu'une cellule a déterminé son appartenance au trajet à tracer, elle ne change plus d'état. La construction d'un tel réseau nécessite de réaliser un compromis entre les différents paramètres suivants:

- Complexité en temps
- Complexité de la structure de donnée
- Complexité matérielle

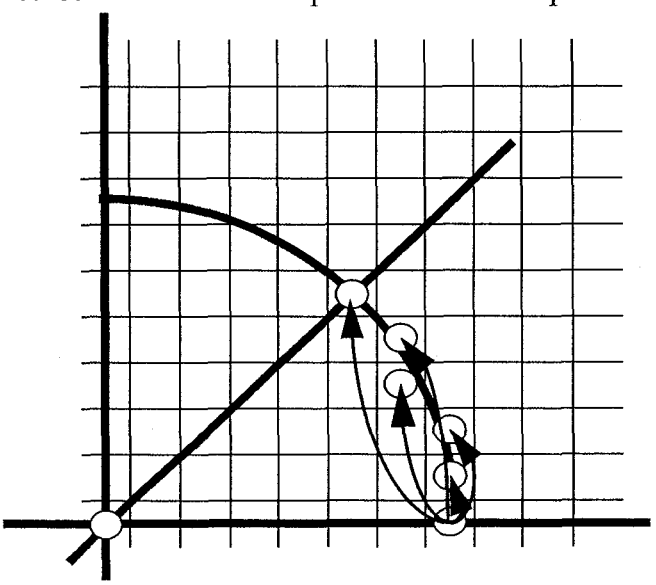
Les paramètres qui entrent en compte pour l'évaluation de la complexité matérielle sont:
- Le nombre de cellules
- La longueur totale des liens
- Le fan-out ou connectivité de sortie pour laquelle il existe une limite fixée par les contraintes technologiques.

Le temps sera évalué en terme de nombre de portes à traverser.

2•4•1 Tracés indépendants: Le cercle

L'étude se limite aux cercles de centre O et de rayon R. Seul est tracé l'arc de cercle situé dans le premier octant, les autres pouvant être obtenus par symétrie.
Etant donné qu'un cercle de rayon R et un cercle de rayon R+1 n'ont aucun pixel commun, chaque cercle peut être tracé indépendamment les uns des autres.
Le pixel (R,0) est activé, nous désirons alors que tous les pixels correspondant à l'arc de cercle à tracer soient à leur tour activés. Pour chaque schéma de connexion possible, nous donnons les différentes complexités correspondantes:

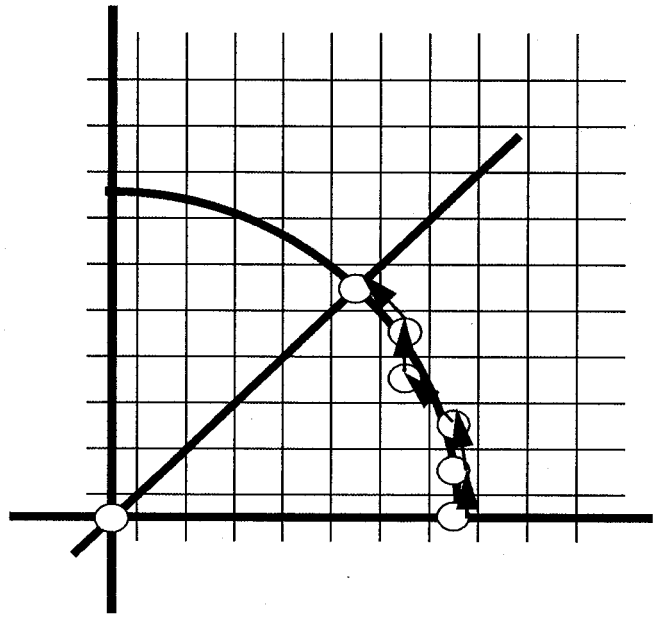
- 1ere méthode: Relier tous les pixels à activer au pixel initial.



Complexité temps : O(1)
Complexité matérielle: $O(R^2)$ en terme de longueur des connexions.
Longueur max des liens: O(R)

Nombre de liens: O(R)
Fan-out nécessaire: O(R) => incompatible avec les contraintes technologiques

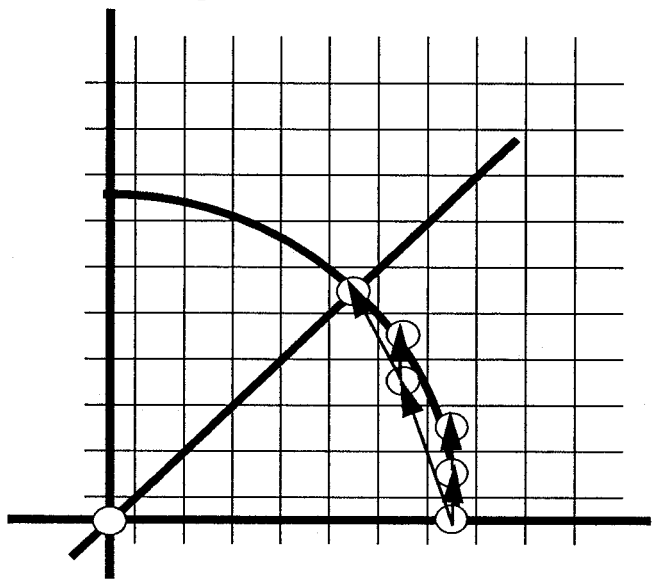
- 2nde méthode: Relier le pixel initial au pixel suivant et ainsi de suite....



Complexité temps: O(R)
Complexité matérielle: O(R)
Longueur max des liens: O(1)
Fan-out nécessaire: 1

La complexité temps est comparable à celle d'une implantation logicielle et donc n'est pas très intéressante puisqu'elle n'apporte rien de plus.

- 3eme méthode: solution optimale basée sur une structure en arbre binaire



Complexité temps: O(log R)
Complexité hard: O(R log R)
Fan-out: 1 ou 2

Nombre de liens: O(R)

-4eme méthode: Extension de la méthode précédente en utilisant des arbres F-aires (F>2).

Complexité temps: O(log R / log F)
Complexité Hard: O(R.log R / log F)
Fan-out: F>2
Nombre de liens: O(R)

Les tracés de cercles différents étant indépendants, le temps mis pour tracer plusieurs cercles en même temps est égal au temps mis pour tracer le cercle de rayon le plus grand. Une autre caractéristique intéressante est le fait que les liens ne se chevauchent pas et forment un graphe planaire, ce qui implique une réalisation physique facile à effectuer.

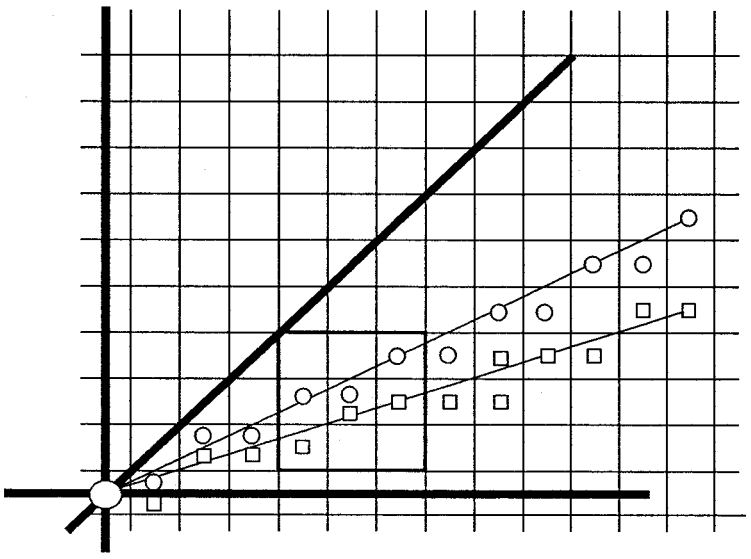
2•4•2 Tracés dépendants: Le segment

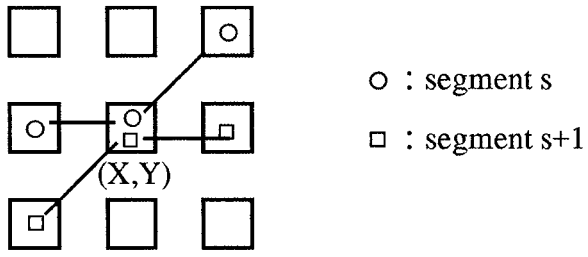
Nous allons maintenant aborder le problème du tracé de segment sur réseau neuronal. Ici aussi nous allons restreindre l'étude aux segments d'origine O et d'extrémité dans la dernière colonne. L'extrémité du segment qui appartient à la dernière colonne est activée. Le réseau doit alors allumer les pixels appartenant au segment. A l'inverse du tracé de rayons, deux segments distincts peuvent avoir des pixels en commun. Il existe donc des pixels qui appartiennent à plusieurs segments.
Si P(x,y,s) désigne le fait que le pixel de coordonnées (x,y) appartient au segment s, nous avons le théorème suivant:

Il existe des pixels tels que:

P(x,y,s) et P(x,y,s+1)
et P(x+1,y,s)
et P(x+1,y+1,s+1)
et P(x-1,y,s+1)
et P(x-1,y-1,s)

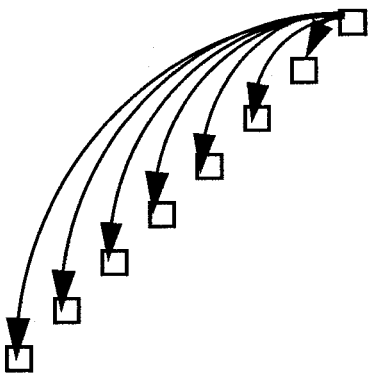
La figure suivante illustre ceci. Nous avons tracé 2 segments d'origine (0,0) ayant respectivement pour pentes 1/3 et 1/2. Les pixels correspondant à la discrétisation du premier segment sont marqués par un carré, ceux correspondant à l'autre segment, par un rond.





Un lien ne peut être créé entre le pixel (X_0,Y_0) et le pixel (X,Y) que si l'ensemble des segments qui passent par (X_0,Y_0) passent également par (X,Y). Nous dirons alors que la cellule (x,y) dépend de la cellule (X_0,Y_0). Comme dans le cas du tracé de rayons nous proposons différents schémas de connexion possibles et donnons les complexités correspondantes.

1ere méthode: Chaque cellule ne dépend que de l'extrémité

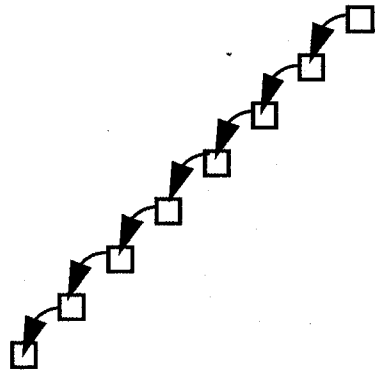


Complexité temps: O(1)
Complexité matérielle: $O(N^2)$
Fan-out nécessaire: N

Chaque point P_i appartient à un nombre k_i de segments ayant leur extrémité dans la dernière colonne. Chaque cellule C_i sera donc reliée aux extrémités des k_i segments auxquels elle appartient. On dira que la cellule C_i dépend de ces cellules extrémités.

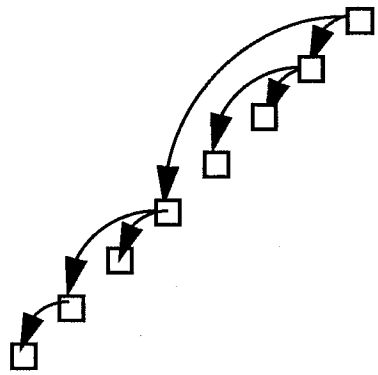
Nous voyons tout de suite apparaître un problème: la connectivité d'entrée d'une cellule augmente au fur et à mesure que l'on se rapproche de l'origine. Chaque cellule extrémité est reliée aux n cellules qui composent le segment correspondant.
Remarque: pour les cellules situées à droite de la colonne "milieu" (d'indice ENT(n/2)), la connectivité d'entrée ne dépasse pas 2 (il passe au plus 2 segments par chacune de ces cellules).

2nde méthode: Chaque cellule ne dépend que de sa voisine directe de droite



Complexité temps: O(N)
Complexité matérielle: O(N)
Fan out nécessaire: 1

3eme méthode: Arbre binaire gauche



Complexité temps: $O(\log_2 N)$
Complexité matérielle: $O(N.\log_2 N)$
Fan-out nécessaire: 2

En fait, le réseau construit à partir de la solution théorique crée des liens incorrects. Un lien allant d'une cellule C_1 vers une cellule C_2 est incorect s'il existe au moins un segment qui passe par C_1 et non par C_2. Nous nous autorisons de déplacer les liens d'un certain seuil par rapport au schéma théorique en remplaçant chaque lien incorrect par le lien valide le plus proche. Les simulations ont donné les résultats suivants: [UZU 89]

Temps: Cste + log(N)
Longueur moyenne: $Cste*N^{0.588}$
Seuil: Cste*N
Connectivité: $Cste*N^{0.42}$

CHAPITRE 3

Discrétisation de facettes

TABLE DES MATIERES

CHAPITRE 3 : Discrétisation de facettes

3•1 Objectif

Etant donnée une scène modélisée à l'aide de facettes, l'objectif est de découper ces facettes en cubes élémentaires appelés *Voxels* qui les approximent le mieux possible. La présente étude se limitera au cas des facettes polygonales, planes et convexes.
La scène sera supposée bornée, entièrement comprise à l'intérieur d'un cube.

Définitions: (en espace Euclidien)
-Une facette polygonale est définie par la suite des coordonnées (X_i,Y_i,Z_i) de ses sommets.
-Une facette est plane si tous les sommets appartiennent au même plan, défini par son équation AX+BY+CZ+D=0
-Une facette plane est convexe si, dans le plan de la facette, les segments reliant 2 sommets quelconques ne sortent pas de la facette (ne coupent pas le contour).

Nous allons voir que la méthode de l'approximation bi-linéaire pour la discrétisation de facettes polygonales planes convexes ne donne pas de bons résultats et conduit à la construction de surfaces comportant des trous. Nous étudierons donc deux autres méthodes pour remédier aux inconvénients de celle-ci. La première découpe la surface en tranches horizontales qui peuvent alors être remplies par un algorithme 2D classique. Nous lui préférerons toutefois la seconde méthode qui consiste à découper la surface en dexels de direction OZ. Nous déterminerons ainsi un ensemble de voxels situés à une distance du plan réel de la facette inférieure à un seuil donné. Nous étudierons également la possibilité d'implanter cet algorithme de découpage sur réseau cellulaire et faire ainsi calculer à chaque cellule (x,y) le "dexel" qui lui correspond.
Désirant implémenter un lancer de rayons discrétisé sur la scène ainsi découpée, nous aborderons enfin les notions de connectivité d'un rayon et d'une surface. Nous nous t apercevrons que l'intersection rayon/surface dans le cas discret peut ne pas toujours être détectée, et nous créerons un nouveau type de surfaces qui élimine ce problème: les surfaces blindées.

3•2 Méthode de l'approximation bi-linéaire

La méthode de remplissage de polygones 2D par balayage en lignes horizontales de haut en bas peut être étendue en 3D de la façon suivante. Une fois le plus haut sommet du polygone déterminé, pour chacun des cotés partant de ce point, deux pentes sont calculées: Dx/Dy et Dz/Dy. Le polygone est ensuite balayé en lignes (interpolation selon y) et sur chacune des lignes la valeur correspondante en Z est interpolée (selon x) puis arrondie à l'entier le plus proche. A chaque rencontre d'un nouveau sommet, les 2 pentes correspondantes sont recalculées.
Cette méthode convient parfaitement pour afficher à l'écran des objets 3D (par exemple en utilisant un Z-Buffer), puisque seul nous importe le Z réel des facettes. Cependant, dans le cas d'une représentation discrète en 3D, elle laisse apparaître des trous en Z, à l'intérieur de la surface discrétisée.

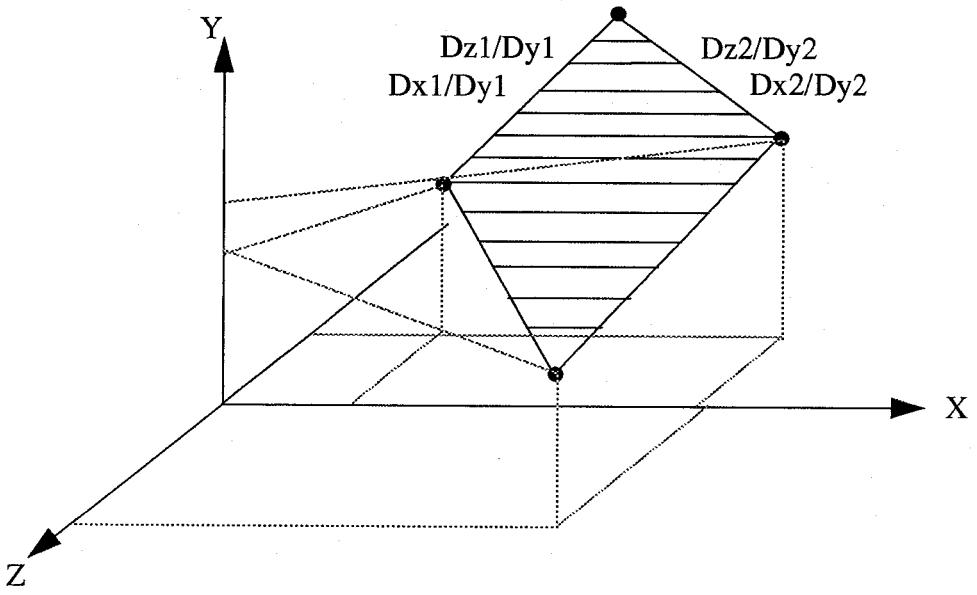


Figure 3.1 : Approximation bi-linéaire

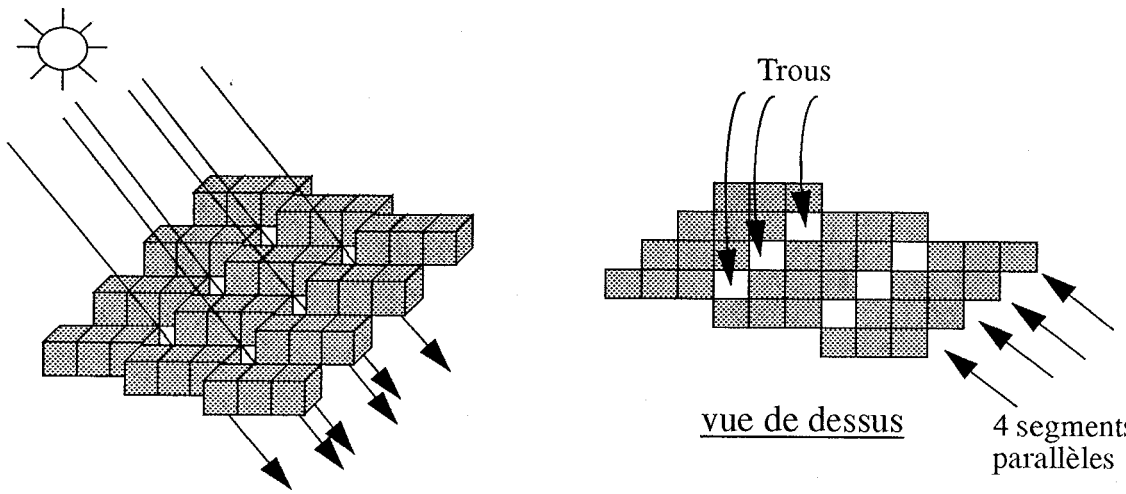
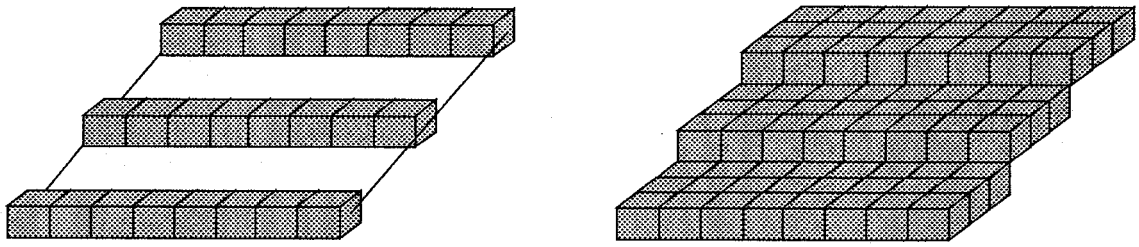


Figure 3.2 : Surfaces trouées

Le problème provient en partie de la discrétisation des segments. Deux segments parallèles discrétisés, contigus en un point peuvent ne plus être contigus en certains autres points, d'où l'apparition de trous.

Le problème provient également du choix de la direction de balayage utilisée dans l'interpolation bi-linéaire.



Selon Y puis selon X

Selon Z puis selon X

Figure 3.3 : directions de balayage

On peut penser que la méthode d'Amanatides & Woo permettrait de résoudre le problème des trous. En fait il n'en est rien, même si dans certains cas elle y remédie, comme le montre l'exemple suivant:

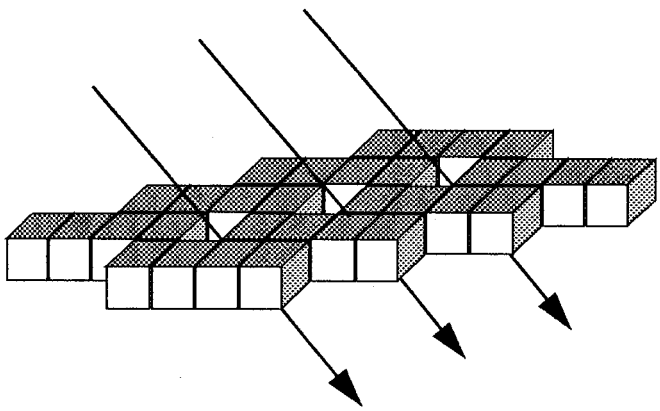


Figure 3.4 : Le problème des trous

3•3 Le découpage en tranches horizontales

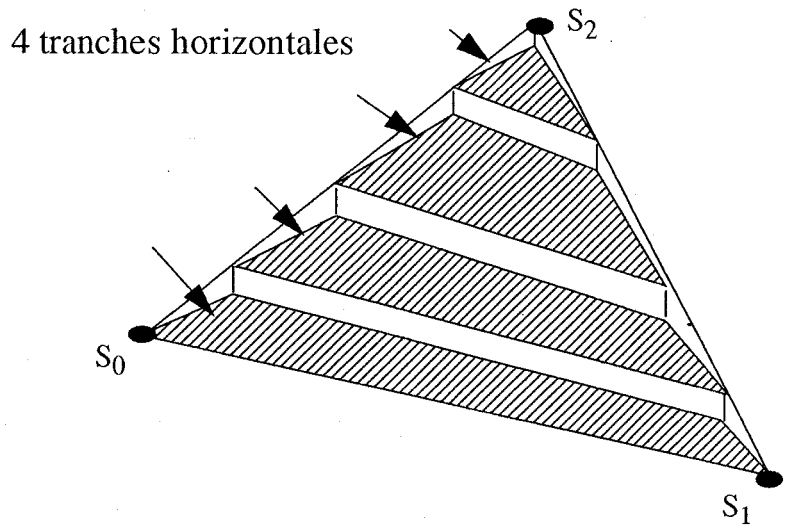
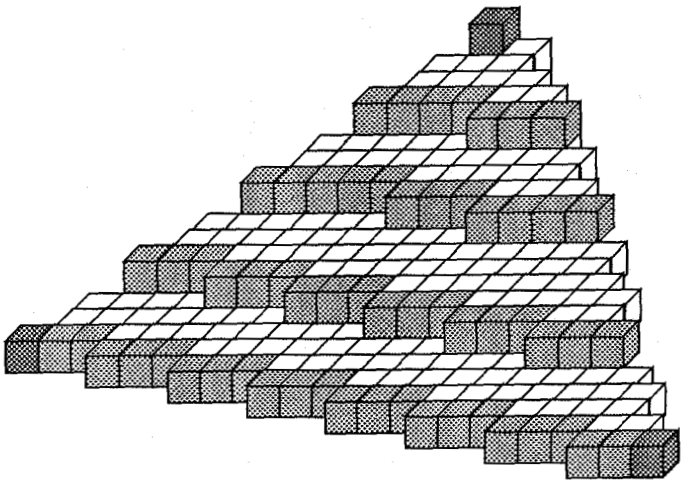


Figure 3.5 : Découpage en tranches d'une facette



Nous avons donc envisagé une première méthode qui consiste à découper la facette en tranches horizontales (Y=cste) formant des quadrilatères (ou triangles) horizontaux et d'utiliser un algorithme de remplissage 2D classique pour construire chacune de ces tranches. Nous supposerons la facette décrite par un tableau S[0..Nbs-1] contenant la liste des coordonnées de ses sommets donnés dans le sens trigonométrique.

