Représentation de paysages et tracé de rayon

Sabine Coquillart

ECOLE NATIONALE
SUPERIEURE DES MINES
DE SAINT-ETIENNE

INSTITUT NATIONAL
POLYTECHNIQUE
DE GRENOBLE

N°d'ordre: 43 I S

THESE

présentée par

Sabine COQUILLART

pour obtenir le diplôme de

DOCTEUR DE 3 ème CYCLE
EN INFORMATIQUE

REPRESENTATION DE PAYSAGES ET TRACE DE RAYON

Soutenue à Saint-Etienne le 3 Décembre 1984
devant la Commission d'Examen

Président	M.	J.C. LATOMBE
Examinateurs	MM.	G. ALLAIN
		M. GANGNET
		M. HABIB
		M. LUCAS
		F. MARTINEZ

ECOLE NATIONALE
SUPERIEURE DES MINES
DE SAINT-ETIENNE

INSTITUT NATIONAL
POLYTECHNIQUE
DE GRENOBLE

N° d'ordre: 43 I S

THESE

présentée par

Sabine COQUILLART

pour obtenir le diplôme de

DOCTEUR DE 3 ème CYCLE
EN INFORMATIQUE

REPRESENTATION DE PAYSAGES ET TRACE DE RAYON

Soutenue à Saint-Etienne le 3 Décembre 1984
devant la Commission d'Examen

Président	M.	J.C. LATOMBE
Examinateurs	MM.	G. ALLAIN
		M. GANGNET
		M. HABIB
		M. LUCAS
		F. MARTINEZ

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1983-1984

résident: Daniel BLOCH

ice-Présidents: René CARRE
Hervé CHERADAME
Jean-Pierre LONGEQUEUE

Professeur des Universités

NCEAU	François	E.N.S.I.M.A.G
ARIBAUD	Michel	E.N.S.E.R.G
ARRAUD	Alain	E.N.S.I.E.G
AUDELET	Bernard	E.N.S.I.E.G
ESSON	Jean	E.N.S.E.E.G
LIMAN	Samuel	E.N.S.E.R.G
LOCH	Daniel	E.N.S.I.E.G
OIS	Philippe	E.N.S.H.G
ONNETAIN	Lucien	E.N.S.E.E.G
ONNIER	Etienne	E.N.S.E.E.G
OUVARD	Maurice	E.N.S.H.G
RISSONNEAU	Pierre	E.N.S.I.E.G
UYLE BODIN	Maurice	E.N.S.E.R.G
AVAIGNAC	Jean-François	E.N.S.I.E.G
HARTIER	Germain	E.N.S.I.E.G
HENEVIER	Pierre	E.N.S.E.R.G
HERADAME	Hervé	U.E.R.M.C.P.P
HERUY	Arlette	E.N.S.I.E.G
HIAVERINA	Jean	U.E.R.M.C.P.P
OHEN	Joseph	E.N.S.E.R.G
OUMES	André	E.N.S.E.R.G
URAND	Francis	E.N.S.E.E.G
URAND	Jean-louis	E.N.S.I.E.G
ELICI	Noël	E.N.S.I.E.G
ONLUPT	Jean	E.N.S.I.M.A.G
OULARD	Claude	E.N.S.I.E.G
ANDINI	Alessandro	U.E.R.M.C.P.P
AUBERT	Claude	E.N.S.I.E.G
ENTIL	Pierre	E.N.S.E.R.G
UERIN	Bernard	E.N.S.E.R.G
UYOT	Pierre	E.N.S.E.E.G
VANES	Marcel	E.N.S.I.E.G
ALINIER	Jean-Michel	E.N.S.I.E.G
AUSSAUD	Pierre	E.N.S.I.E.G
JOUBERT	Jean-Claude	E.N.S.I.E.G
JOURDAIN	Geneviève	E.N.S.I.E.G
LACOUME	Jean-Louis	E.N.S.I.E.G
LATOMBE	Jean-Claude	E.N.S.I.M.A.G
LESIEUR	Marcel	E.N.S.H.G
LESPINARD	Georges	E.N.S.H.G
LONGEQUEUE	Jean-Pierre	E.N.S.I.E.G
LOUCHET	François	E.N.S.E.E.G
MASSELOT	Christian	E.N.S.I.E.G
MAZARE	Guy	E.N.S.I.M.A.G
MOREAU	René	E.N.S.H.G
MORET	Roger	E.N.S.I.E.G
MOSSIERE	Jacques	E.N.S.I.M.A.G
PARIAUD	Jean-Charles	E.N.S.E.E.G
PAUTHENET	René	E.N.S.I.E.G
PERRET	René	E.N.S.I.E.G
PERRET	Robert	E.N.S.I.E.G
PIAU	Jean-Michel	E.N.S.H.G
POLOUJADOFF	Michel	E.N.S.I.E.G
POUPOT	Christian	E.N.S.E.R.G
RAMEAU	Jean-Jacques	E.N.S.E.E.G
RENAUD	Maurice	U.E.R.M.C.P.P
ROBERT	André	U.E.R.M.C.P.P
ROBERT	François	E.N.S.I.M.A.G
SABONNADIERE	Jean-Claude	E.N.S.I.E.G
SAUCIER	Gabrielle	E.N.S.I.M.A.G
SCHLENKER	Claire	E.N.S.I.E.G
SCHLENKER	Michel	E.N.S.I.E.G
SERMET	Pierre	E.N.S.E.R.G
SILVY	Jacques	U.E.R.M.C.P.P
SOHM	Jean-Claude	E.N.S.E.E.G
SOUQUET	Jean-Louis	E.N.S.E.E.G
VEILLON	Gérard	E.N.S.I.M.A.G
ZADWORNY	François	E.N.S.E.R.G

Professeurs Associés

BLACKWELDER	Ronald	E.N.S.H.G
HAYASHI	Hirashi	E.N.S.I.E.G
PURDY	Gary	E.N.S.E.E.G

Professeurs Université des Sciences Sociales (Grenoble II)

BOLLIET	Louis	
CHATELIN	Françoise	

Chercheurs du C.N.R.S

CARRE	René	Directeur de recherche
FRUCHART	Robert	Directeur de recherche
JORRAND	Philippe	Directeur de recherche
VACHAUD	Georges	Directeur de recherche
ALLIBERT	Michel	Maître de recherche
ANSARA	Ibrahim	Maître de recherche
ARMAND	Michel	Maître de recherche
BINDER	Gilbert	Maître de recherche
BORNARD	Guy	Maître de recherche
DAVID	René	Maître de recherche
DEPORTES	Jacques	Maître de recherche
DRIOLE	Jean	Maître de recherche
GIGNOUX	Damien	Maître de recherche
GIVORD	Dominique	Maître de recherche
GUELIN	Pierre	Maître de recherche
HOPFINGER	Emil	Maître de recherche
JOUD	Jean-Charles	Maître de recherche
KAMARINOS	Georges	Maître de recherche
KLEITZ	Michel	Maître de recherche
LANDAU	Ioan-Dore	Maître de recherche
LASJAUNIAS	Jean-Claude	Maître de recherche
MERMET	Jean	Maître de recherche
MUNIER	Jacques	Maître de recherche
PIAU	Monique	Maître de recherche
PORTESEIL	Jean-Louis	Maître de recherche
THOLENCE	Jean-Louis	Maître de recherche
VERDILLON	André	Maître de recherche
SUERY	Michel	Maître de recherche

Personnalités habilitées à diriger des travaux de recherche
(Decision du Conseil Scientifique)

E.N.S.E.E.G.

ALLIBERT	Colette	DIARD	Jean Paul	NGUYEN TRUONG	Bernadette
BERNARD	Claude	EUSTATHOPOULOS	Nicolas	RAVAINE	Denis
BONNET	Roland	FOSTER	Panayotis	SAINFORT	(CENG)
CAILLET	Marcel	GALERIE	Alain	SARRAZIN	Pierre
CHATILLON	Catherine	HAMMOU	Abdelkader	SIMON	Jean Paul
CHATILLON	Christian	MALMEJAC	Yves (CENG)	TOUZAIN	Philippe
COULON	Michel	MARTIN GARIN	Régina	URBAIN	Georges (Labora des ultra-réfract ODEILLO).

E.N.S.E.R.G.

BARIBAUD	Michel	CHEHIKIAN	Alain	HERAULT	Jeanny
BOREL	Joseph	DOLMAZON	Jean Marc	MONLLOR	Christian
CHOVET	Alain				

E.N.S.I.E.G.

BORNARD	Guy	KOFMAN	Walter	MAZUER	Jean
DESCHIZEAUX	Pierre	LEJEUNE	Gérard	PERARD	Jacques
GLANGEAUD	François			REINISCH	Raymond

E.N.S.H.G.

ALEMANY	Antoine	MICHEL	Jean Marie	ROWE	Alain
BOIS	Daniel	OBLED	Charles	VAUCLIN	Michel
DARVE	Félix			WACK	Bernard

E.N.S.I.M.A.G.

BERT	Didier	COURTOIS	Bernard	FONLUPT	Jean
CALMET	Jacques	DELLA DORA	Jean	SIFAKIS	Joseph
COURTIN	Jacques				

U.E.R.M.C.P.P.

		CHARUEL	Robert		

C.E.N.G.

CADET	Jean	JOUVE	Hubert (LETI)	PERROUD	Paul
COEURE	Philippe (LETI)	NICOLAU	Yvan (LETI)	PEUZIN	Jean Claude (LE
DELHAYE	Jean Marc (STT)	NIFENECKER	Hervé	TAIEB	Maurice
DUPUY	Michel (LETI)			VINCENDON	Marc

Laboratoires extérieurs :

C.N.E.T.

DEMOULIN	Eric	GERBER	Roland	MERCKEL	Gérard
DEVINE	R.A.B.			PAULEAU	Yves

I.N.S.A. Lyon

		GAUBERT	C.		

••••••••••

ECOLE NATIONALE SUPERIEURE DES MINES DE SAINT ETIENNE

Directeur : M. M. MERMET
Directeur des Etudes et de la formation : M. J. CHEVALIER
Secrétaire Général : Melle M. CLERGUE

PROFESSEURS DE 1ère CATEGORIE

MM.	COINDE	Alexandre	Gestion
	GOUX	Claude	Métallurgie
	LEVY	Jacques	Métallurgie
	LOWYS	Jean-Pierre	Physique
	MATHON	Albert	Gestion
	RIEU	Jean	Mécanique - Résistance des Matériaux
	SOUSTELLE	Michel	Chimie
	FORMERY	Philippe	Mathématiques Appliquées

PROFESSEURS DE 2ème CATEGORIE

MM.	HABIB	Michel	Informatique
	PERRIN	Michel	Géologie
	VERCHERY	Georges	Matériaux
	TOUCHARD	Bernard	Physique Industrielle

DIRECTEUR DE RECHERCHE

M.	LESBATS	Pierre	Métallurgie

MAITRES DE RECHERCHE

MM.	BISCONDI	Michel	Métallurgie
	DAVOINE	Philippe	Géologie
Mle	FOURDEUX	Angeline	Métallurgie
MM.	GUILHOT	Bernard	Chimie
	KOBYLANSKI	André	Métallurgie
	LALAUZE	René	Chimie
	LANCELOT	Francis	Chimie
	LE COZE	Jean	Métallurgie
	PLA	Jean Marie	Mathématiques
	THEVENOT	François	Chimie
	TRAN MINH	Canh	Chimie

PERSONNALITES HABILITEES A DIRIGER LES TRAVAUX DE RECHERCHE

MM.	COURNIL	Michel	Chimie
	DRIVER	Julian	Métallurgie
	MAGNIN	Thierry	Métallurgie
	THOMAS	Gérard	Chimie

PROFESSEUR A L'U.E.R. DE SCIENCES DE SAINT ETIENNE

M.	VERGNAUD	Jean Marie	Chimie des Matériaux

REMERCIEMENTS

Je tiens à remercier Monsieur Jean-Claude LATOMBE qui m'a fait l'honneur d'accepter de présider le jury de cette thèse.

J'exprime ma gratitude à Monsieur Michel LUCAS qui a bien voulu participer au jury.

Je remercie vivement Monsieur Gérard ALLAIN pour les discussions fructueuses que nous avons eu lors de la définition de ce travail et pour avoir accepté de participer à ce jury.

Les conseils de Monsieur Francis MARTINEZ m'ont permis d'améliorer grandement la présentation de ce rapport. Je lui suis très reconnaissante d'avoir accepté de juger ce travail.

Je tiens à remercier Monsieur Michel HABIB pour l'intérêt qu'il a manifesté à l'égard de ces recherches. Ses conseils et ses suggestions ont toujours été extrêmement précieux.

Monsieur Michel GANGNET m'a initiée à la recherche et m'a constamment encouragée dans la réalisation de ce travail. Je lui suis très redevable pour les conseils qu'il a su me prodiguer ainsi que pour ses nombreuses et pertinentes suggestions. Qu'il trouve ici l'expression de mes chaleureux remerciements.

J'exprime aussi mon amitié à tout le personnel du département dont les réflexions à propos des images produites ont toujours été extrêmement profitables. Je remercie tout particulièrement les membres de l'équipe "Communications Visuelles", Djamchid GHAZANFARPOUR, Dominique MICHELUCCI et Jean-Claude MOISSINAC pour leur collaboration amicale.

Je profite de l'occasion pour exprimer ma reconnaissance envers les personnes du centre de calcul qui participent à l'entretien de la configuration matérielle.

Je remercie enfin Mesdames AVONDO et VINCENT et Messieurs BROSSARD, DARLES et LOUBET qui ont participé avec beaucoup de soins et de gentillesse à la réalisation de cet ouvrage. La présentation de ce rapport doit beaucoup au logiciel GROFF (Girardot's Ridiculous Output File Formatter) développé par Jean-Jacques GIRARDOT.

PLAN

PLAN

INTRODUCTION

INTRODUCTION

L'image est un moyen de communication universel qui permet de diffuser une quantité très importante d'informations. Après la peinture, le dessin et la photo, la synthèse d'image par ordinateur constitue un nouvel outil pour la création d'images. La puissance et la souplesse des machines informatiques donnent à cet outil des performances exceptionnelles. Ainsi, les applications de l'informatique graphique sont parfois considérées comme ayant peu de limites. On peut aujourd'hui caractériser principalement trois types d'applications:

* la bureautique. Ce domaine connaît actuellement un développement considérable outre-Atlantique. L'édition de graphiques d'aide à la décision, ou de graphiques d'affaire permet une présentation compacte et claire de grandes quantités de données [BER77].

* la simulation. Nous englobons sous ce terme un grand nombre d'applications allant de la conception assistée par ordinateur à la simulation de phénomènes physiques (ex. déformation de solides), la simulation de conduite des véhicules, les études d'impact en urbanisme ainsi que la synthèse d'éléments naturels comme la végétation [AON84], ou la modélisation du corps humain.

* la production audiovisuelle. Ce domaine rassemble deux grands marchés: la publicité et le cinéma d'animation.

Le type des images générées peut être très variable. Les principaux critères caractérisant une image de synthèse sont:

- image fixe ou animée. La production d'images animées suppose de calculer, en temps réel ou différé, jusqu'à 25 images par seconde. Les temps de calcul deviennent alors une contrainte importante.

- image produite à l'aide de vecteurs ou par remplissage en couleur au niveau des points élémentaires. Les images au trait, ou fil de fer permettent un affichage plus rapide et généralement des temps de calcul plus faibles. Cependant, dans ce cas, la représentation demeure très schématique. Cette visualisation est à l'heure actuelle utilisée pour les graphiques d'affaire, la simulation de certains phénomènes et dans les logiciels de CAO.

- images 2D ou 3D. Les images tri-dimensionnelles sont générées à partir d'un modèle tri-dimensionnel de la scène à représenter. Si l'ajout d'une dimension au modèle permet de

modéliser des phénomènes intrinséquement tri-dimensionnels tels que l'éclairement, il permet aussi de se déplacer dans la scène et de modifier le point de vue.

En synthèse tri-dimensionnelle, on peut citer quelques autres critères caractérisant une image:
- l'élimination des parties cachées,
- les modéles d'éclairement,
- la gestion des transparences,
- l'ajout de textures
- l'antialiassage.

La sophistication de l'image produite entraîne généralement l'accroissement des coûts de production.

On peut distinguer trois types de contraintes temporelles en fonction du domaine d'application concerné:

* la production d'images en temps réel. La simulation pour la conduite de véhicules est un exemple où le temps réel est imposé.

* la production d'images en un temps compatible avec l'interaction. Cette contrainte est imposée pour tout logiciel interactif tel que les logiciels de CAO, les logiciels d'édition de graphiques d'affaire...

* la production d'images en temps différé. Les moyens logiciels et matériels ne permettent pas toujours de créer des images de qualité suffisante sous les contraintes précédentes, temps réel ou compatible avec l'interaction. Par conséquent, si la qualité de l'image prime sur le temps d'exécution, on choisira ce mode de production. La majorité des images spectaculaires sont produites en temps différé.

Les travaux présentés dans ce rapport sont à associer à la synthèse tri-dimensionnelle d'images réalistes. Ces images sont produites en temps différé. On peut cependant espérer que l'utilisation de machines spécialisées ou de systèmes parallélisés pourront rapidement permettre de produire ces images avec des temps d'exécution compatibles avec l'interaction.

Le tracé de rayon est actuellement considéré comme l'algorithme le mieux adapté à la synthèse d'images réalistes. Les résultats les plus importants de ce rapport concernent l'optimisation de cet algorithme adapté à différents types de données. Plusieurs des techniques exposées montrent comment exploiter la cohérence des données. Il s'agira soit d'une cohérence intrinsèque des données soit d'une cohérence générée artificiellement pour faciliter les calculs ultérieurs.

Le premier chapitre contient une étude bibliographique sur la technique du tracé de rayon et une présentation des nombreuses possibilités de rendu et des optimisations propres à cette

méthode. La notion de rendu d'une image, empruntée au vocabulaire des dessinateurs et des architectes, est employée ici pour caractériser le degré de finition d'une image.

Nous avons choisi le tracé de rayon pour les possibilités de rendu qu'il offre, en contre partie, cette méthode est très coûteuse en temps de calcul. Par conséquent, nos objectifs sont doubles:

* exploiter certaines des possibilités de cette technique pour créer des effets spéciaux qui ne sont pas toujours faciles à obtenir avec les méthodes classiques. Dans ce domaine, nous avons concentré nos efforts sur des effets comme l'ajout de bleu atmosphérique ou la simulation de brume. Le paragraphe 6 du chapitre 2 présente les résultats obtenus.

* mettre au point des techniques permettant de diminuer les temps de calcul. Trois résultats sont présentés:

  - au chapitre 2 paragraphe 3 une méthode spécifique appliquée à une base de données altimétriques définie sur une grille rectangulaire permet de réduire le nombre de tests d'intersection.

  - au chapitre 2 paragraphe 3.3.2 un modèle de surfaces formées d'hyperboloïdes est considéré. Les tests d'intersection entre un rayon et la surface sont environ deux fois plus rapides que pour une surface formée de facettes triangulaires planes.

  - le chapitre 3 est consacré à l'étude d'une structure de données permettant d'accélérer de façon importante la recherche du premier point d'intersection entre un rayon et une scène d'objets. Cette méthode est très générale et permet de diminuer sensiblement le nombre de tests d'intersection nécessaires.

Le quatrième chapitre présente un algorithme de génération de textures. Cette méthode s'est avérée intéressante dans le cas des textures naturelles. Tous les ciels présentés sur les différentes images de terrains sont réalisés avec cette méthode sauf celui de la photo 4. Certains autres résultats sur des textures de bois par exemple semblent prometteurs.

Le dernier chapitre de ce rapport expose la solution apportée à un problème rencontré dans la production des images illustrant les chapitres précédents. La mémoire d'image utilisée dispose de 12 plans de visualisation et d'une table de couleur. Cette configuration permet d'afficher 4096 couleurs différentes simultanément. Le choix des couleurs à introduire dans la table de couleur n'est cependant pas facile. Il a donc été nécessaire de mettre au point un programme d'initialisation des tables de couleur. L'algorithme exposé détermine une table de couleur permettant de représenter de façon correcte les couleurs de l'image à afficher compte tenu de la limitation sur le nombre de

couleurs simultanément représentables.

Avant d'aborder le premier chapitre, nous ferons deux remarques liées à la présentation et à la discussion des algorithmes.

Expliciter un algorithme est une tâche délicate où l'auteur se trouve partagé entre le désir de présenter une solution simple et celui de montrer tous les pièges rencontrés lors de la programmation. Il nous a paru difficile d'exprimer tous les algorithmes sous la même forme. Nous avons par conséquent choisi deux formes de présentation:

- la présentation sous forme d'un algorithme rédigé en français. Cette version ne fait appel à aucun langage de programmation et paraît être dans certains cas une solution simple pour dégager les points importants de l'algorithme.

- une présentation beaucoup plus précise, écrite en pseudo-Pascal dans laquelle les séquences moins détaillées continueront d'être exprimées en français.

Une description détaillée de la configuration matérielle utilisée est présentée en annexe. Les travaux décrits dans les chapitres 2, 3 et 4 sont indépendants du système de visualisation. En effet, comme la mémoire d'image utilisée ne permet que de coder la couleur d'un point sur 12 bits, nous avons choisi de générer des images 24 bits mémorisées en mémoire virtuelle. Par contre, les temps d'exécution sont très dépendants du matériel, en particulier l'image stockée en mémoire virtuelle entraîne des défauts de pages non maîtrisables; de plus, le système d'exploitation utilisé est très sensible à la charge de la machine. Il est donc difficile dans les conditions actuelles d'obtenir des évaluations correctes, nous préférons par conséquent, fournir un nombre d'opérations ou une estimation de la complexité des algorithmes lorsque cela est possible.

CHAPITRE 1

ETAT DE L'ART EN TRACE DE RAYON

ETAT DE L'ART EN TRACE DE RAYON

1. INTRODUCTION

La modélisation des rayons lumineux est utilisée depuis longtemps pour la simulation de phénomènes optiques. C'est cependant en 1968 [APP68] et en 1971 [GOL71] que l'on trouve les premières références au tracé de rayon comme technique de visualisation graphique. Le tracé de rayon (ray-tracing), ou suivant les auteurs, le lancer de rayon (ray-casting), ne connait un réel développement que depuis 1980, lorsque T. Whitted à publié "An Improved Illumination Model for Shaded Display", [WHI80]. Cet article présentait deux images comportant des effets tout à fait inédits de réflexion et d'ombrage sur des objets opaques ou transparents. Cette technique est, comme on le verra par la suite, très puissante et a permis de générer certaines des images de synthèse parmi les plus réussies, sur le plan du nombre et de la qualité de restitution des effets présents dans l'image. En contre partie, cette méthode est très coûteuse en temps CPU. Ce sont les progrès technologiques de ces dernières années qui ont permis de disposer de machines assez rapides pour envisager l'utilisation de ces algorithmes. On peut recenser aujourd'hui plusieurs films présentant des scènes synthétisées par tracé de rayon. Si la production d'images utilisant le tracé de rayon nécessite d'importants moyens de calculs, de nombreuses recherches se développent sur des machines plus modestes, permettant soit de mettre au point des modèles d'éclairement plus sophistiqués [HAL83] soit d'étudier des algorithmes visant à réduire les temps de calcul.

Les paragraphes suivants présentent une synthèse des atouts et des contraintes du tracé de rayon. Après avoir décrit l'algorithme, nous évoquerons les facilités offertes pour obtenir un rendu de qualité puis nous étudierons les différentes techniques employées pour réduire les temps de calcul.

2. ALGORITHME

L'algorithme du tracé de rayon est, comme l'indiquent S.M. Rubin et T. Whitted [RUB80], et en faisant référence à la classification désormais classique proposée par I. Sutherland & al. [SUT74], un algorithme qui agit dans l'espace objet. L'image est construite en lançant, pour chaque pixel, un rayon qui est défini par son origine, l'oeil, et un deuxième point situé au centre du pixel écran considéré. L'algorithme agit ensuite dans l'espace objet en recherchant les points d'intersection du rayon

avec les différents objets de la scène. Un tri de ces points permet de déterminer le point d'intersection le plus proche de l'oeil. C'est le point de la scène, appartenant à la demi-droite constituant le rayon, qui est vu depuis l'oeil. Les caractéristiques physiques de ce point de l'objet (couleur, coefficients de réflexion, réfraction...) permettent de calculer la couleur du pixel à afficher sur l'écran. Cette couleur n'est pas déductible directement mais peut nécessiter, comme on le verra ultérieurement, de lancer de nouveaux rayons pour caractériser les conditions d'éclairement du point. Fondamentalement, la méthode du tracé de rayon modélise le parcours inverse des rayons lumineux et somme pour chaque pixel les intensités lumineuses contribuant à l'éclairement de cette partie de l'image.

Sous sa forme la plus élémentaire, l'algorithme peut s'écrire:

Algorithme 1-1 : Version simplifiée de l'algorithme du tracé de rayon.

```
DEBUT
   POUR chaque point de l'écran FAIRE
   DEBUT
      Définir le rayon issu de l'oeil passant par le point
      courant;
      POUR chaque objet FAIRE
         tester intersections;
      trier les intersections;
      calculer éclairement;
      afficher;
   FIN POUR;
FIN.
```

Nous allons reprendre les différentes actions de cet algorithme pour en donner une description plus détaillée.

2.1. Définition du rayon issu de l'oeil et passant par le point courant de l'écran

Le rayon est défini par deux points: l'oeil et le centre du pixel traité. Nous allons montrer comment, à partir des paramètres de visualisation de l'image, on peut caractériser le rayon à envoyer. Les transformations considérées font appel à des notions de géométrie élémentaire définies dans les ouvrages de base d'informatique graphique, [MOR76], [NEW79], [FOL82]. Il nous paraît cependant important de décrire le processus conduisant à la définition d'un rayon car il est utilisé à plusieurs reprises dans la suite et peut être l'objet d'un calcul incrémental.

Nous considérons les trois repères suivants:

- le repère de définition de la scène ou repère du monde, RM.
- le repère de l'oeil, RO.
- le repère de l'écran, RE.

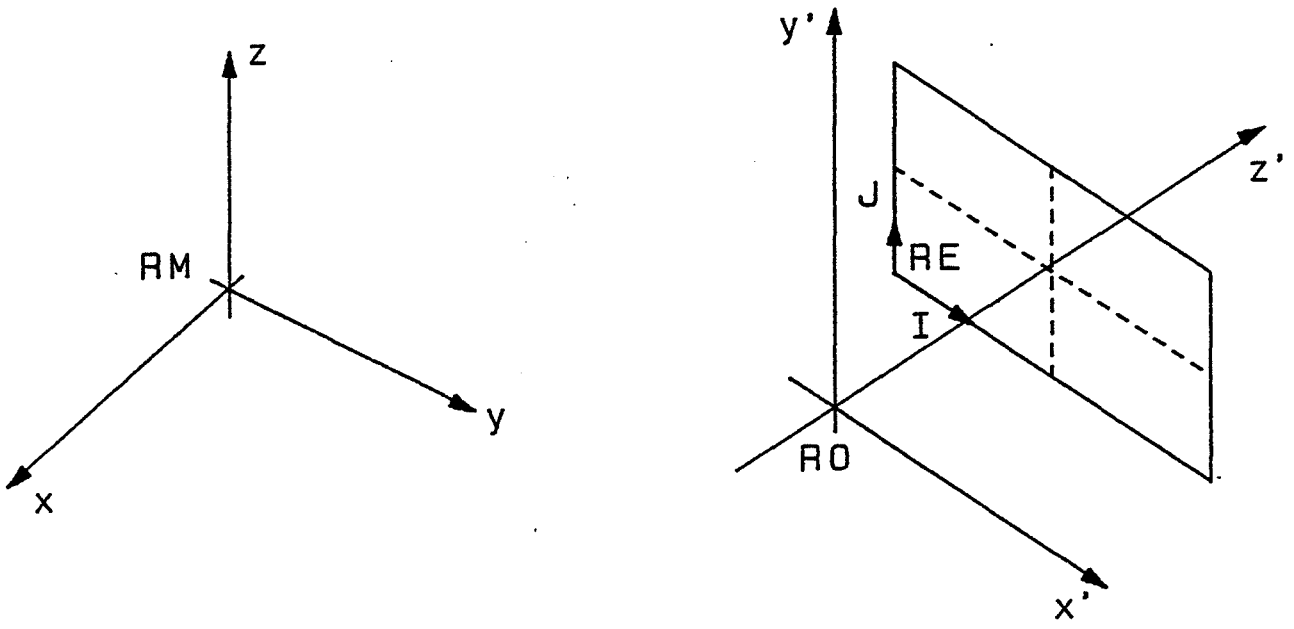


Figure 1-1 : Définition des différents repères

Soit T2 la matrice de passage des coordonnées homogènes définies dans RE aux coordonnées homogènes définies dans RO.

On a pour le pixel de coordonnées I,J:

$$(x'\ y'\ z'\ 1)_{RO} = (I\ J\ 0\ 1)_{RE}\ T2$$

T1 étant la matrice de passage des coordonnées homogènes définies dans RO aux coordonnées homogènes définies dans RM, on a:

$$(x\ y\ z\ 1)_{RM} = (I\ J\ 0\ 1)_{RE}\ T2T1$$

Soit, en posant T=T2T1

$$(x\ y\ z\ 1)_{RM} = (I\ J\ 0\ 1)_{RE}\ T$$

le calcul de T2 et de T1 repose sur les paramètres habituels de calcul d'une vue perspective.

Comme il est préférable de calculer les intersections entre les rayons et les objets dans le repère du monde, la boucle sur les rayons est définie à l'aide des paramètres suivants:

$$(x1\ y1\ z1\ 1) = (IMIN\ JMIN\ 0\ 1)T$$
$$(a\ b\ c\ 1) = (1\ 0\ 0\ 1)T$$
$$(d\ e\ f\ 1) = (0\ 1\ 0\ 1)T$$

IMIN, JMIN, IMAX, JMAX étant la clôture (zone d'affichage) dans l'écran et xo, yo, zo les coordonnées de l'oeil dans le repère RM, l'équation de la demi-droite représentant un rayon étant

$$xo+tu,\ yo+tv,\ zo+tw \text{ avec } t \geq 0 \qquad (1)$$

On aura:

Algorithme 1-2 : Définition incrémentale des rayons

```
u ← x1-xo; v ← y1-yo; w ← z1-zo;
POUR I de IMIN à IMAX FAIRE
DEBUT
   u1 ← u; v1 ← v; w1 ← w;
   POUR J de JMIN à JMAX FAIRE
   DEBUT
      tester intersection de la scène avec le rayon
             (xo+tu,yo+tv,zo+tw);
      u ← u+d; v ← v+e; w ← w+f;
   FIN POUR
   u1 ← u+a; v1 ← v+b; w1 ← w+c;
FIN POUR
```

2.2. Intersection entre un rayon et un objet élémentaire

La majorité des algorithmes d'élimination des parties cachées permettent le traitement de facettes planes. Il est alors soit impossible, soit très pénalisant de traiter des formes plus complexes. Le tracé de rayon étant basé sur des tests d'intersection entre un rayon et l'ensemble des objets d'une scène, la description en facettes planes n'est généralement pas souhaitable car elle conduit à de très nombreux tests élémentaires.

Soit T le temps moyen mis pour un calcul d'intersection entre un rayon et un objet élémentaire; soit N le nombre d'objets élémentaires de la scène. Le coût moyen pour un rayon est:

$$N \times T$$

Soit un objet du type i décomposable en Nj objets du type j. Suivant le modèle choisi, le coût sera:

Ti ou Nj×Tj

d'où l'importance du choix du modèle.

L'objectif étant de définir un modèle qui permette de minimiser le coût global, commençons par évaluer l'intersection entre un rayon et différents types d'objets élémentaires.

- **les facettes planes.**
  Le calcul d'intersection entre un rayon et une facette plane est détaillé au chapitre 2 paragraphe 3.3.1

- **les parallélépipèdes rectangles et les quadriques (cône, cylindre, sphère...).**
  On peut définir ces objets dans un repère local [ROT82] permettant:

  . un calcul d'intersection généralement plus facile. Les transformations telles que rotation, translation, homothétie sont effectuées sur le rayon; cela permet d'utiliser une description simple de l'objet dans son repère local.

  . une description des primitives plus homogène. Par exemple, tout ellipsoïde peut être obtenu à partir d'une sphère de rayon 1 définie dans son repère local et d'une affinité.

  Le test d'intersection est alors réalisé sur l'image du rayon par le changement de repère monde - repère local de l'objet.

- **les prismes et les surfaces de révolution.**
  Ce cas peut être résolu soit:

  . en se ramenant à un calcul d'intersection bi-dimensionnel entre un rayon et une courbe pour le prisme ou entre deux courbes pour la surface de révolution. Ce dernier problème peut être résolu en utilisant les strip-trees [KAJ83b].

  . en décrivant chaque section de génératrice de la surface de révolution de façon paramétrique dans un plan contenant l'axe de révolution. La modélisation de la section génératrice étant réalisée à l'aide de segments de droite, de B-splines ou de courbes de Bézier [BOU84].

- **Les surfaces bicubiques.**
  Deux méthodes sont proposées:

  . l'une est basée sur une subdivision des bicubiques [WHI80] utilisant la technique de E.E. Catmull [CAT78] ou celle de J.M. Lane et L.C. Carpenter [LAN80].

  . l'autre effectue un calcul direct des points d'intersection en se ramenant à la recherche des racines d'un polynôme [KAJ82].

- **Les surfaces algébriques.**
  P. Hanrahan [HAN83] ramène aussi ce cas à la recherche des racines d'un polynône.

- **Les surfaces fractales.**
  J.T. Kajia [KAJ83a] a développé un algorithme basé sur une subdivision récursive en triangles. Cet algorithme réalise un tri permettant de ne pas considérer les triangles ne pouvant pas être intersectés par le rayon.

Ainsi, et pour répondre au problème du choix du modèle posé précédemment, il est préférable de ne pas décomposer une sphère en facettes planes alors qu'il peut être intéressant de subdiviser une surface bi-cubique.

Les tests d'intersection entre un rayon et tous les objets de la scène étant renouvelés pour chaque pixel de l'écran, on constate que la complexité de la scène est très pénalisante.

C'est pour cela que de nombreuses recherches sur le tracé de rayon sont consacrées à l'étude d'optimisations visant à diminuer le nombre de tests d'intersection. Une présentation de différentes structures et des méthodes d'optimisation existantes est présentée au paragraphe 4.

Nous allons auparavant terminer la description des différentes actions de l'algorithme 1-1, ce qui nous permettra de constater qu'un rendu de qualité suppose d'étudier plusieurs rayons pour chaque pixel.

2.3. Tri des intersections

On dispose d'un ensemble de points d'intersection entre le rayon et les objets de la scène. Un tri sur cet ensemble fourni le point le plus proche de l'oeil. Pour faciliter ce tri, l'équation du rayon est définie sous forme paramétrique par (1) qui peut aussi s'exprimer par:

$$OM = OA + tu$$

De cette façon, les points du rayon sont caractérisés par le paramètre t proportionnel à la distance du point d'intersection à l'oeil.

En appliquant le changement de repère T,

$$T(OM) = T(OA) + tT(u)$$

t est invariant car T est linéaire. On peut donc comparer des valeurs de t obtenues avec des transformations différentes, cette propriété est utilisée par [ROT82] dans le cadre de la géométrie constructive ("constructive solid geometry").

2.4. Calcul de l'éclairement

Ce sont les possibilités de réaliser des calculs d'éclairement sophistiqués qui ont donné toute son importance au tracé de rayon. Les effets les plus facilement réalisables sont:

- les ombres portées relatives à plusieurs sources lumineuses,
- les réflexions multiples,
- les transparences avec réfraction.

Nous ne détaillerons pas ici les différents modèles utilisés [WHI80],[HAL83],[BOU84], notons simplement que les calculs supposent la connaissance

- des conditions d'éclairement du point considéré,
- de la position de l'oeil par rapport à la surface,
- des caractéristiques physiques des matériaux: rugosité, transparence,
- des caractéristiques géométriques de la surface où est situé le point: normale.

La somme des intensités réfléchies, réfractées, transmises, modulée par les caractéristiques physiques et géométriques de la surface permettent de déterminer la couleur à affecter à un point. Ce calcul doit être exécuté pour chaque point. Cela suppose d'envoyer des rayons secondaires dans la scène.

- Pour les ombres portées, ces rayons sont renvoyés vers chacune des sources lumineuses.

- Pour les réflexions et les réfractions, les rayons secondaires simulent les rayons lumineux réfléchis et réfractés en ce point.

L'affichage est alors une opération triviale qui consiste à affecter la couleur déduite des calculs précédents au pixel courant.

En conclusion, pour chaque pixel, on envoie un rayon primaire qui peut donner naissance à plusieurs rayons secondaires.

3. STRUCTURES DE DONNEES

Nous présentons dans ce paragraphe les différentes structures de données qui ont déjà été combinées avec un algorithme de tracé de rayon. D'autres structures, qui n'ont à notre connaissance jamais été utilisées avec cette méthode seront étudiées au chapitre 3 paragraphe 3

* La juxtaposition d'objets.

La solution la plus simple définit la scène comme une juxtaposition d'objets élémentaires [WHI80], on utilisera évidemment les boîtes ou les sphères englobantes de chaque objet pour minimiser les tests.

* La géométrie solide constructive

Une des structures de données souvent considérée comme la mieux appropriée est dérivée d'une modélisation hiérarchique sous forme d'arbre binaire (arbre CSG soit en anglais constructive solid geometry tree). La structure de base, qui n'a pas été étudiée spécialement pour le tracé de rayon mais qui s'était auparavant révélée un outil intéressant pour la modélisation de pièces mécaniques [CLA76], [REQ80], peut être décrite de la façon suivante.

L'objet est représenté comme une combinaison d'objets élémentaires. L'ensemble des objets élémentaires peut être plus ou moins important suivant les applications. Il peut par exemple être constitué du cube, de la sphère, du cône, du cylindre, du tore, de surfaces de révolution... Les opérateurs de combinaison sont généralement l'union, l'intersection et la différence. La représentation sous forme d'arbre binaire, est réalisée en affectant un objet élémentaire à chaque feuille et un opérateur à chaque noeud non terminal qui correspond donc à la description d'un objet composite. L'objet décrit par l'arbre est situé à la racine. (cf. figure 1-2).

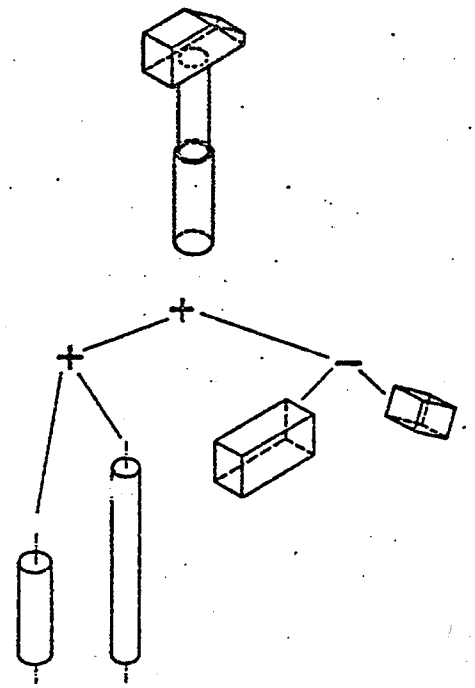


Figure 1-2 : Représentation d'un arbre CSG
Cette figure est extraite de [BRO84].

Nous avons vu comment déterminer les points d'intersection entre un rayon et chacun des objets élémentaires. Nous allons montrer comment en déduire les points d'intersection entre le rayon et la scène.

Ceci est réalisé en combinant pour chaque noeud les points d'intersection du sous arbre droit et ceux du sous arbre gauche en accord avec l'opérateur présent sur le noeud. On imagine aisément un algorithme récursif permettant de réaliser cette tâche. L'algorithme décrit ci-dessous est extrait de l'article de S.D. Roth [ROT82].

Algorithme 1-3 : Détermination des points d'intersection entre un rayon et une scène.

```
FONCTION inter(noeud);
DEBUT
   SI noeud n'est pas une feuille de l'arbre ALORS
      inter ← combine(inter(fils-gauche(noeud)),
              inter(fils-droit(noeud));
   SINON
      inter ← point d'intersection du rayon avec
              l'objet élémentaire
FIN.
```

L'arbre CSG présente principalement deux avantages:

- le traitement des objets intersectants est facilité.
- il est possible d'enrichir la structure avec des notions de boîtes englobantes par exemple.

*** la méthode de S.M. Rubin et T. Whitted.**

S.M. Rubin et T. Whitted [RUB80] proposent un codage des données représentant tous les niveaux sous la forme de parallélépipèdes rectangles pouvant à la limite représenter un point. L'efficacité de ce codage provient de sa hiérarchisation et de l'uniformité des données. La modélisation n'est cependant pas triviale.

4. MINIMISATION DU NOMBRE DE TESTS D'INTERSECTION : LES SOLUTIONS EXISTANTES

En ce qui concerne le problème de l'élimination des parties cachées, les sujets de recherche les plus fructueux pour l'optimisation des algorithmes sont:

*** la suppression.**
Il s'agit de supprimer une partie des éléments à traiter. La solution consiste à appliquer un prétraitement visant à tester un critère de suppression. On peut par exemple citer:

- la suppression des faces arrières,
- les tests d'intersection des boîtes englobantes avec la pyramide de vision.

*** le traitement adaptatif.**
On choisit dans ce cas d'adapter le traitement à la complexité de la scène ou de la région de la scène considérée. L'algorithme de Warnock est un excellent exemple. On verra dans le paragraphe suivant sur la cohérence de l'image que T. Whitted [WHI80] exploite aussi cette méthode.

*** la simplification.**
On souhaite, par un prétraitement, simplifier l'algorithme d'élimination des parties cachées. Par exemple, l'existence d'une relation d'ordre entre les faces ou les objets de la scène simplifie grandement de nombreux algorithmes. Toute technique permettant d'accélérer le tri des éléments est par

conséquent profitable [SUT74], [FUC79], [MAR82].

Nous présentons dans ce paragraphe les différentes méthodes envisagées jusqu'ici pour minimiser le nombre de tests d'intersection entre un rayon et les objets constituants la scène à visualiser.

Cette étude est très importante car nous abordons ici la partie la plus sensible de l'algorithme. D'après T. Whitted [WHI80], entre 75 et 95% du temps CPU de calcul d'une image en tracé de rayon est consacré aux calculs d'intersection. On peut recenser deux techniques principales et pour chacune d'entre elles, un certain nombre de variantes qui ont donné naissance à divers algorithmes. Ces deux techniques sont:

- l'exploitation de la cohérence de l'image,

- l'exploitation de la cohérence des données.

Fondamentalement, l'exploitation de la cohérence de l'image revient à diminuer le nombre de pixels étudiés alors que la cohérence des données conduit à minimiser le nombre d'objets élémentaires traités.

4.1. Exploitation de la cohérence de l'image.

L'idée est la suivante: dans les zones les plus lisses de l'image, il paraît superflu d'envoyer un rayon pour chaque pixel. Au contraire, dans les zones correspondant à une frontière entre deux surfaces en projection sur l'écran, on souhaiterait disposer d'informations supplémentaires. Cette idée énoncée par S.D. Roth [ROT82], a été exploitée par une équipe de l'université de Delft qui à travers deux articles, [JAN83] et [BRO84], montre les avantages et les inconvénients de la méthode. L'exploitation de cette technique est réalisée en sous-échantillonnant l'image, c'est à dire en envoyant un rayon tous les n pixels, n pouvant être fixé par l'utilisateur. Les rayons intermédiaires ne sont envoyés que si la différence entre les intensités des points voisins est supérieure à un certain seuil.

Cette méthode permet donc d'obtenir rapidement des images. Cependant, la vitesse est souvent acquise aux dépends de la qualité. Bien qu'il soit possible de fixer à la fois la valeur de n et le seuil, cette méthode pose le problème des petits objets qui risquent d'être oubliés.

Signalons, que T. Whitted [WHI80] exploite l'effet inverse pour résoudre les problèmes d'aliassage. Il exécute un sur-échantillonnage par subdivision des pixels dans les zones où les changements d'intensité sont brusques.

Malheureusement, les techniques d'optimisation qui tirent parti de la cohérence de l'image ne peuvent être appliquées qu'aux rayons primaires. Par contre, les solutions utilisant la cohérence des données affectent généralement les rayons primaires et secondaires.

4.2. Exploitation de la cohérence des données.

Le but n'est plus, comme pour la cohérence de l'image, d'envoyer moins de rayons dans la scène mais pour chaque rayon envoyé, de définir des sous-ensembles de la scène qui ne peuvent être intersectés de façon à ne considérer que peu d'objets pour chaque rayon. Cette solution suscite généralement un remaniement de la structure de donnée et un prétraitement conduisant à la sélection des objets à traiter. Ce prétraitement peut être global ou appliqué à chaque objet. Plusieurs idées ont été suggérées pour réaliser cette sélection:

- **optimisation de la CSG**
  Une idée simple consiste à ne plus appliquer l'algorithme 1-1 de façon systématique. Si l'opérateur d'un noeud non terminal est la différence ou l'intersection, et que le rayon n'a pas intersecté l'objet décrit par le sous arbre gauche, alors il est inutile d'étudier le sous arbre droit [ROT82].

- **boîtes englobantes**
  S.D. Roth [ROT82] propose de profiter de la structure hiérarchique pour éliminer certaines régions de la scène. Pour cela, il introduit la notion de boîte englobante d'un noeud. Cette boîte est définie pour chaque noeud et englobe tout le sous arbre correspondant. Ainsi, un test d'intersection avec cette boîte permet, si le rayon n'intersecte pas la boîte, de ne pas considérer le sous arbre. S.D. Roth définit les boîtes dans le repère de l''oeil. Si ce choix permet d'accélérer sensiblement les tests d'intersection, il pose deux problèmes:

  . la définition des boîtes dépend du point de vue; les boîtes doivent donc être recalculées systématiquement.

  . cette structure n'est plus valable pour les rayons secondaires.

On remarquera que la notion de boîte englobante peut être appliquée à une structure hiérarchique non définie sous forme de CSG. Elle est ainsi appliquée à un seul objet dont le test d'intersection avec un rayon est couteux en temps.

- **sphères englobantes**
  Pour pallier la difficulté précédente, S.D. Roth [ROT82] et T. Whitted [WHI80] proposent de définir des sphères englobantes qui présentent les avantages suivants.

  . les tests d'intersection entre un rayon et une sphère sont simples

  . le test est aussi valable pour les rayons secondaires

  . la définition des rayons des sphères englobantes est indépendante du point de vue.

- **intervalle englobant**
  W.F. Bronsvoort & al. [BRO84] suggèrent d'améliorer l'efficacité de la boîte englobante parallélépipédique associée à une structure CSG, [ROT 82]. Dans ce cas, la boîte englobante associée à un noeud, est obtenue en appliquant l'opérateur du noeud aux boîtes englobantes des fils. Constatant que dans de nombreux cas, la boîte englobante associée à un noeud est une approximation très grossière, W.F. Bronsvoort & al. proposent d'utiliser la notion d'intervalle englobant défini sur les lignes de balayage. Cet intervalle est obtenu en appliquant l'opérateur du noeud aux intervalles associés aux noeuds fils. Pour une feuille, l'intervalle englobant correspond à l'intersection entre la boîte englobante et la ligne de balayage. Dans le pire des cas, le gain est nul. Généralement l'intervalle englobant constitue une meilleure approximation qui permet d'éliminer des zones de la scène plus importantes.

Plusieurs extensions viennent d'être présentées de façon indépendante, cependant, les essais de combinaison de ces techniques à l'intérieur d'un même algorithme donnent généralement de meilleurs résultats. C'est ainsi que T. Whitted [WHI80] propose de résoudre le problème des petits objets de la méthode de sous-échantillonnage en ajoutant des sphères englobantes. Les sphères englobantes des petits objets qui posent des problèmes dans la méthode de sous-échantillonnage, sont gonflées de façon à être forcement intersectées par au moins un rayon. Le rayon minimum des sphères est fixé en fonction du pas de sous-échantillonnage courant. Cette condition assure de ne pas oublier de petits objets.

La description du tracé de rayon a montré que cet algorithme est très simple dans sa conception et qu'il se prête très bien aux méthodes de parallélisation [KAJ83b], [BOU84]. A notre connaissance, trois techniques de parallélisation sont à l'étude:

* Une parallélisation triviale consiste à subdiviser l'image et à confier chaque région de l'image à un processeur différent.

* Un système pipeline associant à chaque processeur un ensemble d'objets de la scène. Chaque rayon est traité par chaque processeur afin de tester son intersection avec l'ensemble d'objets associés.

* Une structure parallèle associant à chaque processeur une région de la scène. Ainsi, chaque rayon est orienté vers le processeur traitant le volume qu'il traverse.

Le tracé de rayon est une technique récente souvent réservée à la production d'images expérimentales représentant des scènes d'une faible complexité en nombre d'objets. On peut cependant espérer que l'association de l'amélioration des performances du materiel et des extensions ou des optimisations algorithmiques provenant des nombreuses recherches développées actuellement sur le sujet permette d'élargir les domaines d'application de cette technique.

CHAPITRE 2

VISUALISATION DE TERRAINS ET TRACE DE RAYON

VISUALISATION DE TERRAINS ET TRACE DE RAYON

1. REVUE BIBLIOGRAPHIQUE

Nous allons aborder dans ce chapitre les problèmes liés à la représentation de scènes d'extérieur. Les modèles les plus simples décrivent uniquement le relief du terrain. Des visualisations plus réalistes peuvent nécessiter une définition précise des parcelles, des voies, de la végétation, des différents bâtiments...

Le besoin de dispositifs permettant la visualisation réaliste de scènes d'extérieur dans les simulateurs de conduite a suscité le développement de nombreuses recherches et l'industrialisation de plusieurs systèmes opérationnels. Ce domaine d'application n'est cependant pas le seul; d'autres secteurs d'activité sont intéressés. On peut en particulier citer certaines applications de CAO en ingéniérie minière ou en urbanisme pour des études d'impact, ainsi que des applications audiovisuelles ou des études pour la cartographie.

Les systèmes pour les simulateurs de conduite sont contraints par le temps réel, ce qui pour les premiers systèmes a imposé la synthèse d'images très schématiques. Aujourd'hui, les simulateurs utilisent des calculateurs spécialisés permettant de reproduire des scènes très complexes. Cependant, les images les plus réalistes actuellement sont générées pour des applications qui autorisent soit la production d'images en un temps compatible avec l'interactivité soit la production d'images en temps différé.

L'ampleur des marchés concernés par les simulateurs de conduite a permis le développement de nombreuses études. Cependant les résultats, en particulier les algorithmes et les détails d'implémentation ne sont généralement pas publiés ou pas diffusés. Il existe cependant un ouvrage récent "Computer image generation", édité par B.J. Schachter [SCH83] qui fait la synthèse des travaux publiés et présente quelques systèmes existants. Dans l'étude comparative développée ci-dessous, les informations relatives aux systèmes des simulateurs de conduite sont principalement extraites de cet ouvrage et des articles de B. Schachter [SCH81] et de G. Allain [ALL83].

Nous allons exposer dans les paragraphes suivants les différentes techniques mises en oeuvre pour la synthèse de scènes d'extérieur. Nous n'étudierons pas les scènes nocturnes qui sont généralement considérées comme les plus simples à représenter; les premiers simulateurs de vol ne permettaient que la production de scènes nocturnes composées de points lumineux. Après une revue

des différents modèles de description des données, nous exposerons les avantages, les inconvénients et éventuellement les extensions de chacun pour la visualisation. L'insertion d'effets visuels dans les images sera abordé ensuite.

1.1. Modèles de description des données

1.1.1. Le terrain

Les deux descriptions les plus couramment employées sont une représentation par un réseau de facettes planes et une représentation sous forme d'une grille rectangulaire.

* **Réseau de facettes**

  Dans ce cas, la surface du sol est modélisée par un réseau de facettes polygonales planes. Ces facettes peuvent être définies soit à partir de courbes de niveaux soit en recherchant les lignes correspondant à des discontinuités de pentes et en leur associant si possible les arêtes des facettes. Pour simplifier la modélisation puis la visualisation, on considère généralement des facettes triangulaires. Cette solution présente plusieurs avantages:

  . Le découpage en facettes permet d'adapter la précision du découpage au relief de la région sans redondance dans les zones de plaine. Ainsi, une région accidentée sera modélisée par de petites facettes alors qu'une région plus plate sera modélisée par des facettes plus grandes.

  . Le découpage en facettes peut dans certains cas être identifié au découpage en parcelles. Ceci permet de disposer dans la même structure de la définition du relief et de la couleur ou de la texture à associer à chaque région. Le terrain et la description surfacique sont alors représentés par le même modèle.

  . Le découpage en facettes peut être construit avec différents niveaux de précision. C'est la raison pour laquelle les premiers simulateurs de conduite utilisaient tous ce modèle. En effet, au prix d'une représentation très schématique du relief, il est possible de modéliser un terrain avec très peu de facettes. Cette propriété est aussi exploitée pour construire des structures hiérarchiques contenant diverses représentations du terrain avec différents niveaux de détails. General Electric Company a développé plusieurs modèles de ce type.

*** Grille rectangulaire**

Cette description des données fournit une définition de l'altitude en chacun des sommets d'une grille rectangulaire. L'acquisition d'une base de donnée altimétrique ainsi définie peut être réalisée soit automatiquement à partir de relevés aériens soit par interpolation sur une carte de courbes de niveaux [CHR83]. Ce deuxième modèle, bien que généralement plus coûteux à cause du nombre de faces à traiter, présente certains avantages:

. Ce modèle permet une modélisation sous forme de surfaces gauches en définissant des quadriques ou des bi-cubiques sur les éléments de la grille. On appelera élément de la grille ou facette, la surface définie entre les 4 sommets i,j; i+1,j; i+1,j+1; i,j+1 de la grille. Si la surface est gauche on nommera l'élément nappe (patch).

. La régularité et la symétrie induites par la description des données permettent de déduire un ordre sur les éléments de grille ou les nappes. On verra par la suite que cette propriété est à la base de plusieurs algorithmes d'élimination des parties cachées spécifiques à ce type de données.

1.1.2. Le sur-sol

On peut décomposer les informations relatives au sur-sol en deux catégories:

- La description surfacique du sur-sol permettant d'identifier les zones de culture, les voies... Cette description fournit une couleur ou une texture à appliquer sur le sol décrit précédemment.

- La description d'objets tri-dimensionnels tels que les bâtiments. Ces objets sont généralement représentés par des polyèdres qui peuvent éventuellement être définis avec différents niveaux de détails. D'autres modèles ont été étudiés tels que les octrees ou les modèles procéduraux qui permettent généralement de régler plus simplement le problème des niveaux de détails.

1.2. Visualisation

Dans ce paragraphe, nous allons présenter quelques techniques couramment utilisées lors de la visualisation de terrains. Dans un premier temps, nous présenterons brièvement la visualisation de terrains modélisés par des facettes triangulaires, nous étudierons ensuite de façon plus détaillée la visualisation de terrains définis sur une grille rectangulaire.

*** Réseau de triangles**

Les algorithmes d'élimination des parties cachées utilisés sont principalement basés sur une notion de liste de priorité associée à un balayage ligne à ligne de l'image. La construction de la liste de priorité peut faire appel à différentes techniques [SCH83], telles que les plans séparateurs, les priorités relatives mémorisées dans la base de données ou la hiérarchisation des données.

*** Grille rectangulaire**

Les algorithmes d'élimination des parties cachées agissant sur une base de données altimétriques définie sur une grille rectangulaire exploitent tous la notion de plans séparateurs. Soit Eij un élément de la grille. Les plans séparateurs sont des plans verticaux parallèles aux axes de la grille et positionnés entre les éléments. Ces plans permettent de produire un ordre sur les éléments de la grille. Cet ordre est défini de la façon suivante:

Eij < Ei'j' si et seulement si Ei'j' ne peut cacher Eij

Cette condition dépend évidemment des paramètres de visualisation courants. En termes de plans séparateurs, cette condition s'exprime de la façon suivante:

Eij < Ei'j' si et seulement si il existe un plan tel que Ei'j' est situé d'un côté du plan alors que Eij et l'observateur sont situés de l'autre côté du plan.

On distingue deux classes d'algorithmes appliqués à la visualisation de données définies sur une grille rectangulaire:

. **La liste de priorité**

La condition précédente est exploitée pour définir l'ordre de balayage. Pour une projection perspective, cet ordre est dépendant de la région étudiée. On définit quatre régions (éventuellement vides) délimitées par les plans x=i0 et y=j0, i0 et j0 étant les coordonnées entières de l'oeil suivant x et y. (cf figure 2-1). On peut alors définir un sens de balayage à l'intérieur de chaque région, compatible avec l'ordre défini précédemment [AND82], [COQ84a].

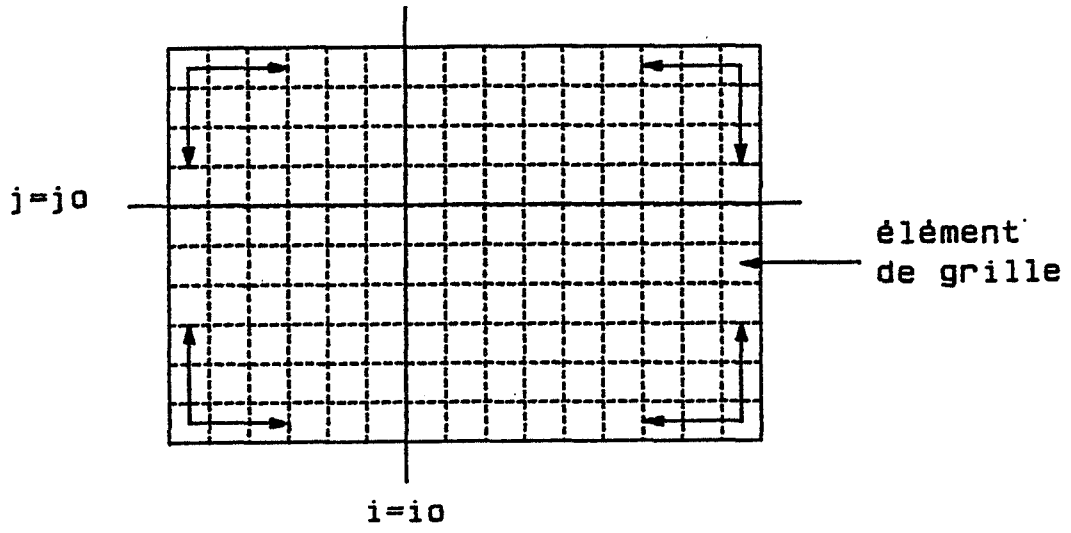


Figure 2-1 : Ordre de balayage de la grille régulière

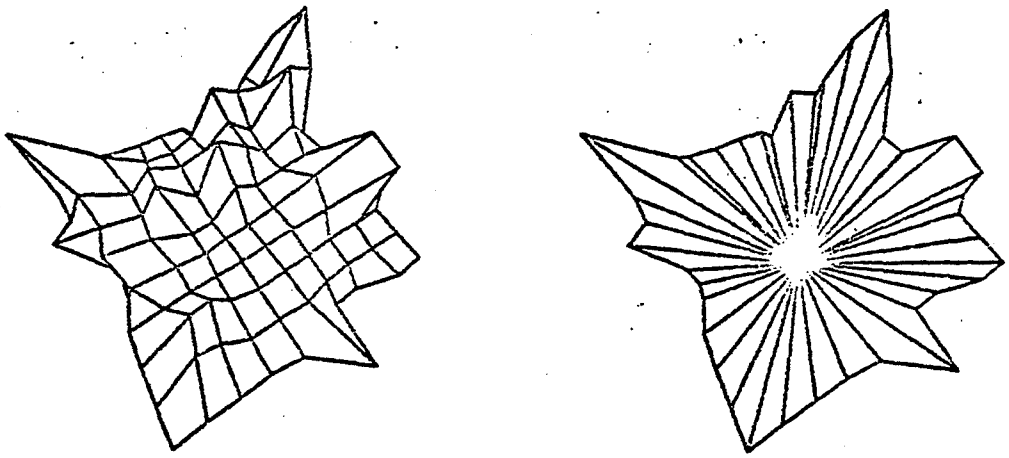
Un ordre entre les différents éléments étant établi, diverses techniques d'affichage peuvent être exploitées. La méthode la plus simple consiste à afficher d'arrière en avant par superposition. Cette méthode suppose un affichage rapide car beaucoup de facettes vont être affichées inutilement.

L'affichage d'avant en arrière suppose d'utiliser un masque. Les masques les plus couramment employés sont des masques linéaires codant la frontière de la tache. Williamson [WIL72] et Wright [WRI73] considèrent deux tampons pour mémoriser les deux lignes de crêtes supérieure et inférieure. La mise en oeuvre de ce modèle est simple mais il n'est pas valable dans tous les cas. Il suppose une cohérence de l'image suivant une direction. C'est à dire qu'il existe une direction telle que toute ligne parallèle à cette direction n'intersecte la tache qu'une seule fois. Ces conditions sont réalisées si la projection est une projection parallèle ou, pour une projection perspective, si la direction de visée est horizontale.

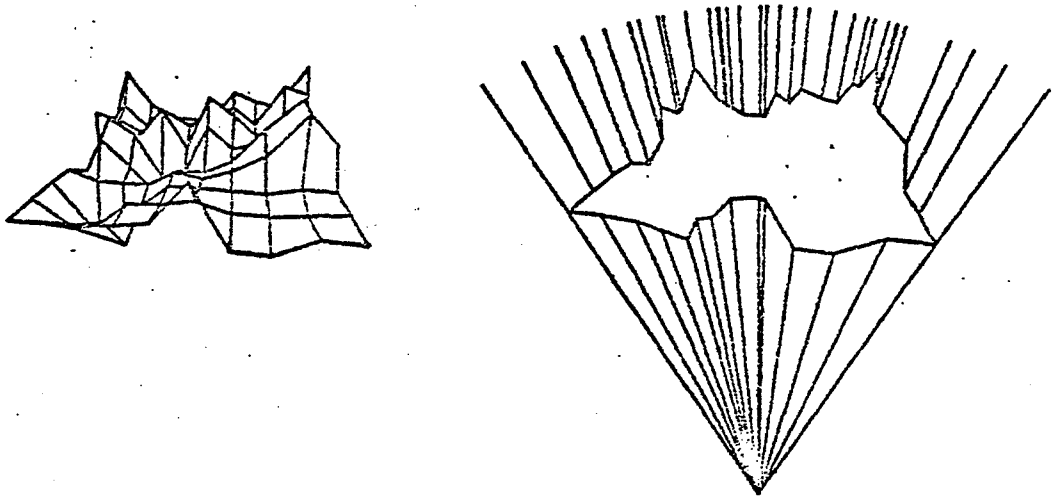
Pour résoudre le cas d'une projection perspective où la direction de visée n'est pas horizontale, D.P. Anderson [AND82] propose un masque étoilé. D.P. Anderson, montre qu'en considérant l'ordre de balayage dépendant des régions définies précédemment, la tache est à tout instant un ensemble connexe étoilé (cf. figure 2-2). Cette solution résout le problème de l'incohérence verticale de l'image. Deux techniques peuvent être envisagées:

- Définir le masque en coordonnées polaires et exprimer les coordonnées des sommets des faces à insérer en coordonnées polaires avant de tester leur inclusion dans le masque. Les différents cas particuliers pouvant se présenter lors du test d'intersection entre une facette et le masque sont étudiés en détail dans [AND82].