On obtient 4 types de quadrilatères possibles selon les positions relatives des différents sommets entre eux:

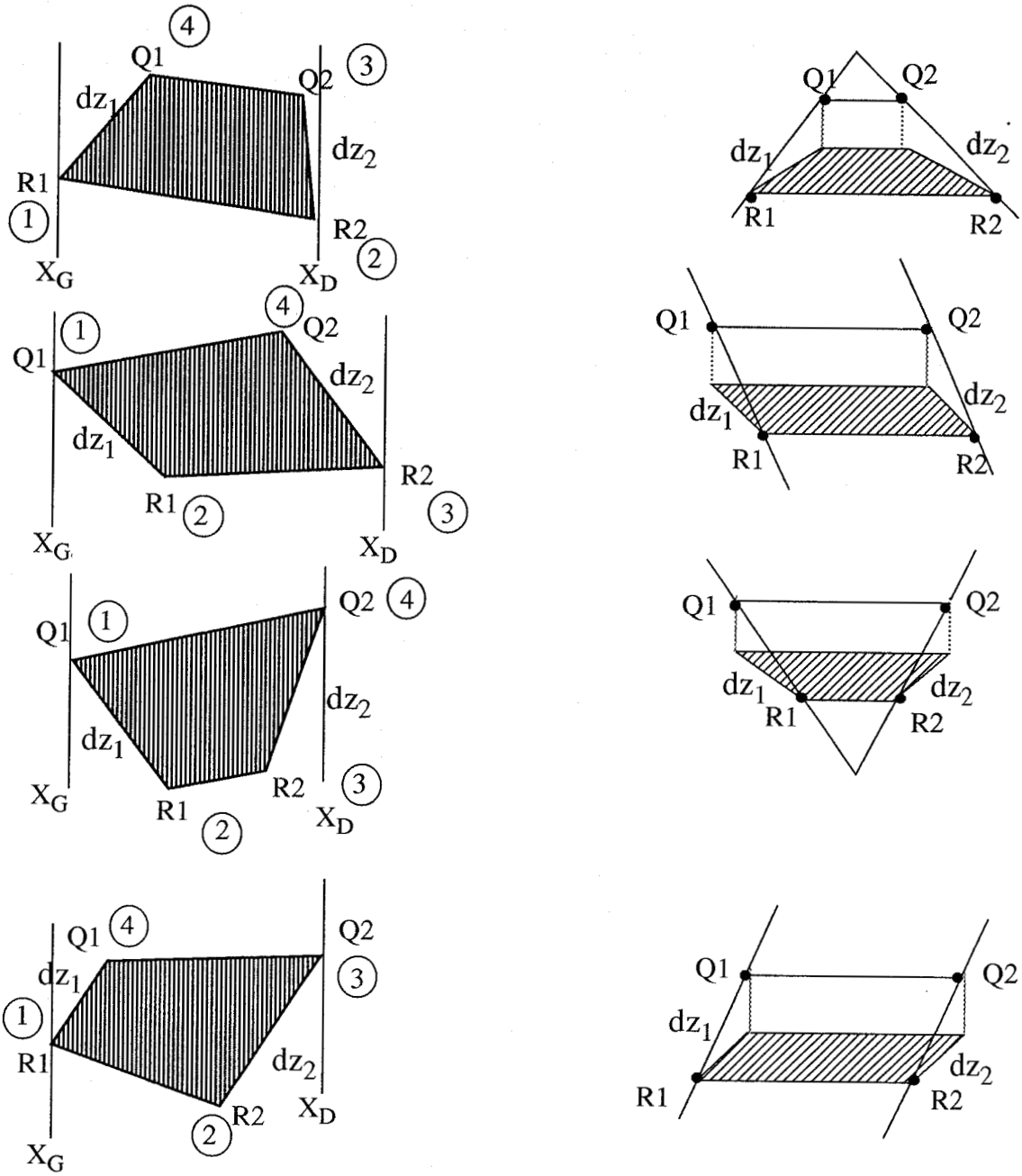


Figure 3.6 : Les différents types de quadrilatères

Algorithme de découpage en tranches horizontales.

```
/* Détermination de Ymin Ymax */
s1=0; Ymin=S[s1].Y;
Pour i de 1 à Nbs - 1
  Si ( S[i].Y > S[s1].Y ) Alors s1=i; Fsi
  Si ( S[i].Y < Ymin ) Alors Ymin=S[i].Y;Fsi
Fpour
s2=s1;Y=S[s1].Y;

/* Recherche sommets de départ s1 s2 */
Tant Que ( S[s1].Y=Y )
  X1=S[s1].X; Z1=S[s1].Z; s1=succ(s1);
FTQ
Tant Que ( S[s2].Y=Y )
  X2=S[s2].X; Z2=S[s2].Z; s2=pred(s2);
FTQ

/* Calcul des pentes initiales */
DX1 = (S[s1].X-X1) / (S[s1].Y-Y);
DZ1 = (S[s1].Z-Z1) / (S[s1].Y-Y);
DX2 = (S[s2].X-X2) / (S[s2].Y-Y);
DZ2 = (S[s2].Z-Z2) / (S[s2].Y-Y);

/* Initialise Tableau quadri. horiz. */
T[0].x=proche(X1); T[0].z=proche(Z1);
T[1].x=proche(X2); T[1].z=proche(Z2);
T[2].x=proche(X2); T[2].z=proche(Z2);
T[3].x=proche(X1); T[3].z=proche(Z1);
in=1; /* sens de remplissage du tableau */
FIN=FAUX;

/* Boucle principale de remplissage */
Tant Que ( Non FIN)
   /* Recherche Xmin, Xmax */
 m1=0; Xmax=T[m1].x;
 Pour i de 1 à 3
   Si (T[i].x<T[m1].x) Alors m1=i; Fsi
   Si (T[i].x>Xmax) Alors Xmax=T[i].x; Fsi
 Fpour
 x=T[m1].x; m2=m1;
 Si (x=Xmax)
   Alors
     z1=T[m1].z; z2=T[m2].z;
   Sinon /* Recherche points de départ m1 m2 */
     Tant Que (T[m1].x=x)
       z1=T[m1].z; m1=succ(m1,4);
     FTQ
     Tant Que (T[m2].x=x)
       z2=T[m2].z; m2=pred(m2,4);
     FTQ

/* boucle de remplissage tranche 2D */
Tant Que (x<=Xmax)
  /* remplir ligne Y=Cste, x=Cste */
  z=z1; incr=sgn(z2-z1);
  Nbp=abs(proche(z2-z1));
  Pour i de 0 à Nbs - 1
    put_voxel(x,Y,proche(z));
    z=z+incr;
  Fpour
  /* mise à jour segment suivant */
  Si ( (x=T[m1].x) et (x<Xmax) )
    Alors
      m1=succ(m1);
      pente1 = (T[m1].z-z1) / (T[m1].x-
  Fsi
  Si ( (x=T[m2].x) et (x<Xmax) )
    Alors
      m2=pred(m2);
      pente2 = (T[m2].z-z2) / (T[m2].x-
  Fsi
  z1=z1+pente1; z2=z2+pente2;
  x++;
FTQ

/* Calcul / recherche sommets quadri s
Si ( (Y=S[s1].Y) et (Y>Ymin) )
  Alors
    s1=succ(s1);
    DX1 = (S[s1].X-X1) / (S[s1].Y-Y);
    DZ1 = (S[s1].Z-Z1) / (S[s1].Y-Y);
Fsi
Si ( (Y=S[s2].Y) et (Y>Ymin) )
  Alors
    s2=pred(s2);
    DX2 = (S[s2].X-X2) / (S[s2].Y-Y);
    DZ2 = (S[s2].Z-Z2) / (S[s2].Y-Y);
Fsi
X1=X1-DX1;
Z1=Z1-DZ1;
X2=X2-DX2;
Z2=Z2-DZ2;
Y--;
Si (Y<Ymin) Alors FIN=Vrai; Fsi

/* Mise à jour Tableau Quadri Horiz */
Si (in=1)
  Alors i1=0; i2=1;
  sinon i1=3; i2=2;
Fsi
```

Cette méthode est beaucoup trop complexe et il n'est pas envisageable d'implanter un tel algorithme sur une machine cellulaire.

3•4 Le découpage en dexels

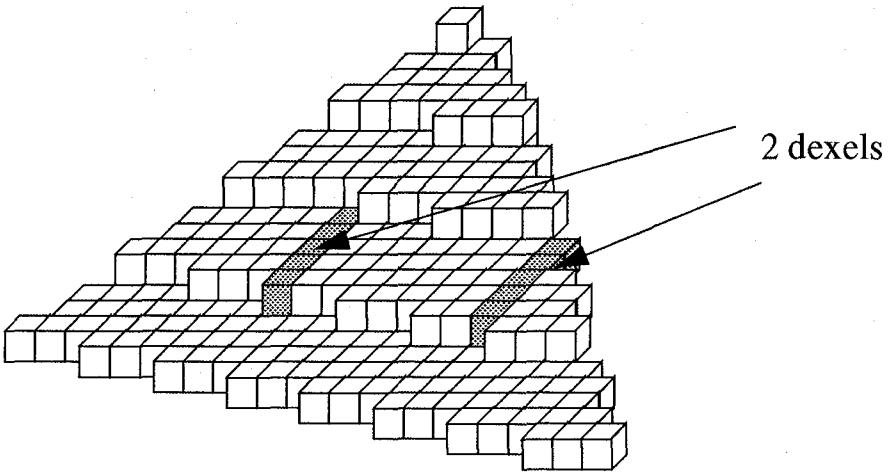


Figure 3.7 : Découpage en "dexels"

Nous nous sommes inspirés de la représentation d'une scène en dexels ("dexel" pour Depth Element [Van Hook 86]) et avons développé un algorithme de découpage de la facette. Celle-ci est décomposée en parallélépipèdes de taille unité selon les directions X et Y et de taille variable selon la direction Z ("Dexel" pour Depth Element [Van Hook 86]). Pour chaque couple de valeurs (x,y) donné, est fabriqué un ensemble de voxels qui approximent une partie de la facette (un "dexel"). Deux valeurs Z_{av} et Z_{ar} sont calculées qui déterminent un intervalle de voxels à remplir en Z. La façon de calculer ces valeurs doit être simple à réaliser et donc comporter très peu d'opérations. D'autre part, elle doit approcher le plus finement possible la facette, c'est à dire garantir l'absence de trous, sans pour autant générer des facettes trop grossières.

3•5 Discrétisation d'un plan

Rappel du problème: Etant donné un plan de l'espace 3D, discrétiser le plan en garantissant l'absence de trous à l'intérieur de la surface obtenue.

Définition:

Soit M un point de l'espace 3D, M(x,y,z) et Π un plan, on appellera **proximité** du point M par rapport au plan Π le réel suivant: $Prox(M/\Pi)=\min(\Delta x,\Delta y,\Delta z)$

avec $\Delta x=\min\{|\delta x|$ tq $(x+\delta x,y,z) \in \Pi\}$

$\Delta y=\min\{|\delta y|$ tq $(x,y+\delta y,z) \in \Pi\}$

$\Delta z=\min\{|\delta z|$ tq $(x,y,z+\delta z) \in \Pi\}$

On dira qu'un point M(x,y,z) est **collé** au plan Π si $Prox(M/\Pi)\leq\sigma$, σ étant un seuil donné qui détermine un écart par rapport au plan considéré ($\sigma\geq 0$).

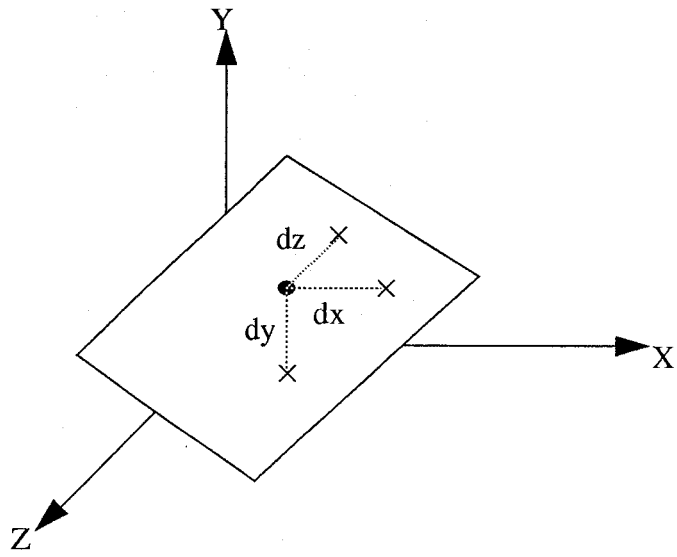


Figure 3.8 : Proximité d'un point par rapport à un plan

Conjecture: Un plan peut être discrétisé par l'ensemble des voxels qui lui sont collés, en prenant un seuil de 1/2. Cette discrétisation permet l'absence de trous à l'intérieur de la surface ainsi discrétisée.

Théorème: Soit Π un plan d'équation $Ax+By+Cz+D=0$ $(A,B,C)\in R^3$
Un point M(x,y,z) est collé au plan Π ssi $-K\sigma\leq(Ax+By+Cz+D)\leq K\sigma$
avec $K=\max(|A|,|B|,|C|)$

Démonstration: $M(x,y,z)$ collé à $\Pi <=> Prox(M/\Pi)\leq\sigma$

$<=> \Delta x\leq\sigma$ ou $\Delta y\leq\sigma$ ou $\Delta z\leq\sigma$

$<=>$ $\begin{cases} (1)\ \exists\ \delta x \in R \text{ tq } |\delta x|\leq\sigma \text{ et } (x+\delta x,y,z) \in \Pi \\ \text{ou} \\ (2)\ \exists\ \delta y \in R \text{ tq } |\delta y|\leq\sigma \text{ et } (x,y+\delta y,z) \in \Pi \\ \text{ou} \\ (3)\ \exists\ \delta z \in R \text{ tq } |\delta z|\leq\sigma \text{ et } (x,y,z+\delta z) \in \Pi \end{cases}$

(1) $<=>\exists\ x_1$ tq $(x_1,y,z)\in \Pi$ et $|x_1-x|\leq\sigma$
donc $Ax1+By+Cz+D=0$
d'où $Ax_1-Ax=-(Ax+By+Cz+D)$
$|Ax_1-Ax|=|Ax+By+Cz+D|$
$(1)<=>|A(x_1-x)|\leq|A|\sigma$
$(1)<=>-|A|\sigma\leq|A(x_1-x)|\leq|A|\sigma$
$(1)<=>-|A|\sigma\leq(Ax+By+Cz+D)\leq|A|\sigma$

On démontre de la même manière que
$(2)<=> -|B|\sigma\leq(Ax+By+Cz+D)\leq|B|\sigma$
$(3)<=> -|C|\sigma\leq(Ax+By+Cz+D)\leq|C|\sigma$
par symétrie des variables x,y,z.

Pour que l'une des 3 propriétés précédentes (1), (2), ou (3) soit vérifiée, il faut et il suffit que l'équation suivante soit vérifiée:
$-\sigma.\max(|A|,|B|,|C|)\leq(Ax+By+Cz+D)\leq\sigma.\max(|A|,|B|,|C|)$

donc $-K\sigma \leq (Ax+By+Cz+D) \leq K\sigma$ avec $K=\max(|A|,|B|,|C|)$.

Définition: Un dexel V de coordonnées (x,y) est un parallélépipède qui regroupe tous les voxels de l'espace 3D de coordonnées (x,y,z_i) avec z_i décrivant R. En pratique z_i variera entre 2 limites Z_{min} et Z_{max}.

Problème: Etant donnés un plan Π et un dexel V(x,y), déterminer
$H=\{z_i \in R$ tq $M(x,y,z_i)$ collé au plan $\Pi\}$.
Le plan Π est déterminé par les coefficients de son équation A,B,C,D.

$H=\{z_i \in R$ tq $-K\sigma \leq (Ax+By+Cz_i+D) \leq K\sigma\}$
Il faut et il suffit que le système suivant soit vérifié:

$$(I) \begin{cases} Ax+By+Cz_i+D \leq K\sigma \\ Ax+By+Cz_i+D \geq -K\sigma \end{cases} \Leftrightarrow \begin{cases} Cz_i \leq K\sigma - (Ax+By+D) \\ Cz_i \geq -K\sigma - (Ax+By+D) \end{cases}$$

1er cas: C>0

$$(I) \Leftrightarrow \begin{cases} z_i \leq \dfrac{K\sigma - (Ax+By+D)}{C} \\ z_i \geq \dfrac{-K\sigma - (Ax+By+D)}{C} \end{cases}$$

$$(I) \Leftrightarrow \begin{cases} z_i \leq Z_{AV} \\ z_i \geq Z_{AR} \end{cases} \quad \text{avec} \quad \begin{cases} Z_{AV} = \dfrac{K\sigma - (Ax+By+D)}{C} \\ Z_{AR} = \dfrac{-K\sigma - (Ax+By+D)}{C} \end{cases}$$

2nd cas: C<0

$$(I) \Leftrightarrow \begin{cases} z_i \geq \dfrac{K\sigma - (Ax+By+D)}{C} \\ z_i \leq \dfrac{-K\sigma - (Ax+By+D)}{C} \end{cases}$$

$$(I) \Leftrightarrow \begin{cases} z_i \geq Z_{AR} \\ z_i \leq Z_{AV} \end{cases} \quad \text{avec} \quad \begin{cases} Z_{AR} = \dfrac{K\sigma - (Ax+By+D)}{C} \\ Z_{AV} = \dfrac{-K\sigma - (Ax+By+D)}{C} \end{cases}$$

3ème cas: C=0
Pour un dexel donné, vérifier si $-K\sigma \leq (Ax+By+D) \leq K\sigma$ avec $K=\max(|A|,|B|)$.
Si la double inéquation est vérifiée pour le dexel (x,y) alors le plan Π coupe le dexel: $Z_{av}=+\infty$; $Z_{ar}=-\infty$
Si la double inéquation n'est pas vérifiée pour le dexel (x,y) alors le plan Π ne coupe pas le dexel: $Z_{av}=\perp$; $Z_{ar}=\perp$

3•6 Discrétisation d'une facette

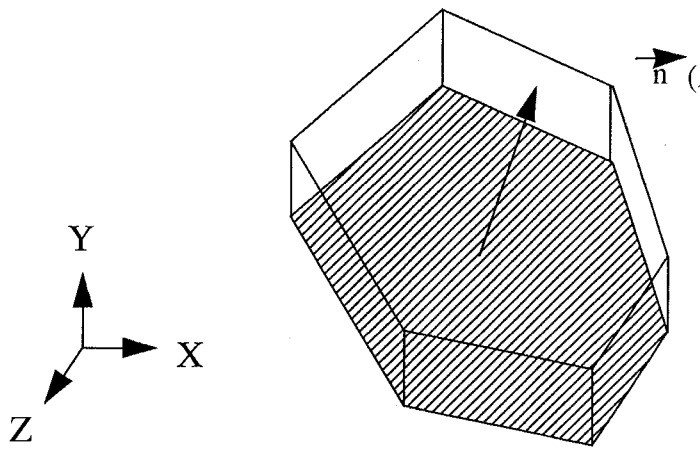
•Une facette peut être déterminée par l'équation du plan support, et des inéquations des différents bords formant le contour. La facette peut être représentée par un système d'inéquations:

$$\begin{cases} -K\sigma \le Ax + By + Cz + D \le K\sigma \\ A_1x + B_1y + C_1z + D_1 \ge 0 \\ A_2x + B_2y + C_2z + D_2 \ge 0 \\ \vdots \\ A_nx + B_ny + C_nz + D_n \ge 0 \end{cases}$$

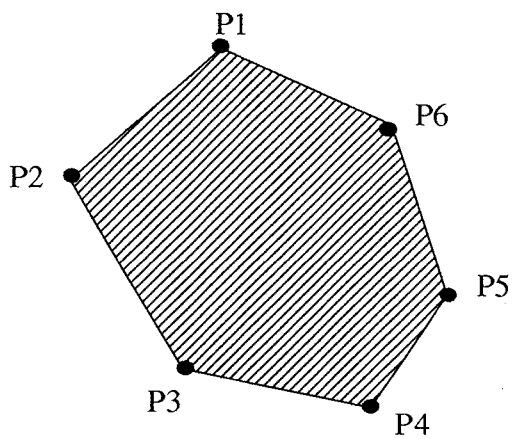
II

équation du plan support

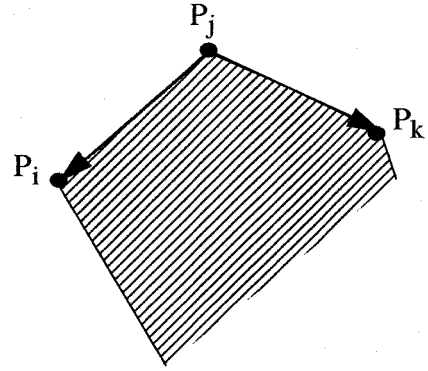
inéquations du contour définissant l'intérieur de la surface



•Facette déterminée par les coordonnées de ses sommets



1) Comment déterminer l'équation du plan support Π



$$\begin{bmatrix} (X-X_j) & (X_i-X_j) & (X_k-X_j) \\ (Y-Y_j) & (Y_i-Y_j) & (Y_k-Y_j) \\ (Z-Z_j) & (Z_i-Z_j) & (Z_k-Z_j) \end{bmatrix} = 0$$

$$(X-X_j)\begin{bmatrix} (Y_i-Y_j) & (Y_k-Y_j) \\ (Z_i-Z_j) & (Z_k-Z_j) \end{bmatrix} - (Y-Y_j)\begin{bmatrix} (X_i-X_j) & (X_k-X_j) \\ (Z_i-Z_j) & (Z_k-Z_j) \end{bmatrix} + (Z-Z_j)\begin{bmatrix} (X_i-X_j) & (X_k-X_j) \\ (Y_i-Y_j) & (Y_k-Y_j) \end{bmatrix} = 0$$

$$Ax + By + Cz - Ax_j - By_j - Cz_j = 0$$

$$A = \begin{bmatrix} (Y_i-Y_j) & (Y_k-Y_j) \\ (Z_i-Z_j) & (Z_k-Z_j) \end{bmatrix} = (Y_i-Y_j)\,(Z_k-Z_j) - (Y_k-Y_j)\,(Z_i-Z_j)$$

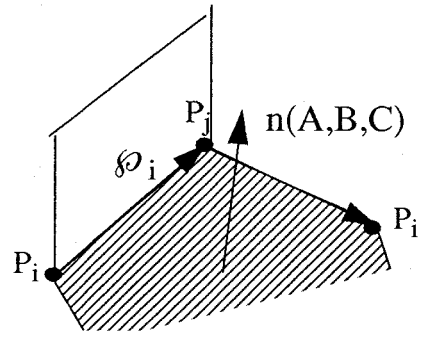
$$B = \begin{bmatrix} (X_i-X_j) & (X_k-X_j) \\ (Z_i-Z_j) & (Z_k-Z_j) \end{bmatrix} = -\ (X_i-X_j)\ (Z_k-Z_j) + (X_k-X_j)\,(Z_i-Z_j)$$

$$C = \begin{bmatrix} (X_i-X_j) & (X_k-X_j) \\ (Y_i-Y_j) & (Y_k-Y_j) \end{bmatrix} = (X_i-X_j)\ (Y_k-Y_j) - (X_k-X_j)\,(Y_i-Y_j)$$

$$D = -AX_j - BY_j - CZ_j$$

2) Calcul des inéquations de contour

La facette est délimitée par des plans $\wp_i$ perpendiculaires au plan support qui contiennent des segments de droite formés par 2 sommets consécutifs.



Le plan $\wp_i$ recherché est engendré par les vecteurs n et P_iP_j
Equation de $\wp_i$:

$$\begin{bmatrix} (X-X_i) & A & (X_j-X_i) \\ (Y-Y_i) & B & (Y_j-Y_i) \\ (Z-Z_i) & C & (Z_j-Z_i) \end{bmatrix} = 0$$

$$(X-X_i)\begin{bmatrix} B & (Y_j-Y_i) \\ C & (Z_j-Z_i) \end{bmatrix} - (Y-Y_i)\begin{bmatrix} A & (X_j-X_i) \\ C & (Z_j-Z_i) \end{bmatrix} + (Z-Z_i)\begin{bmatrix} A & (X_j-X_i) \\ B & (Y_j-Y_i) \end{bmatrix} = 0$$

$$\alpha X + \beta Y + \gamma Z - \alpha X_i - \beta Y_i - \gamma Z_i = 0$$

$$\alpha X + \beta Y + \gamma Z + \delta = 0$$

$$\alpha = \begin{bmatrix} B & (Y_j-Y_i) \\ C & (Z_j-Z_i) \end{bmatrix} = B(Z_j-Z_i) - C(Y_j-Y_i)$$

$$\beta = -\begin{bmatrix} A & (X_j-X_i) \\ C & (Z_j-Z_i) \end{bmatrix} = -A(Z_j-Z_i) + C(X_j-X_i)$$

$$\gamma = \begin{bmatrix} A & (X_j-X_i) \\ B & (Y_j-Y_i) \end{bmatrix} = A(Y_j-Y_i) - B(X_j-X_j)$$

$$\delta = -\alpha X_i - \beta Y_i - \gamma Z_i$$

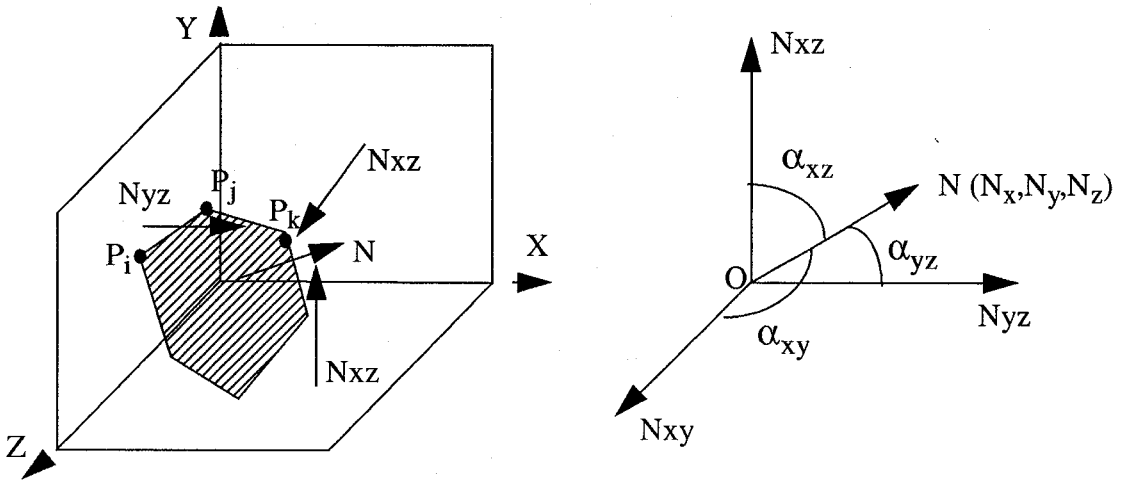
Détermination de l'inéquation délimitant le contour
Soit P_k un sommet de la facette $P_k \neq P_i$ et $P_k \neq P_j$
Si $\alpha X_k+\beta Y_K+\gamma Z_k+\delta \geq 0$
alors $A_i=\alpha$; $B_i=\beta$; $C_i=\gamma$; $D_i=\delta$
Si $\alpha X_k+\beta Y_K+\gamma Z_k+\delta < 0$
alors $A_i=-\alpha$; $B_i=-\beta$; $C_i=-\gamma$; $D_i=-\delta$
On obtient dans tous les cas une inéquation de la forme
$A_iX+B_iY+C_iZ+D_i \geq 0$

3) Simplification:

Dans le calcul des inéquations de contour de la facette, plutôt que d'utiliser les équations des plans $\wp_i$, il serait intéressant d'utiliser la projection de la facette sur l'un des plans de bases (XOY, XOZ, YOZ) et déterminer les équations des segments limitant la projection de la facette. De cette façon, dans chaque inéquation finale du système II, l'un au moins des coefficients A_i, B_i, ou C_i sera nul.

Pour avoir des surfaces sans trous sur les bords, il faut projeter les bords sur le plan de de base (XOY, XOZ, YOZ) dont la normale fait un angle minimal avec la normale de la facette.

Pour chaque bord de la facette, on récupère une inéquation de la forme
$A'_i X + B'_i Y + C'_i Z + D'_i \geq 0$ avec $A'_i=0$ ou $B'_i=0$ ou $C'_i=0$



$\alpha=\min(\alpha_{xy},\alpha_{xz},\alpha_{yz})$
$m=\max(|N_x|,|N_y|,|N_z|)$
Si $\alpha=\alpha_{xz}$ (si $m=|N_y|$)
alors projection de la facette sur XOZ
inéquation $A_iX+C_iZ+D_i \geq 0$
Si $\alpha=\alpha_{xy}$(si $m=|N_z|$)
alors projection de la facette sur XOY
inéquation $A_iX+B_iY+D_i \geq 0$
Si $\alpha=\alpha_{yz}$ (si $m=|N_x|$)
alors projection de la facette sur YOZ
inéquation $B_iY+C_iZ+D_i \geq 0$

- Projection sur XOZ

$$\begin{bmatrix} (X-X_i) & (X_j-X_i) \\ (Z-Z_i) & (Z_j-Z_i) \end{bmatrix} = 0$$

$$(X-X_i)\,(Z_j-Z_i) - (Z-Z_i)\,(X_j-X_i) = 0$$

$$A_iX + C_iZ - (A_iX_i + C_iZ_i) = 0$$

$$A_i = Z_j-Z_i \qquad B_i = 0 \qquad C_i = -(X_j-X_i) \qquad D_i = -X_iZ_j+Z_iX_j$$

- Projection sur XOY

$$\begin{bmatrix} (X-X_i) & (X_j-X_i) \\ (Y-Y_i) & (Y_j-Y_i) \end{bmatrix} = 0$$

$$(X-X_i)\,(Y_j-Y_i) - (Y-Y_i)\,(X_j-X_i) = 0$$

$$A_iX + B_iY - (A_iX_i + B_iY_i) = 0$$

$$A_i = Y_j - Y_i \qquad B_i = -(X_j-X_i) \qquad C_i = 0 \qquad D_i = -X_iY_j + Y_iX_j$$

- projection sur YOZ

$$\begin{bmatrix} (Y-Y_i) & (Y_j-Y_i) \\ (Z-Z_i) & (Z_j-Z_i) \end{bmatrix} = 0$$

$$(Y-Y_i)\,(Z_j-Z_i) - (Z-Z_i)\,(Y_j-Y_i) = 0$$

$$B_iY + C_iZ - (B_iY_i + C_iZ_i) = 0$$

$$A_i = 0 \qquad B_i = Z_j - Z_i \qquad C_i = -(Y_j-Y_i) \qquad D_i = -Y_iZ_j + Y_jZ_i$$

- Normalisation de l'inéquation

Si $A_iX_k+B_iY_k+C_iZ_k+D_i<0$
alors $A_i=-A_i$
$B_i=-B_i$
$C_i=-C_i$
$D_i=-D_i$
Fsi

Pour chacun des segments du contour, est déterminée une équation de la forme AX+BY+CZ+D=0 (avec l'un au moins des coefficients A,B,C nul).
Soit P_k (X_k,Y_k,Z_k) la projection d'un 3e sommet de la facette, différent des 2 premiers (P_i,P_j), avec l'un au moins des X_k,Y_k,Z_k nul, le signe de $AX_k+BY_k+CZ_k+D=R$ est calculé.
Si $R\geq 0$, les coefficients sont laissés tels quels; l'équation devient une inéquation

$$AX+BY+CZ+D\geq 0$$

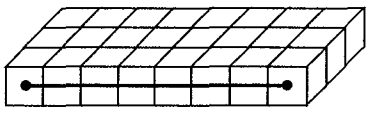
Si $R<0$, les coefficients (a,b,c,d) sont changés en leurs opposés; l'équation devient une inéquation : $-AX-BY-CZ-D\geq 0$
Le processus est réitéré pour chacun des côtés suivants
Un point M est donc à l'intérieur de la facette ssi
1° M appartient au plan support Π discrétisé
2° Projection(M) est à l'intérieur de la projection de la facette

3•7 Algorithme séquentiel de découpage en dexels

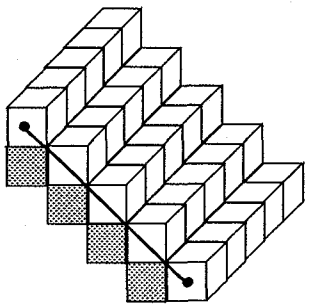
```
Début
  K=max(|A|,|B|,|C|)
  pour Y de Ymin à Ymax
    pour X de Xmin à Xmax
      inter=Vrai
      T=Ax+By+D
      Si C=0
        alors
          Si (T>Kσ ou T<-Kσ)
            alors
            inter=Faux
            sinon
            Zav=+∞
            Zar=-∞
          Fsi
        sinon
          Zav=(Kσ-T)/C
          Zar=(-Kσ-T)/C
          Si (C<0)
            alors échanger(Zav,Zar)
          Fsi
      Fsi
      i=1
      Tant Que (Inter=Vrai et i≤Nb_cotés)
        i=i+1
        Ti=AiX+BiY+Di
        Si (Ci=0 et Ti<0)
          alors  inter=Faux
        Fsi
        Si (Ci>0)
          alors Zar=max(Zar,Zar,-Ti/Ci)
        Fsi
        Si (Ci<0)
          alors Zav=min(Zav,-Ti/Ci)
        Fsi
      FTQ
    Fpour
  Fpour
Fin
```

3•8 Création de surfaces blindées

Les études sur la topologie ont montré que pour éviter qu'un rayon discrétisé traverse une surface discrétisée par effet "passe-muraille", il fallait que les rayons ou les surfaces soient construits en connectivité 6 voisins.
Nous avons choisi la connectivité 6 pour les surfaces car ceci nous laisse la possibilité d'utiliser une connectivité quelconque pour les rayons et ainsi de choisir parmi les algorithmes de suivi, celui qui nous semble le mieux adapté.
Nous allons donc modifier le seuil σ dans l'algorithme de discrétisation de facettes de façon à créer des surfaces "blindées" (de connectivité 6) empêchant l'effet passe-muraille de se produire.
Pour fabriquer des surfaces sans trous, le seuil de 1/2 est suffisant et cette constante est indépendante de la surface à tracer. Par contre, pour fabriquer des surfaces blindées, une étude empirique nous a montré qu'il fallait augmenter le seuil σ. Celui-ci dépend alors de la surface à discrétiser car, comme le montrent les figures suivantes, il ne peut être le même quelque soit la pente.
σ varie entre 1/2 et 1.