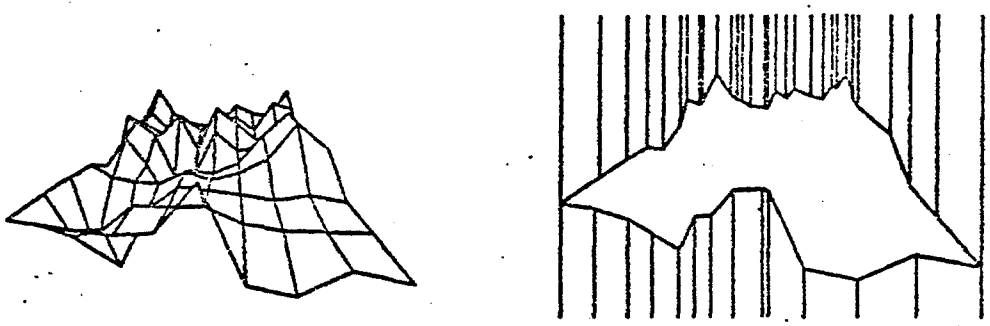
- Définir les lignes de crête par des segments de droites comme dans l'algorithme de Williamson. Le test de visibilité est alors un peu plus complexe. Il suppose de tester l'intersection entre le contour de la tache définie par les segments de droite et la droite joignant le point étudié au centre. Le centre est le point du plan de l'écran de coordonnées xoeil, yoeil.



cas 1 : vue de dessus



cas 2 : vue oblique



cas 3 : vue horizontale

Figure 2-2 : Représentation de la tache: les différents cas (cette figure est extraite de [AND82])

. **La sélection dans la base de données**

Une deuxième méthode, présentée par B. Fishman [FIS80] puis reprise par N.L. Max [MAX81] permet de sélectionner les éléments de grille à étudier. Supposons que la surface ne présente pas de surplomb, que la direction de visée soit horizontale et l'angle de roulis nul. L'angle de roulis est l'angle de l'écran autour de l'axe passant par son centre qui lui est perpendiculaire. On considère pour chaque colonne de l'image, le plan vertical dans le monde contenant l'oeil et la colonne de l'écran. Ce plan intersecte la surface topographique suivant une courbe C et le plan z=0 suivant la droite D. Par projection perspective, C a pour image la colonne de l'écran. Pour déterminer la valeur à affecter à chaque point de la colonne, B. Fishman sélectionne différents points uniformément répartis sur la droite D et étudie la projection sur l'écran du point de la surface associé. Pour réaliser une étude précise près de l'observateur sans calcul inutile dans les régions plus éloignées, N.L. Max propose de ne pas choisir d'étudier des points uniformément répartis le long de D mais de fixer l'écart entre deux points consécutifs en fonction de la distance à l'observateur.

On peut envisager d'exploiter cette technique avec un angle de roulis non nul, en exécutant une rotation de l'image. Cependant, pour une direction de visée non horizontale, les plans verticaux dans le monde ne correspondent plus à des lignes ou des colonnes de l'écran.

Nous allons présenter dans la suite de ce chapitre une troisième méthode qui au lieu d'étudier la projection de certains points de la surface, enverra dans la scène un rayon qui recherchera les éléments de surface qu'il est susceptible d'intersecter.

2. INTRODUCTION

La suite de ce chapitre est consacrée à l'étude de la représentation des terrains par la méthode du tracé de rayon. Les données sont fournies sous forme d'une base de données altimétique définie sur une grille rectangulaire. A chaque sommet (i,j) de la grille est associé une valeur z(i,j) définissant l'altitude du point. La base de données que nous avons utilisée représente une partie de la région des Vosges. Les dimensions de la grille sont 256x256 et les pas sur chacune des trois dimensions sont: 60m en x, 90m en y et 1m en z. Une unité élémentaire de la grille mesure donc 60x90m. On nommera ces unités les éléments de grille.

La structuration des données par une grille rectangulaire n'est pas économique en nombre d'éléments à traiter. En effet, on a vu qu'une modélisation sous forme de faces triangulaires est généralement composée de beaucoup moins d'éléments. Sachant que ce sont les tests d'intersection entre un rayon et un élément de

la base de données décrivant la scène qui sont pénalisants dans le tracé de rayon, associer l'algorithme du tracé de rayon à une base de données altimétrique définie sur une grille rectangulaire peut paraître dangereux. Cependant, la régularité et la cohérence induite par la grille rectangulaire vont permettre d'éviter de nombreux tests. Seuls les éléments de la grille traversés par le rayon courant et situés entre l'oeil et le premier point d'intersection devront être considérés.

Parmi les autres critères qui nous ont conduit à choisir le tracé de rayon comme algorithme d'élimination des parties cachées, le plus important est certainement le fait de pouvoir représenter de nombreux effets comme: les ombres portées, les réflexions dans un lac, le bleu atmosphérique, le flou dans les lointains..

Il est souvent difficile de mixer plusieurs scènes contenant différents types d'objets, [MAR79], [WHI82]. Contrairement aux algorithmes spécifiques à un type de données, le tracé de rayon peut être appliqué à des données variées. Le mixage des données issues de différentes scènes est alors plus facile; il suffit pour cela d'envoyer chaque rayon dans les différentes scènes et de considérer le point d'intersection le plus proche de l'oeil. Cette idée est exploitée à la fin du chapitre 3 où une base de données altimétrique et un ensemble d'objets polyèdriques sont associés. (cf. photo 2)

Cette étude commence par la description de l'algorithme général de synthèse. Afin d'accélérer l'exécution du programme, nous étudions ensuite une méthode permettant d'éviter un grand nombre de tests. Une évaluation des gains entrainés par cette optimisation suit. Avant de conclure cette présentation, certains effets spéciaux seront intégrés dans le processus de synthèse.

3. DESCRIPTION DE LA METHODE

Le squelette de cet algorithme est identique à celui de l'algorithme du tracé de rayon classique décrit dans le premier chapitre. Alors que dans le cas classique, la scène est composée d'objets variés non structurés, la surface topographique que l'on considère ici est constituée d'un grand nombre d'éléments de même type structurés par la grille rectangulaire sous-jacente. Cette structuration des données va permettre de n'étudier, pour chaque rayon, que les éléments dont la projection sur le plan z=0 est traversée par la projection du rayon sur ce même plan.

Algorithme 2-1 : Tracé de rayon appliqué à la visualisation d'une surface topographique définie sur une grille rectangulaire.

```
DEBUT
   saisie des paramètres de visualisation
   POUR chaque point de l'écran FAIRE
      calculer les paramètres définissant le rayon issu de
      l'oeil et passant par le point courant
      TANTQUE non intersection et
              non en dehors de la grille FAIRE
         rechercher l'élément de grille suivant
         tester l'intersection entre l'élément et le rayon
      FIN TANTQUE
      SI intersection ALORS
         traiter le point d'intersection
      FIN SI
   FIN POUR
FIN
```

Nous allons reprendre chaque module pour le décrire plus précisément.

3.1. Saisie des différents paramètres de visualisation

Il s'agit de toutes les données relatives à la vue courante. On distingue trois classes de paramètres:

* **les paramètres relatifs à la description géométrique de la vue.**
  Ils sont utilisés pour définir et calculer incrémentalement chacun des rayons primaires issus de l'oeil. L'équation de la demi-droite représentant un rayon est alors:

$$x0+tu,\ y0+tv,\ z0+tw \quad \text{avec } t>0$$

  Les calculs ont été détaillés au premier chapitre paragraphe 2.1.

* **les paramètres de description de la base de données altimétrique.**
  Voici la liste des paramètres définis dans l'implémentation réalisée:

  . choix d'une base de données et d'une fenêtre à l'intérieur de cette base.

. choix du pas de la grille suivant chacune des trois dimensions. Ces pas sont fournis avec la description des données. Cependant, un réajustement de la valeur du pas en z permet d'affecter aux altitudes un facteur multiplicatif. On remarquera sur les documents photographiques présentés dans ce rapport que nous avons abondamment usé de cet artifice afin d'accroître le relief de la région considérée.

* **les paramètres de description du rendu.**
Nous ne considérons pas, pour le moment, les effets spéciaux qui peuvent être intégrés dans le processus de synthèse. Nous décrirons simplement une visualisation sans ombres portées. Les seuls paramètres nécessaires sont les coefficients d'intensité ambiante et diffuse. Le modèle utilisé est le suivant [FOL82]:

- soit P un point de la surface,
- soit ps le produit scalaire entre le vecteur normal à la surface en P et le vecteur dirigé vers la source lumineuse à l'infini,
- soient Is l'intensité de la source lumineuse, Ia l'intensité ambiante, kd le coefficient de réflexion diffuse du sol et ka le coefficient de réflexion ambiante du sol.

La règle de calcul de l'intensité au point P est :

$$I = Ia\ ka + ps\ Is\ kd.$$

L'affectation d'une couleur à chaque point de la surface est dans la version actuelle très simple. Nous avons choisi la solution adoptée pour les cartes géographiques: la couleur d'un point est déduite de son altitude.

3.2. Recherche de l'élément de grille suivant

Nous allons exploiter la régularité de la grille. Pour chaque rayon, seuls les éléments de la grille traversés par la projection du rayon sur le plan z=0, contenant la base de la grille et situés entre l'oeil de l'observateur et le premier point d'intersecion sont considérés. Il est alors nécessaire de savoir générer une liste ordonnée de chacun des éléments de la grille traversés par le rayon.

Considérons:

* la grille rectangulaire définissant l'altimétrie,

* la projection PR du rayon sur le plan z=0 contenant la base de la grille.

Les éléments de grille à étudier sont les éléments traversés par PR dans l'ordre dans lequel ils sont traversés. On retrouve ici un des problèmes de base de l'informatique graphique: le tracé d'un segment de droite dans une mémoire de trame, [MOR76], [NEW79], [FOL82]. Notre application présente cependant trois particularités:

* nous ne disposons pas d'un segment de droite mais d'une demi-droite. Nous ne connaissons donc pas les deux extrémités du segment mais son origine et son équation.

* nous avons besoin de tous les éléments de grille traversés par le segment. Les algorithmes de tracé de segment de droite ne fournissent pas toujours tous les pixels traversés par le segment de droite.

* nous souhaitons connaître les coordonnées des points d'entrée et de sortie du rayon dans l'élément de grille.

Nous avons implémenté un algorithme incrémental adapté à la particularité de notre application.

Soit Dx le déplacement en x pour un déplacement en y égal au pas en y,

soit Dy le déplacement en y pour un déplacement en x égal au pas en x.

La méthode est basée sur la mémorisation des coordonnées de deux points: le premier point de changement d'élément en x, PX et son homologue en y, PY, (cf. figure 2-3). Le passage à l'élément suivant est réalisé en considérant le premier des deux points PX et PY. Soit par exemple PX ce point, l'élément suivant est donc l'élément adjacent à l'élément courant en x. Avant d'étudier cet élément, il est nécessaire de mettre à jour PX en incrémentant PX_x du pas en x et PX_y de Dy.

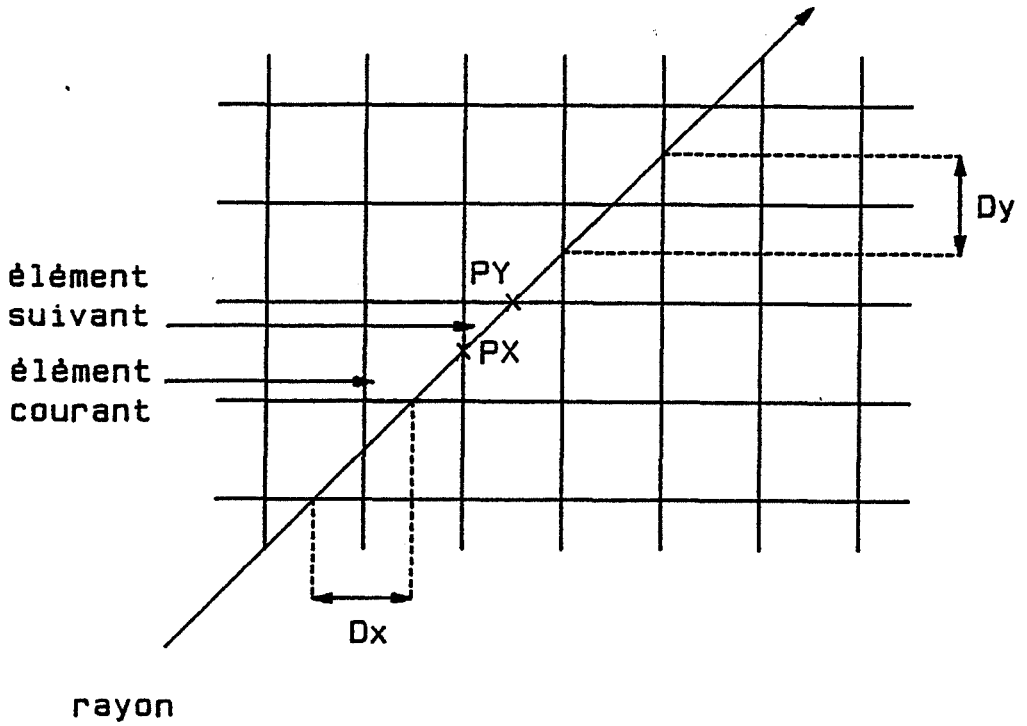


Figure 2-3 : Sélection des éléments de grille traversés par le rayon.

3.3. Tests d'intersection entre le rayon et un élément de grille

Cette opération étant renouvelée un grand nombre de fois, nous avons recherché un modèle de surface tel que les tests d'intersection entre un rayon et la surface soient rapides. Avant de présenter le modèle de surface choisi, nous allons étudier un modèle très simple: la modélisation par des facettes triangulaires planes. Deux facettes triangulaires planes sont définies sur chaque élément de la grille en construisant le plan passant par trois des sommets.

3.3.1. Test d'intersection entre un rayon et une facette plane triangulaire

Algorithme 2-2 : Test d'intersection entre la facette plane ABD et le rayon R, (cf. figure 2-4 pour les notations).

```
DEBUT
   SI la projection PR du rayon intersecte A'B'D' ALORS
      soient IE et IS les points d'entrée et de sortie de
             PR dans A'B'D'
      soient RE et RS les points du rayon qui se projettent
             en IE et IS
      soient FE et FS les points de la face plane qui se
             projettent en IE et IS
      SI (FE(z)<RE(z) et FS(z)>RS(z)) ou
             (FE(z)>RE(z) et FS(z)<RS(z)) ALORS
         le point d'intersection est le point d'intersection
         des deux segments de droite RE RS et FE FS
         (cf.figure 2-4)
      FIN SI
   FIN SI
FIN.
```

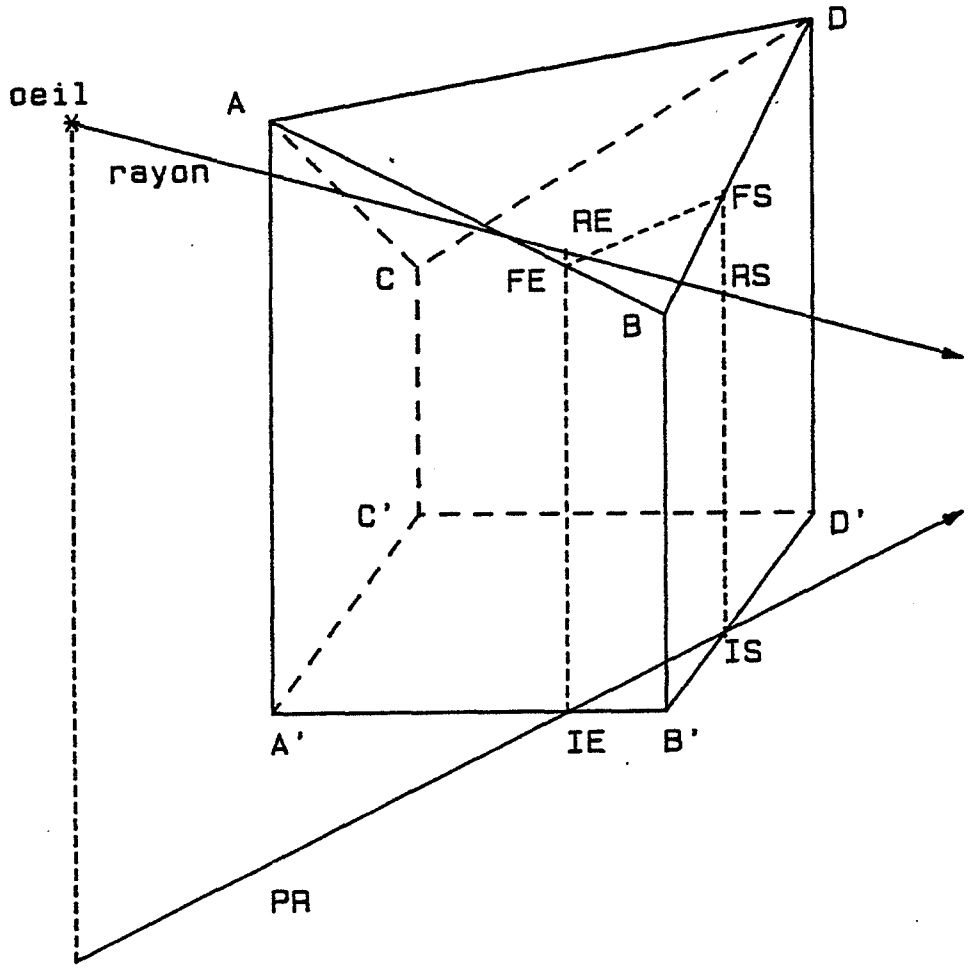


Figure 2-4 : Intersection entre une facette plane et un rayon

On notera qu'un test simple permet de ne pas considérer les faces arrières.

3.3.2. Le modèle de surface choisi

En accord avec la façon dont s'est déroulée l'étude, nous allons commencer par présenter la méthode, nous caractériserons ensuite la surface générée.

L'algorithme est extrèmement proche du précédent. Au lieu de modéliser la surface d'un élément de grille par deux faces planes nous allons considérer une seule surface A B C D à laquelle nous appliquons le même traitement que précédemment. FE et FS sont deux points appartenant aux segments AB ou BD ou CD ou AC ils se projettent en IE et IS (cf. figure 2-5). Comme précédemment, le point d'intersection I est le point d'intersection entre FE FS et RE RS s'il existe. En outre, la première action de l'algorithme:

le test d'intersection entre PR et la projection de la surface sur le plan z=0 n'est plus utile car, par définition, il y a intersection si le module est appelé.

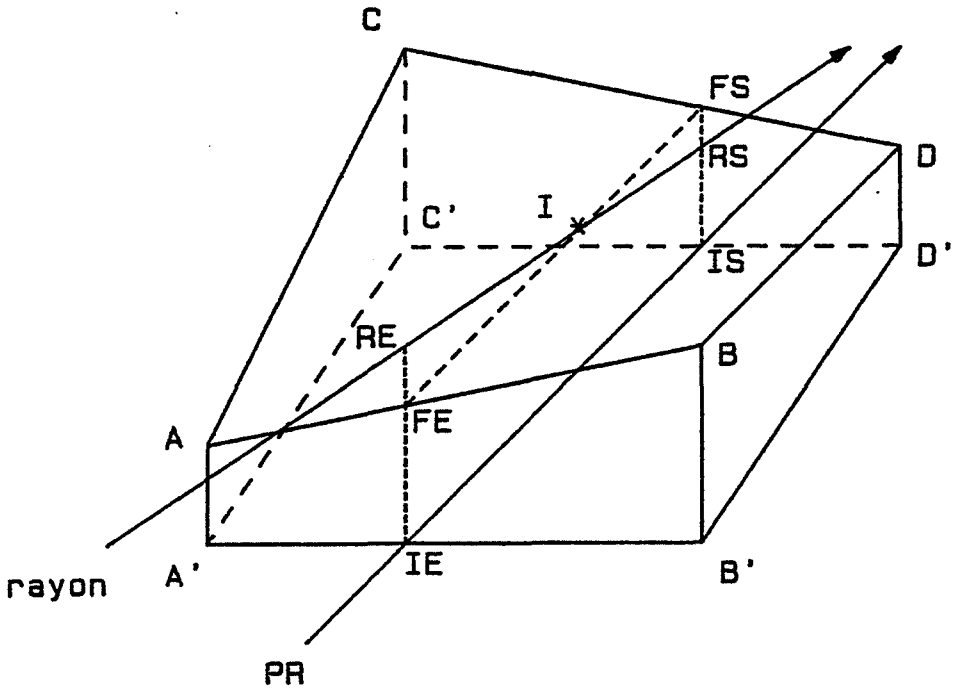


Figure 2-5 : Intersection entre un rayon et un élément de surface

La surface générée sur un élément de grille est constituée de la juxtaposition de deux types de surface:

* La surface définie sur les zones pour lesquelles IE et IS sont situés sur deux arêtes adjacentes est un plan (cf. figure 2-6). Ce plan est défini par les deux arêtes contenant FE et FS; ces arêtes étant deux arêtes successives d'un élément de grille, elles se coupent en un point qui est le sommet commun et ne sont pas confondues, elles définissent donc bien un plan. Si IE et IS sont situés sur deux arêtes opposées coplanaires alors la surface générée est aussi un plan défini par ces deux arêtes coplanaires non confondues.

* La surface définie sur les zones pour lesquelles IE et IS sont situés sur deux arêtes EE et ES opposées non coplanaires, est plus complexe. Elle est engendrée par l'ensemble des droites glissant sur EE et ES et situées dans un des plans verticaux passant par l'oeil. On peut donc affirmer que cette surface est engendrée par une droite s'appuyant sur les trois droites EE, ES et la droite

verticale contenant l'oeil. La surface engendrée est donc une quadrique réglée nommée hyperboloïde réglé. On trouvera une description complète des propriétés des hyperboloïdes dans l'ouvrage de A. Gheorghiu et V. Dragomir intitulé: "La Représentation des Structures Constructives", [GHE68]. Signalons cependant que si une des trois droites appartient au plan de l'infini alors, l'hyperboloïde est dégénéré en paraboloïde. Ce sera le cas pour une axonométrie. La surface engendrée a été choisie pour la simplicité des tests d'intersection; on constate cependant que les résultats sont visuellement très convenables (cf. photo 2,3). On peut par contre formuler quelques inquiétudes pour les applications d'images animées. En effet, la surface engendrée est dépendante de la position de l'oeil; on peut donc craindre que les déformations du terrain induites par les déplacements de l'oeil génèrent un tremblement de l'image gênant.

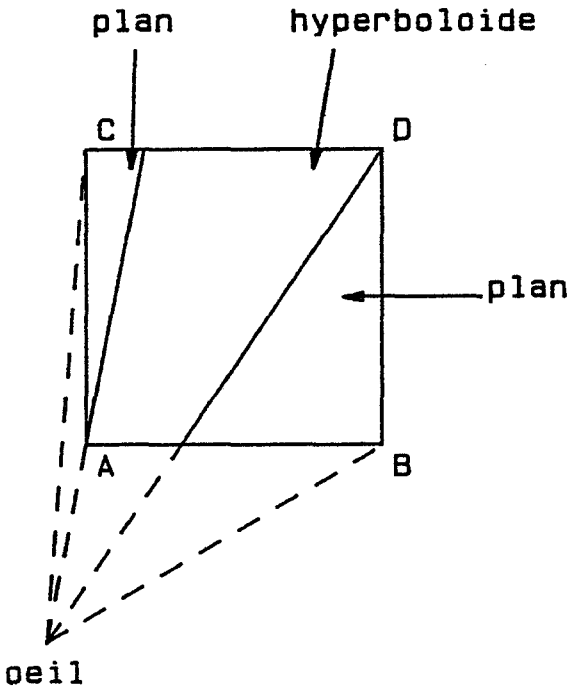


Figure 2-6 : Premier modèle de surface

Extensions

L'idée présentée dans ce paragraphe n'a pas été testée mais paraît cependant intéressante.

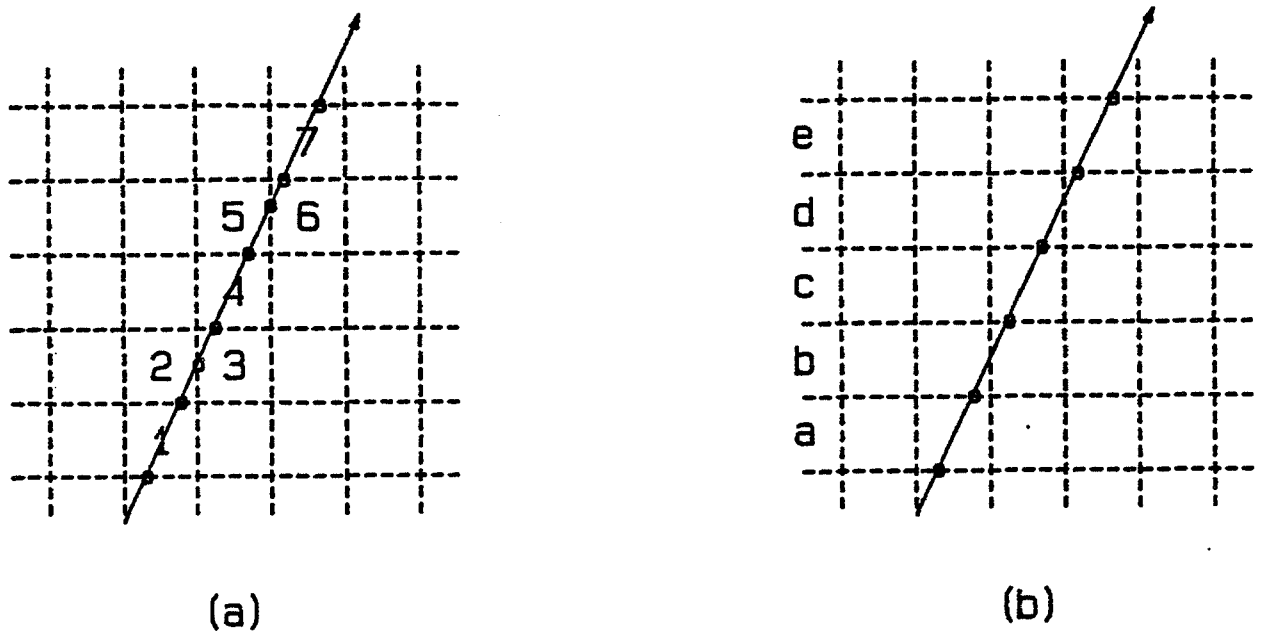


Figure 2-7 : Test d'intersection entre le rayon et
a : les éléments de la grille
b : les lignes de la grille

On constate sur la figure 2-7a que la projection du rayon traverse successivement les éléments 1,2,...,7. Toujours dans l'espoir de diminuer le nombre de tests, il serait avantageux de ne pas faire avancer le rayon d'élément en élément mais de ligne en ligne ou de colonne en colonne suivant son orientation (cf. figure 2-7b). Ainsi, on considère successivement l'intersection du rayon avec les lignes a,b,...,f. On doit ainsi obtenir un gain sur le nombre de tests d'intersection allant de 0 à 50%.

Avec cette solution, la surface générée n'est plus la même. Elle est définie par bandes, les points IE et IS sont donc situés sur deux arêtes opposées, n'appartenant pas forcément au même élément. Chaque bande est composée d'une succession d'hyperboloïdes réglés, à la limite dégénérés en plans si les deux arêtes contenant FE et FS sont coplanaires. (cf. figure 2-8).

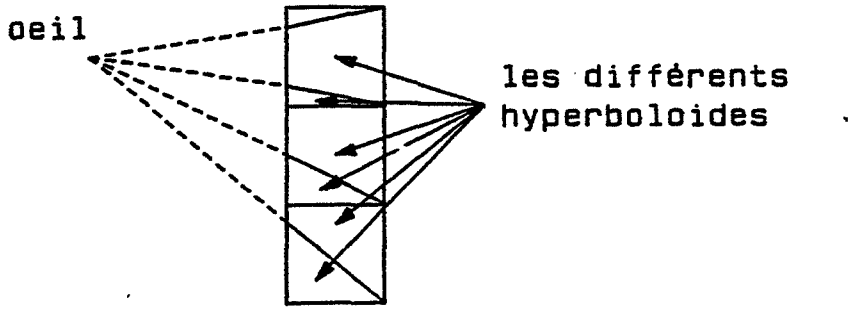


Figure 2-8 : Deuxième modèle de surface

4. COHERENCE VERTICALE

L'étude que nous allons présenter permet d'optimiser l'algorithme du tracé de rayon appliqué à la représentation de données altimétriques dans le cas où il n'existe pas de surplomb; c'est à dire, si à chaque couple (x,y) est associé une seule valeur z(x,y) représentant l'altitude du point.

Revenons tout d'abord à l'algorithme général décrit au paragraphe 3. Pour chaque rayon, cet algorithme réalise un test d'intersection avec chacun des éléments de la grille traversés par le rayon entre l'oeil et le premier point d'intersection. Nous allons montrer comment il est possible d'éviter certains tests.

4.1. Présentation de la méthode

Soit PV l'ensemble des plans verticaux dans le repère du monde passant par l'oeil. Nous allons considérer l'ensemble de toutes les demi-droites issues de l'oeil. Ces demi-droites sont classées en fonction du demi-plan vertical auquel elles appartiennent sauf si la demi-droite est verticale elle-même. Soient Di l'ensemble des demi-droites appartenant au demi-plan Pi.

Définition1:
Soient D et D' deux demi-droites appartenant à Di; D' est superposée à D si:

$$\forall (x,y,0) \in Pi,\ z'(x,y) \geq z(x,y)$$

z(x,y) (resp. z') représente la cote de la demi-droite D (resp. D') au point x,y.

Cet ordre est licite car D et D' sont dans un même plan vertical et elles ne s'intersectent pas sur l'intervalle considéré. (cf. figure 2-9).

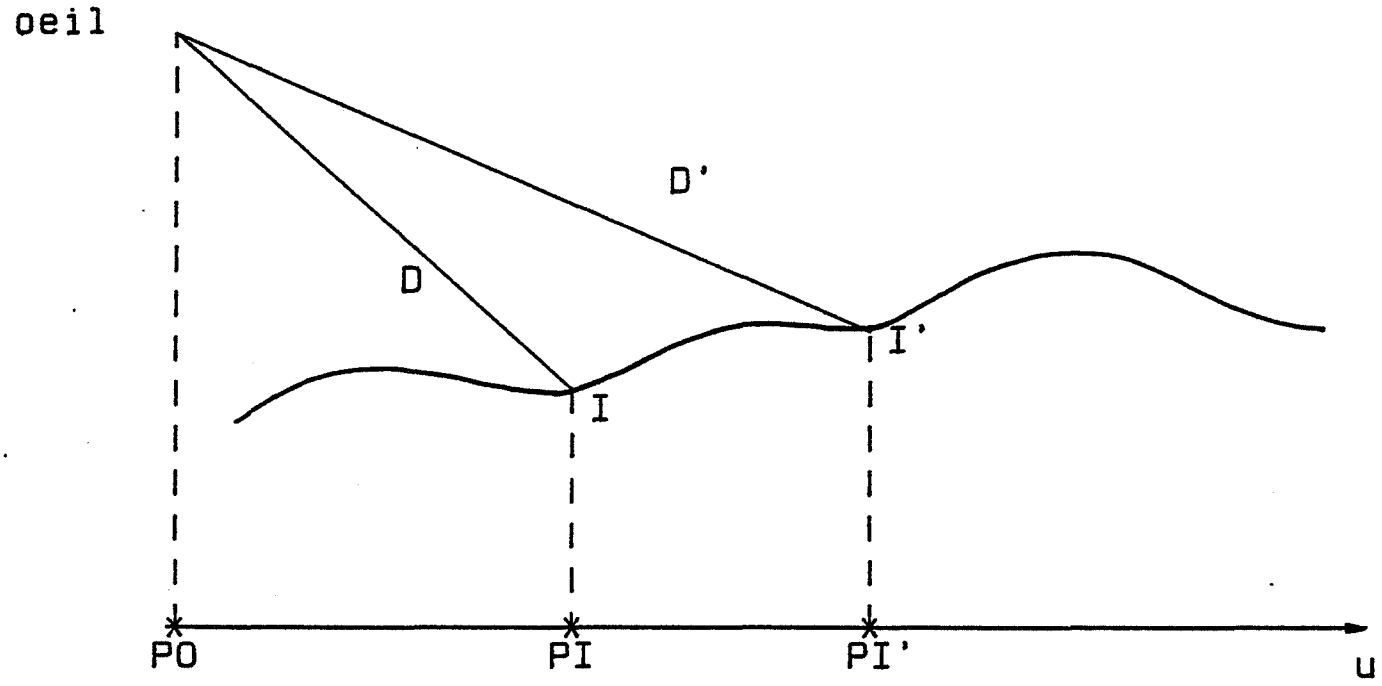


Figure 2-9 : Représentation du demi-plan Pi

On déduit de la définition 1 la propriété suivante:

Propriété1:
Soient deux demi-droites D et D' appartenant à Di.
Soient I (resp. I') le premier point d'intersection entre la demi-droite D (resp. D') et la courbe z=f(u) définie comme l'intersection de la surface et du plan Pi.
Soient PO, PI, PI' les projections de l'oeil et des points I et I' sur le plan de base z=0 parallèlement à l'axe des z.
Alors

$$D < D' \Longrightarrow d(PO,PI) < d(PO,PI')$$

où d représente la distance euclidienne.

Soient deux rayons R et R' issus de l'oeil, identifiés aux deux demi-droites D et D'. Sachant que R intersecte le terrain en I, on sait que R' ne peut intersecter le terrain en un point se projetant entre PO et PI. Ainsi, pour R', au lieu de commencer l'étude au point PO, on peut commencer en PI.

Pour appliquer ce résultat à l'algorithme du tracé de rayon défini précédemment, nous allons devoir faire certaines hypothèses supplémentaires. En effet, le tracé de rayon suppose d'envoyer des rayons traversant le centre des pixels. Deux rayons traversant deux pixels adjacents de la grille n'appartiennent généralement pas au même demi-plan vertical Pi. Dans ces conditions, il est difficile d'exploiter la notion de superposition telle qu'elle a été définie. Nous allons donc devoir élargir les notions précédentes.

Définition2:
Soient R1, R2 et R3, trois rayons issus de l'oeil, R2 et R3 traversent deux pixels adjacents de l'écran appartenant soit à la même ligne soit à la même colonne.
Soit E2 (resp. E3) le demi-espace contenant R3 (resp. E2) et délimité par le plan vertical contenant R2 (resp. R3).
Soit $E = E2 \cap E3$
Soit Q le plan contenant R2 et R3.
Soit Q^+ le demi-espace contenant le point à l'infini en z et délimité par le plan Q (cf. figure 2-10).

$$R1 \text{ est superposé à } R2 \text{ et } R3 \text{ si } R1 \in E \cap Q^+$$

La propriété 1 prend alors la forme suivante:

Propriété2:
Soient R1, R2 et R3 avec R1 superposé à R2 et R3. Soient I1, I2 et I3 les points d'intersection des rayons R1, R2 et R3 avec la surface et PO, PI1, PI2 et PI3 les projections de l'oeil et des points I1, I2 et I3 sur le plan de base z=0 (cf. figure 2-10). On a alors:

$$d(PO,PI1) \geq \min(d(PO,PI2),d(PO,PI3))$$

Cette propriété est valable si on suppose que R2 et R3 sont très proches, de façon à ce que la condition suivante soit vérifiée.

Condition
Soit T le triangle défini par l'oeil, I2 et I3. Soit PO, PI2, PI3 les projections des sommets de T sur le plan de base z=0 (cf. figure 2-10).

$$\forall xi,yi,zi \in T,\ zi \geq z(xi,yi)$$

Nous considérerons dans la suite que cette condition est vérifiée si R2 et R3 traversent deux pixels adjacents de l'écran, appartenant soit à la même ligne soit à la même colonne.

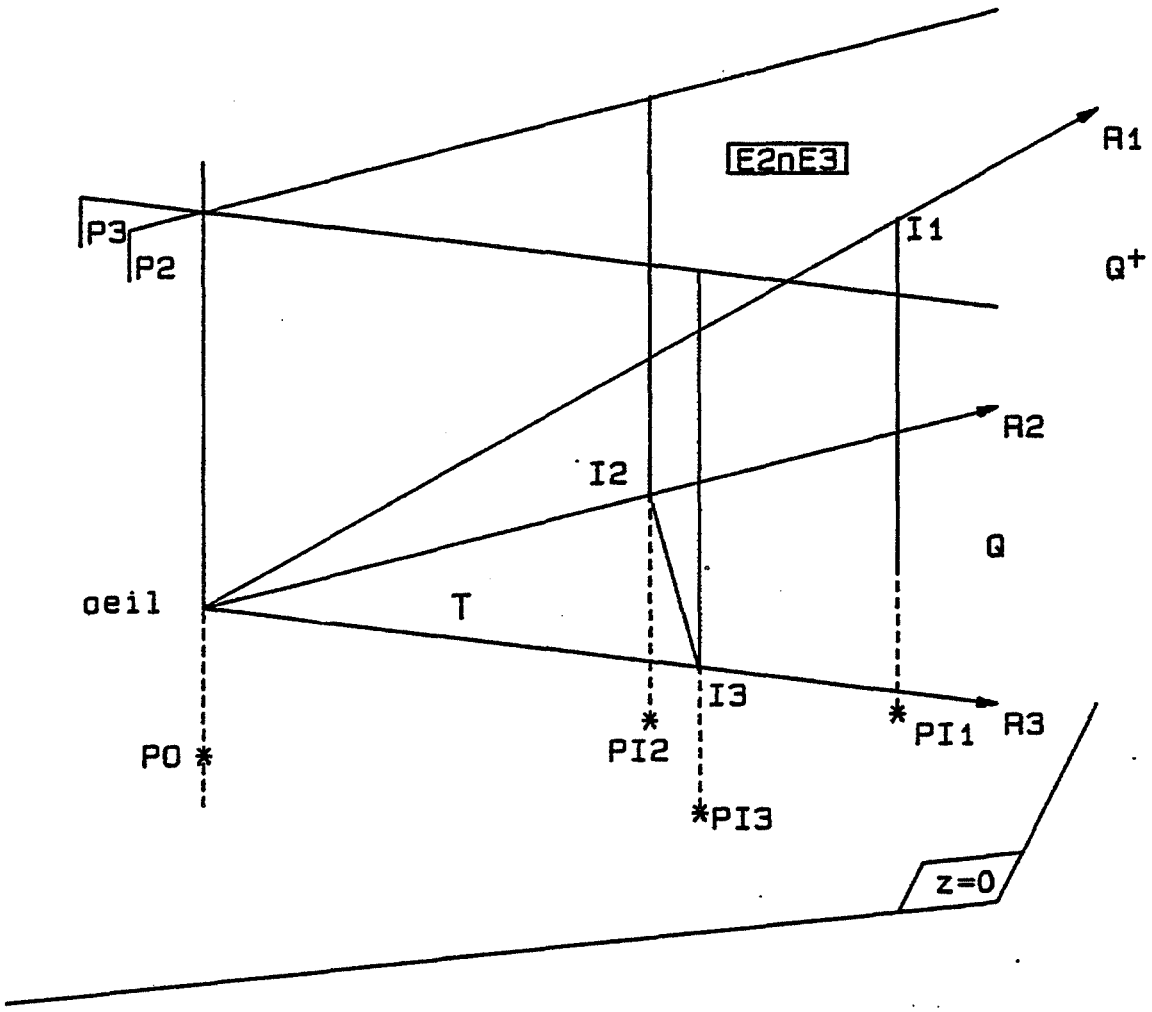


Figure 2-10 : Cohérence des points d'intersection

Un calcul rapide et simple montre que l'erreur induite par cette hypothèse est négligeable pour l'application concernée. Considérons un demi-angle de vision horizontal de 35° et un nombre de pixels par ligne d'écran égal à 640. Dans ce cas, l'angle horizontal maximum entre deux rayons consécutifs est proche de 1.125°. A une distance de 1000 mètres, la distance séparant deux rayons consécutifs est donc sensiblement égale à 2 mètres.

Appliquer le résultat précédent revient donc à considérer qu'à une distance de 1000m, on risque de négliger un pic dont la largeur est inférieure à 2m dans le monde.

Avant d'étudier l'implication de ces propriétés sur l'algorithme de visualisation des terrains, nous allons présenter quelques résultats préliminaires qui seront repris dans les démonstrations qui suivent.

4.2. Résultats préliminaires

Notons x',y',z' les coordonnées d'un point dans le repère du monde, l'équation d'un plan P élément de PV est:

$$a'x'+b'y'+d' = 0 \text{ avec } a'\neq 0 \text{ ou } b'\neq 0$$

Soit (xo,yo) les coordonnées de l'oeil dans le monde,

$$(xo,yo) \in P \Longrightarrow a'xo+b'yo+d = 0$$

L'équation des plans éléments de PV s'écrit donc:

$$a'x'+b'y'-(a'xo+b'yo) = 0$$

Cette équation définit un faisceau de plans.

Soit M la matrice 4×4 de passage du repère du monde au repère de l'oeil. Notons x,y,z les coordonnées d'un point dans le repère de l'oeil, on a:

$$(x\ y\ z\ 1) = (x'\ y'\ z'\ 1)M$$

et

$$(x'\ y'\ z'\ 1)MM^{-1}\begin{bmatrix} a' \\ b' \\ 0 \\ d' \end{bmatrix} = 0$$

donc

$$(x\ y\ z\ 1)M^{-1}\begin{bmatrix} a' \\ b' \\ 0 \\ d' \end{bmatrix} = 0$$

comme

$$M^{-1}\begin{bmatrix} a' \\ b' \\ 0 \\ d' \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix}$$

L'équation des plans de PV dans le repère de l'oeil s'écrit:

$$ax+by+cz=0$$

Dans la suite, nous travaillons dans le repère de l'oeil,

soit

$$x=V1t$$
$$y=V2t$$
$$z=V3t$$

l'équation paramétrique de la droite verticale passant par l'oeil, définie dans le repère de l'oeil. V3=0 correspond à une direction de visée horizontale.
Les plans de PV contiennent tous la droite verticale passant par

l'oeil donc:

$$aV1+bV2+cV3=0$$

L'équation des plans de PV peut donc s'exprimer sous l'une des trois formes suivantes:

$$aV3x + bV3y - (aV1+bV2)z = 0 \qquad \text{si } V3\neq 0$$
$$-(bV2+cV3)x + bV1y + cV1z = 0 \qquad \text{si } V1\neq 0$$
$$aV2x - (aV1+cV3)y + cV2z = 0 \qquad \text{si } V2\neq 0$$

Soit DV le faisceau des droites obtenues en intersectant les plans éléments de PV avec le plan de l'écran.
Soit k la distance oeil-écran, supposons V3≠0, alors:

$$ax+by = k\left(\frac{aV1+bV2}{V3}\right)$$

est l'équation du faisceau DV. Le sommet du faisceau est le point de fuite des droites verticales du monde dans la projection perspective.

Si V3≠0, ce point est défini par

$$OE = \begin{cases} x = k\frac{V1}{V3} \\ y = k\frac{V2}{V3} \\ z = k \end{cases}$$

Si V3=0 la projection des droites verticales du monde est un faisceau de droites parallèles sur l'écran, le point de fuite étant rejeté à l'infini.

4.3. Enoncé des propriétés pour les pixels de l'écran

Dans ce paragraphe, nous allons expliciter la relation d'ordre appliquée aux rayons traversant les pixels de l'écran. Si R1 est superposé à R2 et R3 alors R1 devra être étudié après R2 et R3. Pour ne pas alourdir l'énoncé, nous considérons la relation "est à étudier après", qui est la restriction de la relation de superposition aux rayons R1 adjacents à R2 ou R3 et situés soit sur la même ligne soit sur la même colonne que R2 ou R3. La relation "est à étudier après", est symbolisée par une flèche: $\rightarrow$. On écrira:

$$R1 \rightarrow R2;R3$$
ou
$$(i,j) \rightarrow (i+1,j+1);(i+1,j)$$

en identifiant le rayon et son point d'intersection avec l'écran.

Par abus de notation, on notera donc (i,j) au lieu de (i,j,k) les coordonnées des points de l'écran.

4.3.1. Direction de visée non horizontale (V3≠0)

Pour expliciter la relation d'ordre entre les rayons traversant les pixels de l'écran, il est nécessaire de considérer différentes régions de l'écran. Ces régions sont caractérisées par leur position par rapport au point OE. Après avoir énoncé les propriétés pour chacune des régions, nous démontrerons les résultats pour la région1.

4.3.1.1. Propriétés

[1] Région1:

$$i > OE(x)$$
$$j > OE(y)$$
$$i-OE(x) > j-OE(y)$$

. direction de visée descendante: $i-1 > OE(x)$
  ==> (i,j) → (i-1,j);(i-1,j-1)

. direction de visée ascendante
  ==> (i,j) → (i+1,j+1);(i+1,j)

[2] Région2:

$$i > OE(x)$$
$$j > OE(y)$$
$$i-OE(x) < j-OE(y)$$

. direction de visée descendante: $j-1 \geqslant OE(y)$
  ==> (i,j) → (i-1,j-1);(i,j-1)

. direction de visée ascendante
  ==> (i,j) → (i,j+1);(i+1,j+1)

[3] Région3:

$$i < OE(x)$$
$$j > OE(y)$$
$$|i-OE(x)| < |j-OE(y)|$$

. direction de visée descendante: $j-1 \geqslant OE(y)$
  ==> (i,j) → (i,j-1);(i+1,j-1)

. direction de visée ascendante
  ==> (i,j) → (i-1,j+1);(i,j+1)

[4] Région4:

$i < OE(x)$
$j > OE(y)$
$|i-OE(x)| > |j-OE(y)|$

. direction de visée descendante: $i+1 \leqslant OE(x)$
==> $(i,j) \rightarrow (i+1,j-1);(i+1,j)$

. direction de visée ascendante
==> $(i,j) \rightarrow (i-1,j);(i-1,j+1)$

[5] Région5:

$i < OE(x)$
$j < OE(y)$
$|i-OE(x)| > |j-OE(y)|$

. direction de visée descendante: $j+1 \leqslant OE(y)$
==> $(i,j) \rightarrow (i+1,j+1);(i+1,j)$

. direction de visée ascendante
==> $(i,j) \rightarrow (i-1,j);(i-1,j-1)$

[6] Région6:

$i < OE(x)$
$j < OE(y)$
$|i-OE(x)| < |j-OE(y)|$

. direction de visée descendante: $j+1 \leqslant OE(y)$
==> $(i,j) \rightarrow (i,j+1);(i+1,j+1)$

. direction de visée ascendante
==> $(i,j) \rightarrow (i-1,j-1);(i,j-1)$

[7] Région7:

$i > OE(x)$
$j < OE(y)$
$|i-OE(x)| < |j-OE(y)|$

. direction de visée descendante: $j+1 < OE(y)$
==> $(i,j) \rightarrow (i-1,j+1);(i,j+1)$

. direction de visée ascendante
==> $(i,j) \rightarrow (i,j-1);(i+1,j-1)$

[8] Région8:

$i > OE(x)$
$j < OE(y)$
$|i-OE(x)| > |j-OE(y)|$

. direction de visée descendante: $i-1 > OE(y)$
==> $(i,j) \rightarrow (i-1,j);(i-1,j+1)$

. direction de visée ascendante
==> $(i,j) \rightarrow (i+1,j);(i+1,j-1)$

[9] Cas particuliers

$$i=OE(x) \text{ ou } j=OE(y) \text{ ou } |i-OE(x)|=|j-OE(y)|$$

Dans ce cas, tous les points de la colonne, de la ligne ou de la diagonale, dans le dernier cas, passant par le point i,j appartiennent au même plan de PV. On peut donc appliquer les résultats présentés au paragraphe 4-1. Ainsi, pour chacun de ces points, la relation "est à étudier après" (cf. → sur la figure 2-11) est monovaluée.

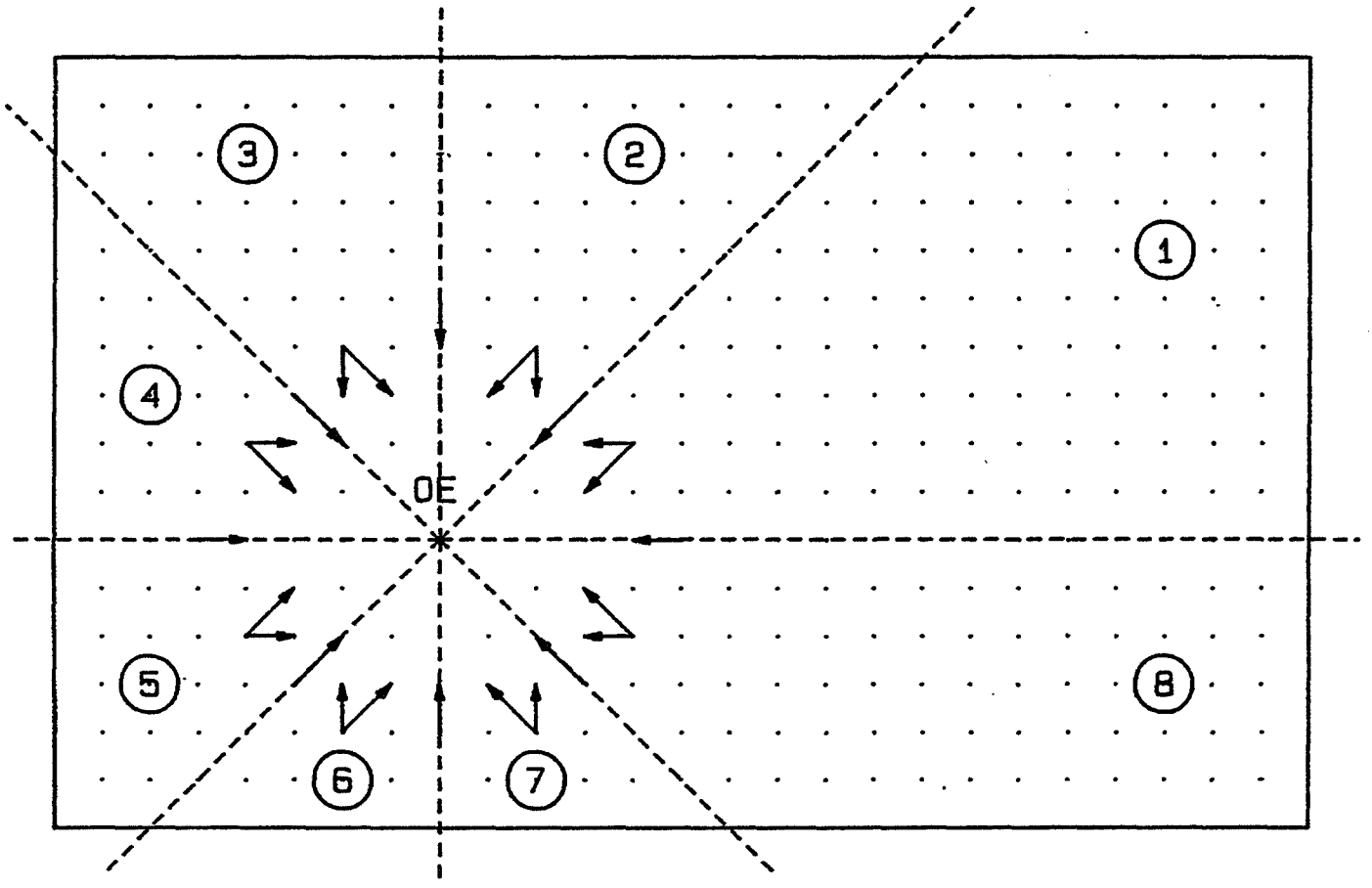


Figure 2-11 : Relation → : "est à étudier après" pour une direction de visée descendante

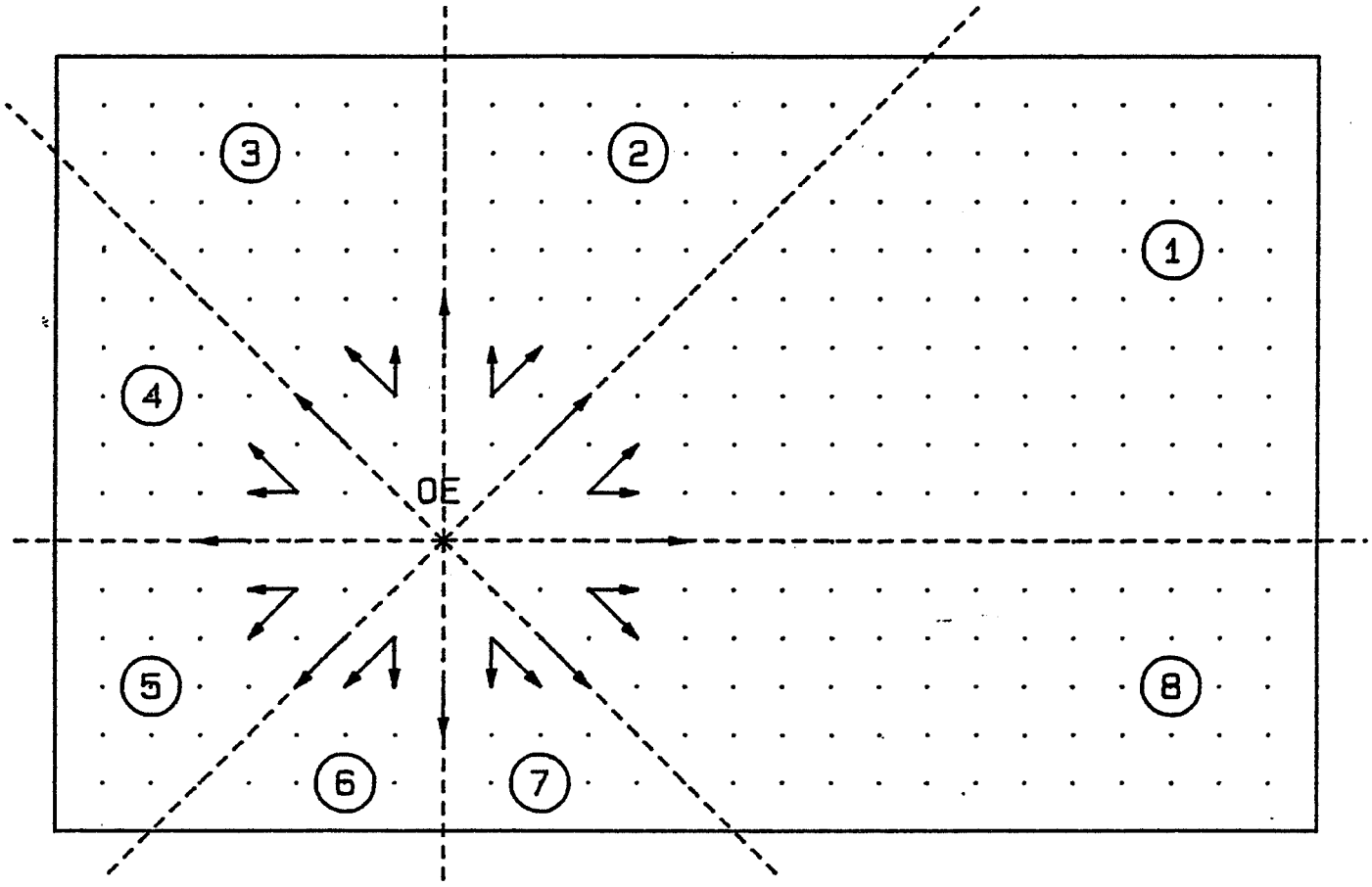


Figure 2-12 : Relation → : "est à étudier après" pour une direction de visée ascendante.

4.3.1.2. Démonstration

Démontrons les propriétés énoncées au paragraphe 4.3.1.1.

Equation dans le repère de l'oeil du plan de PV passant par le point i,j de l'écran:

$$\begin{vmatrix} x & i & V1 \\ y & j & V2 \\ z & k & V3 \end{vmatrix} = 0$$

$$x(jV3-kV2)-y(iV3-kV1)+z(iV2-jV1) = 0 \qquad (1)$$

[1] **Région 1 et direction de visée descendante**

Hypothèses:

direction de visée descendante (2)

$$j > OE(y) \iff j > k\frac{V2}{V3} \qquad (3)$$

$$i-OE(x) > j-OE(y) \iff i-k\frac{V1}{V3} > j-k\frac{V2}{V3} \qquad (4)$$

$$i-1 > OE(x) \iff i-1 > k\frac{V1}{V3} \qquad (5)$$

Montrons que:

$$(i,j) \rightarrow (i-1,j);(i-1,j-1) \qquad (6)$$

c'est à dire que (i,j) est à étudier après (i-1,j) et (i-1,j-1). Comme (i,j) et (i-1,j) sont adjacents, il suffit de montrer que

(i,j) est superposé à (i-1,j) et (i-1,j-1)

Soit P1 (resp. P2) le plan vertical contenant (i-1,j-1) (resp. (i-1,j)).

Soit E1 (resp. E2) le demi-espace contenant P2 (resp. P1) et délimité par P1 (resp. P2).

*** Montrons que (i,j) ε E1∩E2**

+ Montrons que (i,j) ε E1

Equation du plan P1 d'après (1),

$$x((j-1)V3-kV2) - y((i-1)V3-kV1) + z((i-1)V2-(j-1)V1) = 0$$

(i,j) ε E1 si Puiss((i,j)/P1) est du signe de Puiss((i-1,j)/P1) où Puiss((i,j)/P1) représente la puissance du point (i,j) par rapport au plan P1.

$$\underline{Puiss((i,j)/P1)} = i((j-1)V3-k(V2)-j((i-1)V3-kV1)+k((i-1)V2-(j-1)V1) = (j-i)V3 + k(V1-V2)$$

d'après (4),

$$j-i+k\frac{(V1-V2)}{V3} < 0$$

donc

(j-i)V3 + k(V1-V2) est du signe de -V3.

$$\underline{Puiss((i-1,j)/P1)} = (i-1)((j-1)V3-kV2)-j((i-1)V3-kV1)+k((i-1)V2-(j-1)V1) = V3(1-i)+kV1$$

d'après l'hypothèse 5,

$$(1-i)+k\frac{V1}{V3} < 0$$

donc

V3(1-i)+kV1 est du signe de -V3

+ **Montrons que (i,j) ∈ E2**

Equation du plan P2

$$x(jV3-kV2)-y((i-1)V3-kV1)+z((i-1)V2-jV1)=0$$

(i,j) et (i-1,j) sont situés du même côté du plan P2 si et seulement si Puiss((i,j)/P2) est du signe de Puiss((i-1,j-1)/P2).

$$\underline{Puiss((i,j)/P2)} = i(jV3-kV2)-j((i-1)V3-kV1)+k((i-1)V2-jV1)$$
$$= jV3-kV2$$

d'après (3),

$$j-k\frac{V2}{V3} > 0$$

donc

jV3-kV2 est du signe de V3.

$$\underline{Puiss((i-1,j-1)/P2)}$$
$$= (i-1)(jV3-kV2)-(j-1)((i-1)V3-kV1)+k((i-1)V2-jV1)$$
$$= V3(i-1) -kV1$$

d'après (5),

$$i-1 \geqslant k\frac{V1}{V3}$$

$$(i-1)-k\frac{V1}{V3} \geqslant 0$$

donc

V3(i-1)-kV1 est du signe de V3.

*** Montrons que (i,j) est superposé à (i-1,j) et (i-1,j-1)**

L'équation du plan Q défini par l'oeil et les deux points (i-1,j) et (i-1,j-1) est:

$$\begin{vmatrix} x & i-1 & i-1 \\ y & j & j-1 \\ z & k & k \end{vmatrix} = 0$$

soit $kx+(1-i)z = 0$.

La direction de visée étant descendante, la définition 2 sera vérifiée si OE et (i,j) ne sont pas du même côté du plan Q.

$$Puiss((i,j)/Q) = ki+k(1-i) = k$$

$$Puiss(OE/Q) = k(k\frac{V1}{V3}+(1-i))$$

d'après (5),

$$k\frac{V1}{V3}+(1-i) < 0$$

Puiss((i,j)/Q) n'est pas du signe de Puiss(OE/Q), donc (i,j) et OE ne sont pas du même côté du plan Q.

Par conséquent, sous les hypothèses (2)(3)(4)(5),

(i,j) est superposé à (i-1,j) et (i-1,j-1).

[2] Région 1 et direction de visée ascendante

Hypothèses:

direction de visée ascendante (2')

$$i > OE(x) \iff i > k\frac{V1}{V3} \quad (3')$$

$$j > OE(y) \iff j > k\frac{V2}{V3} \quad (4')$$

$$i-OE(x) > j-OE(y) \iff i-k\frac{V1}{V3} > j-k\frac{V2}{V3} \quad (5')$$

Montrons que

$$(i,j) \rightarrow (i+1,j+1);(i+1,j) \quad (6')$$

c'est à dire que (i,j) est superposé à (i+1,j+1) et (i+1,j).

Soit P1 (resp. P2) le plan vertical contenant (i+1,j+1) (resp. (i+1,j)). Soit E1 (resp. E2) le demi-espace contenant P2 (resp. P1) et défini par P1 (resp. P2).

*** Montrons que (i,j) $\in$ E1$\cap$E2**

La démonstration est analogue à celle du cas [1]. On démontre que (i,j) $\in$ E1$\cap$E2 en appliquant le même raisonnement que dans le cas précédent ou la direction de visée était descendante.

*** Montrons que (i,j) est superposé à (i+1,j+1) et (i+1,j)**

Pour démontrer que (i,j) est superposé à (i+1,j+1) et (i+1,j), on considère le plan Q défini par l'oeil et les deux points de l'écran (i+1,j+1) et (i+1,j) et on vérifie que le point (i,j) est situé du même côté du plan Q que le point OE; la direction de visée étant ascendante.

L'équation du plan Q défini par l'oeil et les deux points (i+1,j+1) et (i+1,j) est:

$$kx-(i+1)z = 0$$

$$Puiss((i,j)/Q) = ki-k(i+1) = -k$$

$$Puiss(OE/Q) = k(\frac{V1}{V3}-(1+i))$$

d'après (3'),

$$i > \frac{kV1}{V3} \Longrightarrow i+1 > \frac{kV1}{V3}$$

$$\Longrightarrow \frac{kV1}{V3}-(i+1) < 0$$

Puiss((i,j)/Q) et Puiss(OE/Q) sont de même signe donc (i,j) et OE sont du même côté du plan Q. Par conséquent, (i,j) est superposé à (i+1,j+1) et (i+1,j).

On démontre, par un raisonnement analogue, les propriétés énoncées au paragraphe 4.4.1.1 pour les autres régions.

4.3.2. Direction de visée horizontale (V3=0)

Dans ce cas, il n'y a qu'une seule région. Cependant, la relation "est à étudier après" est dépendante de l'angle de roulis θ.

[1] $0 < \theta \leqslant 45$
alors (i,j) → (i,j-1);(i-1,j-1)

[2] $45 < \theta \leqslant 90$
alors (i,j) → (i-1,j-1);(i-1,j)

[3] $90 < \theta \leqslant 135$
alors (i,j) → (i-1,j);(i-1,j+1)

[4] $135 < \theta \leqslant 180$
alors (i,j) → (i-1,j+1);(i,j+1)

[5] $180 < \theta \leqslant 225$
alors (i,j) → (i,j+1);(i+1,j+1)

[6] $225 < \theta \leqslant 270$
alors (i,j) → (i+1,j+1);(i+1,j)

[7] $270 < \theta \leq 315$
alors $(i,j) \rightarrow (i+1,j);(i+1,j-1)$

[8] $315 < \theta \leq 360$
alors $(i,j) \rightarrow (i+1,j-1);(i,j-1)$.