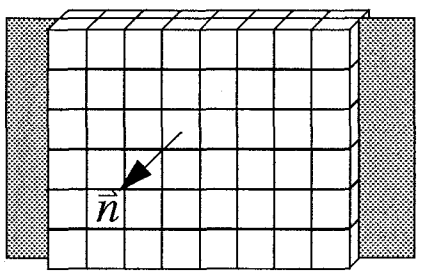


$$0 \leq \sigma < \frac{1}{2}$$



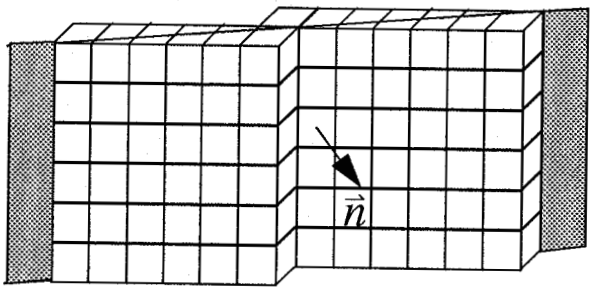
$$1 \leq \sigma$$

A=0
B=0



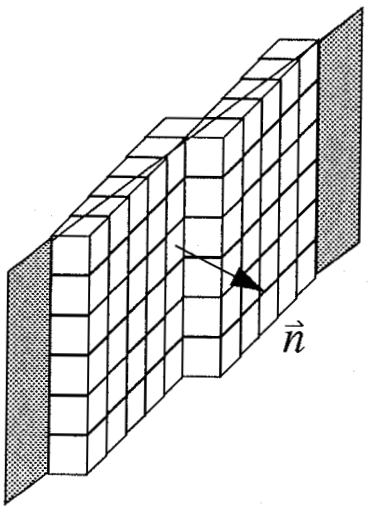
$$\sigma = \frac{1}{2}$$

B=0
|C|>|A|>0



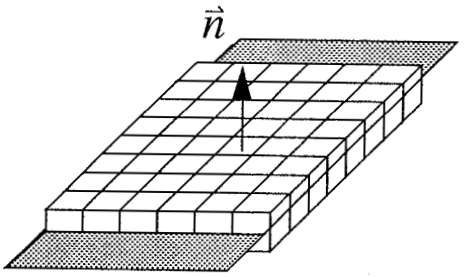
$$\sigma = \frac{1}{2} + \frac{|A|}{2|C|}$$

B=0
|A|>|C|>0



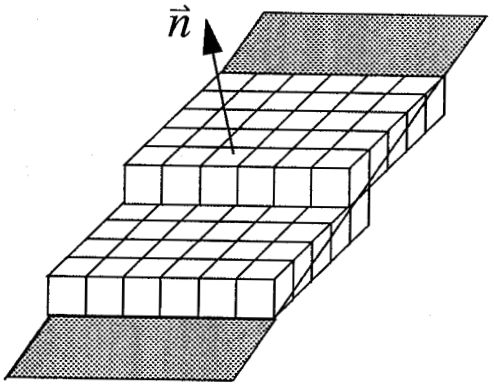
$$\sigma = \frac{1}{2} + \frac{|C|}{2|A|}$$

A=0
C=0



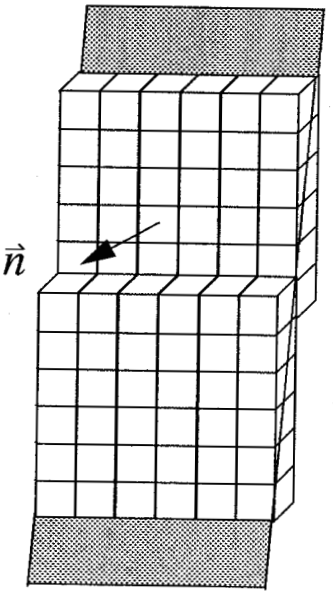
$$\sigma = \frac{1}{2}$$

A=0
|B|>|C|>0



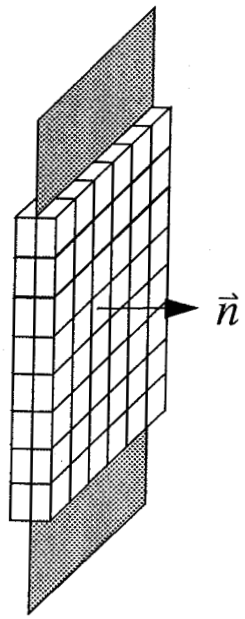
$$\sigma = \frac{1}{2} + \frac{|C|}{2|B|}$$

A=0
|C|>|B|>0



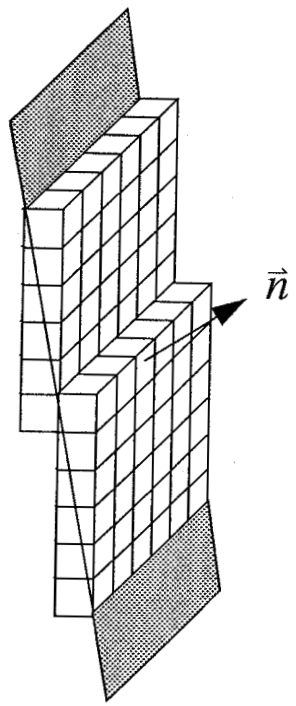
$$\sigma = \frac{1}{2} + \frac{|B|}{2|C|}$$

B=0
C=0

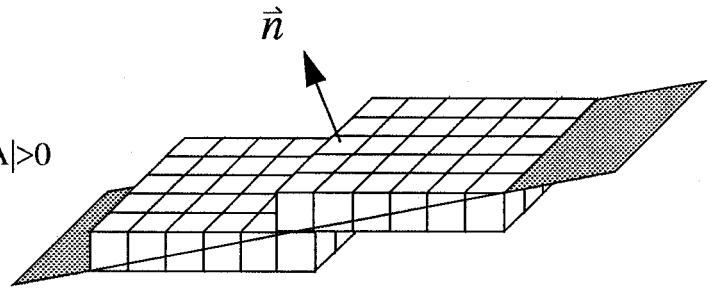


$$\sigma = \frac{1}{2}$$

C=0
|A|>|B|>0



$$\sigma = \frac{1}{2} + \frac{|B|}{2|A|}$$



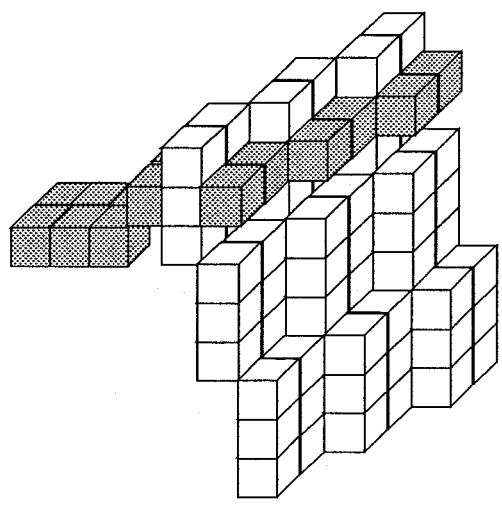
$$\sigma = \frac{1}{2} + \frac{|A|}{2|B|}$$

Un seuil $\sigma>1$ entraîne la création de surfaces trop grossières. Pour fabriquer des surfaces blindées, il faut trouver une valeur convenable de seuil comprise entre 1/2 et 1. Cette valeur doit nécessairement dépendre de la surface à tracer et nous proposons le seuil σ suivant:

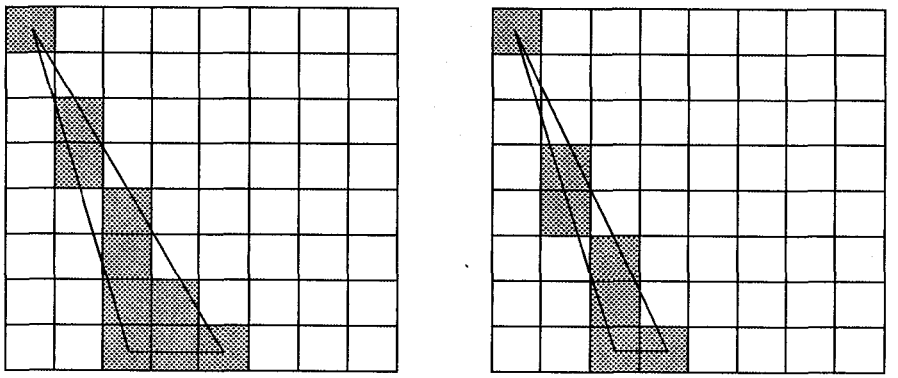
$$\sigma = max(\sigma_1, \sigma_2, \sigma_3)$$

$$\begin{cases} \sigma_1 = \frac{1}{2}\left(1 + \frac{min(|A|,|B|)}{max(|A|,|B|)}\right) \text{si } (A,B) \neq (0,0) \text{ sinon } \sigma_1 = \frac{1}{2} \\ \sigma_2 = \frac{1}{2}\left(1 + \frac{min(|A|,|C|)}{max(|A|,|C|)}\right) \text{si } (A,C) \neq (0,0) \text{ sinon } \sigma_2 = \frac{1}{2} \\ \sigma_3 = \frac{1}{2}\left(1 + \frac{min(|B|,|C|)}{max(|B|,|C|)}\right) \text{si } (B,C) \neq (0,0) \text{ sinon } \sigma_3 = \frac{1}{2} \end{cases}$$

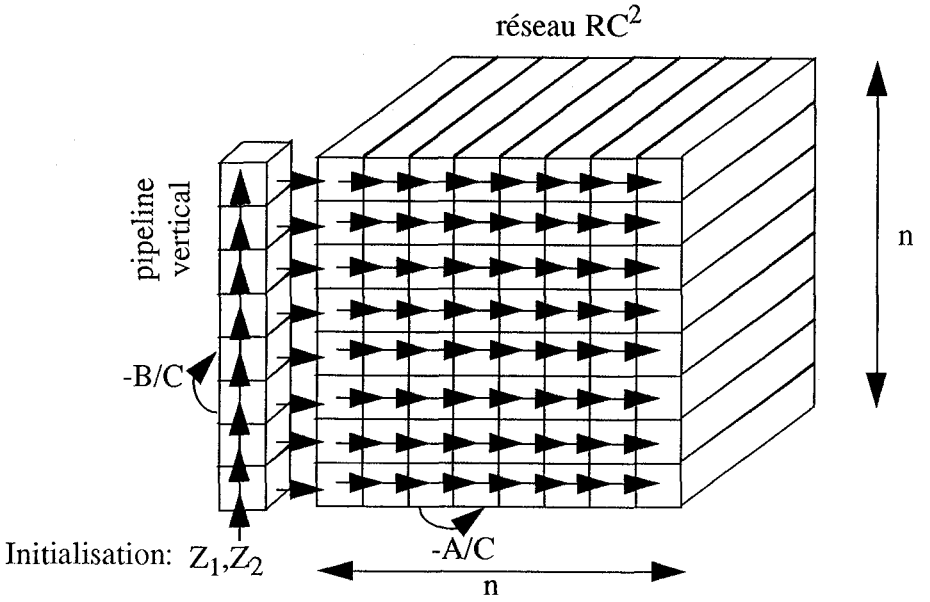
L'algorithme donne des résultats corrects. On ne peut toutefois pas affirmer que l'on obtient la surface de connectivité 6 minimale, car il existe des voxels appartenant à cette surface tel que, si on les enlève, la surface reste 6-connexe. Cependant quelques problèmes subsistent. A l'endroit où deux facettes se rejoignent. l'intersection peut ne pas être juste et les deux facettes ont tendance à se chevaucher.



Un second problème concerne les facettes fines car dans ce cas, le fait de couper la facette par des inéquations de contour peut générer des trous à l'intérieur de celle-ci.



3•9 Perspectives d'application sur architecture cellulaire RC^2.



Dans la phase de discrétisation de la scène, chaque cellule (x,y) du réseau cellulaire RC^2 doit calculer deux valeurs: Z_1 et Z_2. Ces valeurs sont calculées de manière incrémentale. Une cellule ayant reçu (Z_1, Z_2), elle détermine deux nouvelles valeurs Z_1=-A/C et Z_2=-A/C, puis transmet ces 2 nouvelles valeurs à sa voisine de droite.
Un pipeline vertical, fonctionnant lui aussi de manière incrémentale, permet d'initialiser les valeurs de Z_1, Z_2 pour chacune des lignes du réseau RC^2.
La partie initialisation est effectuée par le calculateur hôte qui envoie les bonnes valeurs au pipe vertical, et ceci pour chacune des facettes constituant la scène.

$$Z_1 = \frac{K\sigma - (Ax+By+D)}{C} \qquad Z_2 = \frac{-K\sigma - (Ax+By+D)}{C}$$

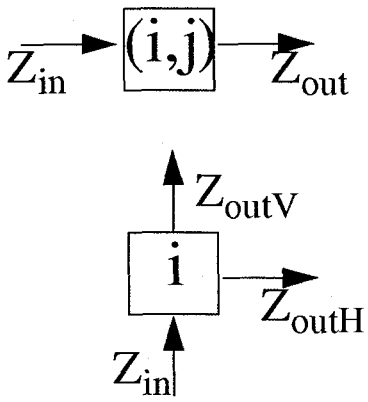
$$(x = x+1) \Rightarrow Z_1 = Z_1 - \frac{A}{C} \qquad (y = y+1) \Rightarrow Z_1 = Z_1 - \frac{B}{C}$$

$(x = x+1) \Rightarrow Z_2 = Z_2 - \frac{A}{C}$ $\qquad$ $(y = y+1) \Rightarrow Z_2 = Z_2 - \frac{B}{C}$

$$(x = y = 0) \Rightarrow Z_1 = \frac{K\sigma - D}{C}, Z_2 = \frac{-K\sigma - D}{C}$$

De toute évidence, il apparait que les calculs effectués par chaque cellule sont des calculs en rationnel.
Il serait intéressant de normaliser les équations de façon à se ramener à des coefficients entiers. L'avantage obtenu est le calcul en entiers pour chacune des cellules $A'x+B'y+C'z+D'=0$ $(A',B',C',D') \in Z^*$.
Nous ne pouvons donc plus avoir des plans supports quelconques, mais de toutes manières il n'existe qu'un nombre fini de plans discrétisés dans un espace 3D borné. Le problème se ramène donc à déterminer une équation de la forme précédente de telle sorte que le plan discrétisé à partir de cette équation soit une 'bonne' approximation du plan initial.

Pour travailler avec des valeurs entières, les valeurs passées sont $T=CZ_1$ et $U=CZ_2$



Chaque cellule (i,j) du réseau reçoit des valeurs Z_{in} et renvoie, après calcul, les nouvelles valeurs Z_{out} à sa voisine.

Chaque cellule i du pipeline vertical reçoit des valeurs Z_{in}. Elle réémet, après calcul, les nouvelles valeurs Z_{outV} à destination de la cellule suivante du pipeline et Z_{outH} vers la première cellule du réseau cellulaire.

Algorithme effectué par les différentes cellules du réseau RC^2

Cellule (i,j)

```
/* recevoir et transmettre */
Recevoir(A); Transmettre(A)
Recevoir(C); Transmettre(C)
Recevoir(T); Transmettre(T-A)
Recevoir(U); Transmettre(U-A)
Recevoir(Nb_cotés); Transmettre(Nb_cotés)
Pour i de 1 à Nb_cotés
  Recevoir(Ai); Transmettre(Ai)
  Recevoir(Ci); Transmettre(Ci)
  Recevoir(Ti); Transmettre(Ti-Ai)
Fpour
/* Calcul intersection */
INTER=Vrai
Si C=0
  alors Si T≥0 et U≤0
    alors Zav=+∞; Zar=-∞
    sinon INTER=Faux
  Fsi
  sinon
    Zav=T/C; Zar=U/C
    Si C<0 echanger (Zav,Zar) Fsi
Fsi
i=1
Tant Que (INTER=Vrai) et (i<=Nb_cotés)
  si Ci=0
    si Ti<0
      INTER=Faux
    Fsi
  Fsi
  Si Ci>0
    Zar=max(Zar,-Ti)
  Fsi
  Si Ci<0
    Zav=min(Zav,-Ti)
  Fsi
FTQ
/* partie affichage */
si inter alors
  pour Z de Zar à Zav
    Afficher_Voxel(i,j,Z)
  Fpour
Fsi
```

Cellule j (pipeline vertical)

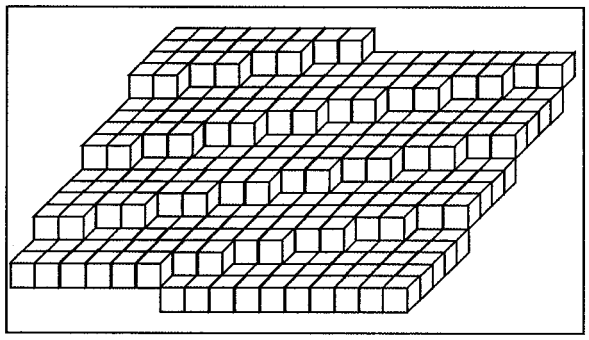
```
Recevoir(A)
Trans_Horiz(A); Trans_Vert(A)
Recevoir(B)
Trans_Vert(B)
Recevoir(C)
Trans_Horiz(C); Trans_Vert(C)
Recevoir(T)
Trans_Horiz(T); Trans_vert(T-B)
Recevoir(U)
Trans_Horiz(U); Trans_vert(U-B)
Recevoir(Nb_cotés)
Trans_Horiz(Nb_cotés); Trans_Vert(Nb_cotés)
Pour i de 1 à Nb_cotés
  Recevoir(Ai)
  Trans_Horiz(Ai); Trans_Vert(Ai)
  Recevoir(Bi)
  Trans_Vert(Bi)
  Recevoir(Ci)
  Trans_Horiz(Ci); Trans_Vert(Ci)
  Recevoir(Ti)
  Trans_Horiz(Ti)
  Trans_Vert(Ti-Bi)
Fpour
```

Initialisation effectuée par le hôte

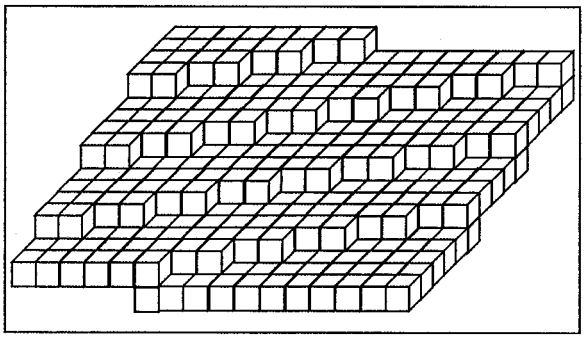
```
K=max(|A|,|B|,|C|); T=Kσ-D; U=-Kσ-D
envoyer (A)
envoyer (B)
envoyer (C)
envoyer (T)
envoyer (U)
envoyer (Nb_cotés)
pour i de 1 à Nb_cotés
  envoyer (Ai)
  envoyer (Bi)
  envoyer (Ci)
  envoyer (-Di)
Fpour
```

DISCRETISATION D'UNE FACETTE

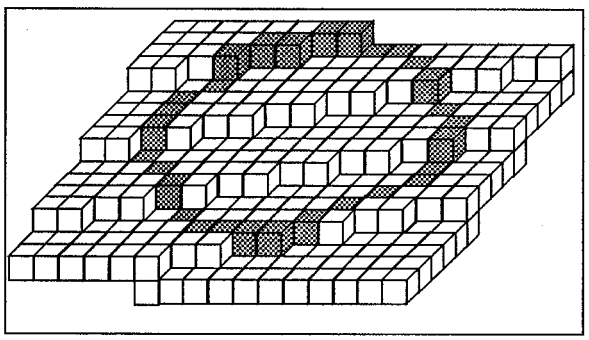
Phase 1 : Discrétisation du plan support de la facette



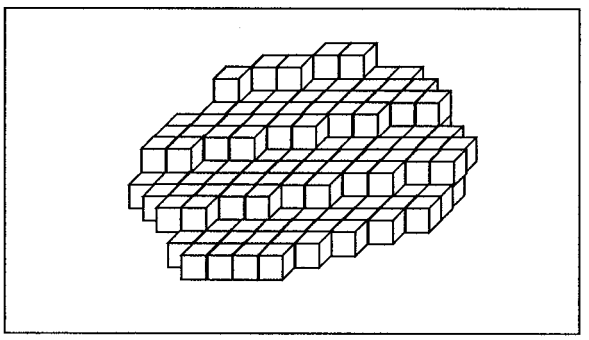
Seuil =1/2



surface blindée



Phase 2 : limitation à l'aide du contour



Résultat : Facette discrétisée

CHAPITRE 4

Diffusion de rayons

TABLE DES MATIERES

CHAPITRE 4 : Diffusion de rayons

4.1 Position du problème

4•1•1 Calcul des rayons d'ombre

L'algorithme de lancer de rayon classique nécessite le calcul, en tout point atteint par un rayon, de l'éclairement correspondant. Pour cela, un rayon est envoyé vers chacune des sources lumineuses pour tester si celle-ci éclaire ou non l'objet. Nous allons étudier dans ce chapitre différentes stratégies pour calculer les rayons d'ombre et ainsi déterminer l'éclairement en un point.

4•1•2 Discrétisation de la scène en voxels

La décomposition de la scène en voxels simplifie énormément le calcul d'intersection puisqu'elle se résume à un test de présence/absence d'objet. L'éclairage principal de chaque voxel est calculé par la méthode du lancer de rayons classique. Pour chacun de ces voxels, un rayon d'ombre est envoyé vers chaque source en utilisant un suivi de rayon incrémental (Fig 4.1 et 4.2). Lorsque le rayon coupe un voxel occupé, une information est retournée au voxel émetteur lui indiquant que la source lumineuse n'est pas visible (voxel dans l'ombre pour cette source là). Si le rayon coupe un voxel vide il poursuit sa route, s'il atteint la source lumineuse il retourne l'indication "éclairé" au voxel initial. Pour un voxel donné, N_{src} rayons sont générés, un pour chacune des Nsrc sources de lumière, et Nsrc rayons retour sont renvoyés.

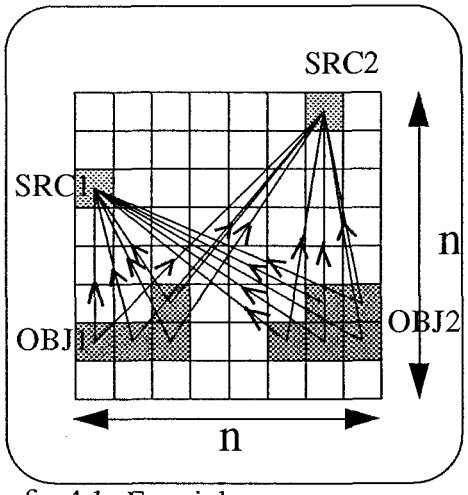


fig 4.1: Envoi de rayons Objets -> Sources en 2D

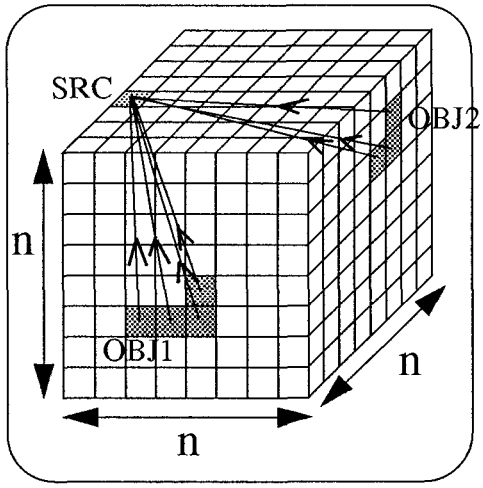


fig 4.2: Envoi de rayons Objets -> Sources en 3D

4•1•3 Fonctionnement du réseau dans la phase de rendu

Chaque cellule du réseau contenant des voxels occupés et atteints par des rayons primaires ou secondaires envoie, pour chacun de ces voxels, un rayon vers chacune des sources lumineuses. Les rayons sont acheminés dans le réseau par un algorithme de suivi de rayons. Chaque cellule recevant un rayon lumineux vérifie si ce rayon coupe un des voxels gérés par elle. Dans RC^2, il suffit de comparer la profondeur en Z du rayon reçu avec le contenu du voxel de même Z. S'il y a intersection, le rayon doit être renvoyé à la cellule émettrice avec l'indication d'éclairement suivante:

- "A l'ombre" si le voxel correspond à un objet
- "Eclairé" si ce voxel correspond à une source lumineuse

Dans le cas où il n'y a pas d'intersection, la cellule doit transmettre le rayon à la cellule suivante en calculant, à l'aide de l'algorithme de suivi de rayons, le voxel suivant atteint par ce rayon.

Dans la phase d'émission, les rayons doivent être acheminés en ligne droite afin de modéliser le comportement de la lumière, mais une fois l'intersection déterminée, les rayons "retour" peuvent être acheminés de manière quelconque et tenir compte éventuellement de la charge du réseau. Les messages contiennent la position de la cellule destinatrice et l'indication d'éclairement. Les messages transitent alors par le routeur intégré à chaque cellule sans passer par l'unité de calcul.

L'inconvénient de cette méthode est la convergence de tous les rayons lancés vers les sources lumineuses. Pour notre architecture, ceci génère un flot de messages important aux abords de la cellule contenant la source lumineuse et peut entraîner un blocage par saturation des cellules voisines de cette source.