4.4. Ordre de balayage

L'ordre de balayage de l'écran est directement déductible des résultats précédents.

* Si la direction de visée est horizontale, le balayage est un balayage ligne à ligne ou colonne à colonne, en accord avec l'angle de roulis.

* Si la direction de visée est descendante, le balayage décrira une spirale autour de OE (cf. figure 2-13).

* Si la direction de visée est ascendante, l'ordre de balayage est l'inverse du précédent; la spirale est parcourue en sens inverse.

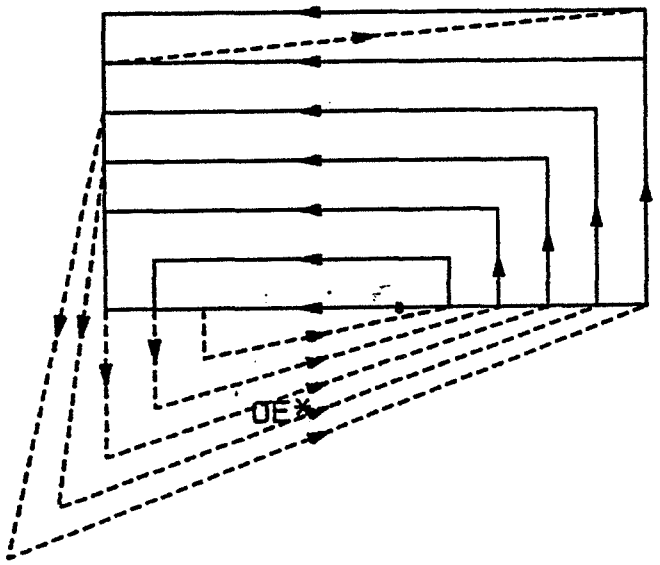


Figure 2-13 : Ordre de balayage de l'écran pour une direction de visée descendante

5. EVALUATION

Cette évaluation est basée sur une étude comparative du nombre de tests nécessaires pour chacune des trois méthodes proposées. Nous considérons le nombre moyen de tests d'intersection par rayon: NTR, obtenu en divisant le nombre total de tests par le nombre de rayons. Un test d'intersection est un test entre un rayon primaire et un élément de grille.

* **Méthode classique sans exploiter la structuration des données fournie par la grille rectangulaire.**

  Nous n'avons pas essayé d'exécuter l'algorithme classique du tracé de rayon sur la base de données altimétriques. Avec cette méthode, le nombre de tests par rayon est égal au nombre d'éléments de la scène. La grille que nous avons utilisée étant composée de 256×256 éléments, on aurait obtenu:

  NTR = 65 536

* **Méthode présentée au paragraphe 3 de ce chapitre.**

  Il est très difficile de calculer NTR dans ce cas. Nous avons préféré étudier les valeurs obtenues pendant l'exécution du programme. Pour cette méthode et pour la suivante, nous donnons les résultats obtenus avec la vue de dessus de la photo 3 et la vallée présentée photo 2.

  Pour la vue de dessus, le nombre de rayons primaires est 273 600 et le nombre de tests 23 234 185. Soit un nombre moyen de tests par rayon primaire égal à:

  NTR = 85

Pour la vallée, le nombre de rayons primaires est 310 554 et le nombre de tests 13 578 127. Le nombre moyen de tests par rayon primaire est égal à:

NTR = 44

La différence entre ces deux valeurs est due à la distance moyenne séparant la surface de l'oeil. La vallée est beaucoup plus près de l'observateur que la chaine de montagne visualisée sur la vue de dessus.

* **Méthode optimisée en exploitant la cohérence verticale des données.**

  Cette méthode à été étudiée en se plaçant dans les mêmes conditions que précédemment, en particulier, le nombre de rayons primaires de chaque image est le même.

  Pour la vue de dessus, le nombre total de tests est 449 747, donc:

$$NTR = 1.45$$

  Pour la vallée, le nombre total de tests est 463 188 donc:

$$NTR = 1.49$$

  Dans ce cas, le nombre de tests nécessaire pour la vallée est supérieur car le nombre de rayons intersectant la surface est plus grand: 169 079 contre 147 923 pour la vue de dessus.

On constate par conséquent que l'exploitation de la cohérence verticale des données permet de diviser par 80 le nombre de tests d'intersection.

6. RESULTATS

Toutes les photos présentées dans ce chapitre visualisent une zone de la région des Vosges. Une vue de la totalité de la base de données est présentée photo 1. Cette image a été réalisée avec l'algorithme du tracé de rayon, sans représentation des ombres portées.

Les tests sont basés sur deux vues différentes: une vue de dessus (cf. photo 3) pour laquelle les altitudes sont multipliées par trois et une vue d'une vallée (cf. photo 2) où les altitudes sont multipliées par deux.

Les photos 2 et 3 sont le résultat de l'exécution de l'algorithme du tracé de rayon avec calcul des ombres portées. Ces deux images ont été réalisées sans exploiter la cohérence verticale. Différents tests comparatifs sur la cohérence verticale ont été réalisés. Les gains en nombre de tests sont généralement très appréciables (cf. évaluation précédente), par contre, aucune différence entre les images n'a été perçue. On peut affirmer que, pour les tests réalisés, l'exploitation de la cohérence verticale ne modifie pas l'image résultante.

La simulation du bleu atmosphérique est un effet facile à réaliser avec la méthode du tracé de rayon. La quantité de bleu ajoutée à la couleur d'un point augmente avec la distance de l'observateur à ce point en suivant l'équation suivante:

$$q \leftarrow \exp(-\text{gamma} \times \text{distance})$$
$$\text{couleur} \leftarrow \text{couleur} \times q + \text{bleu}(1-q)$$

Nous avons aussi essayé d'insérer dans les images du flou dans les lointains. B. Fishman et B. Schachter [FIS80] représentent cet effet en faisant diminuer l'intensité exponentiellement avec la distance. Les meilleurs résultats ont été obtenus en appliquant un filtrage 3×3 dépendant de la distance. Ces essais ne sont qu'embryonnaires, (cf. photo 4) et devront être poursuivis. On peut envisager la démarche suivante: un sur-échantillonnage dans les zones crénelées situées près de l'observateur [WHI80] et un filtrage avec sous échantillonnage dans les zones éloignées de l'oeil. Ces zones doivent apparaître floues, il est donc inutile de les représenter avec une résolution normale et de les filtrer ensuite. Une résolution plus faible suivie d'un filtrage devrait suffire.

CHAPITRE 3

LE TRACE DE RAYON : UNE OPTIMISATION

LE TRACE DE RAYON : UNE OPTIMISATION

1. INTRODUCTION

Nous présentons dans ce chapitre une structure de données et un algorithme permettant d'accélérer le tracé de rayon pour des bases de données abritant un grand nombre d'objets. Nous ne reviendrons pas sur les possibilités de rendu offertes par le tracé de rayon. Rappelons cependant que les coûts induits par les nombreux tests d'intersection impliquent souvent la création d'images de faible complexité, en nombre d'objets présents dans la base de données. Dans la littérature, les images réalisées grace aux méthodes de tracé de rayon ne présentent souvent qu'un petit nombre d'objets. Il y a deux façons de considérer le problème: utiliser des moyens de calcul très puissants, éventuellement avec des architectures parallèles, ou essayer de diminuer les temps de calculs en diminuant le nombre de tests; la meilleure solution étant évidemment de mixer les deux. Les résultats présentés dans ce chapitre contribuent à la deuxième solution.

Après avoir exposé les buts poursuivis par la présente étude, nous montrerons comment se situent les méthodes existantes vis-à-vis de ces objectifs. Ensuite, nous présenterons une méthode basée sur une mémorisation des adjacences entre les objets de la scène.

2. LES OBJECTIFS

Nous souhaitons rendre possible l'exécution d'un algorithme de tracé de rayon sur des scènes composées de nombreux objets. Il est donc nécessaire d'essayer de rendre le nombre de tests d'intersection indépendant de la taille de la base de données.

Deux idées complémentaires doivent être exploitées:

- restreindre l'étude aux objets proches du rayon. La sélection des objets proches du rayon nécessite un prétraitement qui peut être:

  . local à chaque objet; le test sur la boîte englobante des objets est un excellent exemple de prétraitement local. Dans ce cas, le coût du traitement pour chaque rayon est:

$$N \times TP + N(r) \times T$$

où:
N est le nombre d'objets
TP est le temps moyen de prétraitement pour un objet
N(r) le nombre d'objets sélectionnés
T le temps moyen d'une étude d'intersection entre un rayon et un objet

. global; dans ce cas, le prétraitement n'est plus exécuté sur chaque objet mais sur la scène dans son ensemble. Cette méthode suppose généralement l'existence d'une structure permettant d'organiser spatialement les objets de la scène indépendamment du point de vue. La complexité ne dépend plus du nombre d'objets de la scène. L'évaluation du coût pour chaque rayon est:

$$PT + N(r) \times T$$

où PT est le temps moyen du prétraitement global.

- arrêter l'étude au premier point d'intersection détecté. Ce deuxième critère suppose l'existence d'un ordre entre les objets. L'étude des objets, du plus proche au plus éloigné, permet d'arrêter les tests au premier point d'intersection trouvé. Cette deuxième idée est très importante. Supposons que l'on dispose d'une scène infinie; le premier critère ne permet pas de déterminer en un temps fini le premier point d'intersection entre la scène et un rayon, par contre, le deuxième le permet sauf dans le cas où ce point n'existe pas!

La combinaison de ces deux critères s'avère par conséquent intéressante. Par ailleurs, il est important que la méthode adoptée puisse être appliquée aux rayons primaires comme aux rayons secondaires.

3. LES METHODES EXISTANTES

Le chapitre 1 sur l'état de l'art du tracé de rayon a permis de décrire l'algorithme de base du tracé de rayon ainsi que plusieurs extensions. Ce paragraphe est consacré à la présentation de structures de données ou d'algorithmes qui n'ont pas encore à notre connaissance été combinés avec un algorithme de tracé de rayon mais qui permettraient de satisfaire certains des objectifs énoncés précédemment.

Rappelons tout d'abord que l'algorithme développé pour visualiser les terrains présenté au chapitre 2 paragraphe 3 exploite ces deux propriétés. Ceci est rendu possible par la structure de données (grille régulière) qui permet de décrire les facettes traversées par le rayon de façon ordonnée.

De même, ces critères pourraient facilement être exploités pour la visualisation des intérieurs de bâtiments, cela constituerait une extension des algorithmes présentés par C.B. Jones [JON71] et M. Gangnet [GAN83]. Ces algorithmes sont basés sur la notion d'adjacence entre pièces. La structure de données contient un ensemble de pièces structurées par le graphe d'adjacence et un ensemble d'ouvertures définies entre deux pièces adjacentes. Cette structure permet de guider chaque rayon à l'intérieur de la scène. Une pièce sera étudiée si le rayon a traversé l'une de ses ouvertures. Ainsi, seule les pièces traversées par le rayon sont considérées. De plus, le premier point d'intersection entre le rayon et une face permet de conclure. La structure étant indépendante du point de vue, cette technique est aussi valable pour les rayons secondaires. Pour cette application particulière, il est donc possible de satisfaire les objectifs énoncés ci-dessus.

Comme nous souhaitons que cette étude ne soit pas restreinte à un nombre limité d'applications, nous avons examiné des structures de données permettant d'abriter des scènes variées.

Essayons d'envisager l'utilisation de l'algorithme d'élimination des parties cachées de Schumacker & al. décrit dans [NEW79] page 383-384, chapitre 24. Cet algorithme est basé sur la notion de plans séparateurs situés entre les objets. La structure sous-jacente est un arbre binaire. L'arbre est formé de plans séparateurs aux noeuds non terminaux et d'objets aux noeuds terminaux, (cf. figure 3-1).

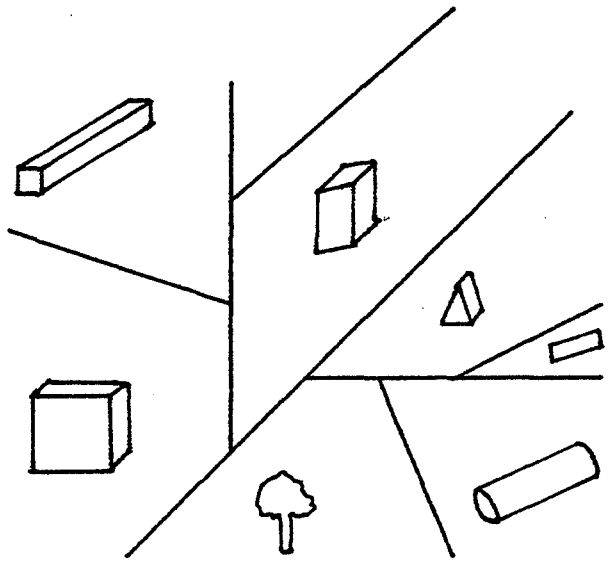


Figure 3-1 : Arbre des plans séparateurs.

Cette structure permet de décrire tous les objets traversés par le rayon de façon ordonnée; elle pose cependant certains problèmes:

- Cette structure n'est pas facile à construire et peut nécessiter de découper, ou de regrouper des objets. La figure 3-2 présente un exemple où il n'est pas possible de positionner des plans séparateurs sans découper les objets.

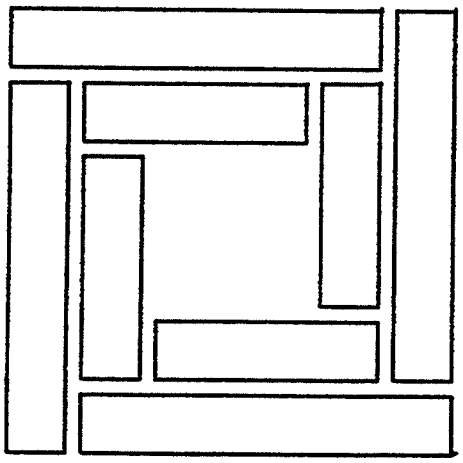


Figure 3-2 : Le positionnement des plans séparateurs est impossible.

- Si les objets sont complexes, par exemple formés de surfaces gauches, le découpage risque d'être très pénalisant.

- L'ordonnancement des objets suppose de nombreux tests d'intersection entre le rayon et des plans quelconques: les plans séparateurs.

Nous avons choisi une solution proche de celle-ci mais plus simple à mettre en oeuvre.

4. STRUCTURE DE DONNEES CHOISIE

Nous sommes parvenus, pour deux applications particulières: les terrains et les intérieurs de bâtiments, à définir une organisation des éléments de la scène telle que les objectifs précédents soient satisfaits. Nous nous sommes appuyés sur ces deux expériences pour développer la méthode décrite dans ce chapitre. Si les objectifs sont les mêmes, le champ d'application est beaucoup plus large. On considère ici une scène formée d'objets répartis dans l'espace. L'organisation et le tri de ces

objets sont fournis par une structure de données décrite ci-dessous. L'idée principale consiste à structurer la scène. On ne présente plus les objets au rayon mais c'est maintenant le rayon qui chemine dans la scène en sélectionnant les objets qu'il est susceptible d'intersecter.

Par la suite, nous allons travailler sur les boîtes englobantes des objets définies dans un même repère, par exemple celui du monde. Ces boîtes sont des parallélépipèdes rectangles tous parallèles entre eux.

Limitons nous pour le moment à un espace de projection bi-dimensionnel. On trouvera dans le dernier paragraphe de ce chapitre une discussion sur le choix de l'espace de projection. Considérons la projection des boîtes englobantes décrites ci-dessus sur le plan de projection défini par deux des axes du repère. Cette projection est une projection orthogonale parallèle au troisième axe. Dans ce cas, les projections des boîtes englobantes seront des rectangles parallèles entre eux, nommés rectangles englobants, définis dans le plan de projection. Nous avons choisi le plan x-y comme plan de projection; la projection est donc réalisée parallèlement à l'axe des z. Ce choix est motivé par une dispersion de la scène qui est généralement plus importante en x,y qu'elle ne l'est en z. Si ce n'était pas le cas, il serait possible de choisir un plan de projection différent.

Le plan de projection est alors décomposé en deux régions: la région [E] composée des zones où sont projetées des boîtes englobantes d'objets (ou rectangles englobants), et son complémentaire, la région [V]. Ces deux régions vont recevoir un traitement différent:

La région [V] va être partitionnée en rectangles parallèles aux axes nommés rectangles vides. Il est alors possible de définir une relation d'adjacence entre les rectangles (englobants ou vides). Cette relation d'adjacence constitue un guide pour gérer les déplacements des rayons dans la scène.

La région [E] va être structurée pour gérér les adjacences et les superpositions entre les rectangles englobants. La gestion des adjacences est commune aux rectangles englobants et vides. La gestion des superpositions peut être réalisée de différentes façons. Nous présentons ici la solution étudiée; une deuxième solution sera décrite dans le paragraphe 8 . On considère l'ensemble des rectangles englobants et pour chacun, on définit une liste des objets dont le rectangle englobant intersecte le rectangle englobant étudié. On définit ainsi une relation de superposition.

4.1. Définition des objets.

Soit une scène à visualiser. Cette scène est composée d'objets; un objet peut représenter tout sous-ensemble de la scène quel que soit son niveau de décomposition. Un objet peut être:

. élémentaire: une sphère, un cube,...

. composé: une table, une chaise,...

La liberté laissée dans la définition d'un objet est primordiale pour la généralité de l'étude. Un objet est défini par:

. un algorithme d'intersection de l'objet avec un rayon dans le résultat est soit un booléen à faux soit le premier point d'intersection.

. sa boîte englobante définie dans le repère du monde.

4.2. Partitionnement de l'espace de projection

Pour rendre possible la description des relations d'adjacence entre les rectangles englobants, nous allons créer des rectangles vides entre les rectangles existants (région [V]) qui formeront un recouvrement du plan. Ces rectangles vides sont des rectangles parallèles aux axes.

- Les rectangles vides sont générés de façon à ce que leur union avec la région [E] recouvre entièrement le plan. Le plan est généralement limité à la projection sur le plan de la boîte englobante de la scène.

- Par définition, un rectangle vide ne doit pas intersecter de rectangles englobants et deux rectangles vides ne doivent pas s'intersecter.

La figure 3-3 montre, sur un exemple, l'ensemble des rectangles englobants et une configuration de rectangles vides associés. Dans la suite, un rectangle Ri représentera soit un rectangle vide soit un rectangle englobant. Ri sera défini comme un produit d'intervalles pour la topologie naturelle de R^2:

$$Ri=[xi1,xi2]\times[yi1,yi2].$$

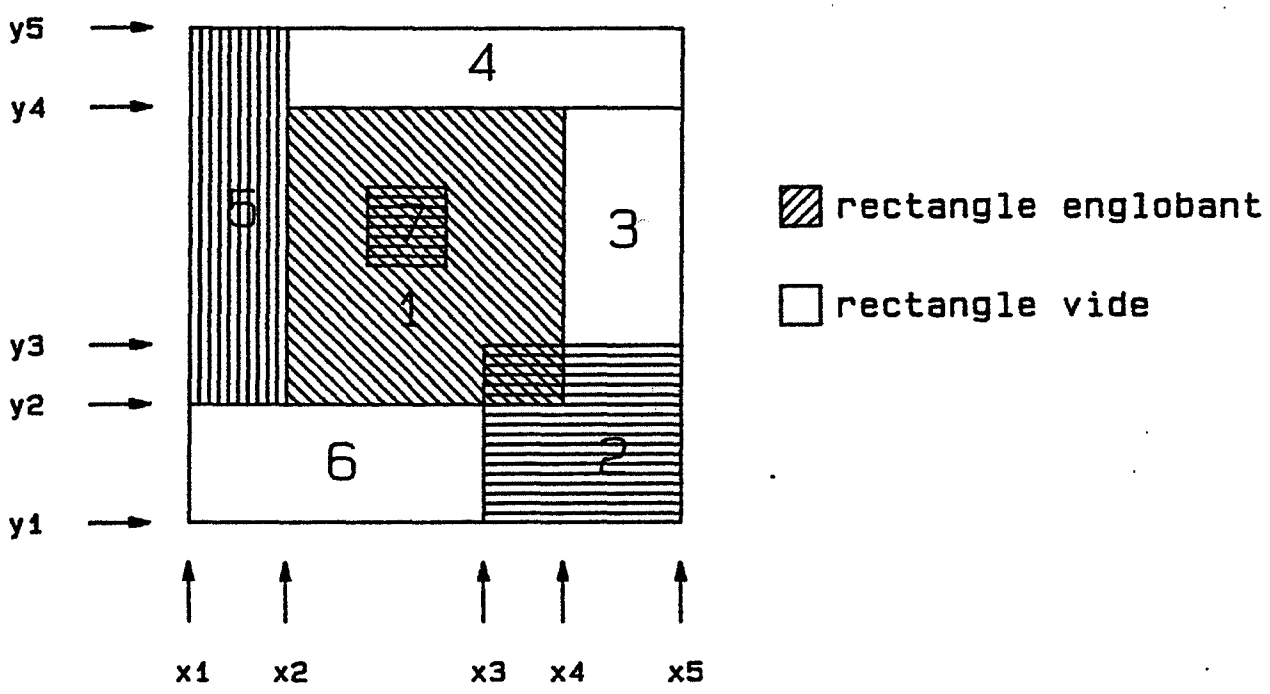


Figure 3-3 : Représentation des rectangles englobants et vides

4.3. Relation d'adjacence.

Il est important pour l'algorithme que l'on va décrire par la suite de connaître toutes les adjacences entre rectangles. Par définition, l'ensemble des rectangles Rj adjacents à un rectangle Ri=[xi1,xi2]×[yi1,yi2] est défini par Rj=[xj1,xj2]×[yj1,yj2] tel que:

$$[xi1,xi2]\cap[xj1,xj2] \neq \emptyset$$
$$[yi1,yi2]\cap[yj1,yj2] \neq \emptyset$$
$$Rj \not\subset Ri$$
$$Ri \neq Rj$$
$$xi2 \neq xj1 \text{ ou } yi1 \neq yj2$$
$$xi2 \neq xj1 \text{ ou } yi2 \neq yj1$$
$$xi1 \neq xj2 \text{ ou } yi2 \neq yj1$$
$$xi1 \neq xj2 \text{ ou } yi1 \neq yj2$$

Ces adjacences sont représentées sous forme d'un graphe. Le graphe d'adjacence de l'exemple de la figure 3-3 est décrit figure 3-4.

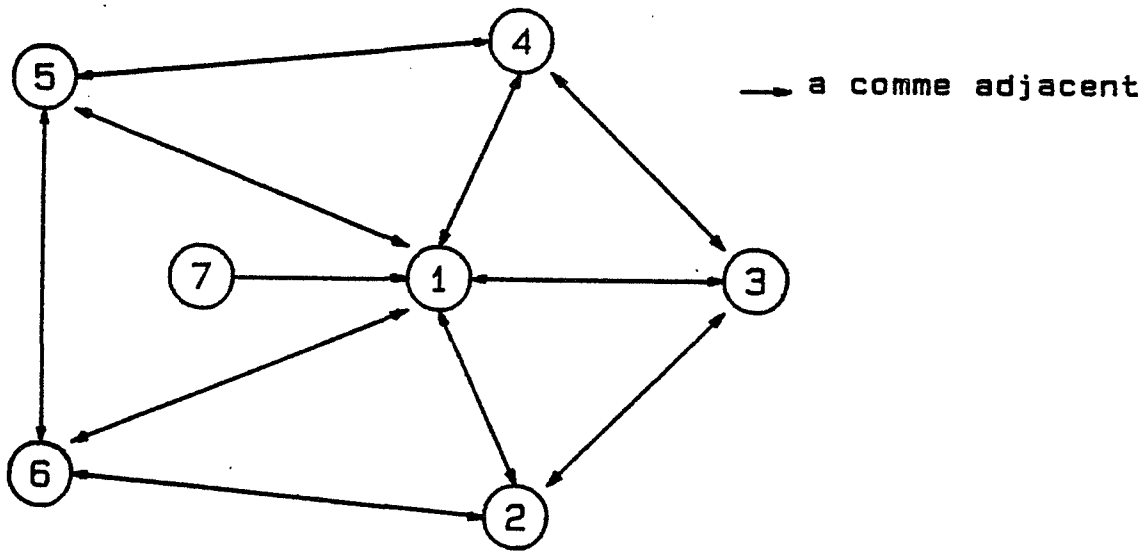


Figure 3-4 : Graphe d'adjacence.

La troisième condition rend la relation non symétrique. On remarque, par exemple, que 1 est adjacent à 7 alors que la réciproque est fausse. Cette définition des adjacences n'est pas suffisante, nous allons définir une nouvelle relation caractérisant la notion de superposition.

4.4. Relation de superposition.

Les superpositions entre deux rectangles vides ou entre un rectangle vide et un rectangle englobant sont interdites par définition, par contre, les superpositions entre deux ou plusieurs rectangles englobants sont licites. Deux rectangles englobants sont superposés si leur intersection est non vide. Ces superpositions sont gérées par la relation de superposition:

Ri superposé à Rj ssi
$Ri \cap Rj \neq \emptyset$

Cette relation est symétrique. La description de la relation de superposition correspondant à l'exemple de la figure 3-3 est présentée figure 3-5.

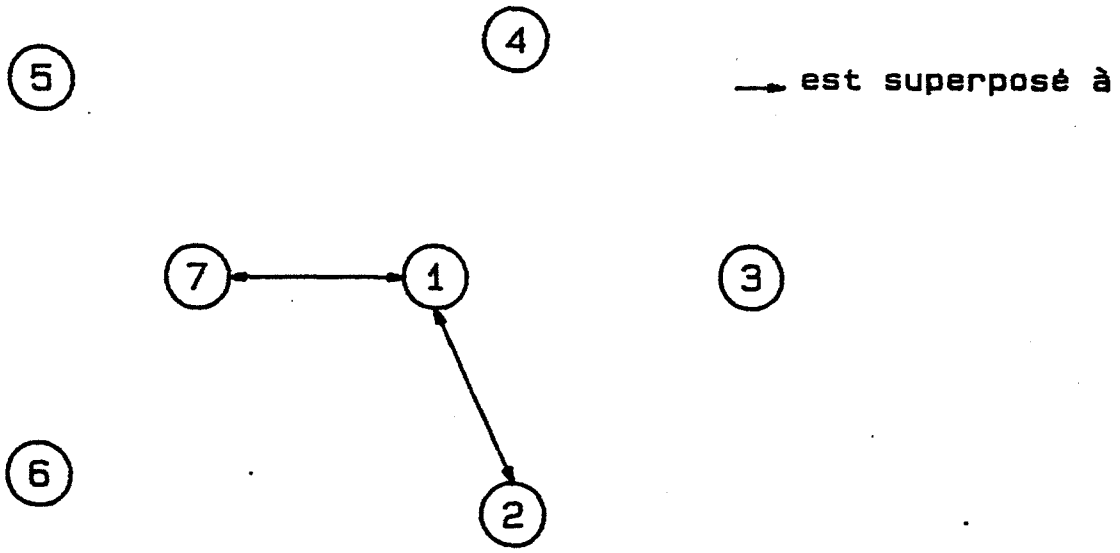


Figure 3-5 : Graphe de superposition.

4.5. Structure de données.

La relation d'adjacence est représentée en associant à chacun des 4 côtés d'un rectangle, une liste des pointeurs vers les rectangles adjacents à ce côté.

La relation de superposition est matérialisée par la présence pour chaque rectangle d'une liste des pointeurs vers les rectangles superposés.

Comme indiqué dans la description Pascal ci-dessous, la définition d'un rectangle suppose en plus l'existence d'un pointeur vers la description de l'objet associé.

Description Pascal de la structure de données

```
type
   pt_rectangle          = ^t_rectangle;
   pt_l_adjacence        = ^t_l_adjacence;
   pt_l_superposition    = ^t_l_superposition;

   t_rectangle = record
       l_superposition   : pt_l_superposition;
                           {pointeur vers la liste des
                            rectangles superposés}
       adjacence         : array[xmin,ymin,xmax,ymax] of
                                          t_adjacence;
       description_objet : pt_des_objet;
                           {pointeur vers la description
                            de l'objet (non décrit ici)
                            nil si le rectangle est vide}
                 end;

   t_adjacence = record
       valeur       : real;
                      {valeur de xmin, ymin, xmax, ymax}
       l_adjacence : pt_l_adjacence;
                      {pointeur vers une liste des rectangles
                       adjacents}
                 end;

   t_l_adjacence = record
       max                 : real;
                             {indice (x ou y) maximum de
                              l'intervalle d'adjacence avec le
                              rectangle pointé par pt_rectangle}
       rectangle_adjacent : pt_rectangle;
                             {pointeur vers le rectangle
                              adjacent}
       suivant             : pt_l_adjacence;
                  end;

   t_l_superposition = record
       rectangle_superpose : pt_rectangle;
                             {pointeur vers le rectangle
                              superposé}
       suivant              : pt_l_superposition;
                     end;
```

Afin de lever les ambiguités qui pourraient subsister, voici une description détaillée de la structure pour le rectangle 1 de l'exemple de la figure 3-3.

Description du rectangle 1

```
rectangle1^.l_superposition                = l_superposition1
rectangle1^.adjacence[xmin].valeur         = x2
rectangle1^.adjacence[xmin].l_adjacence    = l_adjacence1
rectangle1^.adjacence[ymin].valeur         = y2
rectangle1^.adjacence[ymin].l_adjacence    = l_adjacence2
rectangle1^.adjacence[xmax].valeur         = x4
rectangle1^.adjacence[xmax].l_adjacence    = l_adjacence3
rectangle1^.adjacence[ymax].valeur         = y4
rectangle1^.adjacence[ymax].l_adjacence    = l_adjacence4

l_superposition1^.rectangle_superpose      = rectangle2
l_superposition1^.suivant                  = l_superposition2
l_superposition2^.rectangle_superpose      = rectangle7
l_superposition2^.suivant                  = nil

l_adjacence1.max                           = y4
l_adjacence1.rectangle_adjacent            = rectangle5
l_adjacence1.suivant                       = nil
l_adjacence2.max                           = x3
l_adjacence2.rectangle_adjacent            = rectangle6
l_adjacence2.suivant                       = l_adjacence5
l_adjacence3.max                           = y3
l_adjacence3.rectangle_adjacent            = rectangle2
l_adjacence3.suivant                       = l_adjacence6
l_adjacence4.max                           = x4
l_adjacence4.rectangle_adjacent            = rectangle4
l_adjacence4.suivant                       = nil
l_adjacence5.max                           = x2
l_adjacence5.rectangle_adjacent            = rectangle2
l_adjacence5.suivant                       = nil
l_adjacence6.max                           = y4
l_adjacence6.rectangle_adjacent            = rectangle3
l_adjacence6.suivant                       = nil
```

On présentera au paragraphe 6 la méthode utilisée pour construire cette structure. Nous allons maintenant supposer que nous disposons d'une telle structure et étudier l'algorithme permettant de l'exploiter.

5. ALGORITHME

Nous allons étudier comment exploiter la structure de données ainsi définie pour diminuer le nombre de tests d'intersection.

Dans la description de l'algorithme, nous n'étudierons pas les calculs d'intersection entre un rayon et un objet élémentaire. De nombreuses études ont déjà été développées sur ce sujet. Une bibliographie est fournie au premier chapitre présentant l'état de l'art sur le tracé de rayon. Nous nous sommes contenté d'implémenter quelques unes de ces techniques de façon à pouvoir

réaliser les tests. Pour le moment les objets élémentaires connus par le logiciel sont la sphère et le prisme. Nous ne reviendrons pas sur le balayage de l'image et l'initialisation des rayons primaires ou secondaires, ces thèmes ont été développés au chapitre 2 présentant le tracé de rayon pour la synthèse d'images de terrains.

Nous détaillons donc la partie de l'algorithme qui étant donnés un rayon, défini par une origine et un vecteur directeur, et la structure de données précédente permet de déterminer le premier point de la base de données intersecté par le rayon.

5.1. Espace contenant l'origine du rayon

Une première étape consiste à caractériser le rectangle où se trouve l'origine du rayon en projection. Pour les rayons secondaires, l'origine du rayon est généralement le point d'intersection déterminé lors de l'étude du rayon primaire, dans ce cas, le rectangle associé est déduit de l'étude du rayon primaire. Pour les rayons primaires l'origine du rayon est constituée par l'oeil il est donc nécessaire, avant de commencer l'étude de déterminer quel est le rectangle contenant la projection de l'oeil.

Cette opération n'étant exécutée qu'une seule fois par image, nous avons choisi un algorithme simple qui étant donné un rectangle quelconque de la structure, se déplace de ce rectangle vers l'origine du rayon. Le déplacement s'effectue en 2 temps: un déplacement suivant l'axe des x, jusqu'à ce que l'abscisse du point soit atteinte puis un déplacement à x constant suivant les y permettant d'atteindre le point cible.

5.2. Structure générale de l'algorithme

Nous supposons connus

- le rectangle contenant l'origine du rayon
- les paramètres géométriques définissant le rayon

L'algorithme recherchant l'intersection du rayon avec les objets de la scène va exécuter un cheminement dans la structure décrite précédemment jusqu'à ce qu'une intersection entre le rayon et un objet soit détectée. Ce module va être constitué de deux actions principales:

* Le déplacement dans la structure: limité à une succession d'appels à une procédure permettant de passer d'un rectangle intersecté par le rayon au rectangle suivant le long du rayon. Procédure rectangle_suivant.

* le test d'intersection du rayon avec un rectangle, c'est à dire avec les objets pouvant être situés dans ce rectangle. Procédure inter_rectangle.

Algorithme 3-1 : Structure générale de l'algorithme d'intersection d'un rayon avec les objets de la scène

```
VAR
   rectangle : pt_rectangle;
   premier   : boolean;
   résultat  : RECORD
                  rec_inter     : pt_rectangle;
                  intersection  : boolean;
                  info          : {informations relatives
                                   au point d'intersection}
               END;

BEGIN
   {localisation du rectangle contenant l'origine du rayon}
   localiser_rectangle(origine,rectangle);
   premier := true;
   résultat.intersection := false;
   WHILE not résultat.intersection and rectangle ≠ nil DO
   BEGIN
      inter_rectangle(premier,rectangle,résultat);
      premier := false;
      IF not résultat.intersection THEN
         rectangle_suivant(rectangle);
   END;
   IF résultat.intersection and rectangle≠résultat.rec_inter THEN
      inter_rectangle(premier,résultat.rec_inter,résultat);
END
```

Rectangle représente le rectangle étudié, il est modifié par rectangle_suivant.

Comme on le verra plus précisément lors de la description de inter_rectangle, s'il ne s'agit pas du premier appel de la procédure inter_rectangle, un traitement spécial permet de ne pas réétudier les rectangles déjà considérés précédemment.

Lorsqu'une intersection est détectée, il est nécessaire de rappeler inter_rectangle afin de s'assurer que l'intersection est bien la plus proche de l'oeil. En effet, inter_rectangle recherche tous les points d'intersection entre le rayon et les objets associés aux rectangles superposés à rectangle et fournit au programme appelant le point le plus proche de l'oeil. Il se peut donc que ce point, i1, ne soit pas le plus proche de l'oeil, (cf.figure 3-6). Après détection d'un point d'intersection, inter_rectangle est donc rappelé avec le rectangle associé à l'objet qui intersecte le rayon.

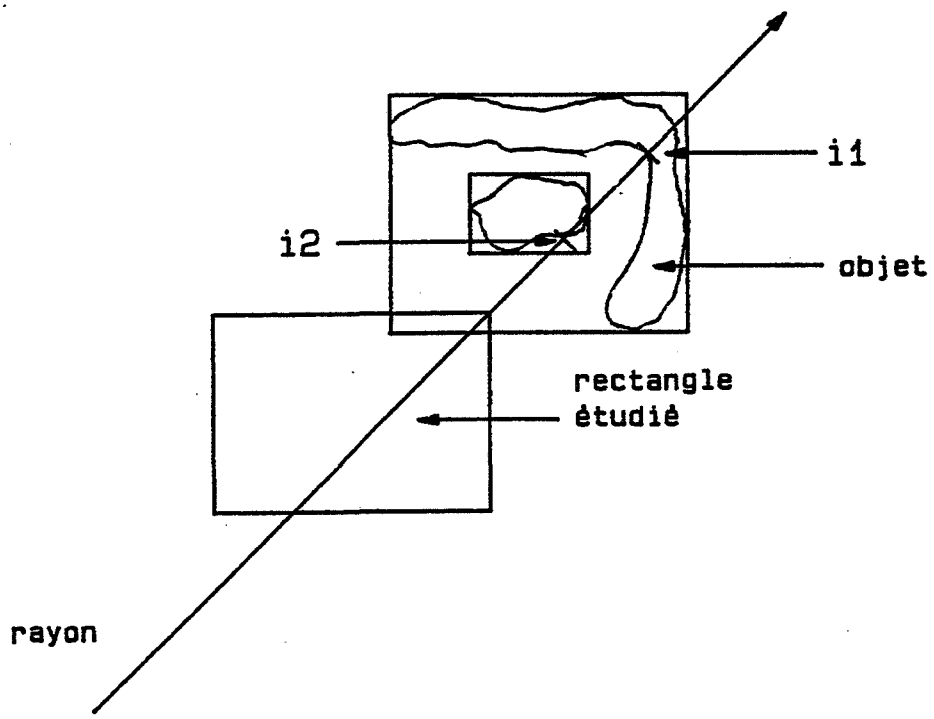


Figure 3-6 : Recherche du premier point d'intersection.

La figure 3-6 montre sur un exemple qu'un point i2, plus proche de l'oeil que i1 peut ne pas appartenir à l'un des rectangles superposés au rectangle étudié. Ce dernier appel à inter_rectangle est évidemment inutile dans le cas de rayons secondaires destinés à connaître la contribution d'une source lumineuse car la seule information recherchée est: le rayon intersecte-t'il oui ou non l'objet?.

5.3. Cheminement dans la scène.

Le cheminement se fait par des appels successifs à la procédure rectangle_suivant. Cette procédure permet, à partir d'un rectangle, de déterminer le rectangle suivant le long du rayon. On suppose que le rectangle courant est traversé par le rayon, soit I le point de sortie: deuxième point d'intersection du rayon avec la frontière du rectangle; on veut déterminer le rectangle adjacent au rectangle courant au point I.

- rayon représente la projection du vecteur directeur du rayon.
- origine contient les coordonnées en projection du point origine du rayon; il s'agira en général de l'oeil pour les rayons primaires et du point d'intersection pour les rayons secondaires.

. indx est initialisé à xmax si rayon[x]>0 et à xmin si rayon[x]≤0.

. indy est initialisé à ymax si rayon[y]>0 et à ymin si rayon[y]≤0.

(xmax,xmin,ymax,ymin sont les valeurs de type scalaire énuméré indiçant le tableau des adjacences).

Cette procédure se décompose en deux parties: la recherche du rectangle adjacent le long de l'une des deux frontières x=constante puis en cas d'échec, une recherche le long de l'une des deux frontières y=constante. En théorie, si on suppose que les conditions initiales sont satisfaites, c'est à dire que le rayon traverse le rectangle et que la structure de données abritant les adjacences entre rectangles est cohérente, la procédure décrite précédemment doit permettre de déterminer le rectangle adjacent.

Algorithme 3-2 : Détermination du rectangle suivant.

```
PROCEDURE rectangle_suivant (var rectangle:pt_rectangle);
VAR
   ptadj   : pt_l_adjacence;
   borne_y : real;
   borne_x : real;
   fini    : boolean;
BEGIN
   IF rayon[x] = 0 and rayon[y] = 0 THEN
      rectangle := nil
   ELSE
   BEGIN
      IF rayon[x] ≠ 0 THEN
         borne_y := origine[y]+(rectangle^.adjacence[indx].valeur
                    - origine[x])*rayon[y]/rayon[x];
      IF (borne_y ≤ rectangle^.adjacence[ymax].valeur) and
         (borne_y ≥ rectangle^.adjacence[ymin].valeur) and
         (rayon[x] ≠ 0) THEN
      BEGIN
         ptadj := rectangle^.adjacence[indx].l_adjacence;
         fini := false;
         WHILE (ptadj ≠ nil) and (not fini) DO
            IF ptadj^.max ≥ borne_y THEN
            BEGIN
               rectangle := ptadj^.rectangle-adjacent;
               fini := true
            END
            ELSE
               ptadj := ptadj^.suivant;
         IF ptadj = nil THEN
               rectangle := nil;
      END
      ELSE
      BEGIN
         borne_x := origine[x]+(rectangle^.adjacence[indy].valeur-
            origine[y]) * rayon[x]/rayon[y];
             { rayon[y] ne peut être nil }
         ptadj := rectangle^.adjacence[indy].l_adjacence;
         fini := false;
         WHILE (ptadj ≠ nil) and (not fini) DO
            IF ptadj^.max ≥ borne_x THEN
            BEGIN
               rectangle := ptadj^.rectangle_adjacent;
               fini := true
            END
            ELSE
               ptadj := ptadj^.suivant;
         IF ptadj = nil THEN
            rectangle := nil;
      END;
   END;
END;
```

En fait, la précision des calculs peut poser des problèmes. Les erreurs sur les calculs induites par les arrondis peuvent provoquer des incohérences. Par exemple, pour un rayon intersectant le rectangle près d'un de ses sommets, il se peut qu'aucune intersection ne soit détectée. Nous avons par conséquent dû rajouter une tolérance aux tests pour régler les cas ambigus.

5.4. Intersection entre le rayon et un rectangle

Cette procédure recherche le premier point d'intersection entre un rayon et les objets liés à un rectangle et ceux de ses rectangles superposés. Afin de ne pas faire de tests d'intersection inutiles entre le rayon et des objets, un traitement spécial permet de savoir si le rectangle a déjà été étudié. Pour cela, nous ne considérerons que les rectangles superposés dont la boîte englobante intersecte le rayon et ne contient pas le point d'entrée du rayon dans le rectangle initial, (cf. figure 3-6). Cette deuxième condition permet d'assurer que le rectangle considéré n'a pas encore été étudié.

Algorithme 3-3 : Intersection entre un rayon et un rectangle.

```
PROCEDURE inter_rectangle(premier:booleen;rectangle:pt_rectangle;
                          résultat:t_résultat);

VAR
   superpose : pt_l_superposition;

BEGIN
   IF rectangle^.decrit_objet ≠ nil THEN
   BEGIN
      (* le rectangle n'est pas vide *)
      intersect(rectangle,résultat);
      superpose := rectangle^.l_superposition;
      WHILE ((superpose ≠ nil) and (not rayon_d'ombre or
            not résultat.intersection)) DO
      BEGIN
         IF (inter_rect(superpose^.rectangle_superpose) and
            (not entree(rectangle,superpose^.rectangle_superpose)
             or premier)) THEN
            intersect(superpose^.rectangle_superpose,résultat);
         superpose := superpose^.suivant;
      END
   END
END.
```

inter_rect(rectangle) est une fonction dont la valeur est un booléen à vrai si rectangle est intersecté par le rayon et à faux sinon.

entree(rectangle1,rectangle2) est une fonction dont la valeur est un booléen à vrai si rectangle2 contient le point d'entrée du rayon dans rectangle1.

intersecte(rectangle,resultat) est une procédure qui choisit

puis exécute un algorithme d'intersection adapté au type d'objet à étudier. S'il y a intersection et que le rayon est un rayon primaire alors, toutes les informations relatives au point d'intersection sont fournies dans résultat: rectangle, coordonnées, normale, couleur et éventuellement, caractéristiques du matériau de l'objet au point d'intersection.

Un test sur le z du rayon par rapport au ZMIN et au ZMAX de la scène permet d'éviter certains tests inutiles.

6. INITIALISATION DE LA STRUCTURE DE DONNEES

Nous disposons de la projection des boîtes englobantes des objets sur le plan x-y, parallèlement à l'axe des z. Soit donc un ensemble de rectangles de côtés parallèles aux axes et éventuellement superposés (cf figure 3-3). Nous allons décrire le choix d'une configuration de rectangles vides qui recouvre le plan. La disposition des rectangles vides n'a pas d'influence sur le nombre de tests d'intersection entre les rayons et les objets. Par contre, elle intervient pour les déplacements dans la structure. L'objectif est donc d'essayer de minimiser le nombre moyen de rectangles vides traversés par un rayon.

La solution que nous avons développée, sans prétendre à l'optimalité, a permis de construire toutes les bases de données utilisées pour les tests présentés au paragraphe suivant. Ce programme se décompose en deux parties principales:

- le choix d'une configuration de rectangles vides
- la mise à jour de la structure et des adjacences.

* Choix d'une configuration de rectangles vides

Le choix d'une configuration de rectangles vides peut se ramener à un problème de convexification dans un cas particulier où toutes les arêtes sont parallèles aux axes. Examinons la méthode que nous avons utilisée pour choisir les arêtes à ajouter.

Les arêtes ajoutées sont toutes issues d'un sommet d'un rectangle englobant. La figure 3-7 montre, pour chacun des quatre sommets d'une boîte, les deux arêtes qu'il est possible d'ajouter.

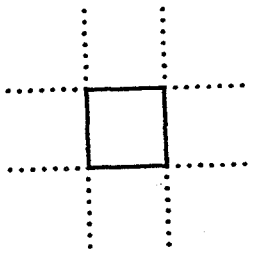


Figure 3-7 : Arêtes qu'il est possible d'ajouter

Deux paramètres interviennent dans le choix d'une arête: son orientation et sa longueur. Guidé par l'objectif précédent, nous avons essayé de minimiser la somme des longueurs des arêtes. Pour ce faire, et avec comme second objectif de mettre au point une solution robuste permettant de réaliser facilement quelques tests, nous avons choisi de sélectionner les nouvelles arêtes en prenant pour chaque sommet, l'arête de longueur minimum. Pour une arête donnée issue d'un sommet donné, sa longueur est égale à la distance séparant le sommet de la première paroi rencontrée dans la direction de l'arête. La figure 3-8 montre sur un exemple que le sommet en haut à gauche de la boite A peut donner naissance à deux arêtes 1 et 2, 1 étant la plus courte, elle sera choisie. Cette solution ne permet pas de garantir une borne minimale de la somme des tailles des arêtes. La recherche d'une telle borne est un problème combinatoire que nous n'avons pas abordé.

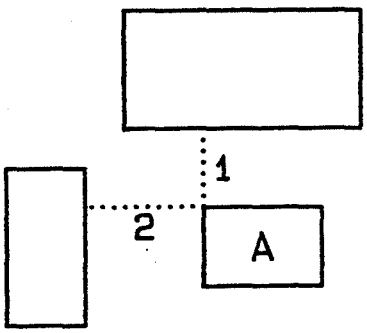


Figure 3-8 : Choix des arêtes à ajouter

L'algorithme réalisant le choix des arêtes est donc très simple. Disposant de deux listes d'arêtes: une liste pour chaque orientation, chaque sommet est étudié et l'arête résultante rajoutée à l'une des listes. La taille de l'arête étant déterminée en recherchant l'arête perpendiculaire la plus proche.

*** Construction de la structure et des adjacences**

Nous avons dans la phase précédente sélectionné les arêtes à ajouter. Il faut maintenant

. construire les rectangles vides définis par ces arêtes

. initialiser le graphe d'adjacence. Pour ce faire, nous avons utilisé un logiciel de calcul d'une partition du plan développé par M. Gangnet et D. Michelucci [GAN84]. Ce logiciel se présente comme un outil graphique de description d'une partition du plan. Il accepte en entrée un ensemble d'arêtes, calcule les points d'intersection de celles-ci et fournit en sortie une représentation des zones du plan délimitées par ces arêtes. L'information disponible est essentiellement d'ordre géométrique et topologique. Les entitées connues sont:

les sommets (sommets des arêtes ou points d'intersection)
les arêtes
les bords qui représentent les zones délimitées par un contour formé d'arêtes

De nombreuses relations entre ces entitées sont définies:
arête → sommets extrémités de l'arête
sommet → liste des arêtes ayant pour extrémité ce sommet
bord → liste des arêtes frontières du bord
arête → les deux bords ayant pour frontière l'arête (il se peut qu'il n'y en ait qu'un)

On trouvera donc tous les renseignements nécessaires à la définition des rectangles vides et à la construction du graphe d'adjacence.

Nous allons fournir au logiciel l'ensemble des arêtes définies précédemment. Un balayage de la structure fournie par ce logiciel permettra d'identifier tous les rectangles et de mettre à jour le graphe d'adjacence. Le graphe de superposition est directement initialisé à partir de la définition des rectangles englobants.

Les figures (3-10,3-12,3-14), présentent les résultats de cette méthode. La construction, telle qu'elle vient d'être exposée n'est qu'expérimentale et un travail important devrait être réalisé pour approfondir cette étude. Deux axes sont dictés par cette préétude:

- la recherche d'un critère d'optimalité satisfaisant l'objectif défini précédemment.

- le développement d'un algorithme permettant de générer une structure satisfaisant le critère précédent.

7. RESULTATS ET EVALUATIONS

Les images présentées dans ce paragraphe ont été réalisées en utilisant l'algorithme décrit précédemment. Les seuls objets élémentaires disponibles dans la version actuelle du programme sont la sphère et le parallélépipède rectangle dont deux faces sont situées dans un plan parallèle au plan x-y. Avant de présenter les résultats, nous allons fournir les temps d'exécution

[1] de la recherche de l'intersection entre un rayon et une sphère

[2] de la recherche de l'intersection entre un rayon et un prisme

[3] du passage d'un rectangle au suivant

Ces temps ont été évalués en isolant les procédures concernées dans des programmes indépendants. Ce sont de petits programmes n'utilisant pas la mémoire virtuelle et exécutés sur une machine peu chargée. Au delà des temps réels qui comme cela a été indiqué dans l'introduction, sont extrêmement dépendants de la machine et de sa charge, ce sont les temps relatifs que nous retiendrons et que nous utiliserons dans les évaluations qui vont suivre.

Les temps pour 100 000 exécutions successives sont les suivants:

[1] avec intersection effective :	1mn 46.7s
[1] sans intersection :	31.1s
[2] avec intersection effective :	1mn 15.2s
[2] sans intersection :	1mn 04.6s
[3]	16.6s

Les temps avec intersection comprennent l'initialisation des paramètres du point d'intersection: coordonnées, normale, couleur, distance. Les temps sans intersection sont valables pour les rayons secondaires ou pour les rayons primaires s'il n'y a pas d'intersection. Les temps [1] et [2] avec intersection peuvent paraître surprenants sachant que l'intersection entre le rayon et

la sphère est la moins coûteuse. Cette différence en faveur du prisme est causée par une évaluation de la normale (normée) qui est incluse pour la sphère alors que pour le parallélépipède, la normale est fournie par les données.

Les résultats de l'algorithme qui vient d'être décrit sont présentés à travers trois exemples:

- une scène constituée de sphères
- une scène constituée de prismes
- une scène constituée de prismes insérés dans les images de terrain décrites au chapitre 2.

Pour tous les exemples présentés dans ce paragraphe, nous avons considéré une seule source lumineuse. Les seuls rayons secondaires pris en compte sont les rayons nécessaires au calcul des ombres portées.

a. Scène de sphères

La scène [S] représentée sur la photo 5 contient 200 sphères générées de façon aléatoire. La seule contrainte imposée par le programme de génération est l'inclusion de toute la scène dans une boîte dont la taille est fixée interactivement. Le programme de génération fournit un point aléatoire à l'intérieur de cette boîte, ensuite, le rayon de la sphère est calculé aléatoirement de façon à ce que la sphère soit entièrement incluse dans la boîte. Ces valeurs sont le résultat de tirages aléatoires suivant une loi uniforme. Ce mode de construction explique la présence de petites sphères sur les bords et de sphères plus grosses au centre de la boîte. La couleur des sphères est, elle aussi, aléatoire. La figure 3-9 présentent l'ensemble des boîtes englobantes des sphères telles qu'elles sont fournies au programme d'initialisation de la structure. Sur la figure 3-10 sont représentées les rectangles vides et les projections des projections des boîtes englobantes des sphères dont l'intérieur a été noirci. Cette scène contient 497 rectangles vides soit 697 rectangles au total.

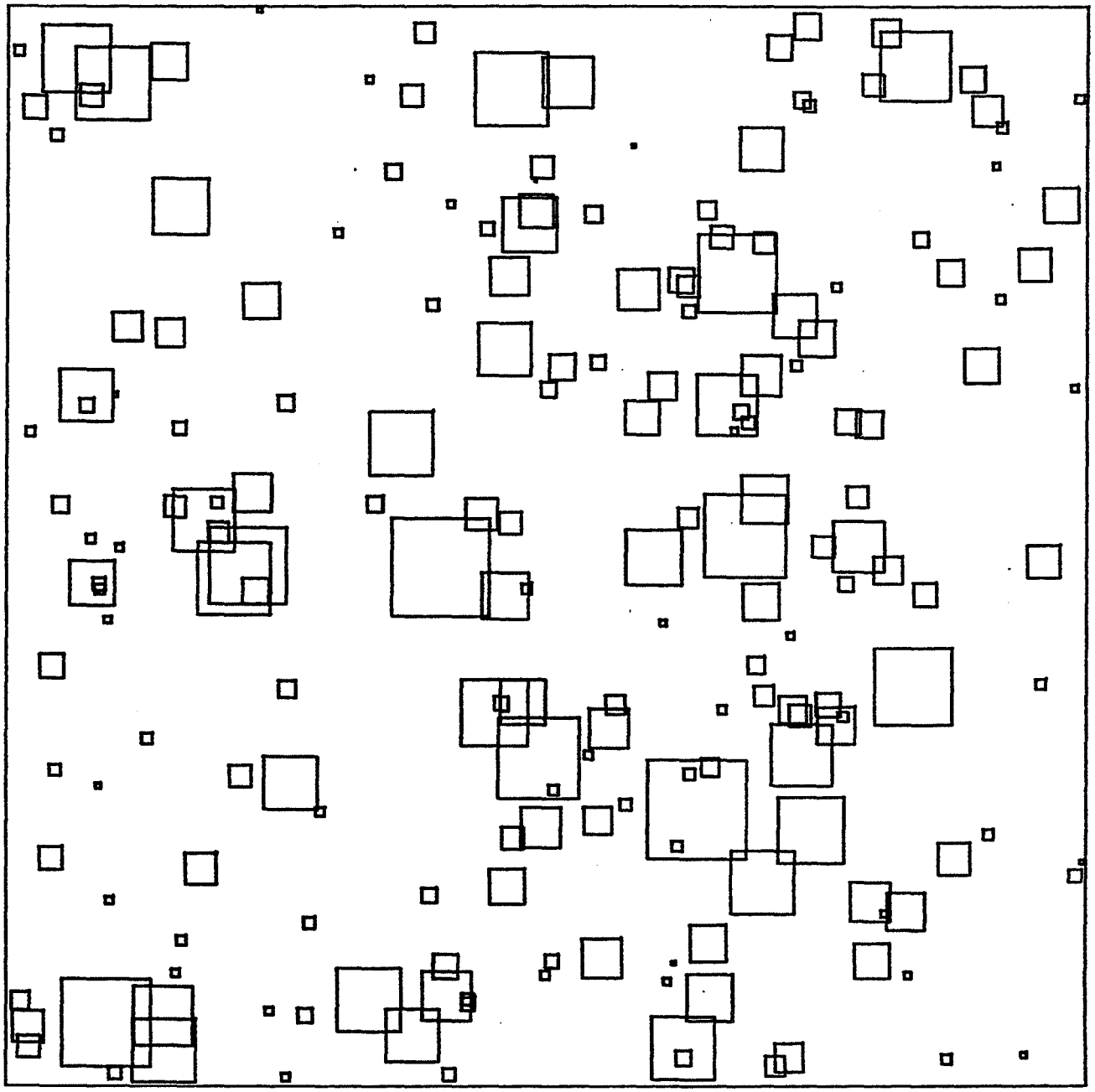


Figure 3-9 : projection des boîtes englobantes des sphères

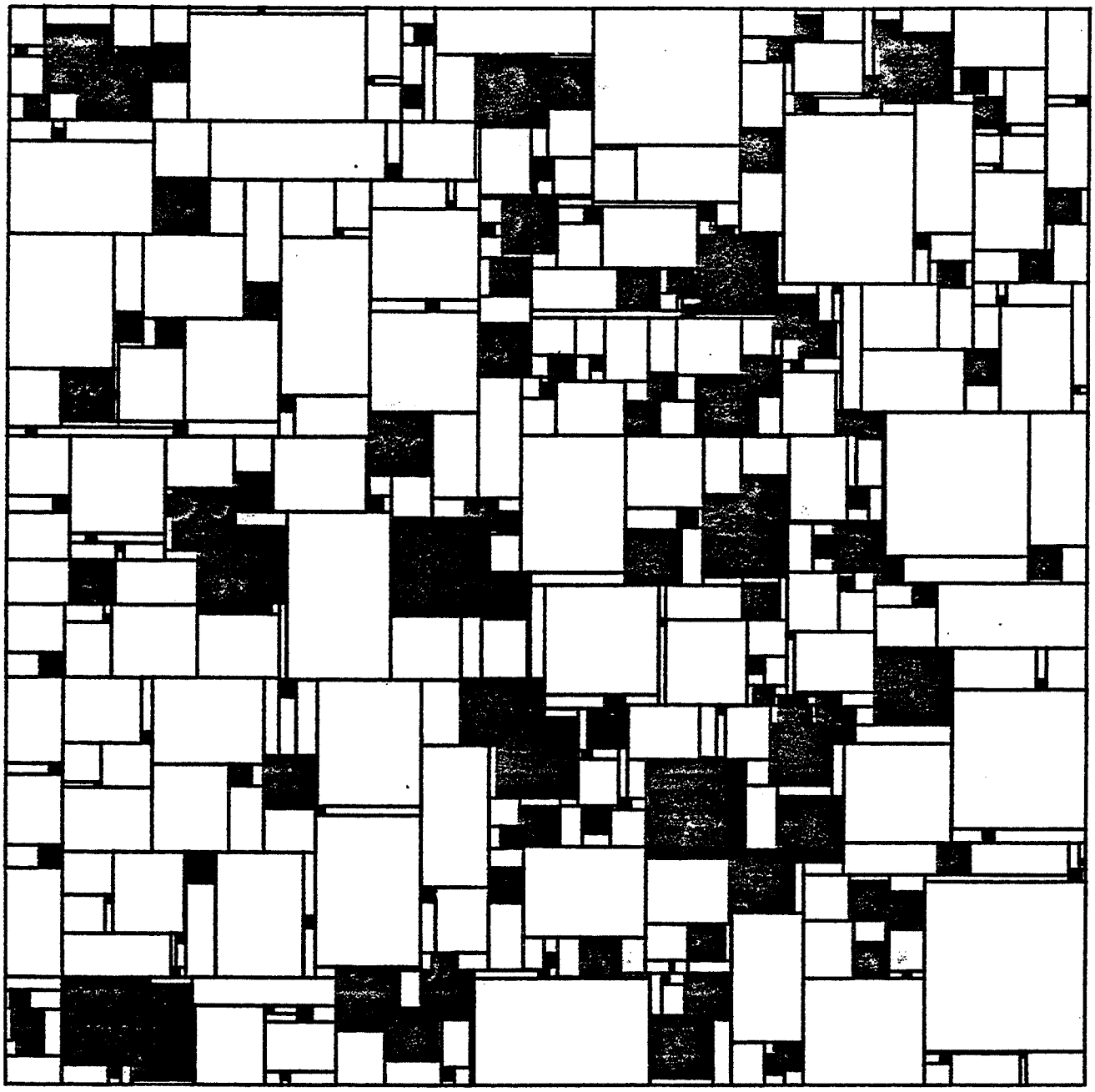


Figure 3-10 : Une configuration de rectangles vides associée à la scène de sphères.

b. Scène de prismes

La scène de prismes [P] visualisée sur la photo 6 contient 225 prismes générés sur une grille régulière. La hauteur et la couleur des prismes sont aléatoires. La figure 3-11 montre l'ensemble des boîtes englobantes des prismes; la boîte englobante du prisme est ici égale à la projection du prisme sur le plan x-y.

La figure 3-12 visualise le découpage généré par le logiciel d'initialisation de la structure de données. Cette scène contient 242 rectangles vides soit 467 rectangles au total.

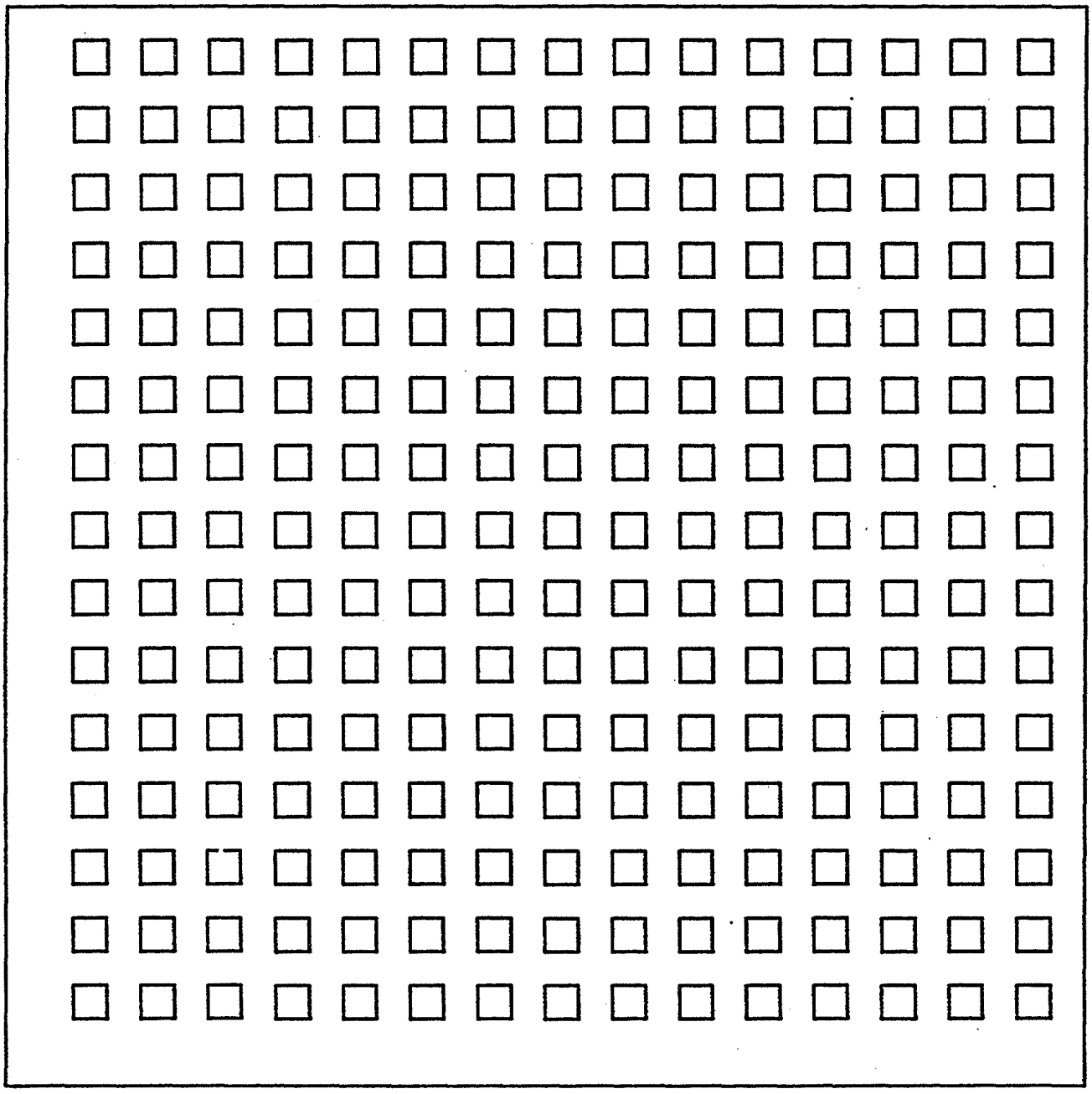


Figure 3-11 : Projection des boîtes englobantes des prismes.