4•1•4 Diffusion à partir des sources de lumière

Afin de limiter le goulot d'étranglement créé par les messages aux abords de la source, nous avons envisagé une autre méthode qui consiste à émettre des rayons à partir des sources de lumières et à diffuser ces rayons dans tout l'espace (Fig 4.3 et 4.4). Cette approche reproduit plus fidélement le comportement observé dans la nature où les photons sont émis depuis les sources. Pour chaque source lumineuse, des rayons sont envoyés à destination de tous les voxels de l'espace. Un rayon qui coupe un voxel vide poursuit son chemin, un rayon qui rencontre un voxel occupé est tout simplement stoppé. L'éclairage principal en ce point est donc mis à jour à l'aide des informations reçues.

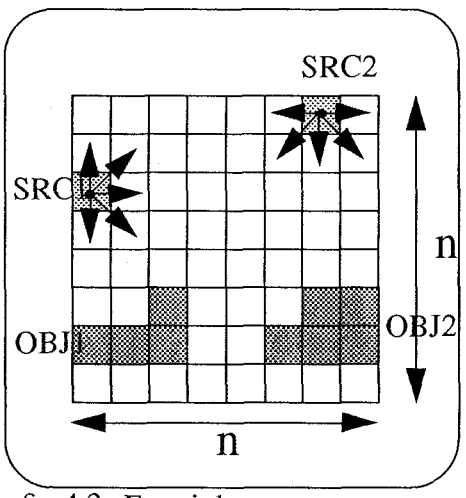


fig 4.3: Envoi de rayons Sources -> Objets en 2D

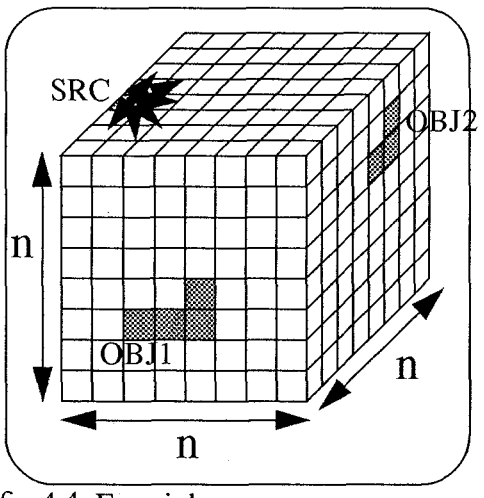


fig 4.4: Envoi de rayons Sources -> Objets en 3D

L'inconvénient de cette méthode est de générer un nombre important de rayons ($O(n^2)$ rayons en 2D, $O(n^3)$ rayons en 3D). Pour limiter ce nombre, il est possible de mémoriser la position des objets et n'envoyer que les rayons qui couperont effectivement ces objets. Il faut mémoriser une grande quantité d'information, ce qui est inenvisageable vu le nombre d'objets qui constituent généralement une scène. Pour réduire encore le nombre de rayons et remédier à l'inconvénient de la méthode précédente, nous pouvons nous

limiter à n'envoyer qu'un seul rayon pour chaque cellule appartenant à l'enveloppe de la scène (Fig 4.5 et 4.6). En 2D il suffira d'envoyer un rayon pour chaque cellule de bord, et en 3D un rayon pour chaque cellule à la surface du cube englobant. Le nombre de rayons sera donc 4n en 2D et $6n^2$ en 3D et ce nombre est indépendant du nombre d'objets constituant la scène.

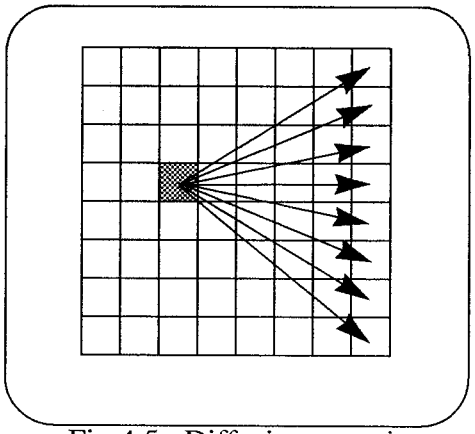


Fig 4.5 : Diffusion restreinte 2D

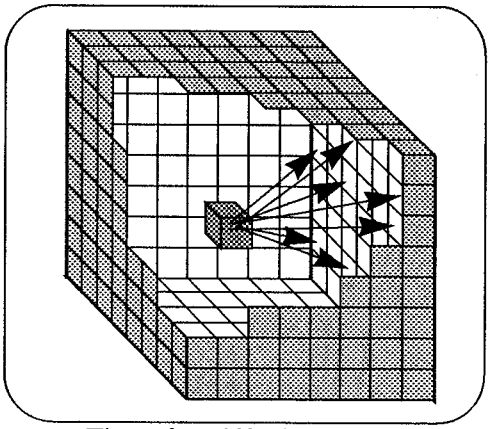
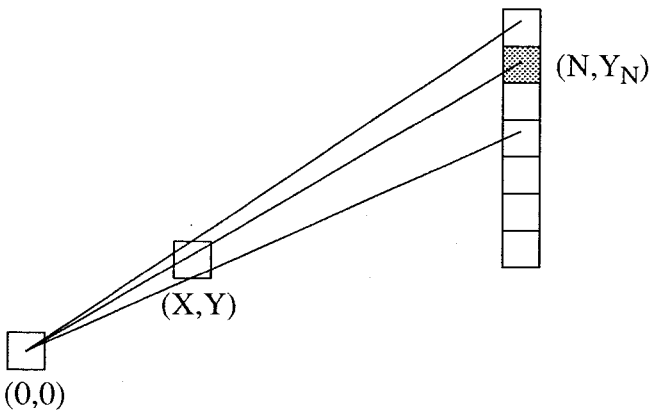


Fig 4.6 : Diffusion restreinte 3D

Même si cette méthode ne génère pas tous les rayons source-objets possibles, elle garantit que l'ensemble des rayons lancés remplit complètement la scène. Dans le cas d'une scène vide, tout voxel est atteint par au moins un rayon en provenance de la source.

$\forall (X, Y) \exists (N, Y_N)$ tel que le segment passant par (0,0) et par (N,Y_N) passe également par (X,Y)

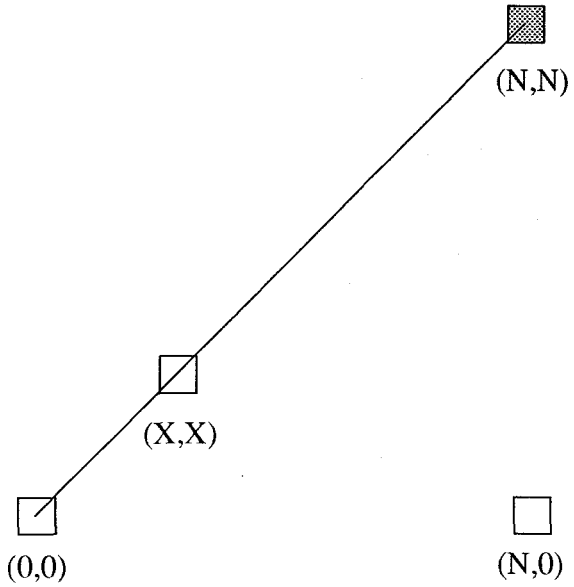


Démonstration:

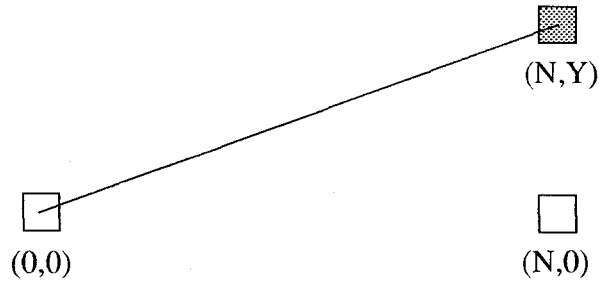
Nous supposons que nous travaillons dans le premier octant.

- Si Y=0 alors le segment issu de (0,0) et passant par (N,0) passe également par (X,0).

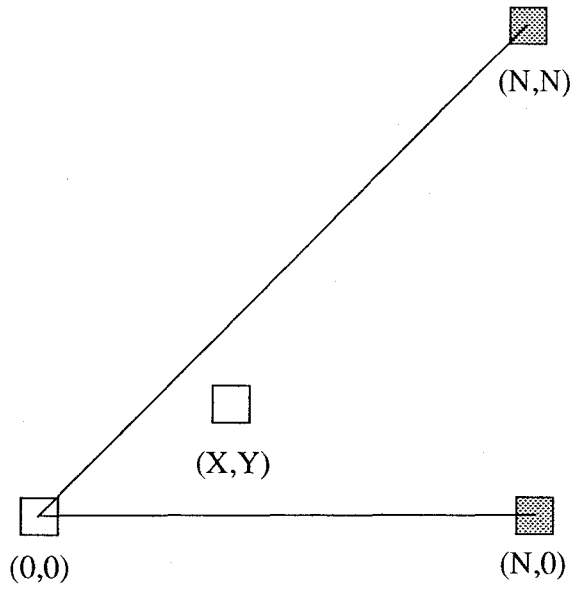
- Si Y=X alors le segment issu de (0,0) et passant par (N,N) passe également par (X,X).



- Si X=N alors le pixel appartient à la dernière colonne et la propriété est donc vraie (le pixel étudié est confondu avec le pixel extrémité).



- Dans tous les autres cas, nous avons le segment [(0,0)-(N,0)] qui passe en dessous du centre du pixel (X,Y) et le segment [(0,0)-(N,N)] qui passe au dessus.



Supposons qu'il n'existe pas de segment [(0,0)-(N,Y_N)] passant par le pixel (X,Y), alors il existe un Y tel que le segment S_1 [(0,0)-(N,Y)] passe au dessus du pixel (X,Y) et le segment S_2 [(0,0)-(N,Y-1)] passe en dessous du même pixel.

L'équation du segment S_1 est: $y = Y \times \frac{x}{N}$

L'équation du segment S_2 est: $y = (Y-1) \times \frac{x}{N}$

Le point de S_1 ayant pour abscisse celle du pixel étudié (X,Y) a donc pour ordonnée:

$$Y_1 = Y \times \frac{x}{N}$$

Le point de S_2 ayant pour abscisse celle du pixel étudié (X,Y) a donc pour ordonnée:

$$Y_2 = (Y-1) \times \frac{x}{N}$$

Or nous supposons que ni S_1 ni S_2 ne passe par le pixel (X,Y), ce qui donne :

$$\text{et} \quad \begin{aligned} Y_1 &> Y + \frac{1}{2} \\ Y_2 &< Y - \frac{1}{2} \end{aligned}$$

ce qui entraîne: $Y_1 - Y_2 > 1$

or $Y_1 - Y_2 = \frac{x}{N}$ et $x < N$

puisque le pixel n'appartient pas à la dernière colonne

d'où la contradiction $Y_1 - Y_2 < 1$

4•1•5 Le problème de l'éclairage multiple par une ou plusieurs sources

La densité des rayons passant par un voxel n'est pas constante dans tout le réseau. La méthode qui consisterait à effectuer la somme des intensités reçues en un point de l'espace amène à des erreurs puisque dans certains cas, suivant le nombre de rayons reçus, l'éclairement variera fortement. Dans le cas de plusieurs sources, un ordre de diffusion distinct sera envoyé pour chaque source lumineuse et il faudra attendre que la diffusion soit terminée pour lancer l'ordre suivant. La règle sera donc : "un voxel est éclairé par la source S s'il lui arrive un rayon en provenance de S".

4•1•6 Comparaison entre méthode classique et méthode par diffusion

DIFFUSION	LANCER CLASSIQUE
sources -> objets	objets -> sources
- les rayons ne voyagent que dans un seul sens	- les rayons sont plus longs puisqu'il faut renvoyer l'information à la cellule ayant émis le rayon d'ombre.
- Pas de mémorisation des positions des sources lumineuses	-Nécessité de mémoriser dans chaque cellule les positions des sources lumineuses.
$Nbrayons = N_{src} \times 6 \times n^2 = N_{rd}$	$Nbrayons = \left(\sum_{i \in objets} S_i \right) \times N_{src} \times 2 = N_{rc}$
n^3 : taille du réseau (Nb de voxels)	Si: surface de l'objet i en nombre de voxels
	Nsrc: nombre de sources lumineuses
Nrd indépendant du nombre d'objets et de leur taille	Nrc dépend du nombre d'objets et de la taille de ces objets
- pas d'interblocage si on diffuse à partir d'une seule source à la fois	-possibilité non négligeable d'interblocage à proximité des sources lumineuses

$$\Re = \frac{N_{rd}}{N_{rc}} = \frac{N_{src} \times 6 \times n^2}{\left(\sum_{i \in objets} S_i \right) \times N_{src} \times 2} = \frac{3 \times n^2}{\sum_{i \in objets} S_i}$$

4•1•7 Comparaison des performances

Pour différentes tailles de réseaux, nous calculons le pourcentage d'occupation de la scène à partir duquel la méthode par diffusion génère moins de rayons que la méthode de lancer classique ($\Re$<1)

$$PC_{occup} = 100 \times \frac{\text{Nb Voxels occupés}}{\text{Nb Voxels total}}$$

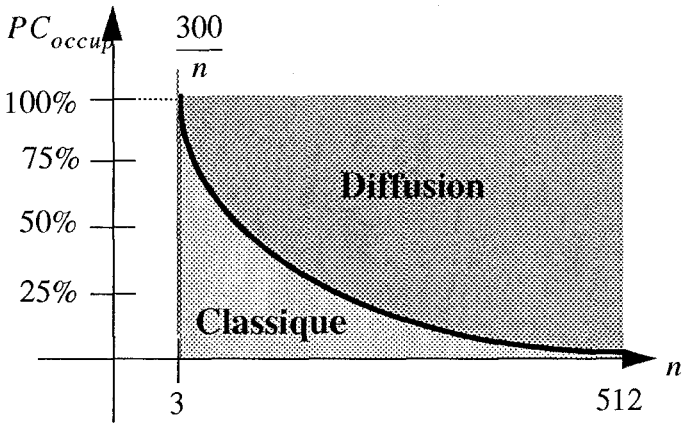
$$PC_{occup} = 100 \times \frac{\sum_{i \in objets} S_i}{n^3}$$

$$\Re < 1 \Leftrightarrow \sum_{i \in objets} S_i > 3 \times n^2$$

$$\Re < 1 \Leftrightarrow PC_{occup} > 100 \times \frac{3 \times n^2}{n^3}$$

$$\Re < 1 \Leftrightarrow PC_{occup} > \frac{300}{n}$$

n	PC_{occup}
n=32	9.4%
n=64	4.7%
n=128	2.3%
n=256	1.2%
n=512	0.6%



Pour une scène standard d'environ 100 000 facettes occupant en moyenne 100 voxels chacune, et pour un réseau de 1K x 1K x 1K, le pourcentage d'occupation est de 1%.
Conclusion: Pour telle résolution, la méthode par diffusion est plus intéressante que la méthode classique puisqu'elle génère moins de rayons.

4.2 Les méthodes de diffusion

4•2•1 Les contraintes à respecter

Pour modéliser le plus fidélement possible l'émission de lumière et sa diffusion dans l'espace, les algorithmes proposés doivent vérifier certains critères:

1°- Il faut garantir la propagation en ligne droite des rayons lumineux.

2°- Si la scène ne contient pas d'objets, l'ensemble des rayons issus d'une source doit remplir la totalité de l'espace.

3°- Lors du passage à l'implantation sur machine, il faut optimiser les temps de calcul et de routage, et donc limiter le nombre de rayons générés tout en vérifiant les 2 contraintes précédentes.

4•2•2 Une méthode de diffusion élémentaire mais erronée

Pour plus de simplicité, l'étude est faite en 2D dans le 1er octant d'un réseau n x n. La source lumineuse est supposée placée en (0,0). Une cellule (x,y) recevant un rayon lumineux le réémet à ses voisines (x+1,y) et (x+1,y+1). Ce qui donne le schéma suivant (Fig 4.7):

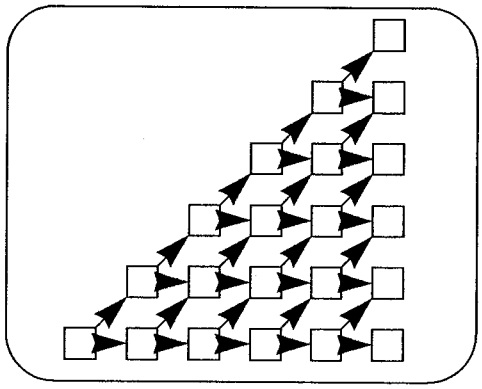


Figure 4.7

Cette méthode ne vérifie pas la contrainte n^{o} 1. En effet il est possible de construire des rayons permettant de contourner les obstacles et d'éclairer ainsi à tort un objet normalement dans l'ombre (Fig 4.8 et 4.9).

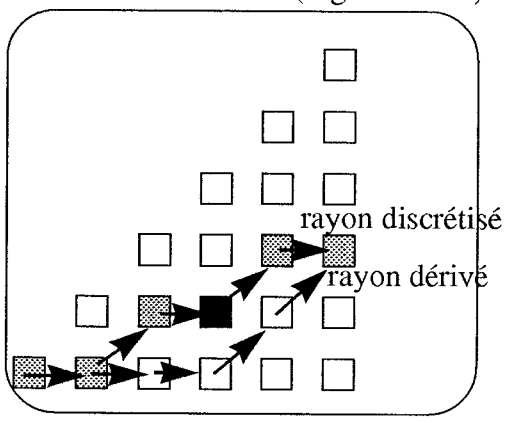


Figure 4.8

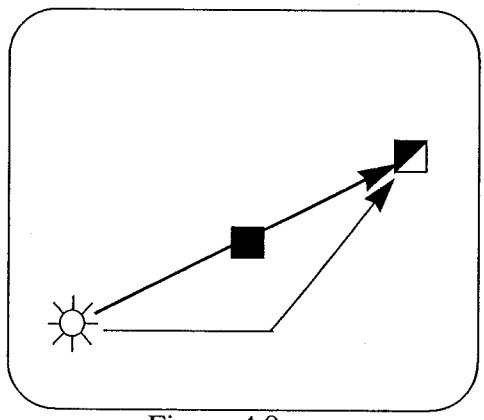


Figure 4.9

4•2•3 Les différents modes de diffusion de rayons

On peut envisager d'envoyer les rayons à partir de la source, l'un après l'autre (envoi séquentiel). Ceci limite le nombre de conflits sur les cellules mais présente l'inconvénient de prendre un temps de l'ordre de N^2.

Partant de la constatation que des rayons de pente "assez proche" empruntent des chemins voisins, il peut être intéressant de traiter simultanément un ensemble de rayons en les regroupant à l'intérieur d'un faisceau ou cône. Le faisceau initial émis à partir de la cellule source se subdivise successivement dans chaque cellule de façon à atteindre tous les points de la surface visée. C'est cette méthode qui a été retenue.

Nous envoyons donc 6 cônes, correspondant chacun à l'une des 6 faces du cube englobant. Bien que le nombre de conflits sur les cellules soit plus important, les simulations ont montré que le temps d'exécution est en N.logN (jusqu'à N=128).

4•2•4 Envoi de rayons les uns après les autres

La méthode séquentielle consiste à envoyer à partir de la source tous les rayons les uns après les autres. A l'instant **t**, la source émet le rayon $\mathbf{R_t}$; chaque cellule ayant reçu un rayon détermine la cellule voisine vers laquelle il faut renvoyer ce rayon. Sachant qu'il y a N_T rayons à générer au total, il faudra donc un temps de N_T+n unité de temps pour acheminer tous les rayons (N_T: nb de rayons à envoyer (4n ou $6n^2$), n: temps mis par le dernier rayon pour parvenir à destination). A chaque instant suivant, les rayons progressent d'une cellule dans la direction de propagation. Au temps **t**, chaque rayon $\mathbf{R_i}$ se trouve donc à une distance (t-i) de la source (t-i = nb d'itérations pour arriver à cette cellule). Comme 1 seul rayon est envoyé à la fois, il ne peut donc y avoir 2 rayons simultanément dans la même cellule.

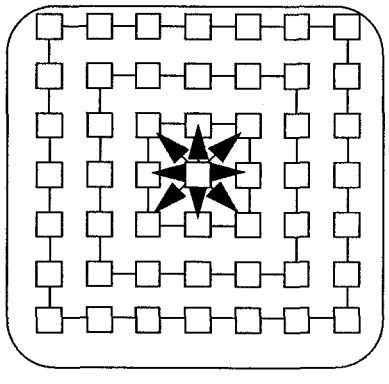


Fig 4.10: diffusion 2D en connectivité 4

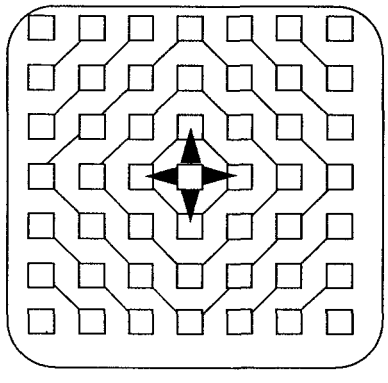
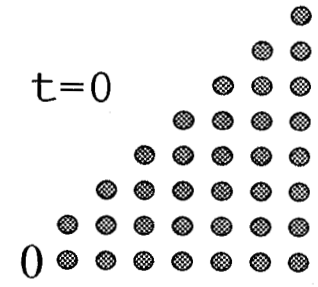
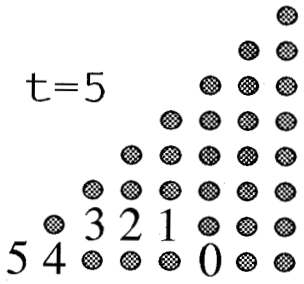
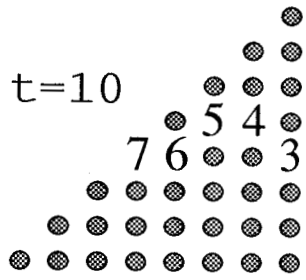


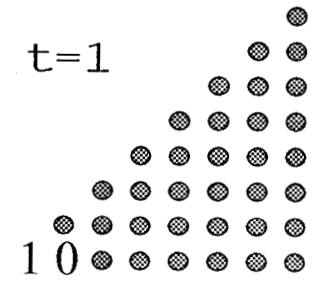
Fig 4.11: diffusion 2D en connectivité 8

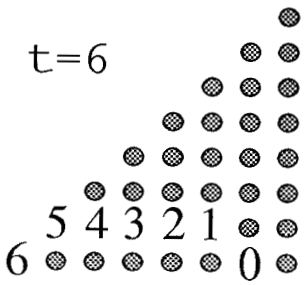
L'exemple suivant montre la diffusion séquentielle de rayons sur un réseau 8x8.
Pour simplifier, l'étude est menée en 2D dans le premier octant. L'extension aux autres octants et à la troisième dimension s'effectue aisément.
Les rayons à envoyer sont numérotés de 0 à 7 et nous suivons leur évolution au cours du temps.
Il faut remarquer qu'aucun conflit ne se produit puisqu'il y a au plus un rayon par cellule à un instant donné. Au bout de 14 itérations, dans notre exemple, les 8 rayons sont arrivés à destination:

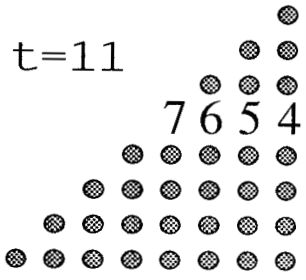

t=0
0

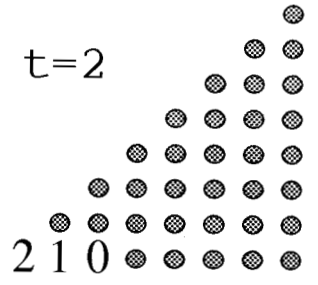

t=5
3 2 1
5 4 0

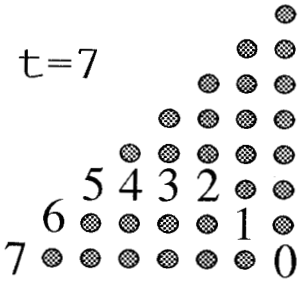

t=10
5 4
7 6 3

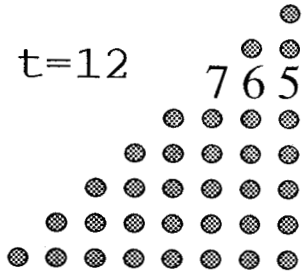

t=1
1 0

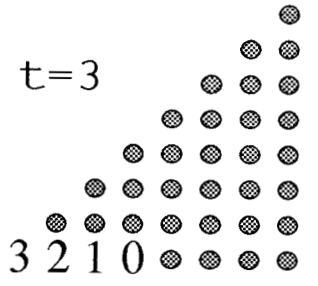

t=6
5 4 3 2 1
6 0

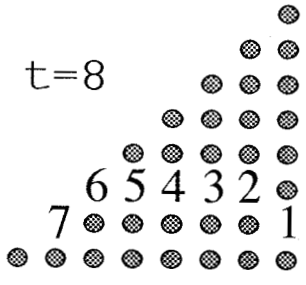

t=11
7 6 5 4

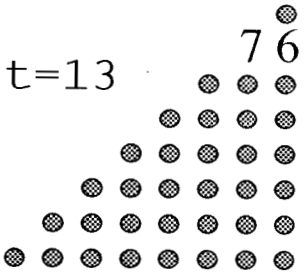

t=2
2 1 0

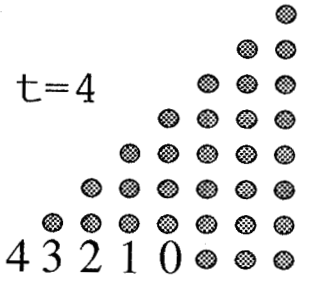

t=7
5 4 3 2
6 1
7 0

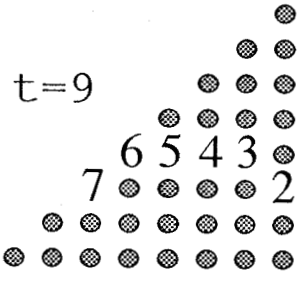

t=12
7 6 5

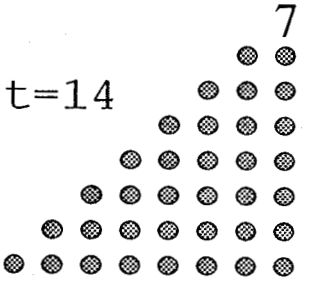

t=3
3 2 1 0


t=8
6 5 4 3 2
7 1


t=13
7 6


t=4
4 3 2 1 0


t=9
6 5 4 3
7 2


t=14
7

4•2•5 Mode subdivision de faisceaux

La méthode précédente met en évidence le fait qu'une cellule traite successivement un ensemble de rayons "proches" les uns des autres (de pente quasi-identique). L'idée est de faire traiter par la cellule simultanément l'ensemble des rayons qui passent effectivement par elle. On obtient donc un faisceau de rayons [AMA 84]. La cellule émettrice envoie donc un nombre de faisceaux nettement inférieur au nombre réel de rayons. Chaque cellule ayant reçu un faisceau découpe ce faisceau de manière à n'envoyer aux cellules voisines que la partie du faisceau qui les concerne. Le faisceau initial est découpé en 2 faisceaux F1 et F2 (Fig 4.12 et 4.14). Le faisceau F1 est envoyé à la cellule (1,1), le faisceau F2 est envoyé à la cellule (1,0). Le processus est réitéré sur les nouveaux faisceaux. F1 se découpe en F11 et F12, F2 se découpe en F21 et F22 (Fig 4.13 et 4.15). Le temps idéal de propagation dans le réseau est ramené à n unités de temps. Cependant, à cause de la discrétisation, des rayons distincts peuvent se confondre sur certains pixels. De ce fait des faisceaux distincts peuvent passer par les mêmes pixels. Si une cellule ne peut traiter qu'un seul faisceau par unité de temps, apparaissent alors des conflits d'accès aux cellules qui vont ralentir le fonctionnement et donc augmenter le temps de calcul. Des simulations ont été faites et semblent montrer que sur des réseaux de taille *pas trop grande* (n<128) les conflits d'accès multiplient le temps de calcul par un coefficient de l'ordre de $\log_2 n$ (pour des espaces 3D).