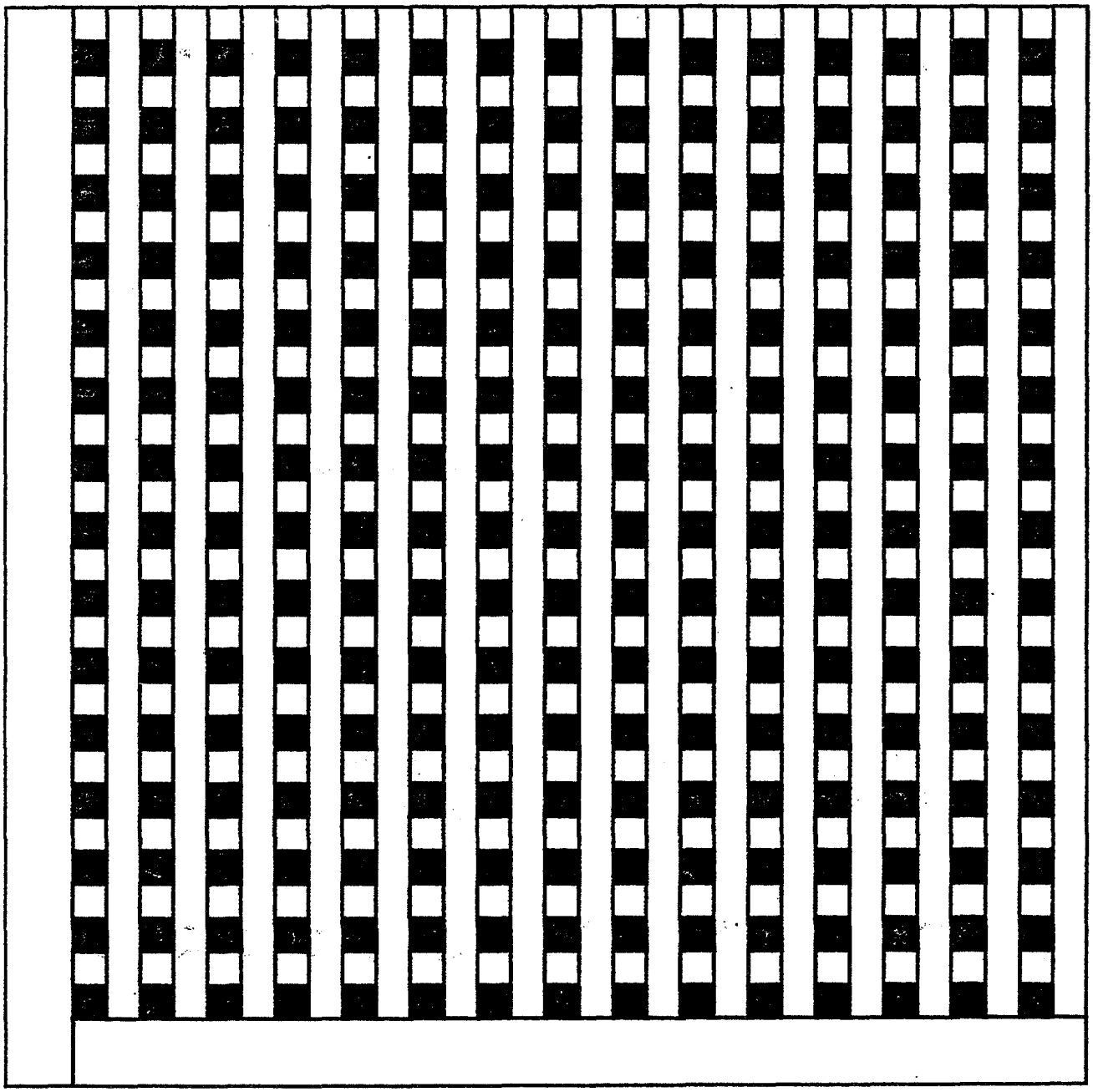


Figure 3-12 : Une configuration de rectangles vides associée à la scène de prismes.

c. Insertion de prismes dans les images de terrain

Cet exemple représente une première approche vers la représentation du sur-sol. Outre les effets qu'il permet de synthétiser, le tracé de rayon a de surcroît l'avantage de

permettre facilement l'élimination des parties cachées entre des types d'objets très divers, à condition qu'ils soient tous traités par le tracé de rayon. Ainsi, l'insertion d'objets dans les scènes représentant des terrains ne pose aucune difficulté. Pour chaque point de l'image, deux rayons sont envoyés, un dans chaque scène. Le point d'intersection retenu est le plus proche de l'oeil. La technique est ensuite réitérée pour les rayons secondaires.

L'image présentée sur la photo 2 a été construite à partir de deux bases de données: la base de données altimétrique utilisée au chapitre 2 et une base de données composée d'une quinzaine de prismes construite pour l'occasion. Cette deuxième base de données, à été construite manuellement c'est à dire qu'aucun outil logiciel n'a été utilisé pour construire et positionner les prismes. La scène de prismes fournie au logiciel d'initialisation de la structure de données, est présentée figure 3-13 alors que le résultat est visualisé figure 3-14. Cette scène contient 15 rectangles.

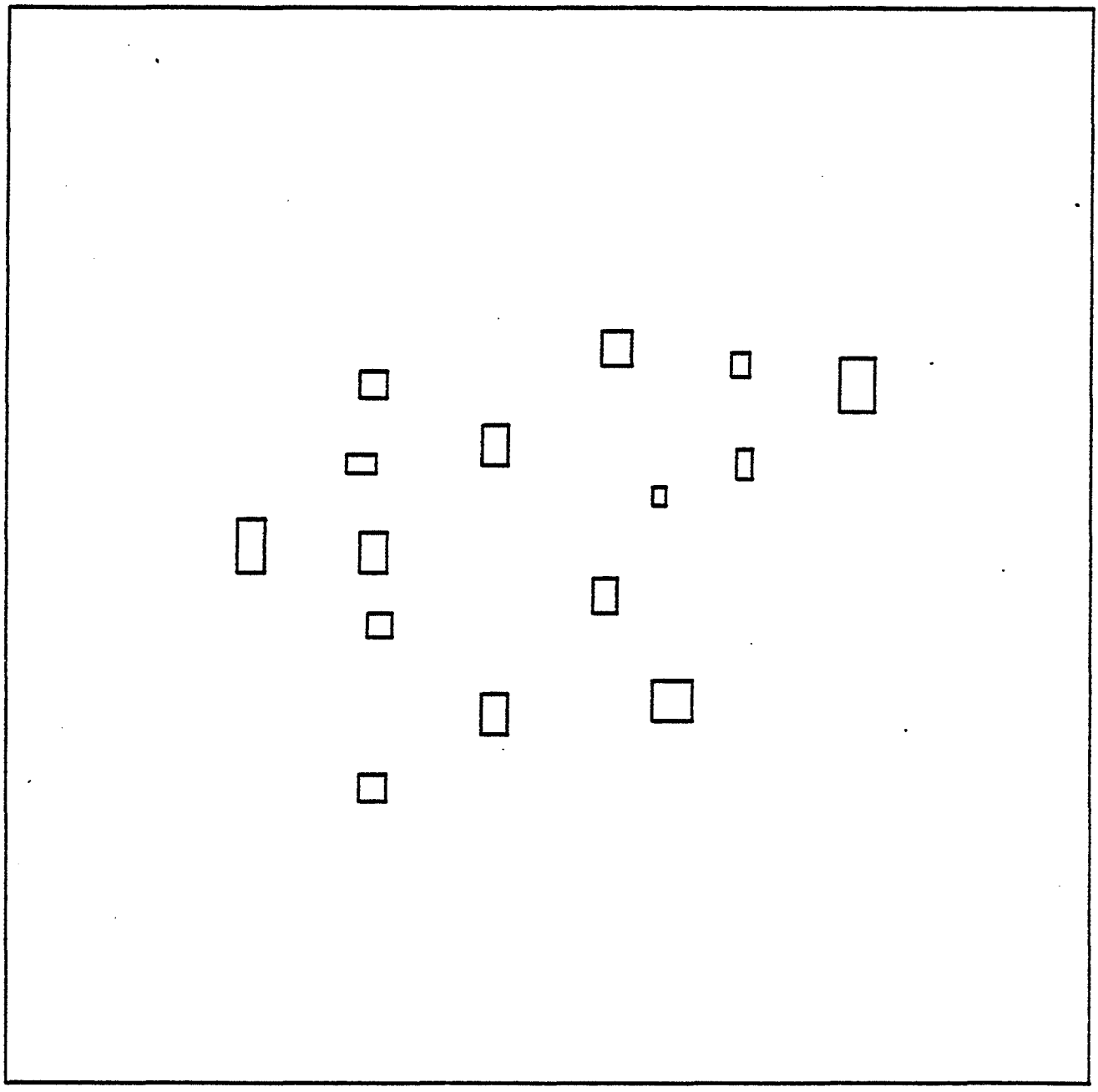


Figure 3-13 : Projection des boîtes englobantes des prismes insérés sur les terrains.

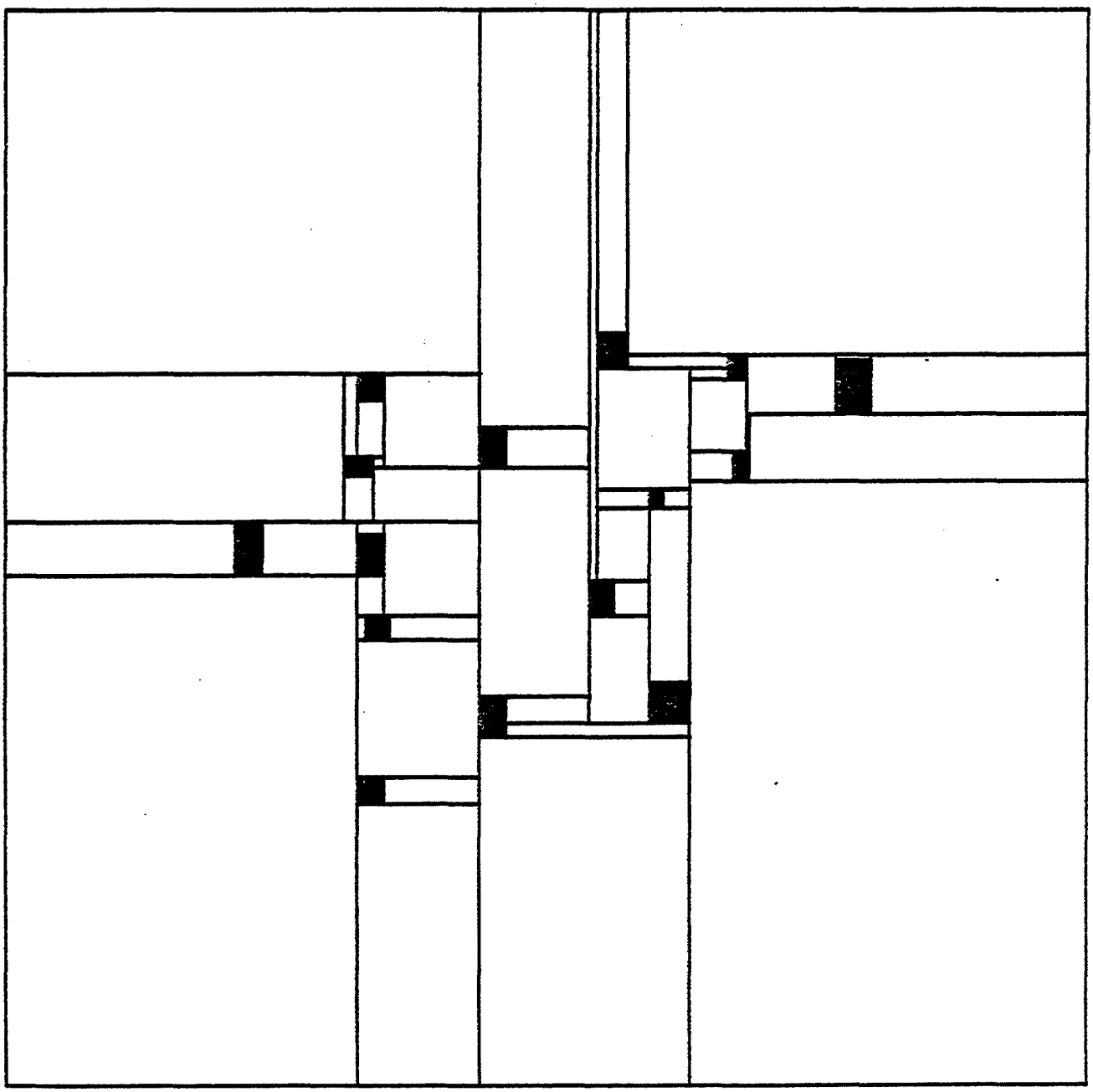


Figure 3-14 : Une configuration de rectangles vides associée à la scène de prismes insérée sur les terrains.

d. Tableau récapitulatif	[S]	[P]	[T]
N :Nombre d'objets de la scène	200	225	15
tn :100 000 tests d'intersection entre un rayon et un objet, sans intersection	31.1s	1mn04.6s	1mn04.6s
ti :100 000 tests d'intersection entre un rayon et un objet, avec intersection	1mn46.7s	1mn15.2s	1mn15.2s
tc :100 000 changements de rectangle	16.6s	16.6s	16.6s
RP :Nombre de rayons primaires envoyés	310 554	261 632	310 554
RS :Nombre de rayons secondaires envoyés	30 362	115 026	166 737
TP :Nombre de tests objet-rayon primaire	642 116	307 445	64 870
TS :Nombre de tests objet-rayon secondaire	114 445	166 990	16 283
IP :Nombre d'intersections objet-rayon primaire	217 207	183 522	16 348
IS :Nombre d'intersections objet-rayon secondaire	5 077	4 997	2 150
NP :Nombre de non intersection objet-rayon primaire	424 909	123 923	48 522
NS :Nombre de non intersection objet-rayon secondaire	109 368	141 993	14 133
CP :Nombre de changements de rectangles d'un rayon primaire	792 529	423 978	604 194
CS :Nombre de changements de rectangles d'un rayon secondaire	90 957	212 540	84 976
TRP :Nombre moyen de tests par rayon primaire = TP/RP	2.06	1.17	0.20
TRS :Nombre moyen de tests par rayon secondaire = TS/RS	3.77	1.45	0.10
TRPC:Nombre moyen de tests par rayon primaire avec la méthode classique = NxRP	200	225	15
TRSC:Nombre moyen de tests par rayon secondaire avec la méthode classique = NxRS	199	224	14
TEP :Temps d'exécution pour les rayons primaires = (NPxtn)+(IPxti)+(CPxtc)	495s	288s	143s
TES :Temps d'exécution pour les rayons secondaires = (NSxtn)+(ISxti)+(CSxtc)	54s	130s	25s
TEPC:Temps d'exécution pour les rayons primaires avec la méthode classique (((NxRP)-IP)xtn)+(IPxti)	19 479s	38 047s	3 011s
TESC:Temps d'exécution pour les rayons secondaires avec la méthode classique (((NxRS)-IS)xtn)+(ISxti)	1 892s	6 718s	1 615s
GEP :Gain sur le nombre moyen de tests par rayon primaire = (TEPC-TEP)100/TEPC	97%	99%	95%
GES :Gain sur le nombre moyen de tests par rayon secondaire = (TESC-TES)100/TESC	97%	99%	98%
GRP :Gain sur les temps d'exécution pour les rayons primaires = (TRPC-TRP)100/TRPC	99%	99%	98%
GRS :Gain sur les temps d'exécution pour les rayons secondaires = (TRSC-TRS)100/TRSC	98%	99%	99%

Les temps d'exécution sont ici des temps calculés en multipliant les nombres d'opérations par les temps moyens d'exécution de ces opérations fournis au début du paragraphe 7. Ces évaluations nous paraissent beaucoup plus fiables et plus faciles à interpréter que les temps d'exécution observés sur la configuration utilisée.

e. Interprétation

On obtient donc des gains d'environ 98%. Ce résultat est valable pour les rayons primaires comme pour les rayons secondaires.

La scène de prismes donne de meilleurs résultats que la scène de sphères principalement pour deux raisons:

- La scène de prismes ne présente pas de superpositions alors que de nombreuses sphères sont superposées. En effet, le nombre moyen de tests d'intersection TRP pour la scène de sphères est 2.06 alors que celui associé à la scène de prismes est 1.175. La densité des superpositions joue un rôle important d'où l'intérêt d'un bon choix du plan de projection. On envisagera au paragraphe suivant d'autres méthodes pour aborder le problème des superpositions.

- Les tests d'intersection entre un rayon et un prisme (sans intersection) sont beaucoup plus longs que les même tests avec des sphères. Ainsi, plus les objets élémentaires sont complexes, plus notre méthode est intéressante.

La scène sur les terrains donne de moins bons résultats en raison de sa faible complexité. La différence entre le nombre de changements de rectangles rajoutés et le nombre de tests d'intersection supprimés est moins importante. Le gain obtenu pour les rayons secondaires est principalement dû à l'orientation du soleil qui est presque situé au zénith. Le nombre de tests d'intersection est donc particulièrement faible avec notre méthode.

8. DEVELOPPEMENTS ET CONCLUSIONS

Avant de présenter les développements envisagés, nous allons présenter une structuration différente des données qui paraît intéressante. La gestion des superpositions va aussi différer.

Nous allons calculer les intersections entre les rectangles englobants puis convexifier les différents éléments résultants. Cette convexification est réalisée en définissant une partition de l'élément par des rectangles parallèles aux axes, (cf. figure 3-15).

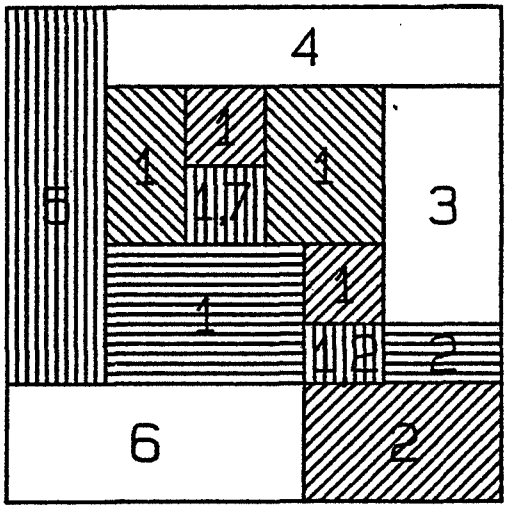


Figure 3-15 : Autre structuration des données

Il est alors possible de définir pour chaque rectangle résultant la liste des objets se projetant sur ce rectangle. Ce sont les objets dont la projection de la boîte englobante recouvre le rectangle.

Cette solution fournit une structure comportant beaucoup plus de rectangles que la première. Le nombre moyen de changements de rectangles pour chaque rayon va donc être sensiblement plus élevé. Cependant, la liste fournie pour chaque rectangle est plus précise: si un rectangle est traversé par le rayon alors, on sait que tous les rectangles associés aux objets de la liste sont aussi intersectés par le rayon. Cette propriété n'est pas vérifiée dans la solution précédente. Ce deuxième critère n'est pas forcémment avantageux car, si la première solution va susciter un nombre important de tests d'intersection entre le rayon et la boîte englobante de l'objet, la deuxième solution nécessite à chaque changement d'espace (ce qui est très courant), de tester si les objets figurant dans la liste de superposition n'ont pas été étudiés auparavant. Nous n'avons pas approfondi cette étude. Seule une évaluation précise des deux solutions permettrait de conclure.

La méthode présentée dans ce chapitre a permis de réaliser un certain nombre de tests et d'évaluations. Cependant, elle est encore assez rudimentaire et devrait être complétée par la suite. Parmi ces compléments, on peut distinguer:

- des extensions de la méthode

- des combinaisons avec d'autres techniques.

* **Extensions de la méthode**

L'aspect bi-dimensionnel de la méthode telle qu'elle vient d'être présentée, ne permet pas, pour certaines bases de données, d'obtenir des gains aussi appréciables que ce que l'on pourrait escompter de la méthode étendue au cas tri-dimensionnel. Cette extension n'est cependant pas immédiate. Deux problèmes sont à résoudre:

- **La gestion des adjacences.**

Alors que dans le cas bi-dimensionnel, la frontière entre deux rectangles est linéaire et par conséquent facile à décrire, dans le cas tri-dimensionnel, cette frontière est planaire. Pour lever d'éventuelles ambiguités, dans le cas tri-dimensionnel, nous ne parlerons plus de rectangle mais d'espace. La structure mémorisant les adjacences doit donc contenir une description complexe de chaque face de chaque espace. Cette description des faces est constituée de différentes zones, et à chaque zone doit être associé l'espace voisin. Malheureusement, la lourdeur de cette structure n'est pas uniquement pénalisante à l'initialisation, elle pénalise aussi les déplacements dans la structure. La recherche de l'espace suivant est par conséquent plus coûteuse.

- **Le découpage en espaces vides.**

Ce découpage est toujours possible, il suffit de se contenter d'un découpage dans une seule dimension. C'est à dire par exemple d'imposer que tous les plans rajoutés soient parallèles au plan x-y. Dans ce cas, le découpage est trivial mais la structure créée est dissymétrique et favorise certaines directions de visée au détriment d'autres. Une solution simple et probablement efficace consiste à utiliser trois structures: une composée de plans parallèles à x,y, la deuxième à y,z, et la troisième à x,z. Dans ce cas, pour

chaque rayon, le choix de la structure exploitée dépend de l'orientation du rayon.

Avant d'envisager une extension tri-dimensionnelle, certaines méthodes intermédiaires peuvent être considérées.

- La cohérence verticale décrite au paragraphe 4 du chapitre 2 sur les terrains peut aussi être exploitée avec la base de données d'objets décrite précédemment. Cette extension est intéressante pour des bases de données particulières sans surplomb. Par exemple, cette méthode conviendrait parfaitement pour la structure de données décrivant les prismes insérés dans les terrains de la photo 2.

- Un tri en z des rectangles superposés éviterait de considérer systématiquement tous les rectangles superposés. Cette solution, éventuellement combinée avec la précédente pour mémoriser le dernier rectangle étudié, semble être un compromis intéressant entre les méthodes bi et tri-dimensionnelles.

* **Combinaison avec d'autres techniques.**
Nous n'avons pas abordé les problèmes de sur et de sous échantillonnage. Le sur-échantillonnage, méthode d'antialiassage proposée par T. Whitted [WHI80] est une des premières extensions à rajouter pour obtenir des images correctes. Le sous-échantillonnage, combiné avec la méthode décrite dans ce chapitre, devrait permettre d'atteindre des temps de calcul acceptables pour des images d'une complexité moyenne.

L'algorithme et la structure de données présentés dans ce chapitre ont permis, sur les trois exemples présentés, d'obtenir des temps de 20 à 100 fois plus faibles qu'avec la méthode traditionnelle. Ce résultat est très prometteur et plusieurs extensions sont envisagées. Ce travail devra donc être poursuivi afin de stabiliser la méthode et d'approfondir les tests d'évaluations sur des scènes plus variées. Outre les performances citées précédemment, cette méthode présente deux atouts majeurs: la possibilité de traiter des scènes variées et d'appliquer la méthode aux rayons primaires comme aux rayons secondaires.

CHAPITRE 4

GENERATION DE TEXTURES

GENERATION DE TEXTURES

1. INTRODUCTION

Les progrès logiciels et matériels de ces dernières années, permettent la création d'images de synthèse de plus en plus réalistes. L'importance donnée à l'aspect et à l'ajout de détails croît parallèlement; un détail est un élément qui n'est pas indispensable à l'interprétation de l'image mais qui la rend plus agréable et plus naturelle. Ainsi, pour les scènes extérieures, la synthèse d'éléments naturels, revêt une grande importance.

La simulation de phénomènes naturels suscite encore de nombreuses recherches. Beaucoup de tels phénomènes n'ont encore jamais été synthétisés de façon satisfaisante. On trouvera dans [CSU83] une définition claire et détaillée des problèmes liés à ce domaine de recherche.

1.1. Synthèse d'images de ciel

Nous nous intéressons, dans ce chapitre, à la synthèse d'images de ciels diurnes; cette étude vient compléter une recherche sur la modélisation d'éléments de végétation [RIB84].

En général, le ciel est représenté soit par une surface unie, soit par un dégradé de bleus s'éclaircissant et se désaturant vers l'horizon [MAX81]. L'aspect lisse et uniforme caractérisant ces images nuit au réalisme du résultat. Notre objectif est donc de représenter des ciels moins monotones.

Des recherches ont déjà été développées pour mettre au point des modèles plus réalistes [SCH83]. On peut distinguer principalement deux approches:

*** la modélisation tri-dimensionnelle.**

L'emploi d'un modèle tri-dimensionnel est justifié par un positionnement de l'observateur dans les nuages. C'est à dire pour toutes les vues où l'on ne peut considérer le ciel et les nuages comme une toile de fond. Il en est ainsi pour les vues aériennes simulant des déplacements dans l'atmosphère et dans les nuages.

Dans ce cas, on différencie les nuages de l'atmosphère dans laquelle ils sont situés. Les nuages sont décrits comme des volumes dans l'espace R^3. Lors de la visualisation, les nuages

sont affichés en utilisant un algorithme d'élimination des parties cachées compatible avec le modèle les décrivant. La profondeur de l'atmosphère est représentée par un fond bleuté.

Les réalisations dans ce domaine sont très rares. On peut cependant citer les travaux de B. Fishman et B. Schachter [FIS80] qui simulent des cumulus par des ellipsoïdes. Par ailleurs, W.T. Reeves [REE83] propose la modélisation de certains phénomènes naturels par des systèmes de particules; les images de feux et d'herbes qu'il présente sont très prometteuses. L'auteur indique qu'il envisage de modéliser des nuages de cette façon.

*** la modélisation bi-dimensionnelle.**

Une représentation bi-dimensionnelle est suffisante dans de nombreuses applications: images fixes où le ciel est situé au dernier plan. On peut envisager deux formes de modélisation bi-dimensionnelle: la représentation des nuages ou celle d'un ciel nuageux.

- Dans le premier cas, les nuages sont modélisés indépendamment. La seule différence avec la modélisation tri-dimensionnelle réside dans la dimension du modèle utilisé pour simuler les nuages.

  Nous avons rapidement abandonné cette première solution à cause de la structure même des nuages; la masse nuageuse est mal délimitée et sa transparence est variable.

- Dans le deuxième cas, on considère un ciel nuageux comme un tout représenté sous forme d'une face texturée qui est affichée en toile de fond de l'image. Cette solution est celle que nous avons choisi et que nous allons maintenant développer.

1.2. Synthèse de textures

L'importance des textures pour l'interprétation des images, notamment pour la profondeur et l'orientation des objets explique l'abondance de la littérature sur le sujet [HAR79]. Si la majorité de ces travaux sont relatifs à l'analyse d'images, certains résultats ont été repris et appliqués à la synthèse. D'autre part, certains modèles de textures ont été étudiés spécialement pour la synthèse. Parmi les méthodes utilisées pour générer des textures, on peut citer:

*** la numérisation de textures existantes.**

Si on dispose de la texture à représenter, l'idée la plus simple consiste à la numériser avec une caméra. Cette technique est en particulier bien adaptée aux problèmes

d'insertion d'objets dans un contexte existant; par exemple: insertion de bâtiments dans un paysage. Ce procédé peut être utilisé avantageusement pour des textures déterministes, par contre, les textures aléatoires sont plus difficiles à obtenir par cette méthode. Deux problèmes sont liés à cette technique:

- Il n'est pas toujours possible de numériser une surface de texture assez grande, on se contente alors de numériser une vignette de la texture choisie.

- Une texture est généralement numérisée de face, il est alors nécessaire d'appliquer une procédure de mise en perspective pour la plaquer sur une face plane dont l'orientation est quelconque.

Des essais ont été réalisés par W.Dungan & al. avec des photos d'herbes et de forêts [DUN78].

*** la méthode structurale.**

L'algorithme de synthèse utilise un modèle définissant la structure de la texture à générer. Cette structure est fournie sous la forme de primitives et des règles de positionnement qui leurs sont associées. Cette définition peut par exemple être décrite par une grammaire des formes. La méthode structurale s'applique particulièrement bien à la description de textures ayant une organisation très régulière: murs de briques, papiers peints, mosaïques. Elle peut cependant être complétée par l'emploi de fonctions aléatoires intervenant soit au niveau de la structure des motifs élémentaires soit au niveau de leur organisation [LU 79], [ZUC76].

*** la méthode stochastique.**

Cette méthode est mieux adaptée aux textures peu structurées comme les textures d'éléments naturels (herbe, sable...). L'algorithme de représentation est basé sur la définition des paramètres statistiques discriminants de la texture. Tout échantillon synthétisé représente une réalisation du processus aléatoire bi-dimensionnel vérifiant les statistiques définies à partir des échantillons analysés, [MAS82], [GAG83], [SON83].

*** la sommation de plusieurs sinusoïdes.**

Dans ce cas, la texture est définie par l'ensemble des sinusoïdes représentant les fréquences de l'image. Cette méthode, bien adaptée à des textures structurées: mur de briques, donne aussi des résultats satisfaisants pour des textures aléatoires comme la surface de l'eau [NOR82],

[SCH80].

Un échantillon de texture est obtenu par sommation de plusieurs sinusoïdes de fréquences et d'amplitudes variables.

Outre la première solution, citée pour mémoire, on peut regrouper les trois autres qui présentent les points communs suivants:

. elles résultent de recherches en analyse d'image.

. elles sont basées sur une définition de quelques paramètres caractéristiques de l'image.

. ces paramètres sont très souvent déterminés lors d'une phase d'analyse, c'est en particulier vrai pour les deux dernières méthodes.

Les deux méthodes suivantes sont plus spécifiques, elles permettent de produire un effet particulier, éventuellement paramètrable.

*** la perturbation de la normale.**

J.F. Blinn [BLI78] a présenté une idée originale consistant à supprimer l'aspect lisse et uniforme des images de synthèse en appliquant une déformation aléatoire pendant le calcul de la normale à une surface. Cette perturbation est ensuite prise en compte dans le calcul d'éclairement. Cette technique a été reprise par W. Dungan pour l'appliquer à la génération de terrains et de nuages [DUN79].

*** la méthode fractale.**

La notion de fractal, pressentie par J. Perrin dès 1912, est depuis quelques années développée activement par B. Mandelbrot [MAN82]. Les fractals représentent une classe d'objets extrêmement irréguliers, peut-être même d'après B. Mandelbrot, infiniment irréguliers. Les structures fractales se retrouvent dans beaucoup de phénomènes naturels et les recherches sur le sujet ont donné naissance à une théorie permettant de définir des modèles les décrivant. De ces modèles, ont été déduits des algorithmes de synthèse de formes fractales et en particulier de textures fractales.

1.3. Critères de choix

Face au nombre de méthodes disponibles, il est nécessaire de se donner les moyens de faire un choix en fonction des objectifs fixés. Dans ce but, voici un certain nombre de critères de sélection:

- **le temps de calcul et la place mémoire.**

- **la fidélité ou le réalisme.**
  Ces deux buts ne sont pas toujours poursuivis conjointement et peuvent même éventuellement être contradictoires. La recherche de l'identité d'une image de synthèse avec une certaine réalité peut nuire à l'aspect stochastique. Parallèlement, une image peut apparaître très réaliste sans être identifiable à une image existante. En d'autres termes, la fidélité est un critère objectif alors que le réalisme est subjectif.

- **la souplesse du modèle.**
  Ceci caractérise l'aptitude de modèle à représenter un grand nombre de formes variées par simple modification de quelques paramètres.