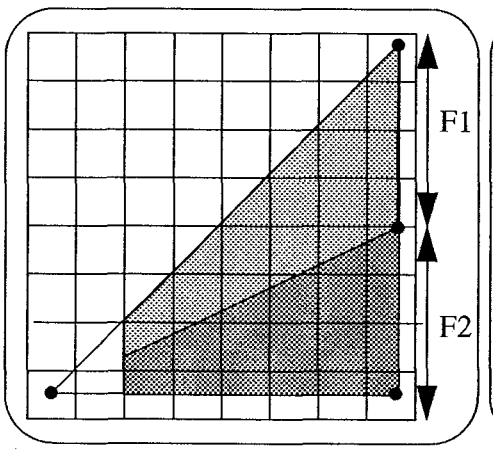


Fig 4.12: Division faisceau

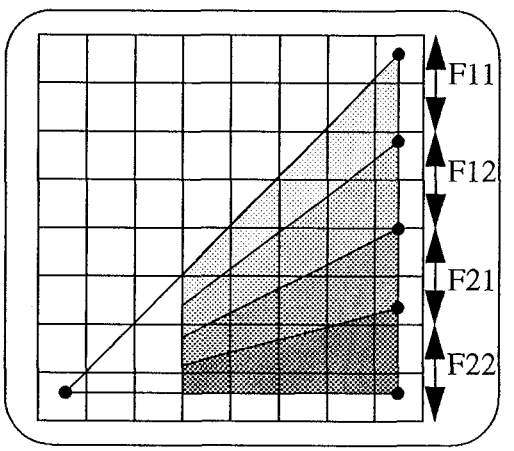


Fig 4.13: Division faisceau

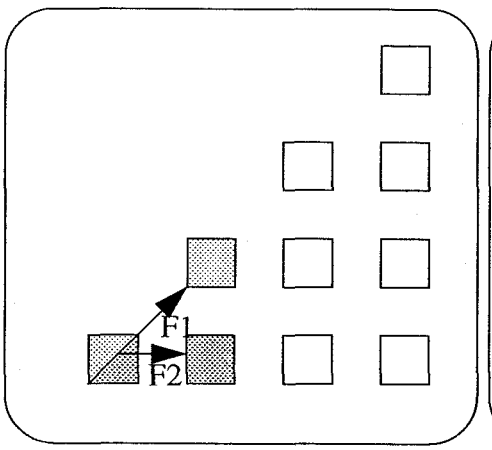


Fig 4.14: Transmission cellule

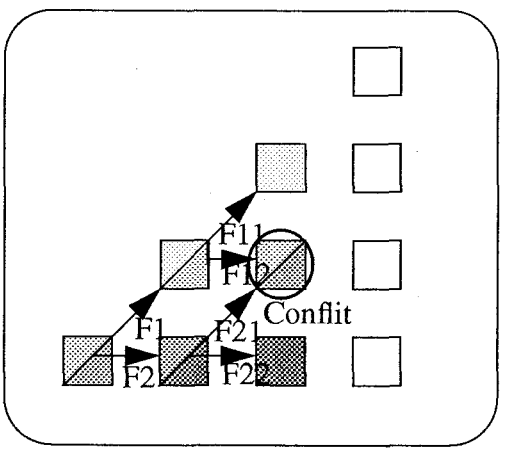


Fig 4.15: Transmission cellule

4•2•6 Comparaison entre les 2 modes d'émission de rayons

	envoi rayons un à un	subdivision faisceau
2D	$T_{diff} = (n+n) \times \tau_p$	$T_{ideal} = n \times \tau_s$
3D	$T_{diff} = (n^2+n) \times \tau_p$	$T_{ideal} = n \times T_s$ $T_{reél} \sim (n \cdot \log n) \times T_s$
	Algorithme ne nécessitant pas trop de calculs au niveau de la cellule. (suivi de rayon incrémental). Pas de conflit d'accès à la cellule	Algorithme "assez" complexe nécessitant des calculs plus important au niveau de la cellule. Conflits d'accès à la cellule

4.3 Algorithme de subdivision de faisceaux

4•3•1 Algorithme de subdivision en 2D

4•3•1•1 Définition d'un faisceau 2D

Un faisceau est déterminé par 2 rayons extrêmes (Fig 4.16). Il correspond à l'ensemble des rayons compris entre ces 2 rayons extrêmes (limites du faisceau). Les limites peuvent être calculées en prenant les valeurs des ordonnées des points de la colonne extrémité par lesquels passent les rayons limites de façon à communiquer des valeurs numériques entières. Un faisceau 2D sera donc défini par 2 valeurs: Y_{min} et Y_{max}.

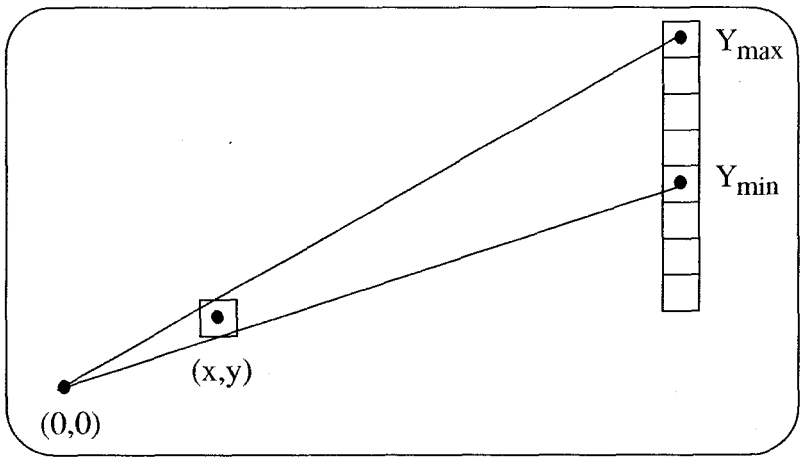


Fig 4.16: structure d'un faisceau

4•3•1•2 Calcul de la séparation des faisceaux

L'étude se limite pour l'instant au 1er octant, la source étant située en (0,0). La cellule (x,y) ayant reçu le faisceau (Y_{min},Y_{max}), elle essaie de le découper en 2 faisceaux (Ymin,Y_{med}) et (Y_{med},Y_{max}) et le cas échéant transmet le 1er faisceau à la cellule (x+1,y) ainsi que le 2nd faisceau à la cellule (x+1,y+1) (Fig 4.17). Y_{med} est déterminé de

la manière suivante: c'est l'ordonnée du point de la colonne extrémité situé sur le rayon passant par le milieu des pixels (x+1,y) et (x+1,y+1) soit

$$Y_{med} = \left[\frac{\left(y + \frac{1}{2}\right) \times (n-1)}{x+1} \right]$$

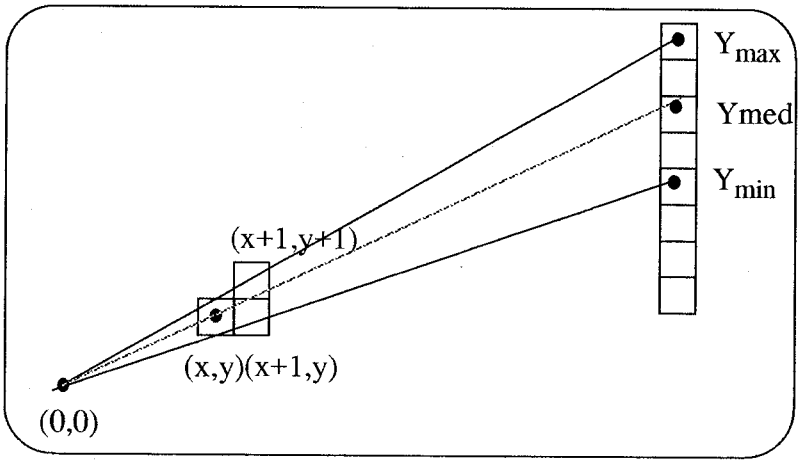


Fig 4.17: subdivision de faisceau

4•3•1•3 Différents cas de réémission possibles

Selon la position de Y_{med} par rapport à Y_{min} et Y_{max}, la cellule enverra 1 ou 2 faisceaux (Fig 4.18). On obtient donc les cas de figure suivants:

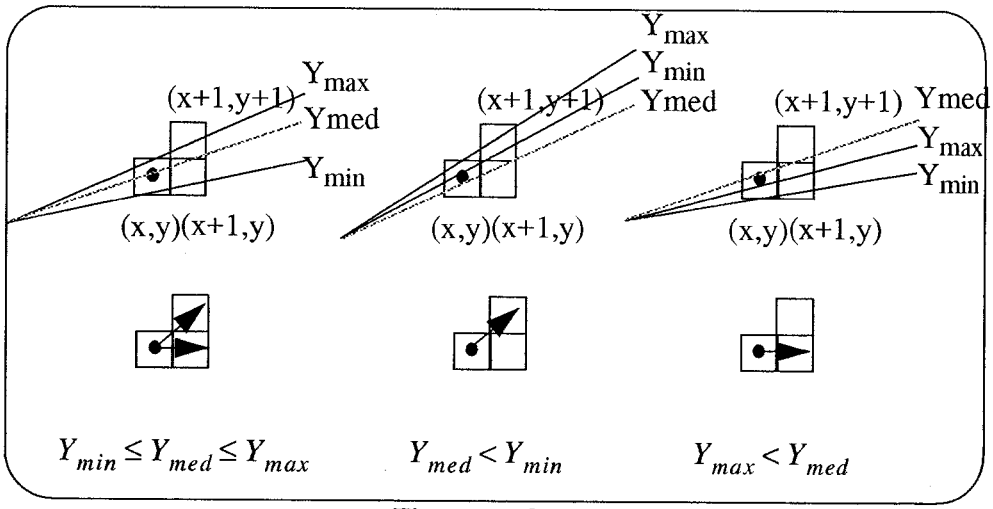
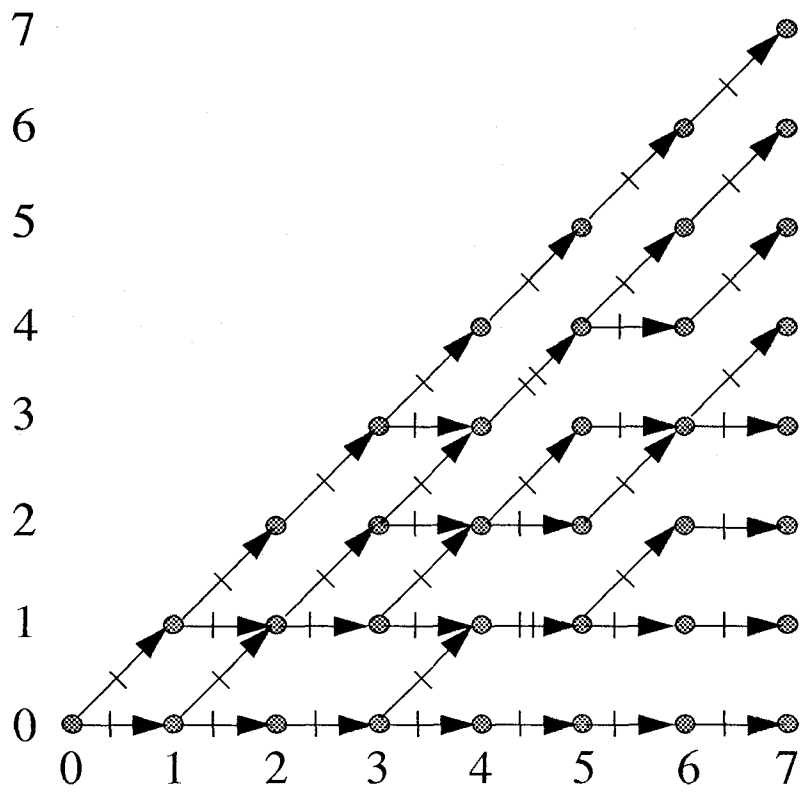


Figure 4.18

Le schéma suivant montre la diffusion des faisceaux en 2D dans le premier octant à partir d'une source de lumière située en (0,0).
Chaque point représente une cellule, les flèches représentent les faisceaux réémis dans cette direction par autant de batonnets correspondants. Nous voyons que plusieurs faisceaux peuvent transiter par une même cellule.



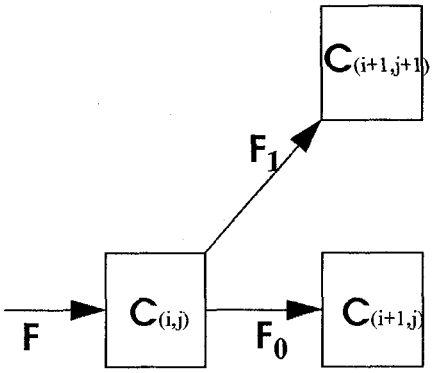
4•3•2 Subdivision de faisceaux à l'aide d'une fonction logique

Nous avons donc étudié la possibilité d'utiliser des fonctions logiques pour déterminer les subdivisions de faisceaux. Nous allons voir que cette méthode très intéressante du point de vue théorique se prête assez mal à une implémentation réelle car elle reste trop complexe. La figure précédente montre qu'un faisceau qui atteint une cellule peut soit continuer son chemin, soit se diviser en deux nouveaux sous-faisceaux.
Etant donné un faisceau F reçu par la cellule C(i,j), si nous notons par F_0 l'envoi d'un faisceau à la cellule C(i+1,j) et par F_1 l'envoi à la cellule C(i+1,j+1), nous avons les trois cas de figures suivants:

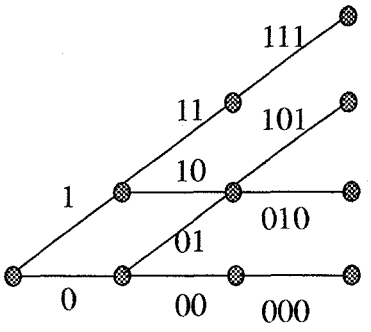
(F_0=1 ; F_1=0) -> Le faisceau continue son chemin vers C(i+1,j)

(F_0=0 ; F_1=1) -> Le faisceau continue son chemin vers C(i+1,j+1)

(F_0=0 ; F_1=1) -> Le faisceau se divise en deux sous-faisceaux:
un vers C(i+1,j) , l'autre vers C(i+1,j+1).



Découpage des faisceaux



Chemins suivis

Les valeurs des fonctions F_0 et F_1 dépendent du chemin suivi par le faisceau pour atteindre la cellule en question et peuvent être précalculées une fois pour toutes, pour tous les chemins possibles, et stockées dans une table intégrée au sein de chaque cellule. Les chemins peuvent être codés par un mot binaire de longueur comprise entre 1 et N dans lequel chaque chiffre B_i représente un pas horizontal (B_i=0) ou diagonal (B_i=1) [BER 87]. Le calcul du suivi de faisceau se réduit alors à une recherche dans la table. Selon les valeurs de F_0 et F_1, on créera un ou deux nouveaux faisceaux en ajoutant 0 ou 1 au chemin correspondant.

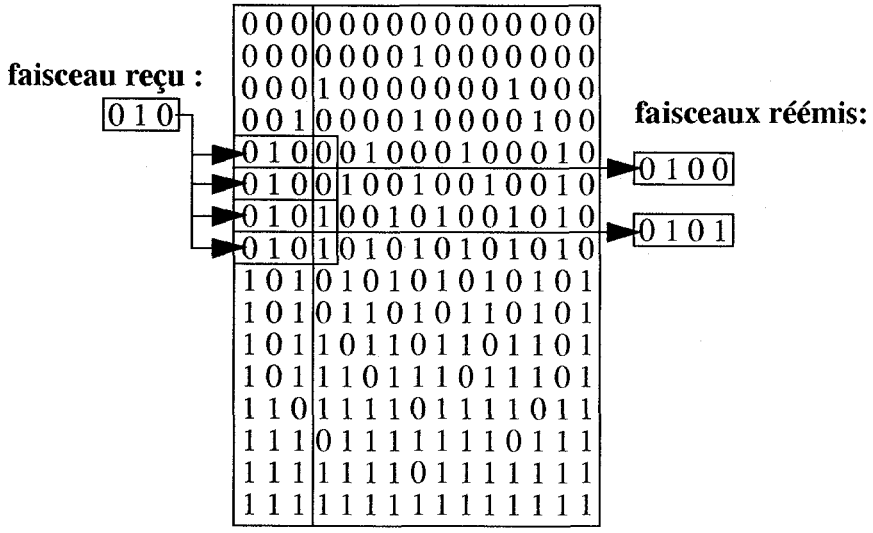
Chemin	F_0	F_1
0	1	1
1	1	1
00	1	0
01	1	0
10	0	1
11	0	1
⋮	⋮	⋮

TABLE

ALGORITHME

- Recevoir (chemin $B_0....B_m$)
- Rechercher chemin dans la table, déterminer F_0 et F_1
- Si F_0=1 générer $B_0....B_m0$
- Si F_1=1 générer $B_0....B_m1$

Le principal inconvénient de cette méthode est de générer une table de taille importante: il y a $2^0+2^1+.....+2^N=2^N-1$ chemins possibles de tailles comprises entre 1 et N bits.
En fait, parmi tous les chemins possibles, seule une partie correspond à des chemins réellement empruntés. Nous avons donc envisagé de ne stocker que ces chemins-là, et nous avons construit une table constituée de N chemins à N bits, chaque chemin correspondant à un rayon différent.



Exemple de table pour N=16

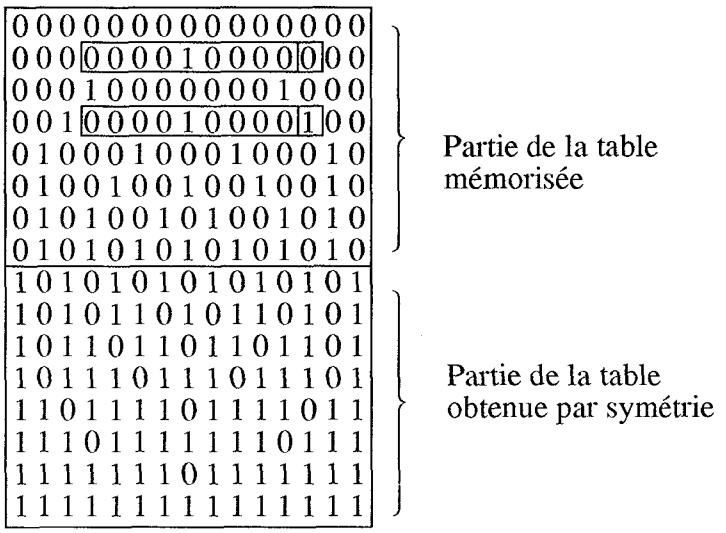
Pour déterminer les faisceaux réémis à partir du faisceau $B_0.....B_m$, le principe est le suivant. Toutes les entrées dont les m premiers bits correspondent au chemin reçu $B_0.....B_m$ sont recherchées dans la table, puis si parmi ces entrées il y en a une qui commence par $B_0.....B_m0$, le faisceau $B_0.....B_m0$ est généré; de même s'il y a une entrée qui commence par $B_0.....B_m1$, le faisceau $B_0.....B_m1$ est généré.
Cette technique permet de simplifier en partie la table mais implique une recherche par comparaison des m premiers bits. Si la table a été précédemment triée par ordre croissant des entrées, le temps d'accès à l'aide d'un algorithme de recherche dichotomique prend au plus un temps de log N (N:taille du réseau).
Nous avons réussi à diminuer encore la taille mémoire nécessaire en tenant compte d'une certaine symétrie des chemins dans la table, en remarquant que :
si $B_0.....B_m$ représente le chemin suivi pour arriver à la cellule considérée et
si $\overline{B0.....Bm}$ représente le chemin obtenu en remplaçant chaque bit par son complément à 1, on a :

$$F_0(B_0.....B_m) = F_1(\overline{B_0.....B_m})$$
$$F_1(B_0.....B_m) = F_0(\overline{B_0.....B_m})$$

et donc si $B_0.....B_m$ est un chemin qui appartient à la table alors le chemin $\overline{B0.....Bm}$ appartient également à la table.Ceci permet de ne garder qu'une moitié de la table, l'autre moitié se retrouvant par complément.
Cette méthode, bien que fort intéressante par le gain de temps qu'elle procure dans la phase de suivi de faisceaux, présente l'énorme inconvénient d'imposer un encombrement mémoire encore trop important (de l'ordre de N^2 bits). Par exemple, pour un réseau 512 x 512 correspondant à une image de même résolution, le coût mémoire est de 16 Koctets par cellule, ce qui ne peut être envisageable technologiquement vues les contraintes d'intégration que cela imposerait.

Une autre technique qui permet de réduire la taille de la table est de ne prendre en compte qu'une partie du chemin. Notre idée était de ne garder que les p derniers bits du chemin emprunté par un faisceau (p à déterminer).

Des simulations ont montré que si l'on veut garantir que les rayons discrets suivent le chemin le plus proche du chemin réel, au sens de Bresenham, il faut mémoriser un grand nombre de bits (p≅2n/3). En effet, des faisceaux peuvent emprunter des chemins parallèles (ayant q bits identiques) sur de longues distances et diverger ensuite. Il faut donc mémoriser un nombre de bits suffisant.
Dans l'exemple ci-dessous, nous remarquons qu'il existe deux chemins identiques sur 9 bits consécutifs mais différents sur le bit suivant.



Exemple de table pour N=16

Nous avons également regardé si les fonctions F_0 et F_1 ne pouvaient pas se simplifier, mais si l'étude est aisée sur des réseaux de petite taille, en revanche sur des réseaux de plus grande taille il est moins évident, sans outil approprié, de trouver une expression simple pour une fonction à 512 variables par exemple.

4•3•3 Cas général: source en un point quelconque.

Supposons maintenant que la source lumineuse soit placée en un point quelconque de l'espace 2D. L'espace est découpé en 4 quadrants (Fig 4.20) et 4 faisceaux distincts sont envoyés, un pour chacune des 4 directions de propagation: +x, +y, -x, -y. L'algorithme de décomposition de faisceaux est légèrement modifié pour tenir compte de cette caractéristique. L'étude est faite pour le 1^{er} quadrant (dir: +x). Pour les autres quadrants la technique est analogue.
La cellule (x,y) ayant reçu le faisceau (Y_{min},Y_{max}), elle essaie de le découper en 3 faisceaux (Y_{min},Y_1), (Y_1,Y_2) et (Y_2,Y_{max}) qu'elle envoie respectivement aux cellules (x+1,y-1), (x+1,y) et (x+1,y+1) selon la position de Y_1 et Y_2 par rapport à Y_{min} et Y_{max}

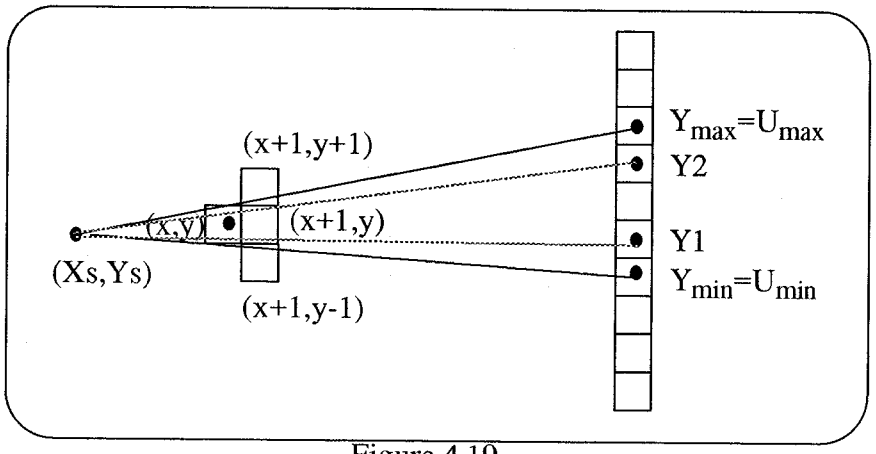


Figure 4.19

Y_1 et Y_2 sont déterminés comme suit:

$$Y_1 = \left[\frac{\left(y - Ys - \frac{1}{2}\right)(n - 1 - Xs)}{(x - Xs + 1)} \right]$$

$$Y_2 = \left[\frac{\left(y - Ys + \frac{1}{2}\right)(n - 1 - Xs)}{(x - Xs + 1)} \right]$$

Pour chaque direction de propagation, les informations à passer d'une cellule à l'autre sont:

Xs, Ys : coordonnées de la source
Ir, Iv, Ib : intensités RVB de la source lumineuse
Umin,Umax: limites du faisceau
D : direction de propagation (0 à 3: +x, +y, -x ou -y)

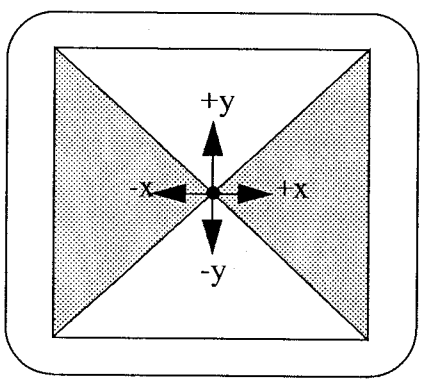


Figure 4.20: directions propagation en 2D

4•3•4 Algorithme de subdivision en 3D

4•3•4•1 Définition d'un faisceau 3D

Un faisceau 3D sera limité par 4 rayons, 2 pour chaque direction orthogonale à la direction de propagation (Fig 4.21).
Ex: un rayon se propageant dans la direction +OZ aura une limite en X (Xmin,Xmax) puis une autre selon Y (Ymin,Ymax).

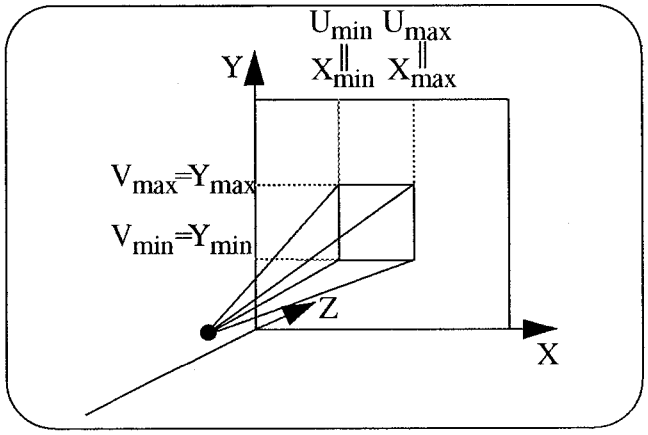


Fig 4.21 : Faisceau 3D

4•3•4•2 Calcul de la séparation des faisceaux.

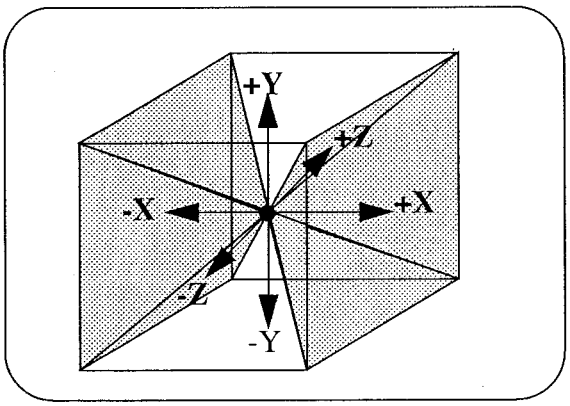


Fig 4.22: directions de propagation en 3D

Six faisceaux initiaux sont envoyés à partir de la source, un faisceau pour chacune des directions +x, -x, +y, -y, +z, -z (Fig 4.22). L'algorithme précédent est appliqué pour chaque direction perpendiculaire à la direction de propagation. Le faisceau initialement reçu est donc découpé en 9 nouveaux faisceaux (Fig 4.23). Dans le cas d'une propagation dans la direction +z U1,U2,V1,V2 sont déterminés de la manière suivante:

$$U_1 = \left[\frac{\left(x - Xs - \frac{1}{2}\right)(n-1)}{(z - Zs + 1)} \right] \qquad U_2 = \left[\frac{\left(x - Xs + \frac{1}{2}\right)(n-1)}{(z - Zs + 1)} \right]$$

$$V_1 = \left[\frac{\left(y - Ys - \frac{1}{2}\right)(n-1)}{(z - Zs + 1)} \right] \qquad V_2 = \left[\frac{\left(y - Ys + \frac{1}{2}\right)(n-1)}{(z - Zs + 1)} \right]$$

Les données à transférer sont:

Xs,Ys,Zs	:	Coordonnées de la source
Ir,Iv,Ib	:	intensité de la source
D	:	direction de propagation (0 à 5)
Umin,Umax	:	Limites du faisceau dans les 2 directions
Vmin,Vmax	:	perpendiculaires à la direction de propagation

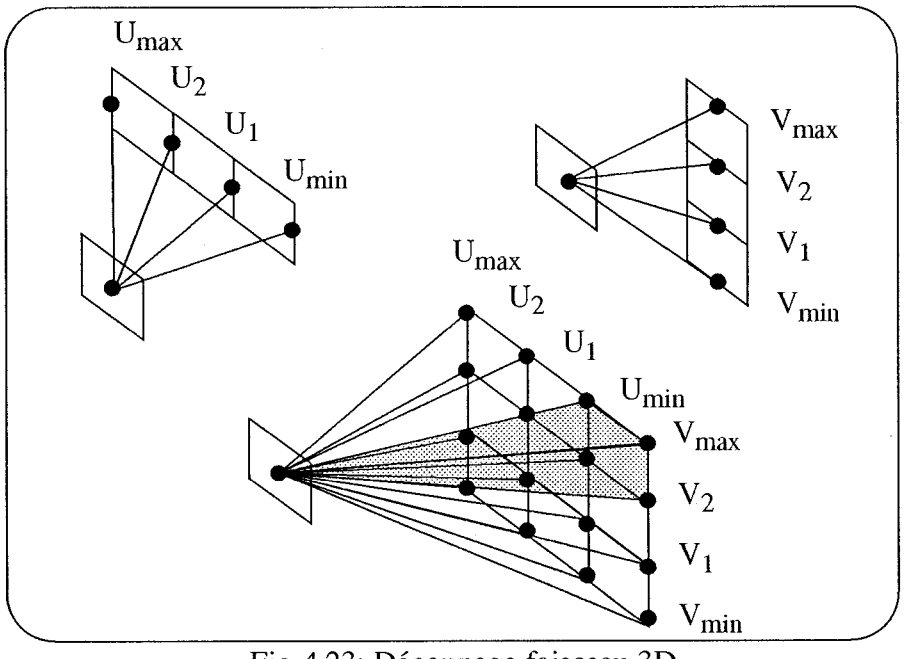


Fig 4.23: Découpage faisceau 3D

4.4 Application au réseau cellulaire RC^2

4•4•1 Implémentation de l'algorithme de diffusion

Nous désirons implanter la méthode de diffusion 3D sur notre réseau RC^2.
Nous rappelons que chaque cellule est supposée avoir 1 buffer de sortie pour chaque direction de communication avec les cellules voisines plus une file d'attente permettant de mémoriser les faisceaux en attente de traitement. Nous verrons par la suite que la taille de cette file conditionne fortement les performances de la diffusion des faisceaux dans le réseau. L'algorithme exécuté par chaque cellule est le suivant:

```
début
  si FA non vide
    alors choisir un faisceau
          si dir propagation libre
            alors découper faisceau
                  renvoyer nouveaux faisceaux
                  aux cellules concernées
            sinon remettre le faisceau dans FA
          fsi
  fsi
fin
```

La cellule reçoit dans la FA des faisceaux en provenance des cellules voisines (si l'état de la FA le permet). Elle choisit un faisceau à traiter dans la FA puis dépose les faisceaux résultants dans les buffers de sortie correspondants. Les buffers seront vidés lors de l'émission des messages à destination des cellules adéquates. Si l'un des buffers de sortie reste occupé (le message n'a pu être envoyé), la cellule passe dans l'état **bloqué**. En effet il se pourrait, à l'étape suivante, qu'elle ait à envoyer un autre rayon à destination de la cellule accessible par le buffer occupé.

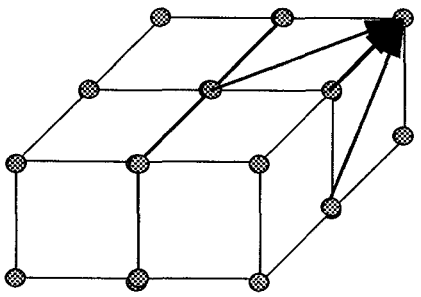
Un message reçu par une cellule est réémis en plusieurs messages envoyés à la cellule elle-même et à ses voisines. Une cellule gérant plusieurs voxels de Z différents, il faut donc mémoriser les messages à destination de tous les voxels de la cellule.
Il peut se produire un ralentissement lorsque la capacité de mémorisation d'une cellule est atteinte. Elle passe momentanément dans l'état "bloqué" et ne peut plus écouler le flux de rayons passant part elle.
Il faut donc réaliser un compromis entre les deux contraintes contradictoires suivantes:
(1) Pour éviter de saturer une cellule, traiter d'abord les rayons réémis dans la même cellule,
(2) Pour diffuser plus rapidement, traiter les rayons qui sortent de la cellule.

4•4•2 Gestion des conflits

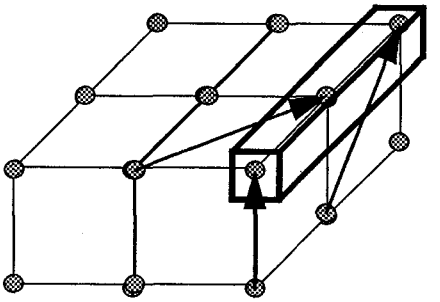
2 types de conflits apparaissent:

- Les conflits inhérents à l'algorithme (collisions naturelles) dûs au fait que 2 faisceaux différents peuvent passer par le même voxel

- les conflits supplémentaires introduits par l'utilisation d'un réseau 2D pour simuler un espace 3D. Une cellule gère un ensemble de voxels, et si on considère qu'une cellule ne peut traiter qu'un seul faisceau à la fois, elle bloque le traitement des faisceaux qui passent par des voxels différents mais gérés par la même cellule (de même x,y mais de z différents).



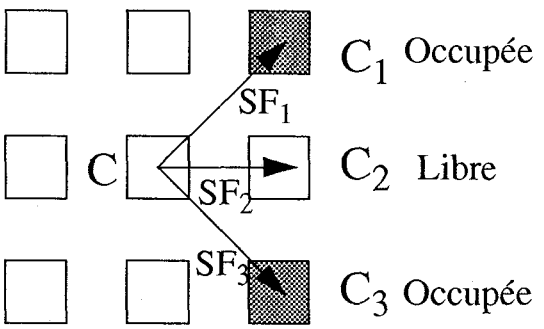
Collision naturelle



Conflits supplémentaires

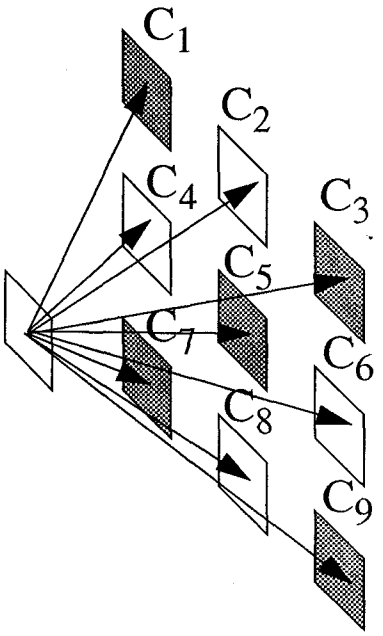
Une cellule qui possède une file d'attente pleine passe dans l'état **bloqué** tant que la FA est pleine. Elle inhibe donc l'émission de messages depuis d'autres cellules vers elle, bloquant ainsi une autre cellule qui ne peut plus émettre puisque en attente d'envoi de message. Nous observons donc un ralentissement de l'activité du réseau mais en aucun cas un dead lock puisque à un moment ou à un autre une cellule bloquée va passer dans l'état **non bloqué** libérant de ce fait les cellules en amont.

Lorsqu'une cellule reçoit un faisceau à découper puis à transmettre à ses voisines, et que l'une d'entre elles au moins est occupée et donc ne peut recevoir le faisceau réémis, plusieurs stratégies peuvent être adoptées:
- soit il faut attendre que les voisines se libèrent pour envoyer les sous-faisceaux du faisceau initial, mais on bloque alors la cellule émettrice.
- soit nous autorisons d'envoyer les sous-faisceaux à destination des cellules libres et il faut donc mémoriser les sous-faisceaux ne pouvant être réémis. Ces derniers sont alors stockés dans la F.A. interne à la cellule qui peut éventuellement être déjà pleine, ce qui entraîne alors la perte de ce faisceau.
En effet, dans l'exemple suivant en 2D, la cellule C doit réémettre 3 sous-faisceaux SF_1, SF_2 et SF_3 à destination des cellules C_1, C_2 et C_3. Or seule la cellule C_2 est libre. Elle peut donc recevoir le faisceau SF_2 mais les faisceaux SF_1 et SF_3 doivent être mémorisés dans la F.A. de la cellule C. Si celle-ci est déjà saturée,l'un des deux faisceaux SF_1 ou SF_3 est perdu.



En 3D, le découpage d'un faisceau peut générer, dans le cas le plus défavorable, jusqu'à 5 nouveaux sous-faisceaux ne pouvant être réémis du fait de la non disponibilité des cellules destinatrices. Le faisceau reçu par la cellule C se divise en 9 sous-faisceaux SF_1, SF_2 .... SF_9. Les cellules C_1, C_3, C_5, C_7 et C_9 sont supposées occupées, et les 5 sous-

faisceaux correspondants SF_1, SF_3, SF_5, SF_7 et SF_9 doivent être mémorisés dans la F.A. et peuvent éventuellement saturer celle-ci:



La stratégie adoptée consistera à déterminer la direction de propagation du faisceau et scruter les cellules susceptibles de recevoir un sous-faisceau. Tant que l'une au moins d'entre elles n'est pas libre, le faisceau ne sera pas traité par la cellule. Cette technique a l'inconvénient de ralentir la diffusion mais elle présente l'énorme avantage d'assurer qu'aucun faisceau ne risque d'être perdu.

4•4•3 Résultats de simulation de RC^2.

Les simulations suivantes montrent l'évolution au cours du temps des voxels atteints par les faisceaux émis depuis une source lumineuse située au centre de la scène. Chaque tranche de 0 à 15 représente une coupe dans l'espace des voxels (Z=constante). La simulation est effectuée sur un réseau 16x16x16 voxels. Nous supposons que la scène ne contient que la source sans aucun autre objet. Le voxel (X,Y,Z) est visualisé par une case noire s'il a reçu au moins un faisceau depuis le temps t_0, instant de l'émission du faisceau initial par la source, et par une case blanche sinon (non encore atteint par aucun faisceau). La tranche de droite représente l'activité du réseau à l'instant t. Les cases noires correspondent aux cellules du réseau qui ont un faisceau à traiter (F.A. non vide), les cases blanches à celles qui sont libres (F.A. vide). Nous obtenons ainsi une photo instantanée de l'état d'activité du réseau. Cette simulation permet de mettre en évidence le fait que tous les voxels sont bien atteints par au moins un faisceau et que le nombre de cellules actives varie fortement au cours du temps.

L'étude est menée sur un réseau de taille n=16 pour 2 tailles de file d'attente (1 faisceau et 16 faisceaux). Nous voyons que l'activité du réseau est plus importante dans le cas où la file d'attente a une longueur de 16 rayons et donc la diffusion plus rapide.

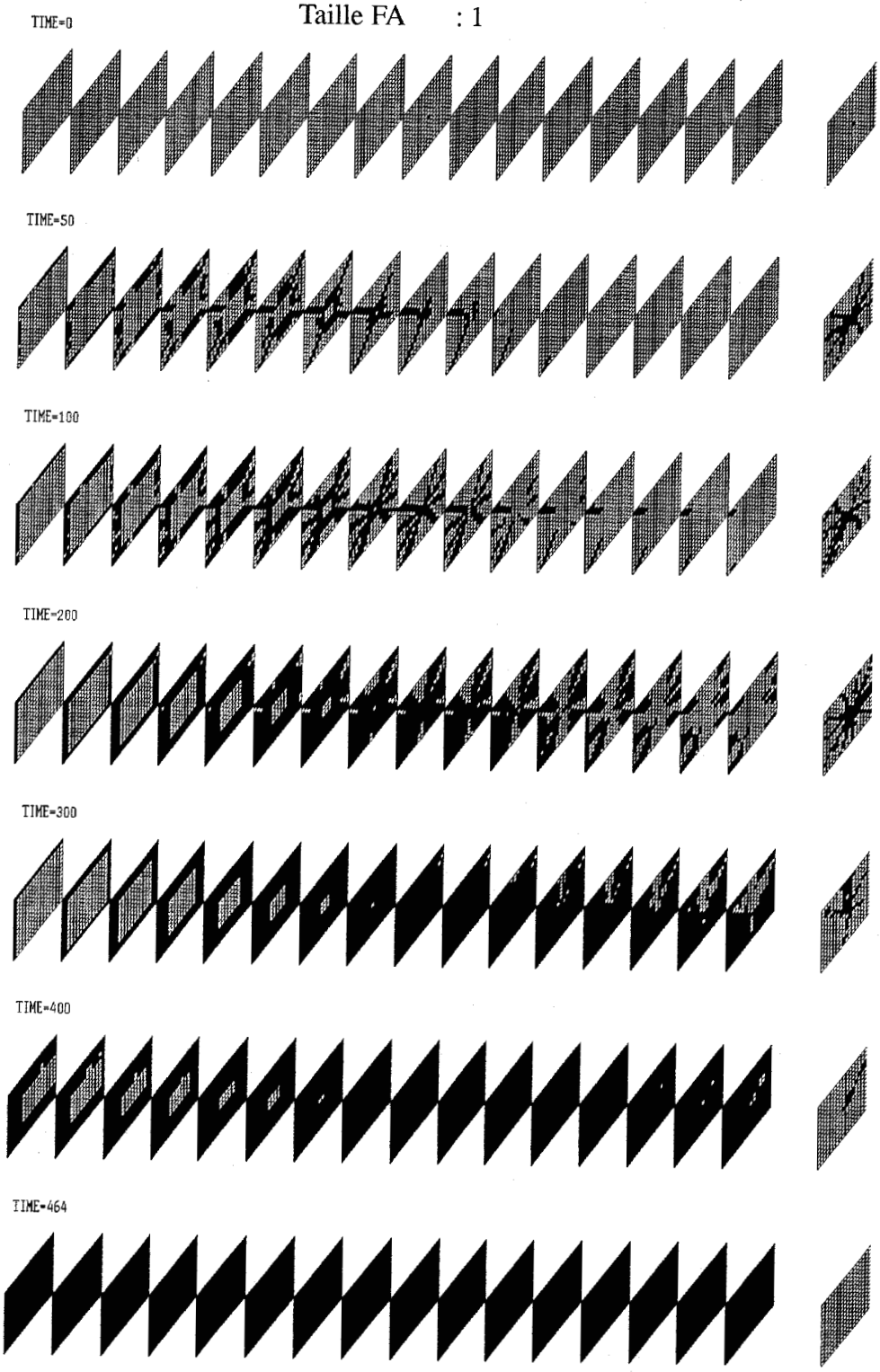
Différentes études ont été effectuées pour des tailles de réseaux différentes.
Si nous considérons qu'une cellule ne peut mémoriser qu'un seul faisceau à la fois (Taille FA=1), nous nous apercevons que la durée d'exécution de l'algorithme (temps au bout duquel tous les faisceaux ont atteint leur objectif) varie en N^2 par rapport à la taille N du réseau. Désirant diminuer cette durée, nous avons introduit des F.A. de longueur

plus importante à l'intérieur de chaque cellule. Le rôle de celles-ci est de mémoriser les faisceaux qui ne peuvent être traités par la cellule de façon à libérer la cellule expéditrice du faisceau.
Nous avons étudié, pour différentes tailles de réseaux, l'allure de la courbe donnant la durée d'exécution en fonction de la taille des F.A. utilisées, et nous nous sommes aperçus que la durée d'exécution chuttait assez vite pour les faibles tailles de F.A. dès que l'on augmentait un peu cette taille.
Nous avons repris nos simulations et avons tracé la courbe donnant le Temps d'exécution en fonction de la taille du réseau, en prenant soin d'avoir dans chaque cas une F.A. de taille suffisante pour mémoriser tous les rayons reçus.
Le temps d'exécution est alors de l'ordre de N.logN, soit un gain de logN par rapport à la première méthode. Il faut cependant remarquer que cette amélioration des performances se fait au détriment d'une occupation mémoire plus importante et il faut garder à l'esprit les contraintes fixées par les procédés d'intégration qui ne nous autorisent pas à utiliser des tailles de F.A. trop grandes.
Nous devons donc réaliser un compromis entre mémoire et vitesse.

L'ordre d'envoi des faisceaux initiaux n'est pas neutre. Il vaut mieux envoyer d'abord en x et y de façon à occuper le maximum de cellules du réseau, puis ensuite en z. En effet, un faisceau se propageant selon la direction z bloque une cellule pendant tout le temps où x et y restent inchangés.

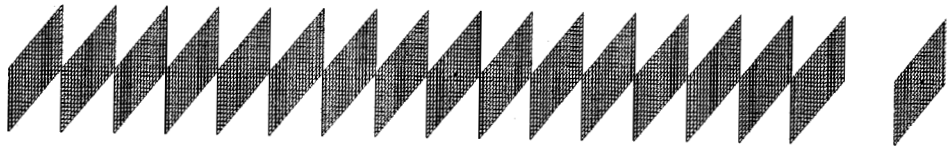
Les courbes suivantes représentent la durée d'exécution T en nombre de cycles en fonction de la taille du réseau N et de celle des files d'attente FA.

Taille réseau : 16
Taille FA : 1

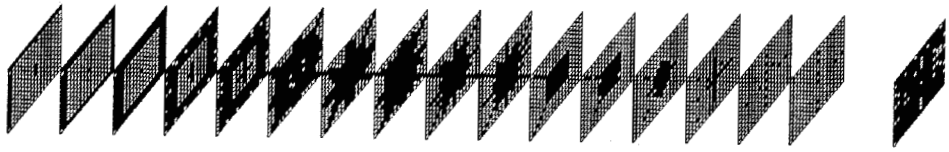


Taille réseau : 16
Taille FA : 16

TIME=0



TIME=25

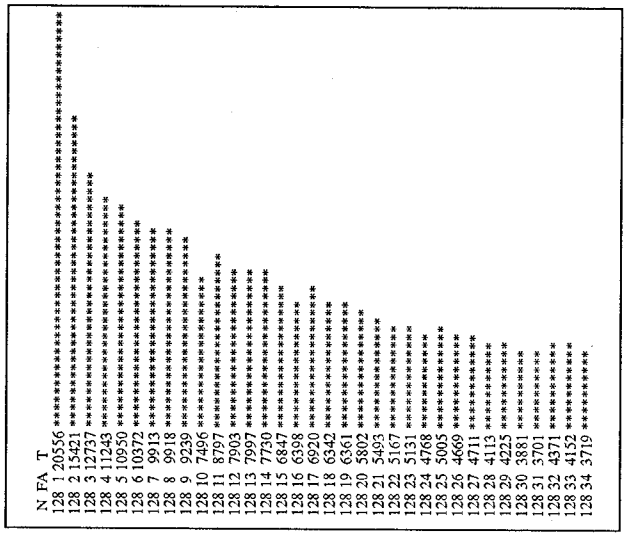


TIME=50

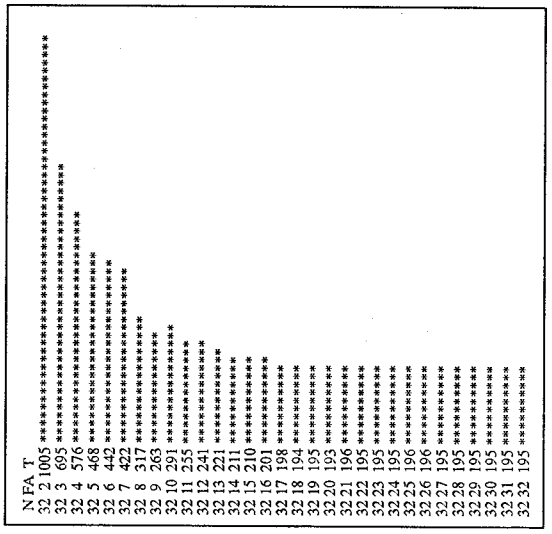
TIME=75

TIME=100

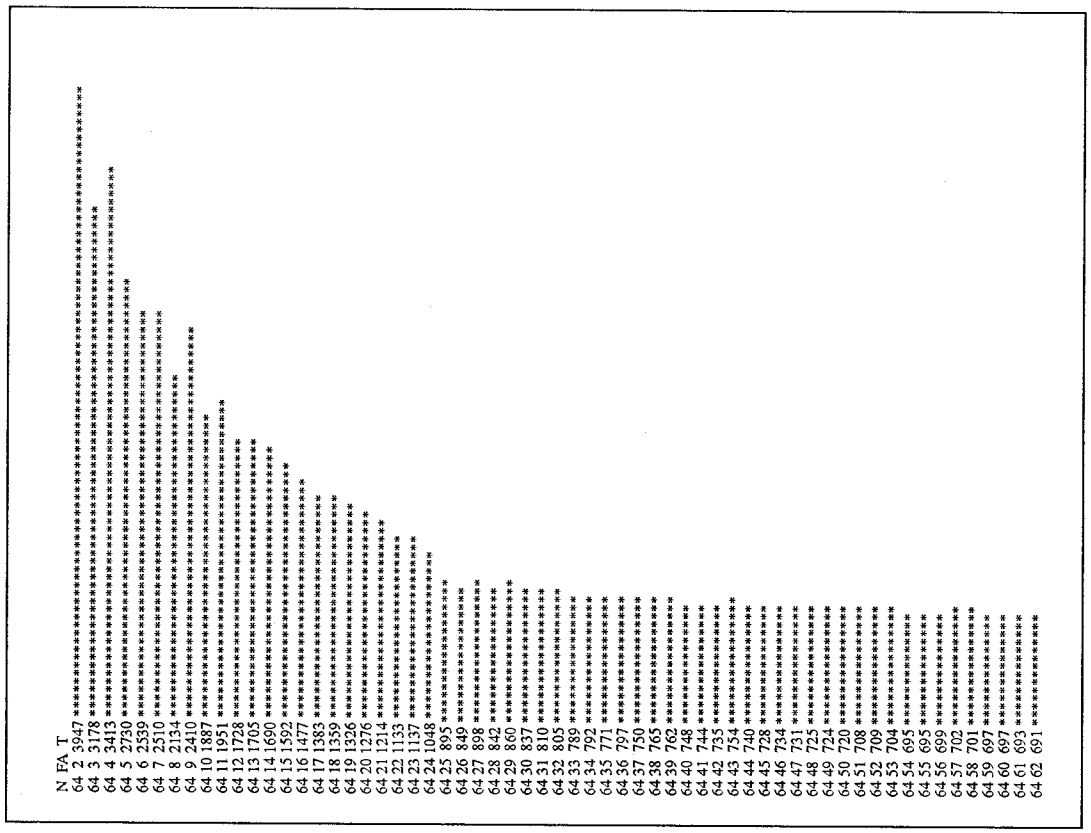
TIME=144



Réseau de taille 128



Réseau de taille 32



Réseau de taille 64

Temps de diffusion en fonction de la taille du réseau (N)
en utilisant des files d'attente de longueur 1

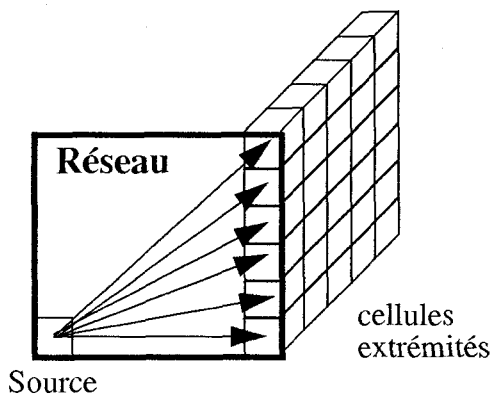
N	FA	T	
2	1	8	*
3	1	10	*
4	1	21	*
5	1	33	*
6	1	49	*
7	1	61	*
8	1	61	*
9	1	99	*
10	1	121	**
11	1	137	**
12	1	163	**
13	1	196	**
14	1	282	***
15	1	280	***
16	1	345	****
17	1	401	*****
18	1	381	****
19	1	440	*****
20	1	500	******
21	1	543	******
22	1	635	*******
23	1	645	*******
24	1	744	********
25	1	814	*********
26	1	887	*********
27	1	933	**********
28	1	993	**********
29	1	1029	**********
30	1	1155	**********
31	1	1223	**********
32	1	1315	**********
33	1	1376	**********
34	1	1536	**********
35	1	1529	**********
36	1	1602	**********
37	1	1735	**********
38	1	1793	**********
39	1	1942	**********
40	1	2121	**********
41	1	2108	**********
42	1	2264	**********
43	1	2352	**********
44	1	2482	**********
45	1	2627	**********
46	1	2767	**********
47	1	2823	**********
48	1	2943	**********
49	1	3138	**********
50	1	3330	**********
51	1	3397	**********
52	1	3604	**********
53	1	3538	**********
54	1	3846	**********
55	1	4049	**********
56	1	4096	**********
57	1	4133	**********
58	1	4432	**********
59	1	4541	**********
60	1	4550	**********
61	1	4829	**********
62	1	5071	**********
63	1	5202	**********
64	1	5537	**********
65	1	5246	**********
66	1	5761	**********
67	1	5805	**********
68	1	5740	**********
69	1	6161	**********
70	1	6548	**********
71	1	6495	**********
72	1	6511	**********
73	1	6975	**********
74	1	7260	**********
75	1	7391	**********
76	1	7723	**********
77	1	7637	**********
78	1	7908	**********
79	1	8109	**********
79	1	8109	**********

Temps de diffusion T en fonction de la taille du réseau N
en utilisant des files d'attentes de longueur maximale.

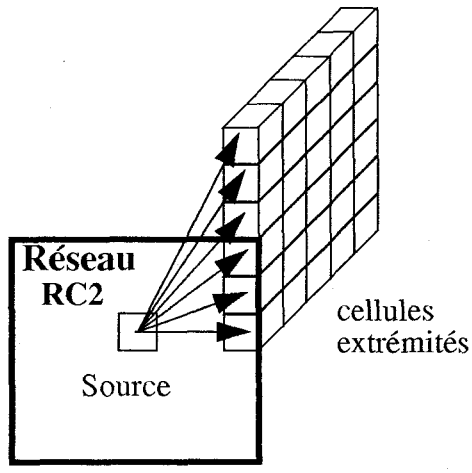
N	FA	T	
1	0	1	*
2	0	2	*
3	0	3	*
4	3	7	*
5	4	11	**
6	4	12	**
7	3	16	**
8	3	18	**
9	6	25	***
10	6	26	***
11	6	31	****
12	6	35	****
13	11	44	*****
14	11	45	*****
15	11	55	******
16	11	59	******
17	7	53	******
18	10	61	*******
19	18	81	*********
20	17	90	**********
21	16	87	*********
22	16	88	*********
23	15	101	***********
24	15	102	***********
25	23	125	************
26	23	126	************
27	23	133	*************
28	23	147	**************
29	23	144	**************
30	23	153	***************
31	19	145	**************
32	19	145	**************
32	19	145	**************
33	28	189	*******************
34	28	193	*******************
35	29	207	*********************
36	29	215	*********************
37	28	197	*******************
38	28	210	*********************
39	28	223	**********************
40	28	226	**********************
41	36	275	*****************************
42	36	281	*****************************
43	34	266	****************************
44	35	280	*****************************
45	35	292	******************************
46	35	304	******************************
47	35	306	******************************
48	35	317	*******************************
49	32	309	*******************************
50	32	313	*******************************
51	44	366	************************************
52	44	375	************************************
53	43	392	***
54	43	397	***
55	42	399	**
56	42	399	**
57	41	394	**
58	41	406	**
59	41	411	**
60	41	427	**
61	53	487	**
62	53	499	**
63	53	505	**
64	53	509	**

4•4•4 Le problème de l'arrêt de l'algorithme

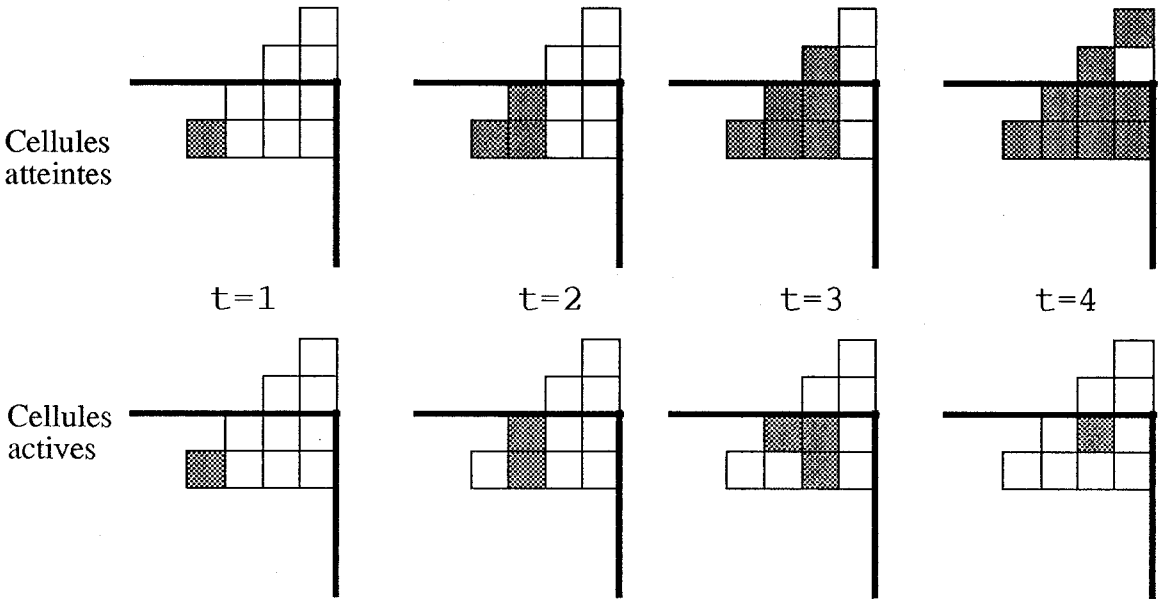
Pour chaque source de lumière, nous envoyons des faisceaux vers tout l'espace de la scène. Le fait d'envoyer des faisceaux à partir de plusieurs sources en même temps peut conduire à une situation de dead-lock. Afin d'éviter ceci, nous devons attendre que tous les faisceaux envoyés dans le réseau, à partir d'une source donnée, soient arrivés à destination et donc aient été consommés par le réseau, avant de lancer la diffusion à partir de la source suivante. Se pose alors le problème de savoir quand l'algorithme est terminé.
Si la source est située à l'origine, nous envoyons autant de rayons que de cellules extrémités et donc le contrôle de l'arrêt est simple puisque, lorsque chaque cellule extrémité a reçu son rayon, la diffusion pour la source en question est terminée.



Par contre, si la source est située en un point quelconque de l'espace, nous visons une extrémité virtuelle en dehors du réseau et une cellule appartenant au bord du réseau peut recevoir plus d'un rayon.



Il se peut donc que, bien que toutes les cellules du bord soient atteintes, des cellules du réseau soient encore actives (celles par lesquelles transitent des rayons dont la destination est extérieure au réseau). Ceci est illustré par l'exemple suivant dans lequel nous montrons l'évolution des cellules atteintes et celle des cellules actives au cours du temps.



Nous ne savons pas déterminer a priori le nombre de faisceaux qui vont transiter dans le réseau. Le seul moyen de savoir quand l'algorithme est terminé est de regarder l'état global du réseau. Lorsque plus aucune cellule ne travaille, c'est que tous les faisceaux sont parvenus à destination. Ceci oblige donc à avoir un mécanisme centralisé permettant de déterminer l'état d'activité du réseau et n'interférant pas avec l'algorithme de diffusion.

4•4•5 Conclusion

Nous avons proposé une méthode de calcul des **rayons d'ombre** dans un espace 3D discrétisé en diffusant les rayons à partir de chaque source lumineuse vers tout l'espace. Cette méthode offre certains avantages par rapport à la méthode de lancer de rayons classique: indépendance par rapport à la complexité de la scène, calcul d'intersection rayon/objets plus simple. La méthode utilise le concept de **faisceau lumineux** qui permet de traiter en une seule fois un paquet de rayons de directions proches, accélérant ainsi l'algorithme de diffusion. Les simulations ont fait apparaître certains conflits d'accès dûs au fait que des faisceaux distincts peuvent emprunter localement les mêmes chemins et ont montré que l'utilisation d'un réseau 2D pour traiter un espace 3D engendre des conflits supplémentaires. Toutefois, en introduisant des files d'attente pour mémoriser les faisceaux en attente à l'intérieur de chaque cellule, la durée de diffusion est de l'ordre de n.log n (n étant la taille du réseau). Enfin il reste encore à résoudre certains problèmes comme celui de l'éclairage multiple ainsi que celui de l'arrêt de l'algorithme.

CHAPITRE 5

Intersection rayon / surface

TABLE DES MATIERES

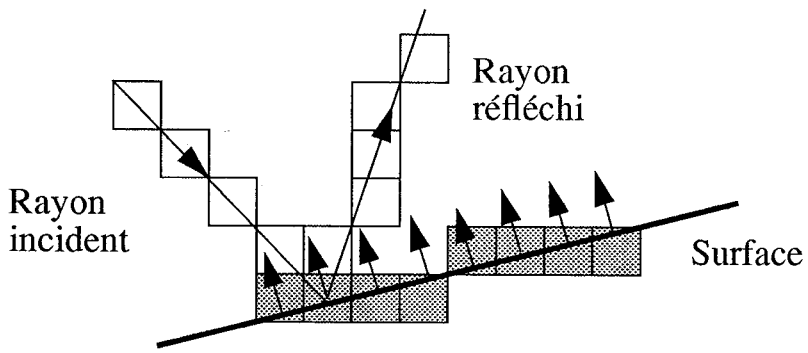
CHAPITRE 5 : Intersection rayon/surface

5.1 Position du problème

A cause de la discrétisation des surfaces et des rayons, l'intersection dans le cas discret n'est pas toujours analogue à celle effectuée dans le cas continu.
Nous allons essayer de mettre en évidence les nouveaux problèmes auxquels nous sommes confrontés. Nous avons déjà mentionné celui de la discrétisation des surfaces, qui nécessite un algorithme garantissant l'absence de trous à l'intérieur de celles-ci. Nous avons également vu que selon le type de suivi de rayons utilisé, les résultats obtenus n'étaient pas les mêmes.
Nous allons maintenant nous préoccuper du calcul de la réflexion des rayons sur une surface. Nous allons étudier différentes méthodes pour déterminer l'intersection rayon/ surface ainsi que le rayon réfléchi par celle-ci et nous mettrons en évidence les problèmes rencontrés.

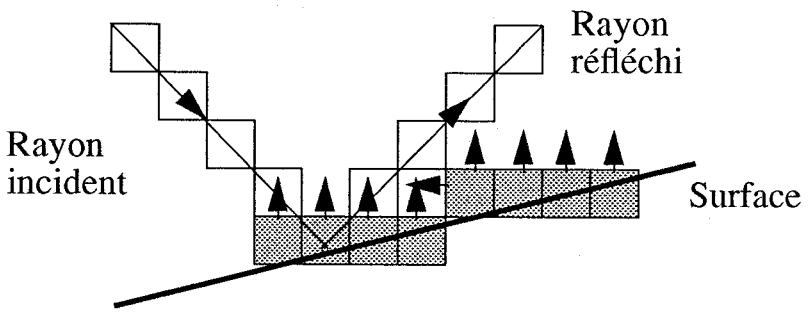
5.2 Calcul du rayon réfléchi - choix de la normale au point d'intersection

5•2•1 Utilisation de la normale à la surface réelle



On mémorise la normale à la surface dans chaque cellule.

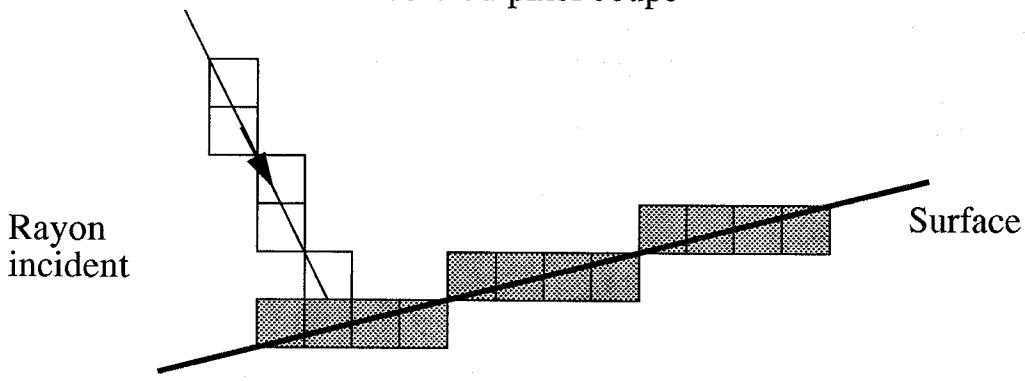
5•2•2 Utilisation de la normale au bord du pixel coupé par le rayon



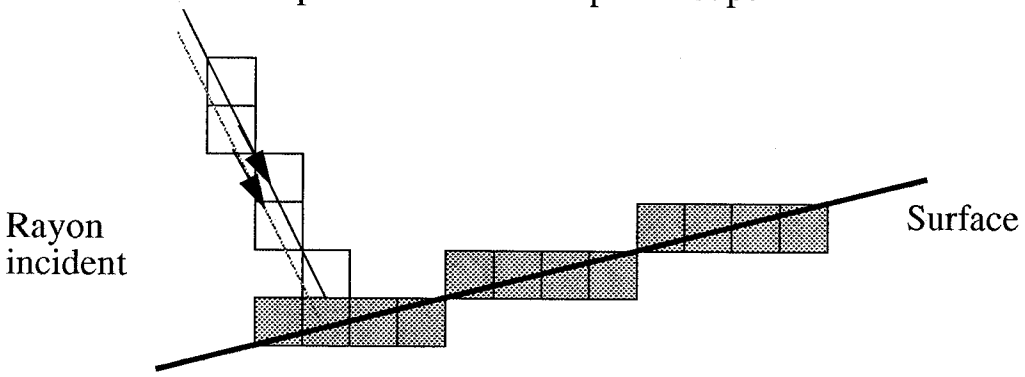
Pas de mémorisation de normale, ne dépend que de la direction incidente du rayon.

5.3 Le choix du calcul de l'intersection entre le rayon et la surface

5•3•1 Intersection avec le bord du pixel coupé



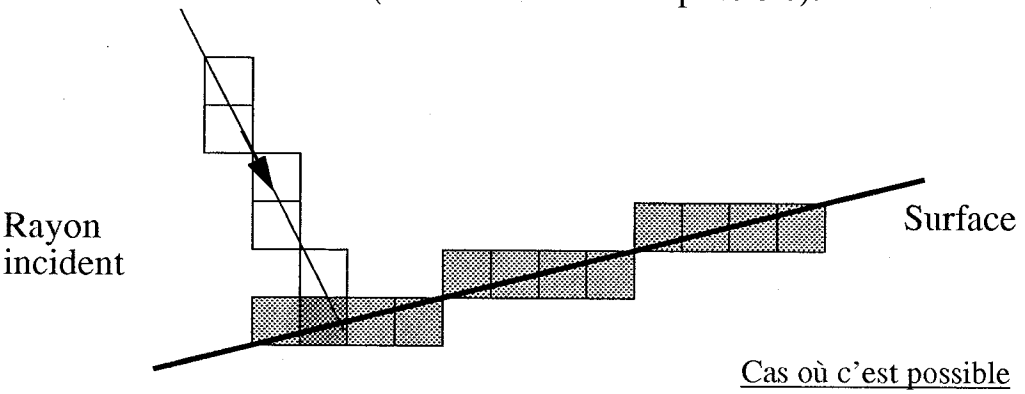
5•3•2 Intersection reportée au centre du pixel coupé



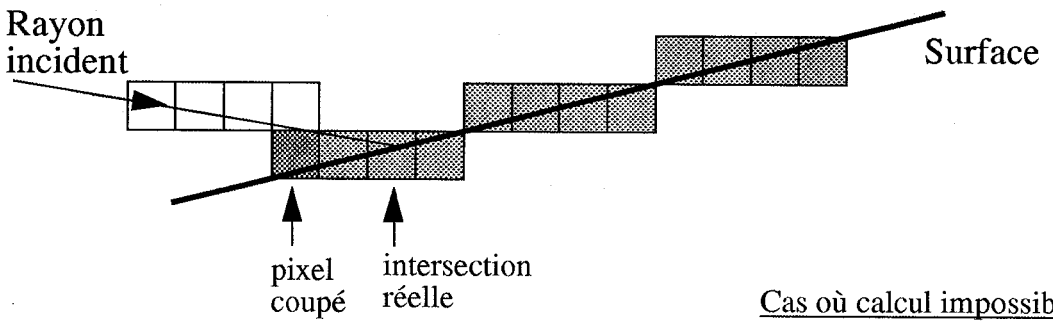
-> Calculs plus simples

->Une "légère" erreur est commise. Le rayon incident est remplacé par un rayon parallèle au précédent, passant par le centre du pixel

5•3•3 Intersection exacte (dans le cas où c'est possible).



Cas où c'est possible

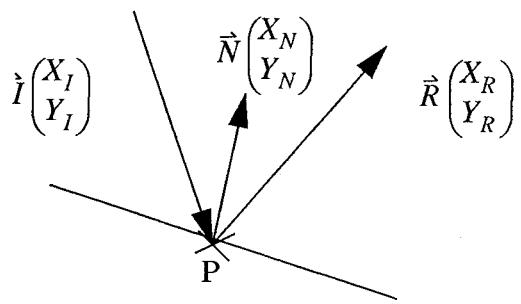


Cas où calcul impossible

L'intersection réelle peut ne pas se trouver dans le pixel coupé.

5.4 Calcul du rayon réfléchi

Dans tous les cas, à partir du point d'intersection P, on calcule le rayon réfléchi R du rayon incident I en utilisant le vecteur normal N à la surface ou au bord du pixel coupé.



En supposant N normé

On a $\vec{R} - \vec{I} = 2\|\vec{I}\|.\cos\theta.\vec{N}$

d'où $\vec{R} = 2\cos\theta.\|\vec{I}\|.\vec{N} + \vec{I}$ avec $\cos\theta = \dfrac{\vec{N}\cdot\vec{I}}{\|\vec{I}\|}$

soit $\vec{R} = 2\,(\vec{N}\cdot\vec{I})\,.\vec{N} + \vec{I}$

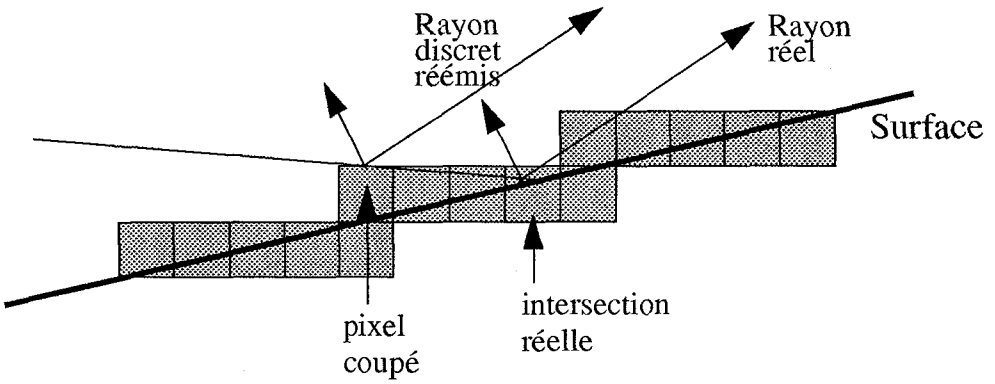
et donc $\begin{cases} X_R = 2\,(X_N.X_I + Y_N.Y_I)\,.X_N + X_I \\ Y_R = 2\,(X_N.X_I + Y_N.Y_I)\,.Y_N + Y_I \end{cases}$

5.5 Problèmes liés à la discrétisation

Nous avons déjà parlé du phénomène "passe-muraille" qui nécessite d'utiliser une connectivité adéquate pour la construction des surfaces et le suivi des rayons. Nous allons maintenant, pour chacun des paramètres précédents (mode de calcul de l'intersection, choix de la normale) noter les avantages et les inconvénients des choix possibles et en particulier nous détaillerons les erreurs introduites ainsi que les problèmes résolus par chacune de ces méthodes.

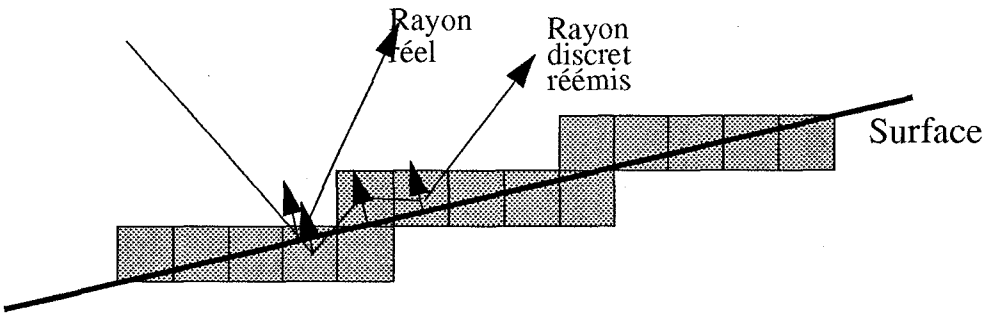
5•5•1 Intersection erronée

Nous avons vu que l'intersection rayon/surface dans le cas discret pouvait être fortement éloignée de l'intersection réelle. Ce qui a pour conséquence de générer un rayon réfléchi décalé par rapport au rayon réfléchi dans le cas réel:

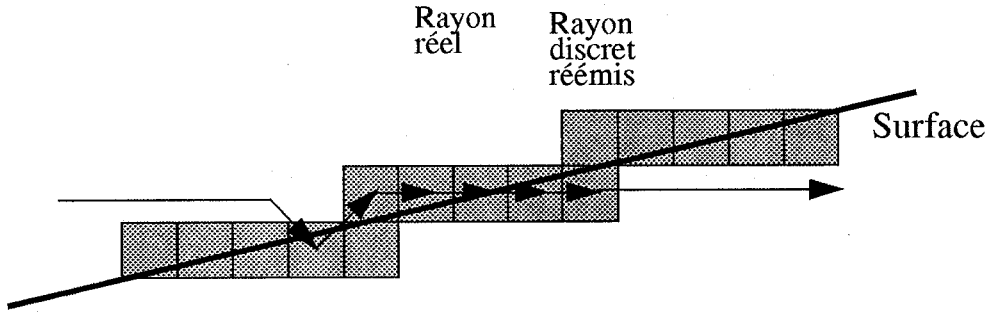


5•5•2 Collision rayon réfléchi / surface

Le rayon réfléchi peut également, du fait de la discrétisation, rencontrer à nouveau la surface dont il provient. Si nous n'en tenons pas compte, ceci peut dévier fortement le rayon réfléchi par rapport à sa direction réelle.

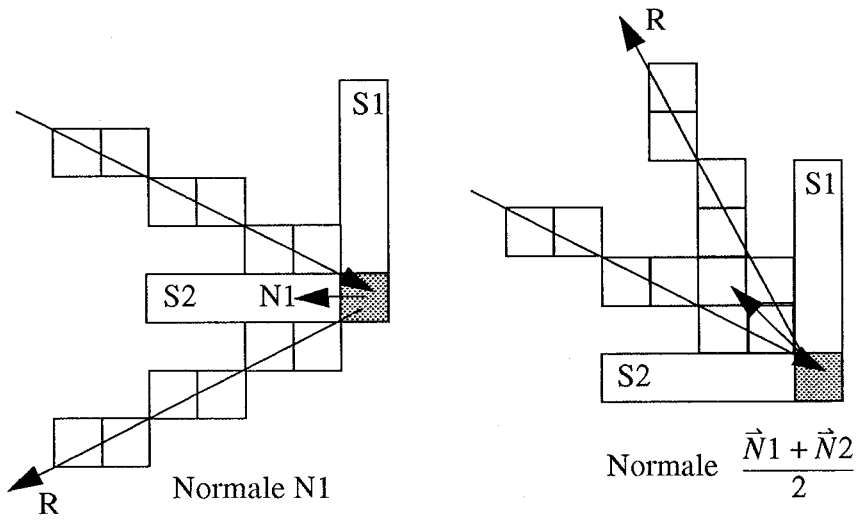


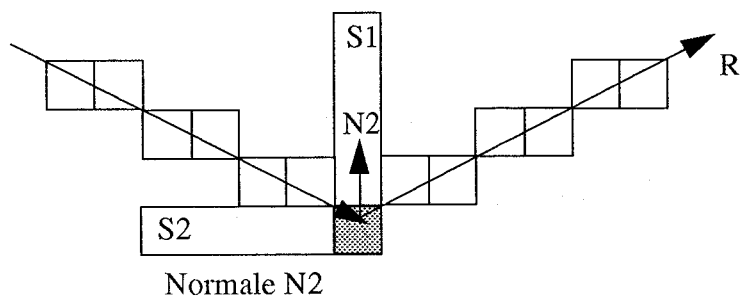
Dans certains cas défavorables, le rayon au lieu d'être réfléchi peut être guidé à l'intérieur de la surface et ressortir du mauvais côté, c'est à dire que des rayons peuvent ainsi traverser une surface même si celle-ci est blindée (construite en connectivité 4 voisins).



5•5•3 Intersection de deux facettes qui se touchent

Un pixel peut appartenir simultanément à deux surfaces (S_1, S_2) d'orientations différentes (N_1,N_2). Si nous mémorisons, pour chaque pixel, la normale à la surface à laquelle il appartient, il est alors difficile, dans le cas présent, de déterminer cette normale car selon la valeur utilisée (N_1,N_2 ou valeur moyenne), les résultats varient fortement:

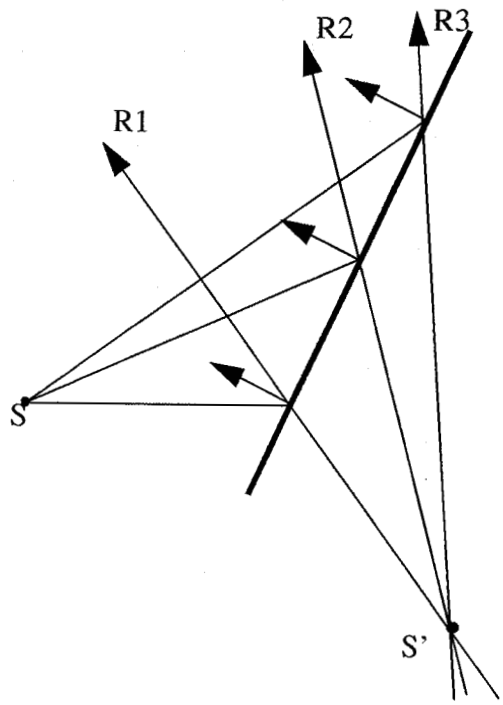




5•5•4 Choix de la normale utilisée

Nous avons étudié la réflexion sur une surface de rayons émis par une source lumineuse et ceci pour chacun des choix possibles de normale. Nous observons alors les différents résultats suivants:

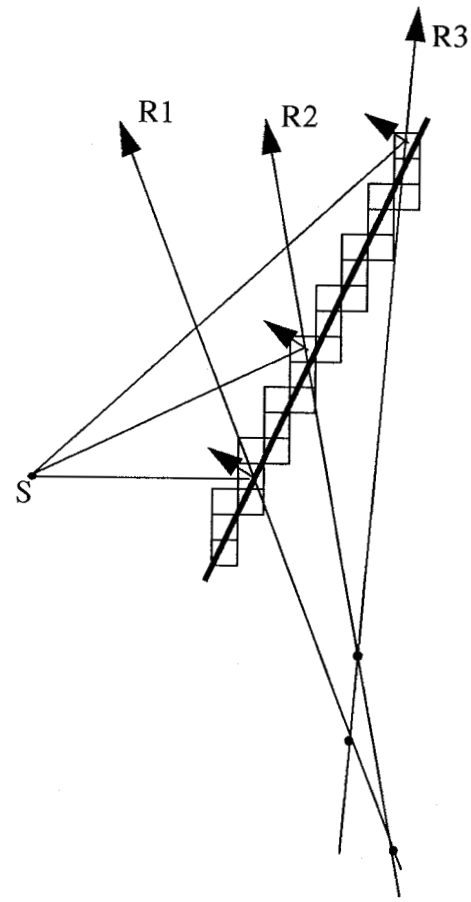
Cas 1



Vraie surface
Vraie normale

Les rayons réfléchis semblent provenir d'un point virtuel unique S'.

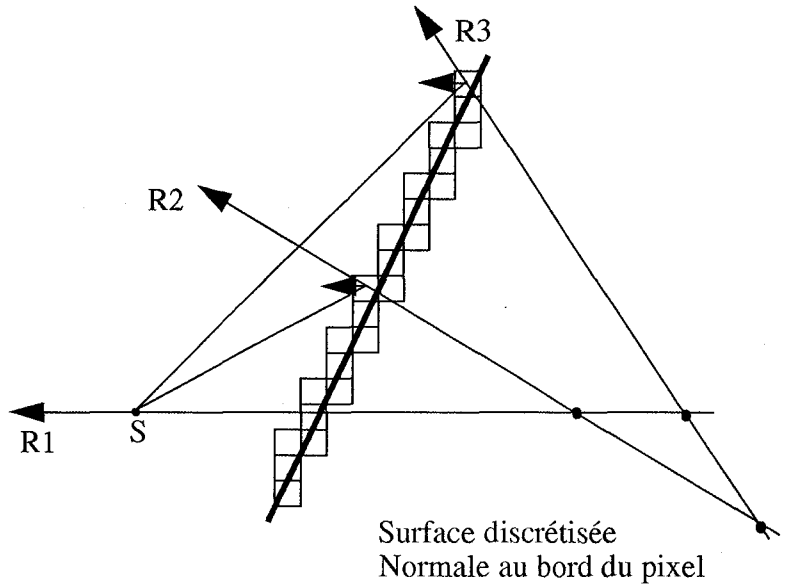
Cas 2



Surface discrétisée
Vraie normale

L'erreur due à l'approximation de la surface implique que les rayons réfléchis ne semblent pas provenir d'un point virtuel unique mais d'un nuage de points situés autour du vrai point

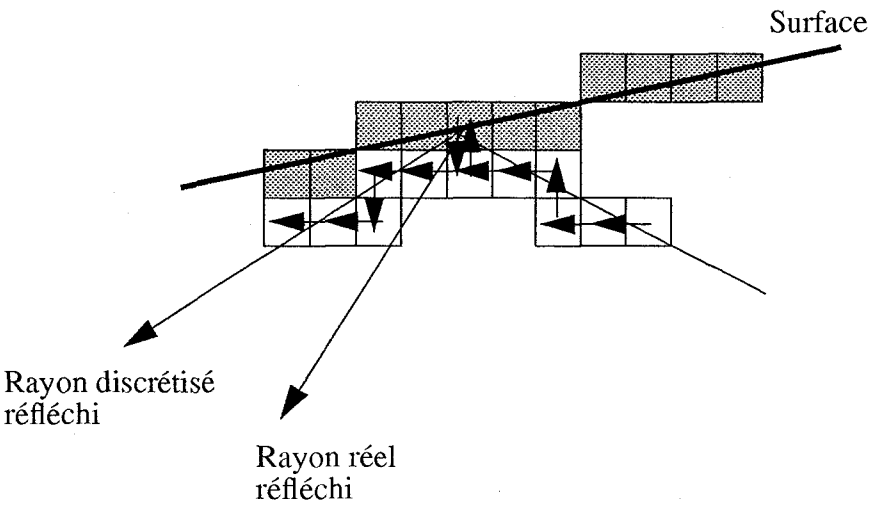
Cas 3

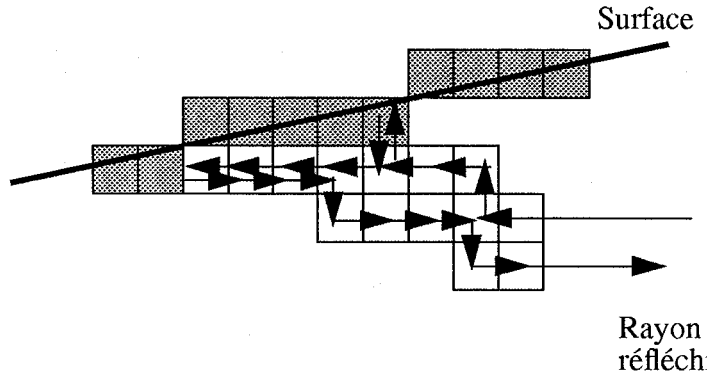


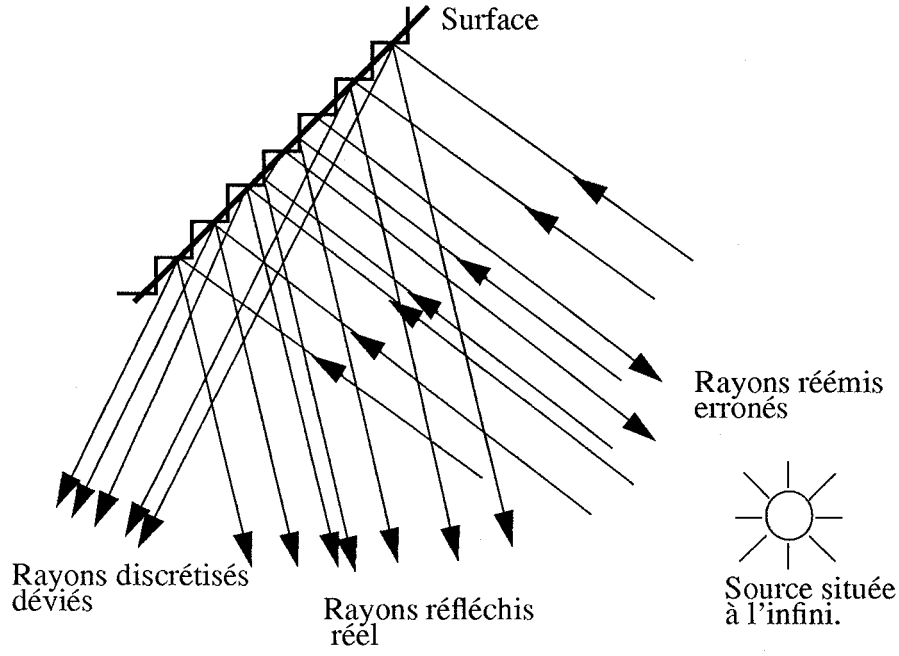
Les rayons réfléchis ne semblent pas provenir d'un point unique mais d'un nuage de points. De plus, le nuage de points virtuel duquel proviennent les rayons réfléchis peut être assez éloigné du vrai point virtuel. Il peut même y avoir plusieurs nuages distincts.

Conclusion: Pour une source ponctuelle, la discrétisation d'une surface donne un rendu tout à fait distinct du rendu appliqué à la surface initiale.

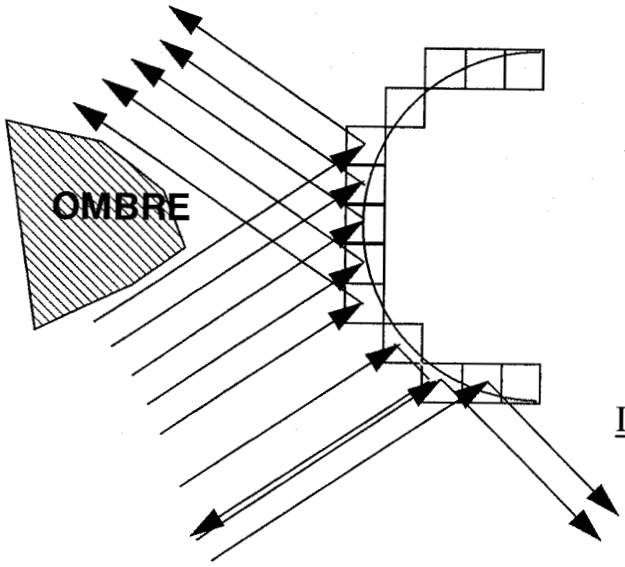
L'avantage du choix de la normale au bord du pixel coupé est de nous dispenser de mémoriser une valeur de normale dans chacun des pixels appartenant à la surface. Cependant nous nous sommes rendus compte que les rayons réfléchis divergent fortement des rayons réels qui auraient été réfléchis. Certains peuvent même être renvoyés vers la source, comme si la surface était en partie éclairée "de face":



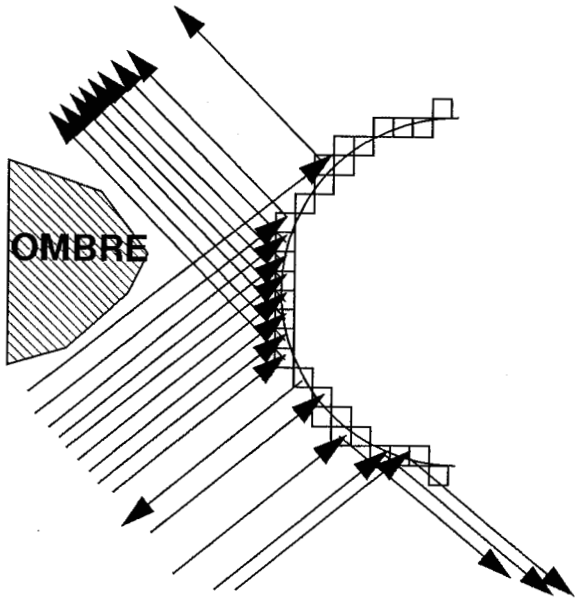




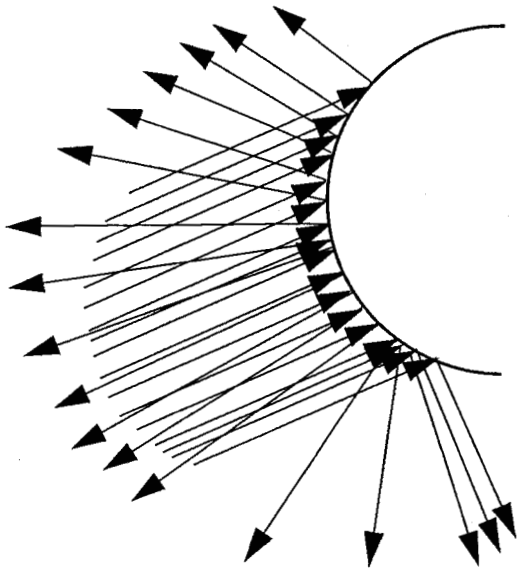
Nous avons également testé différents niveaux de discrétisation plus ou moins fins. Dans chaque cas, nous remarquons une zone d'ombre vers laquelle aucun rayon n'est réfléchi alors que dans le cas continu, tout l'espace serait éclairé:



Discrétisation grossière

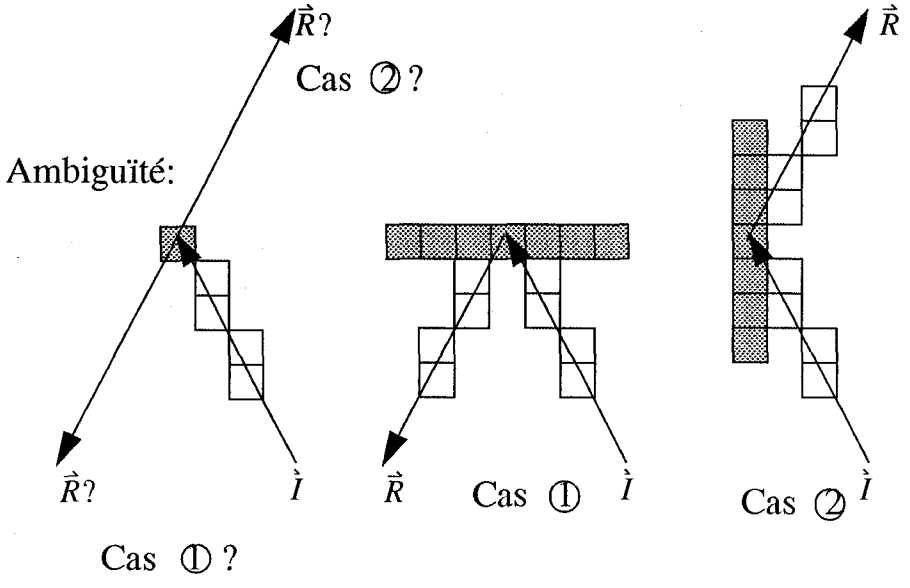


Discrétisation plus fine

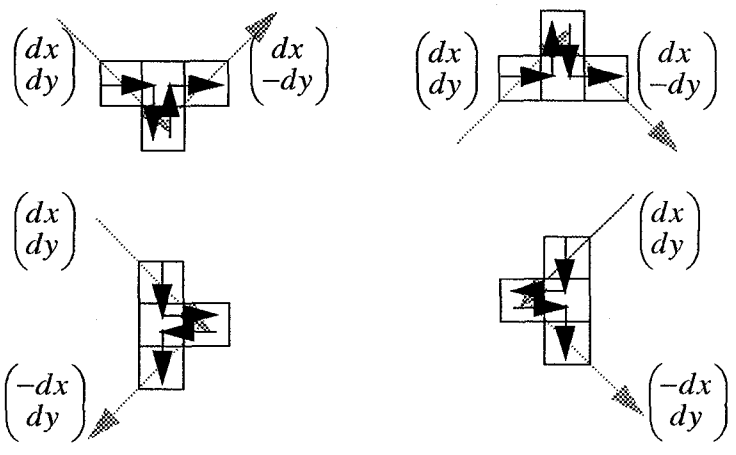


Cas réel

La connectivité 8 voisins pour le suivi de rayons introduit, dans ce cas, des erreurs. En effet, un pixel d'une surface qui reçoit un rayon lui arrivant par le coin d'un pixel voisin, ne peut déterminer de façon correcte le pixel vers lequel il doit réémettre le rayon comme le montre l'exemple suivant, car la réflexion dépend de l'orientation de la surface. Or nous n'avons aucun moyen de connaître celle-ci puisque nous sommes dans l'hypothèse où la normale n'est pas mémorisée.

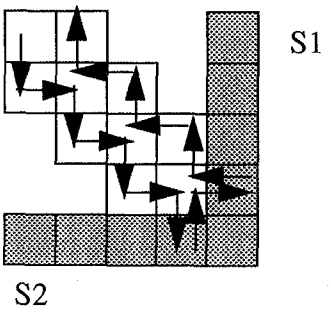


Ce problème peut alors être résolu en utilisant un suivi de rayons en connectivité 4 voisins. Les différents cas de réémission sont les suivants:



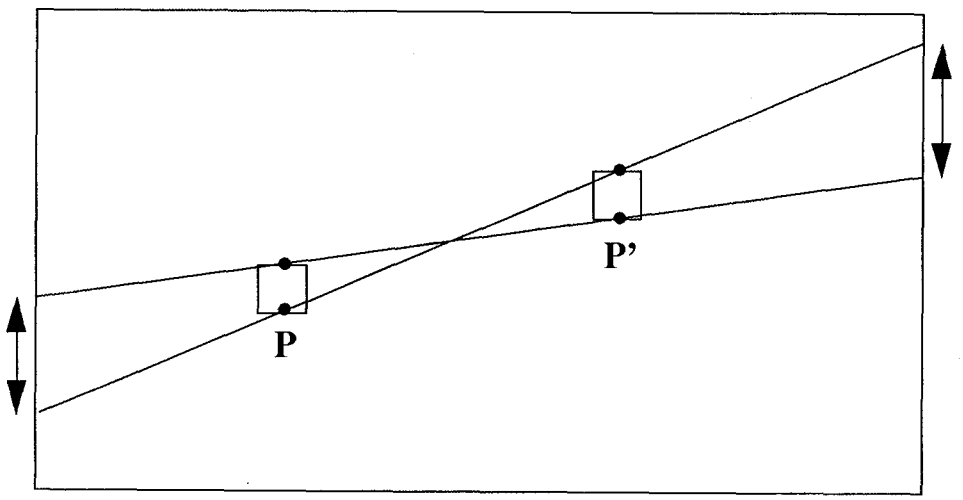
L'avantage est que le calcul de réfléchi est très simple puisqu'il suffit de changer le signe de l'une des composantes du vecteur incident.
Ceci permet également de remédier au problème des pixels appartenant à plusieurs

surfaces.

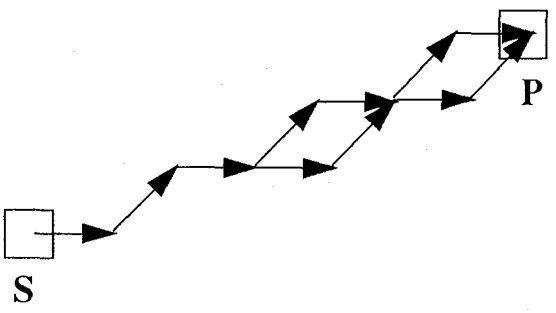


5•5•5 Problème de l'éclairage multiple

Un pixel peut être atteint par plusieurs rayons en provenance de la même source, car dans un espace discret, par deux pixels donnés, il passe plusieurs droites au sens de Bresenham.
Si l'espace est infini, par deux pixels il passe un nombre infini de droites rationnelles.
Si l'espace est fini, par deux pixels P et P' il passe un nombre fini de droites rationnelles (1 ou plusieurs).

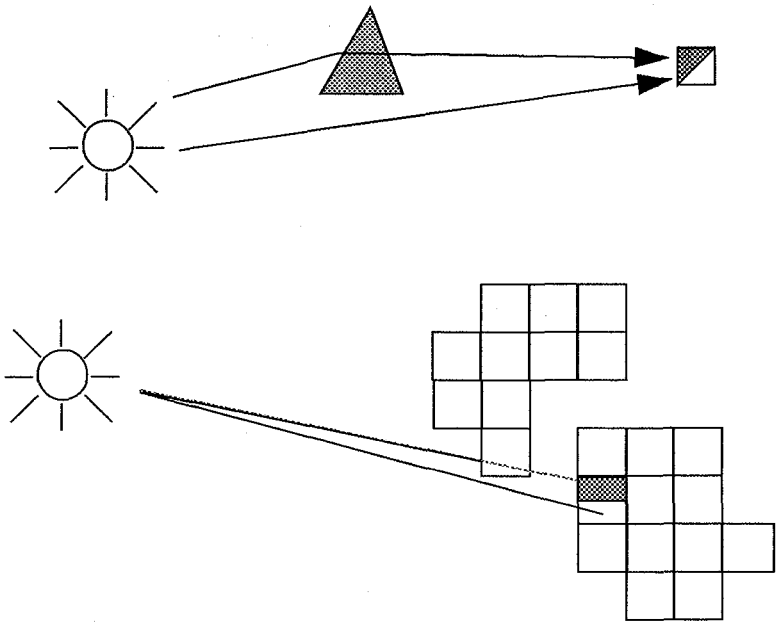


Pour un pixel donné P et une source donnée S il peut donc y avoir deux demi-droites discrètes différentes d'origine la source S et passant par le pixel P.



Supposons qu'un pixel reçoive deux rayons en provenance de la source.

Si, sur le trajet de l'un des deux rayons, nous plaçons un objet opaque, le précédent pixel va recevoir deux informations contradictoires en provenance de la source de lumière. L'une lui indiquera qu'il est à l'ombre, et l'autre lui assurera le contraire.
Quelle stratégie adopter alors pour déterminer l'éclairage correct?

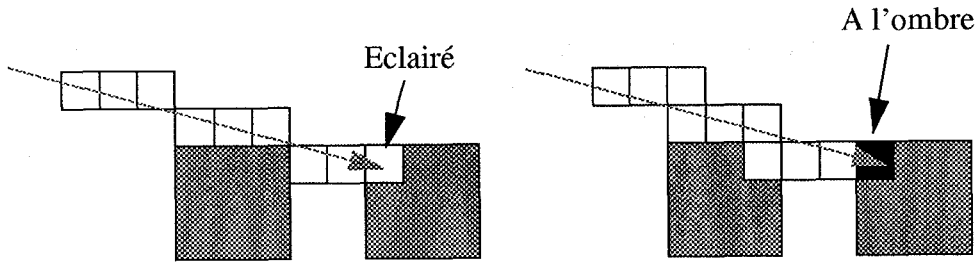


La première solution sera de ne considérer que le rayon issu du centre du pixel source au centre du pixel cible et de dire qu'un pixel sera éclairé si ce rayon ne coupe aucun objet de la scène situé entre la source et lui-même. Ce qui entraîne le calcul exact de tous les rayons source-pixels possibles. Or nous avons vu que ceci génère un nombre trop important de rayons, et est incompatible avec notre algorithme de diffusion de rayons.

Une autre technique sera de dire qu'un pixel sera éclairé si plus de 50% de sa surface est éclairé, ce qui fait intervenir des calculs de surface d'éclairage beaucoup trop compliqués. On pourrait également calculer le rapport entre le nombre de rayons "éclairés" reçus et le nombre de rayons total reçus par le pixel puis déclarer le pixel éclairé si ce terme est supérieur à 0.5.

Il n'y a en fait aucune solution réellement satisfaisante et, par souci de simplification, nous supposerons un pixel éclairé s'il reçoit au moins un rayon lumineux.

Notons également que selon l'algorithme de suivi de rayons utilisé, l'éclairement pourra différer comme le montre l'exemple suivant.

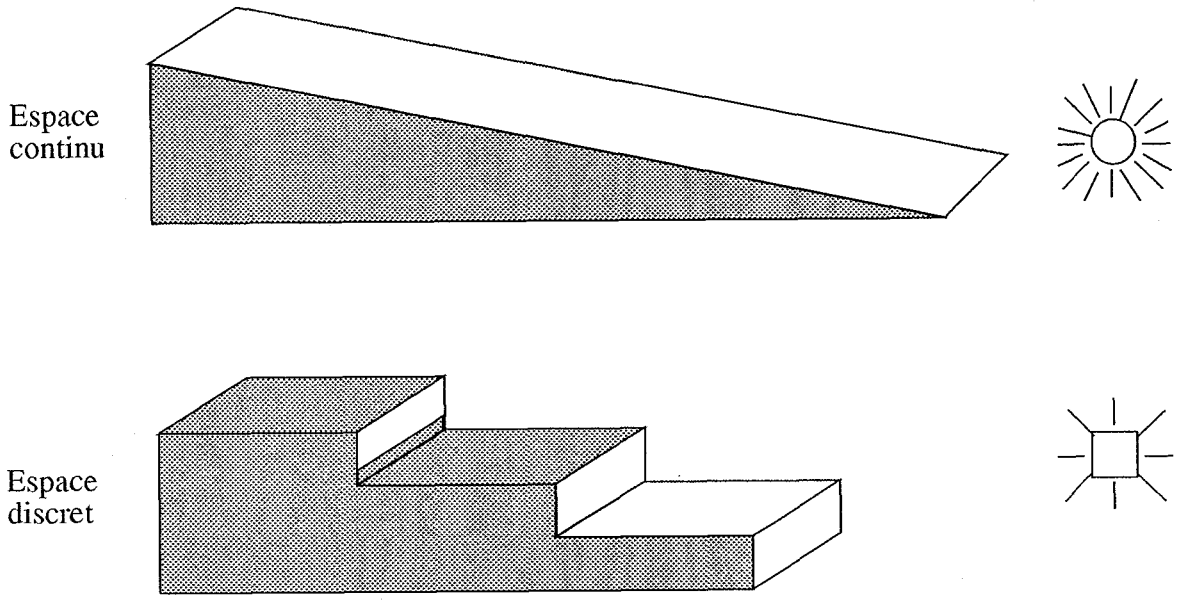


(1) Suivi de rayons de type Bresenham

(2) Suivi de rayons Amanatides & Woo

5•5•6 Eclairage rasant

Enfin, un des problèmes les plus difficiles à résoudre est celui de l'apparition de zones d'ombre dues à la discrétisation des surfaces. Les surfaces éclairées en lumière rasante donneront des résultats fort différents de la réalité:



Problèmes en lumière rasante

Conclusion

Conclusion

Après avoir passé en revue les différentes méthodes qui permettent d'accélérer la technique du lancer de rayons, nous avons proposé une architecture massivement parallèle RC^2 basée sur un réseau cellulaire et une discrétisation de la scène en voxels. Nous avons abordé les problèmes topologiques qui se posent sur ce genre d'architecture nouvelle qui s'appuie essentiellement sur le "tout discret". Nous avons donné plusieurs algorithmes permettant de tracer des segments dans un espace 3D discret. Ces algorithmes, obtenus à partir des algorithmes 2D existants, forment les briques de base de tout algorithme discret 3D. Nous avons ensuite parlé de la discrétisation des facettes et nous avons proposé un algorithme de découpage en "dexels" d'une facette polygonale plane convexe quelconque. L'étude topologique précédemment citée nous a conduit à construire des surfaces discrètes possédant la caractéristique d'empêcher les rayons de passer au travers (effet passe-muraille). Nous avons ensuite montré comment implanter cet algorithme sur notre architecture RC^2.

Nous avons également proposé une nouvelle méthode pour calculer l'éclairage en tout point de la scène et en avons comparé les performances avec celles de l'algorithme de lancer de rayons classique. Cette méthode est basée sur la diffusion de rayons à partir des sources lumineuses vers les objets et permet de reproduire plus fidélement le comportement physique de la lumière. Nous avons exposé différentes possibilités pour diffuser ces rayons à travers l'espace constituant la scène et en avons retenu une: la diffusion par subdivision de faisceaux. Nous avons ensuite donné l'algorithme correspondant et son implémentation sur RC^2. Les simulations effectuées ont fait apparaître des problèmes de conflit d'accès aux cellules et nous avons montré comment améliorer les performances en modulant la taille des files d'attente internes aux cellules. Les résultats sont encourageants puisqu'ils font ressortir une complexité temporelle en N log N. Nous avons toutefois soulevé le problème de l'arrêt de l'algorithme qui ne peut se faire sans l'aide d'un mécanisme centralisé.

Dans le dernier chapitre, nous avons fait une étude exhaustive des problèmes nouveaux liés à la discrétisation des rayons et des surfaces et plus particulièrement concernant l'intersection entre un rayon et une surface. En espace discret, les résultats des théorèmes valables dans les espaces continus ne peuvent pas toujours s'appliquer et il faut trouver des solutions nouvelles pour que la discrétisation du phénomène continu reste cohérente avec celui-ci.

Malgré tous les problèmes posés, l'architecture RC^2est assez bien adaptée à ce genre d'approche puisque les algorithmes exposés qui s'appliquent à une architecture 3D générale peuvent s'implémenter aisément sur RC^2, et lorsque les problèmes théoriques seront tous résolus, on peut espérer obtenir des performances intéressantes.

Références bibliographiques

REFERENCES BIBLIOGRAPHIQUES

[AMA 84]
J. Amanatides
" Ray tracing with cones "
Computer graphics vol 18 N°3 Juillet 1984

[AMA 87]
J. Amanatides & A. Woo
" A fast voxel traversal algorithm for ray tracing "
Eurographic's 87 pp 3-12

[AML 90]
A. Atamenia - M. Meriaux - E. Leprêtre - S. Degrande - B. Vidal
" A cellular architecture for ray tracing "
5th Eurographics workshop on graphics hardware Lausanne, 2-3 sept 90 à paraître chez Springer-Verlag

[ATA 89]
A. Atamenia
" Architectures cellulaires pour la synthèse d'images "
Thèse de Doctorat Université de Lille I Juin 1989

[BER 87]
J. Berstel
" Tracé de droites, fractions continues et morphismes itérés "
paru dans "Mots", mélanges offerts à M.P. Schutzenberger,
Editions Hermes

[BRE 65]
J.E. Bresenham
"Algorithm for Computer Control of a Digital Plotter"
IBM Systems Journal, 4 (1), 1965, pp 25-30

[CAU 88]
R. Caubet
" Voxar : a tridimensional architecture for fast realistic image synthesis "
New trends in computer graphics proceedings of CG International' 88

[CAU 89]
R. Caubet
" Une modélisation unifiée intégrant des informations locales : la discrétisation en atomes "
Actes de Pixim 1989

[CAU 89]
R. Caubet
" Le suivi analytique de rayons : un algorithme incrémental rapide pour la machine Voxar "
Actes MICAD 89 pp 653-664

[CAU 90]
R. Caubet
" A transputer based implementation of the VOXAR project "
Microprocessing and microprogramming 1990

[DUR 83]
Ph. Durif
" Etude d'une machine parallèle de synthèse d'images à découpage par objets "
Thèse de docteur 3ème cycle Université de Lille I 08/12/83

[FIE 85]
D.E. Field
"Incremental Linear Interpolation"
ACM Transaction on Graphics Vol 4 No 1 01/85

[FOL 90]
J.D. Foley - A. Van Dam - S.K. Feiner - J.F. Hughes
" Computer graphics principles and practice " Second edition
Addison - Wesley 1990

[FUJ 86]
A. Fujimoto
" ARTS : Accelerated Ray Tracing System "
IEEE Computer graphics and applications Avril 1996

[GLA 89]
A.S. Glassner
"An introduction to Ray Tracing "
Academic Press 1989

[HEC 84]
P.S. Heckbert - P. Hanrahan
" Beam tracing polygonal objects "
Computer graphics Vol 18, N^{0}3 Juillet 1984

[KAU ??]
D. Cohen - A. Kaufman
" Scan-conversion algorithms for linear and quadratic objects "
Volume visualization IEEE Computer Society Press

[KAU 87]
A. Kaufman
" An algorithm for 3D scan-conversion of polygons "
Eurographics 1987

[KAU 88]
A. Kaufman, R. Bakalash
" Memory and processing architecture for 3D voxel-based imagery "
CG & A , 8(6) , Novembre 1988, pp 10-23

[LEP 89]
E. Leprêtre
" Algorithmes parallèles et architectures cellulaires pour la synthèse d'images "
Thèse de Doctorat Université de Lille I Juin 1989

[LOC 80]
" The line "
Computer Juin 80 pp 57

[MAD 89]
M. Meriaux - A. Atamenia - E. Dufresne - P. Durif - E. Leprêtre - A. Serhrouchni - B. Vidal
" Architectures parallèles pour la synthèse d'images "
Journées graphiques " Groplan " Décembre 1989 Strasbourg
Publié dans Bigre + Globule - 1990

[MER 84]
M. Meriaux
" Contributions à l'imagerie informatique : aspects algorithmiques et architecturaux "
Thèse de Docteur es Sciences mathématiques Lille 07/84

[MER 89]
M. Meriaux - B. Vidal
" Line and circle drawing on a boolean neural network "
CAD & CG 89, Beijing Août 1989

[PEL 85]
M. Pelerin
"Synthèse d'images et parallélismes: algorithmes et architectures"
Thèse Docteur 3e cycle Lille 03/01/85

[PER 88]
B. Peroche - J. Argence - D. Ghazanfarpour - D. Michelucci
" La synthèse d'images "
Editions Hermes 1988

[PER 89]
G. Fertey - B. Peroche
" Sources directionnelles de lumière en tracé de rayons "
Actes de Pixim 1989

[PHO 75]
B.T. Phong
" Illumination for computer generated pictures "
Communication of the ACM Juin 1975, Vol 18, N°6

[REV 88]
J.P. Reveilles
" Les paliers de droites de Bresenham "
Actes de Pixim 1988 Ed. Hermes 10/88

[ROS 89]
T.Y. Kong - A. Rosenfeld
" Digital topology : Introduction and survey "
Computer vision, graphics, and image processing N 48 (1989) pp 357-393

[UZU 89]
D. Uzun
" Simulation de réseaux neuronaux appliqués aux tracés de droites "
Mémoire de DEA 1989

[VAN 88]
M.L.P. Van Lierop - C.W.A.M. Van Overveld - H.M.M. Van de Wetering
" Line rasterization algorithms that satisfy the subset line "
Computer vision, graphics, and image processing 41 (1988) pp 210-218

[VID 91]
B.Vidal - E Leprêtre - M. Meriaux
"Voxelization of convex planar polygonal facets in 3D space"
Compugraphics'91 - First International Conference on Computational Graphics and Visualization techniques , 16-20 Sept 1991 Sesimbra, Portugal

[WHI 80]
T. Whitted
" An improved illumination model for shaded display "
Communication of the ACM Juin 1980, Vol 23, N°6

[WRI 91]
W.E. Wright
" Parallel algorithms for generating the raster representation of straight lines and circles "
Journal of parallel and distributed computing 11 (1991) pp 170-173