- **les raccords entre échantillons.**
  Certaines méthodes ne permettent pas toujours de produire une texture de taille suffisante. Dans ce cas, on a généralement recours à un pavage de la face à texturer par duplication de la vignette synthétisée, ce procédé pose des problèmes de périodicité et de raccords entre vignettes.

- **les transformations géométriques.**
  Une texture est rarement utilisée sans appliquer de transformation géométrique (ex: perspective). Il est alors appréciable de disposer d'un algorithme de génération de texture qui puisse intégrer le calcul de la transformation géométrique sans surcoût ou avec un surcoût faible.

- **l'aliassage.**
  L'application de transformations géométriques, entraîne souvent des problèmes d'aliassage dans les zones fortement comprimées. Quelques méthodes, telle que la sommation de plusieurs ondes, permettent de résoudre ce problème simplement. Sinon, on doit avoir recours à un algorithme d'antialiassage à base de filtrages passe-bas.

- **la résolution variable.**
  Il s'agit de ne pas calculer la texture à une résolution plus fine que nécessaire. Les besoins pouvant dépendre des zones de l'image, il est appréciable de pouvoir générer la texture à des résolutions variables suivant les régions.

1.4. Pourquoi l'approche fractale?

Les deux approches les plus courantes: l'approche structurale et l'approche stochastique n'ont pas été considérées ici; la première est mal adaptée car il est difficile d'exhiber une structure propre à la texture que l'on souhaite représenter. Quant à la deuxième, l'usage d'un modèle stochastique nécessite une quantité d'informations et des temps de calcul assez importants; notons que cette méthode est souvent appliquée à des images comportant relativement peu de niveaux de gris. Par ailleurs, B. Mandelbrot [MAN82] montre l'intérêt de l'approche fractale pour la représentation de phénomènes naturels et mentionne son utilisation pour la modélisation de nuages. Cette approche a retenu notre attention pour les raisons suivantes:

- il n'y a pas de problèmes de raccords.
- l'homothétie interne permet de résoudre simplement le problème de la résolution variable en fonction de la distance.
- Nous verrons dans les paragraphes suivants que les temps de calcul sont faibles, et qu'en outre il est possible de générer la texture directement en perspective sans surcoût. Quant à la place mémoire, l'évaluation montre que bien qu'étant importante, elle reste acceptable pour nos applications.
- En ce qui concerne la qualité des résultats, le lecteur, ou le spectateur, est juge.

1.5. Plan du chapitre

Le deuxième paragraphe présente l'algorithme utilisé pour générer un champ z=f(x,y) "fractal". Cet algorithme constitue une extension des travaux de Fournier & al. [FOU82]. Ces champs sont appelés fractals car les recherches de A. Fournier & al. sont basées sur les travaux de B. Mandelbrot sur les fractals. Cependant, ce dernier a émis quelques objections à propos de ce qui est présenté dans cet article. On trouvera quelques échos de cette discussion dans les Communications de l'ACM d'aout 1982 [ACM82]. Afin de ne pas entrer dans la polémique, nous parlerons plutôt de champs aléatoires.

Le paragraphe 3 applique l'algorithme décrit auparavant à la synthèse de textures et plus particulièrement de textures représentant un ciel nuageux. Quelques extensions et résultats sont présentés en fin de paragraphe.

2. CHAMPS ALEATOIRES

L'algorithme présenté dans ce paragraphe permet de générer un champ de valeurs aléatoires. C'est à dire affecter une valeur à chacun des sommets d'une grille. Nous considérerons dans un premier temps que la grille est régulière et que les éléments de grille sont carrés. Ces valeurs peuvent ensuite être interprétées de différentes façons suivant les applications, synthèse de textures ou champs altimétriques.

. Pour la synthèse de textures, ces valeurs sont exploitées pour calculer la couleur du point correspondant, une interpolation entre les points de la grille fournit le résultat.

. Pour les champs altimétriques, la valeur permet d'évaluer l'altitude du point considéré.

2.1. Présentation du problème

Ce paragraphe est consacré à la génération d'un champ aléatoire $z=f(x,y)$, $x,y \in [0,d]$ utilisant un modèle stochastique récursif.

- Récursivité:
  La méthode consiste à décomposer récursivement chaque région en quatre sous-régions et à évaluer les valeurs en chacun des sommets générés (cf.figure 4-1).

- Aspect stochastique:
  La fonction d'évaluation des valeurs aux sommets est une représentation d'un modèle stochastique prédéfini.

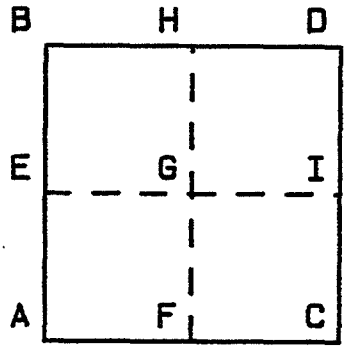


Figure 4-1 : Décomposition de la région A,B,C,D

La décomposition de ABCD en quatre sous-régions AEFG, EBGH, FGCI et GHID nécessite le calcul des valeurs des sommets E, F, G, H et I.

Le procédé d'évaluation des valeurs en E, F, G, H et I est directement déduit de la méthode de subdivision récursive du cas monodimensionnel [FOU82]. Soit le segment s d'extrémités e_1 et e_2 de valeurs respectives v_1 et v_2. La procédure récursive de subdivision s'écrit:

Algorithme 4-1 : Procédure récursive de subdivision

```
PROCEDURE subdivise (n:integer; e1,e2,v1,v2:real);
VAR
   em,vm : real;
BEGIN
   IF n < niveau de récursivité maximum THEN
   BEGIN
      em := (e1 + e2)/2;
      vm := (v1 + v2)/2 + echelle * (2^-h)^n * X; (1)
      subdivise (n+1,e1,em,v1,vm);
      subdivise (n+1,em,e2,vm,v2);
   END
END;
```

La figure 4-2 visualise le déroulement de cet algorithme après 0,1,2,3,4 et 5 niveaux de récursivité.

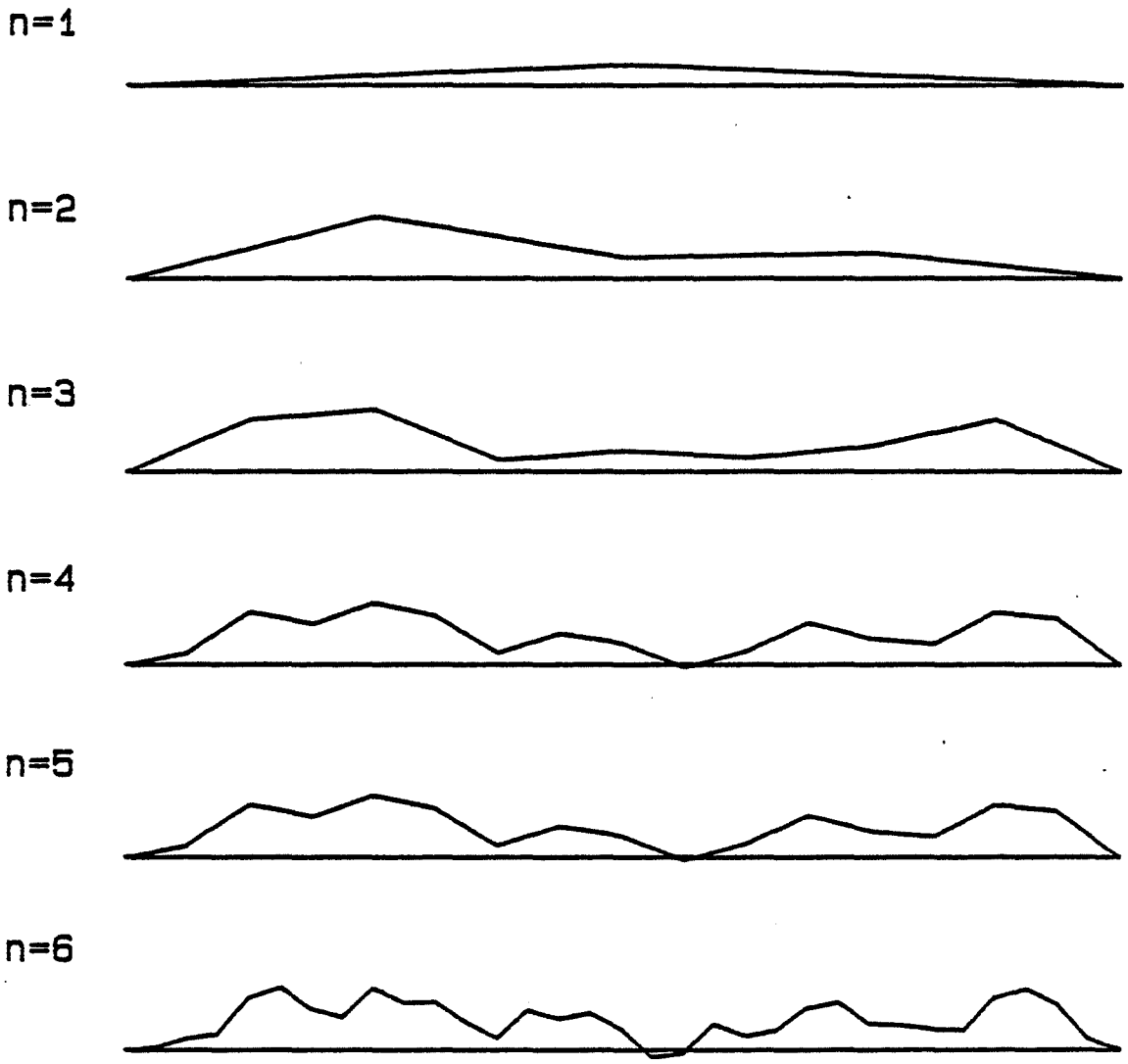


Figure 4-2 : Subdivision récursive d'un segment de droite

Examinons l'instruction (1) que nous allons réutiliser dans le cas bi-dimensionnel.

- échelle est un facteur d'échelle permettant de faire varier l'amplitude de la fonction.
- X représente un tirage d'une variable aléatoire gaussienne de moyenne 0 et de variance 1.
- n est le niveau de récursivité courant.
- h est un paramètre agissant sur l'allure de la courbe. Le rôle de h est décrit plus précisement dans le paragraphe présentant les paramètres du modèle.

La valeur de E (resp F, H et I) est donc calculée en appliquant (1) à A et B (resp A et C, B et D, C et D).

On peut envisager différents procédés de calcul de la valeur du champ en G. Nous avons choisi, de façon arbitraire, de calculer une première fois G en appliquant (1) aux points E et I puis une deuxième fois en l'appliquant à F et H. La valeur définitivement affectée à G est la moyenne des deux valeurs précédentes.

Si l'algorithme de subdivision est simple dans le cas monodimensionnel, le problème de mémorisation ou de recalcul de certaines valeurs frontières se pose dans le cas bi-dimensionnel. Ainsi, sur la figure 4-3 , a,b,c,d et e évalués lors de la subdivision de la région C doivent être réutilisés pour subdiviser D.

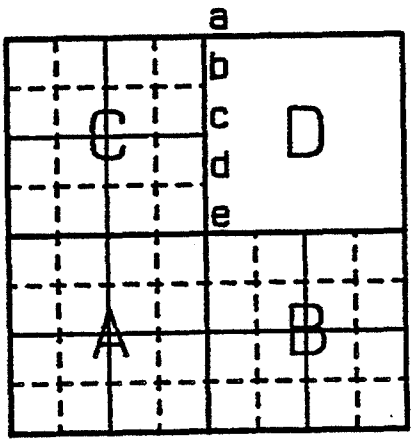


Figure 4-3 : Subdivision récursive: problèmes de continuité

Deux solutions sont envisageables; mémoriser les valeurs ou les recalculer. A. Fournier & al. [FOU82] ont choisi de recalculer ces valeurs en liant le germe du générateur aléatoire à un identificateur du point. Malheureusement, nous ne disposons pas de toutes les informations nécessaires pour discuter cette méthode. Il ne sera donc pas possible d'établir une étude comparative précise avec la solution proposée ici. Pour éviter la solution retenue par A. Fournier & al. qui risque de nuire au caractère aléatoire du processus, nous proposons une structure de donnée permettant de stocker ces valeurs. Cette solution, plus coûteuse en place, devrait être plus rapide car l'évaluation d'un grand nombre de valeurs est remplacée par une lecture en mémoire centrale.

2.2. Description de l'algorithme

2.2.1. Structure de données

La structure doit permettre de stocker les valeurs frontières jusqu'à l'étude de la région adjacente et être compatible avec la récursivité de l'algorithme.

- Un sommet est représenté par ses coordonnées x, y et sa valeur val.
- Une arête est composée
  - . soit de deux arêtes
  - . soit de deux sommets. (cf. figure 4-4)

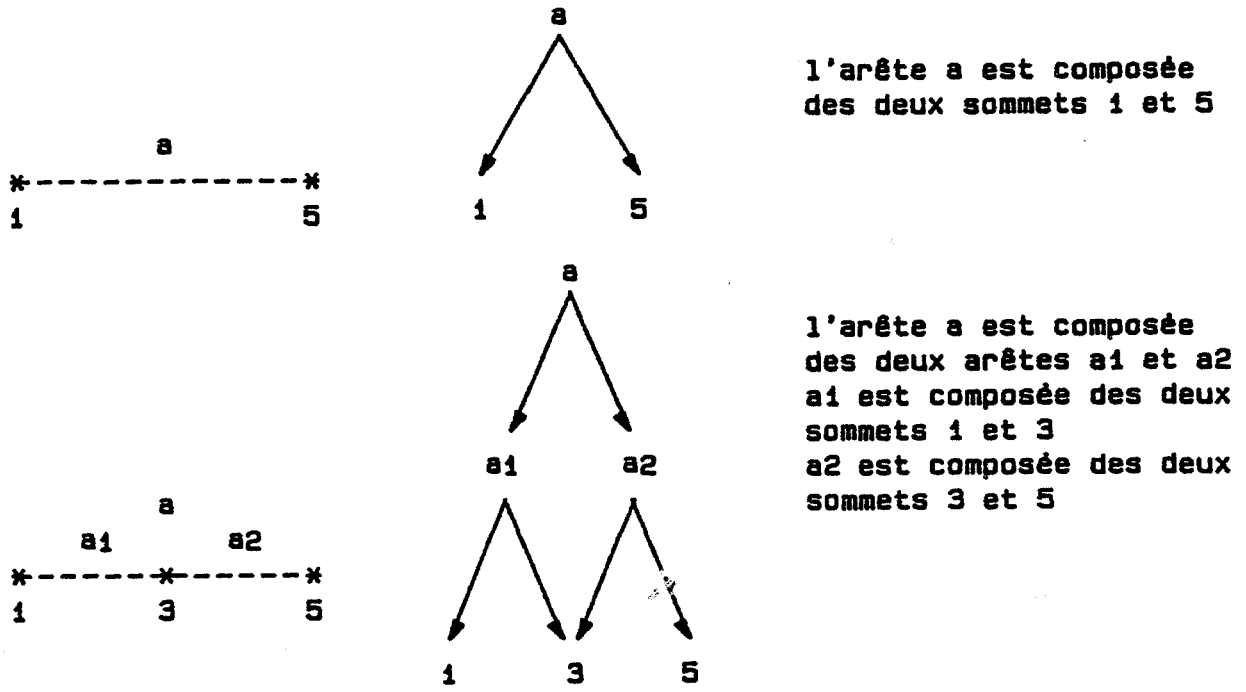


Figure 4-4 : Description d'une arête.

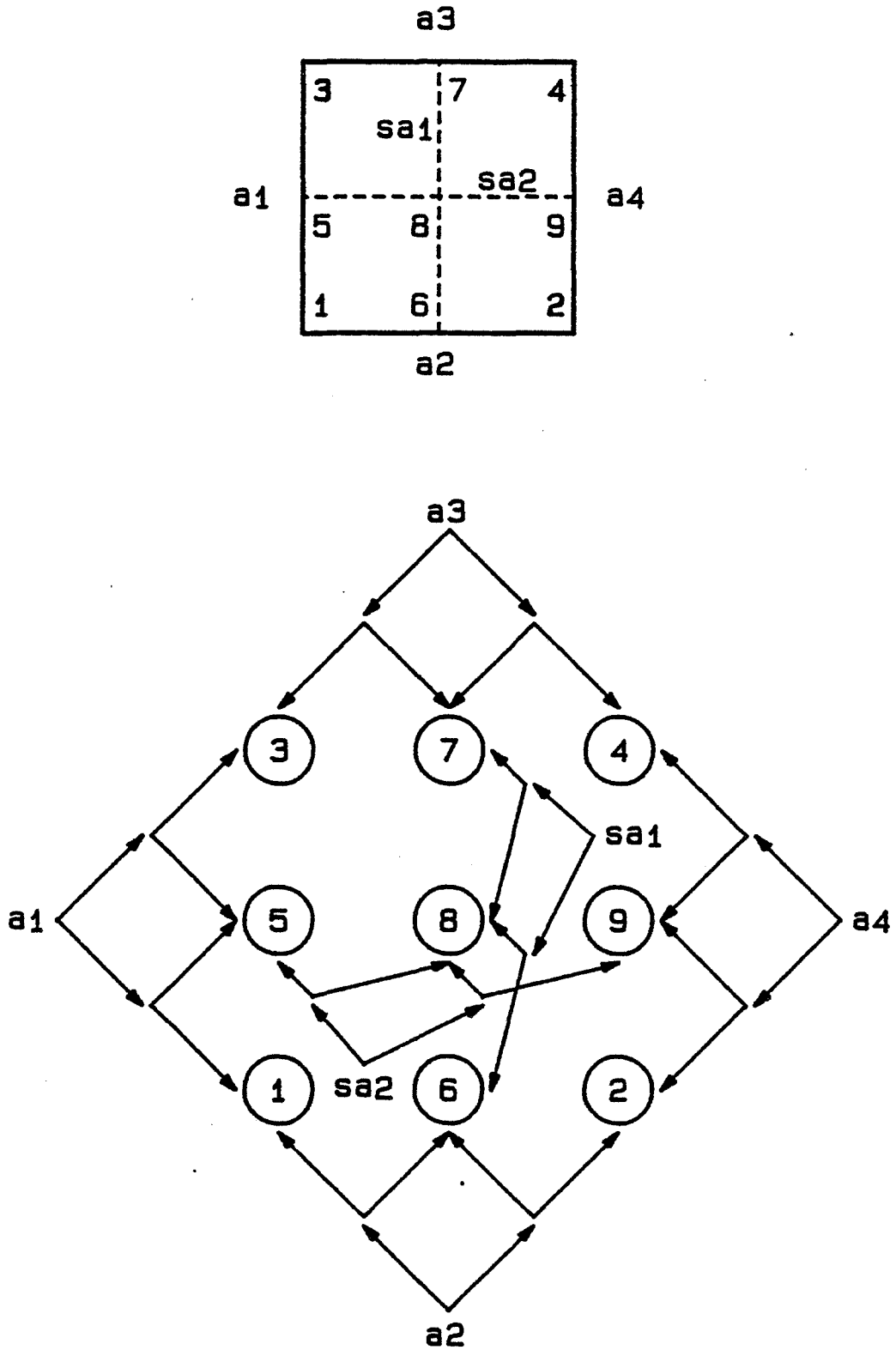


Figure 4-5 : Description d'une région composée

- Une région est identifiée par ses quatre arêtes frontière (cf. figure 4-5)

Aucun sommet n'est dupliqué, un sommet peut donc appartenir à plusieurs arêtes.

Description Pascal des types de la structure de données.

```
TYPE
   type_noeud = (sommet,arête);
   pt_arête = ^type_arête;
   type_arête = RECORD
                    CASE noeud of type_noeud
                       sommet : (x,y : integer; val : real);
                       arête : (fils1,fils2 : pt_arête);
                END;
```

Pour simplifier la description de l'algorithme, il est nécessaire de fixer des conventions sur l'ordre des sommets et des arêtes. Les règles utilisées sont les suivantes (cf. figure 4-6):

- Le fils1 d'une arête horizontale est à l'Ouest.
- Le fils2 d'une arête horizontale est à l'Est.
- Le fils1 d'une arête verticale est au Sud.
- Le fils2 d'une arête verticale est au Nord.

Horizontal (resp. vertical) désigne la direction parallèle à l'axe des x (resp. y).

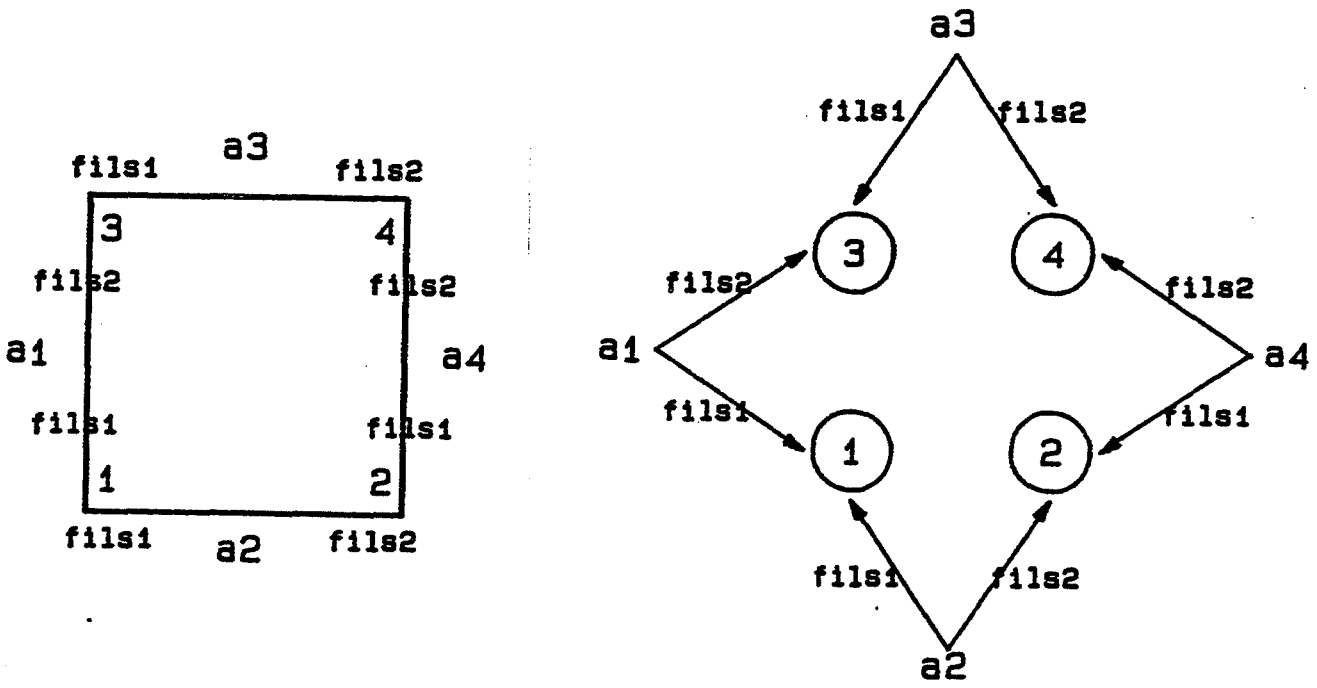


Figure 4-6 : Description d'une région élémentaire

2.2.2. Algorithme

L'algorithme décrit dans ce paragraphe génère un champ aléatoire de valeurs. Les procédures utilisées sont: champ, divise_région, divise_arête et centre.

Champ est une procédure de subdivision récursive. Le traitement appliqué à une région est soit:

- son affichage. La région est affichée quand le niveau de récursivité maximum est atteint. Le mode d'affichage dépend de l'application. Les deux applications les plus courantes sont la génération de champs altimétriques et la synthèse de textures. Nous ne considérerons pas ici la première application; on peut cependant noter que l'ordre de génération des régions est compatible avec l'ordre utilisé par D.P. Anderson [AND82] pour l'affichage d'un champ altimétrique. Le paragraphe suivant décrit l'affichage des textures.

- sa subdivision en 4 régions qui sont ensuite traitées de façon analogue. La subdivision nécessite la mise à jour de la structure c'est à dire:

  . diviser les 4 arêtes de bord,

  . créer 2 nouvelles arêtes,

  . créer 5 nouveaux sommets,

  . calculer les valeurs en ces sommets.

Cette méthode est basée sur une mémorisation des valeurs aux sommets. Rappelons aussi qu'aucun sommet n'est dupliqué. Afin de minimiser la place nécessaire, il est indispensable de détruire les sommets qui ne seront plus utilisés. L'ordre des appels récursifs à la procédure champ étant SO, SE, NO, NE, on peut:

. détruire le sommet SO d'une région affichée

. détruire les arêtes Ouest et Sud d'une région après exécution de la procédure champ

. détruire les arêtes crées par la subdivision d'une région à la fin de l'exécution de la procédure champ.

Description des paramètres de la procédure champ:

- n est le niveau de récursivité courant,

- std est égal à 1 lors du premier appel de la procédure,

- a1, a2, a3 et a4 sont les quatre arêtes définissant la région, elles sont dans l'ordre: Ouest, Sud, Nord et Est.

- rapport est une variable globale égale à (2^{-h}).

Algorithme 4-2 : Subdivision récursive d'une région

```
PROCEDURE champ(std:REAL; n:INTEGER; a1,a2,a3,a4:pt_arête);

VAR
   i : integer;
   sa1, sa2 : pt_arête;

BEGIN
  IF n < niveau_de_récursivité_maximum THEN
  BEGIN
    divise_région(std,a2,a3,sa1);
    divise_région(std,a1,a4,sa2);
    centre(std,sa1,sa2);
    std := std*rapport;
    n := n+1;
    champ(std,n,a1^.fils1,a2^.fils1,sa2^.fils1,sa1^.fils1);
    DISPOSE(a1^.fils1);
    DISPOSE(a2^.fils1);
    champ(std,n,sa1^.fils1,a2^.fils2,sa2^.fils2,a4^.fils1);
    DISPOSE(sa1^.fils1);
    DISPOSE(a2^.fils2);
    champ(std,n,a1^.fils2,sa2^.fils1,a3^.fils1,sa1^.fils2);
    DISPOSE(a1^.fils2);
    DISPOSE(sa2^.fils1);
    champ(std,n,sa1^.fils2,sa2^.fils2,a3^.fils2,a4^.fils2);
    DISPOSE(sa1^.fils2);
    DISPOSE(sa2^.fils2);
    DISPOSE(sa1);
    DISPOSE(sa2);
  END
  ELSE
  BEGIN
    traiter(a1,a2,a3,a4);
    DISPOSE(a1^.fils1);
  END;
END;
```

Divise_région divise une région en deux régions. Après avoir subdivisé les deux arêtes, l'arête de subdivision est générée.

- a1 et a2 sont: soit les arêtes Ouest et Est, soit les arêtes Sud et Nord dans cet ordre.

- sa1 est l'arête de subdivision. sa1 est horizontale si a1 et a2 sont des arêtes verticales (i.e. Ouest et Est) et verticale si a1 et a2 sont des arêtes horizontales (i.e. Sud et Nord).

Algorithme4-3: Subdivision d'une région

```
PROCEDURE divise_région (std : REAL; a1,a2,sa1 : pt_arête);

VAR
   p : pt_arête;

BEGIN
   divise_arête (std,a1);
   divise_arête (std,a2);

   NEW (sa1);
   sa1^.noeud := arête;

   p := a1^.fils1^.fils2;
   WHILE p^.noeud = arête DO
      p := p^.fils2;
   sa1^.fils1 := p;

   p := a2^.fils1^.fils2;
   WHILE p^.noeud = arête DO
      p := p^.fils2;
   sa1^.fils2 := p;
END;
```

Divise_arête divise une arête en deux. Cette opération se déroule en deux temps: une mise à jour de la structure puis un calcul de la valeur du point milieu généré.

Algorithme4-4: Subdivision d'une arête

```
PROCEDURE divise_arête (std:REAL; a:pt_arête);

VAR
   p1, p2 : pt_arête;

BEGIN
   IF a^.fils1^.noeud = sommet THEN
   BEGIN
      p1 := a^.fils1;
      p2 := a^.fils2;
      NEW (a^.fils1);
      NEW (a^.fils2);
      a^.fils1^.noeud := arête;
      a^.fils1^.fils1 := p1;
      a^.fils2^.noeud := arête;
      a^.fils2^.fils2 := p2;
      NEW (a^.fils1^.fils2);
      a^.fils1^.fils2^.noeud := sommet;
      a^.fils2^.fils1 := a^.fils1^.fils2;
      a^.fils1^.fils2^.val := (p1^.val + p2^.val)/2 + std * X;
      a^.fils1^.fils2^.x := (p1^.x + p2^.x)div 2;
      a^.fils1^.fils2^.y := (p1^.y + p2^.y)div 2;
   END;
END; { de divise_arête }
```

Soient deux arêtes a1 et a2 concourantes; la procédure centre calcule la valeur moyenne à attribuer au sommet d'intersection et met à jour la structure.

Algorithme4-5: Calcul du centre

```
   PROCEDURE centre (std:REAL; a1,a2:pt_arête);
  .SP
   VAR
      p1, p2 : pt_arête;
  .SP
   BEGIN
      divise_arête (std,a1);
      divise_arête (std,a2);
      p1 := a1^.fils1^.fils2;
      p2 := a2^.fils1^.fils2;
      p1^.val := (p1^.val + p2^.val)/2;
      DISPOSE (p2);
      a2^.fils1^.fils2 := p1;
      a2^.fils2^.fils1 := p1;
   END;
```

2.2.3. Evaluation en place

La méthode choisie est basée sur la mémorisation des sommets et des arêtes devant être réutilisés. Un sommet ou une arête est crée lors de sa première utilisation et détruit après la dernière. Nous allons évaluer le nombre maximum de sommets et d'arêtes stockés au cours de l'exécution du programme. Ce nombre est évalué en fonction du nombre de niveaux de récursivité requis.

2.2.3.1. Notations et résultats préliminaires

* **Indice d'une région**
  Une région est décomposée en quatre sous-régions sri. L'indice correspond à l'ordre de balayage c'est à dire:

  sr1 : SO
  sr2 : SE
  sr3 : NO
  sr4 : NE

* **Niveau de récursivité**
  Si n est le niveau de récursivité d'une région R, le niveau de récursivité de chacune des quatre sous-régions de R est n-1. Ainsi, le nombre maximum de sommets d'une arête d'une région R de niveau de récursivité n est égal à 2^n+1.

* **Classes de sommets**
  Soit une région R. On rangera les sommets de la région R en trois classes:

  - les sommets de frontière: SF, notés *,
  - les sommets externes: SE, notés o,
  - les sommets internes: SI, notés +.

  La figure 4-7 explicite cette notion de classe pour la région R.

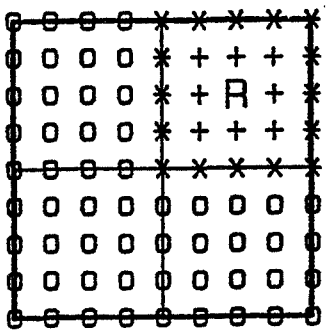
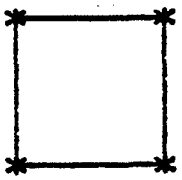


Figure 4-7 : Les différents types de sommets

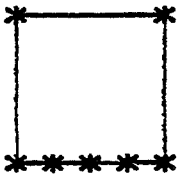
* **Classes de régions**
Les régions sont réparties en différentes classes en fonction des sommets de frontière mémorisés avant l'étude de la région.

On distinguera quatre classes:

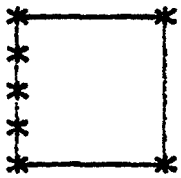
o <u>la classe A</u> : les seuls sommets mémorisés sont les sommets d'angle.



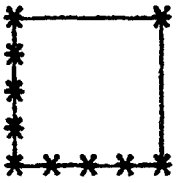
o <u>la classe B</u> : les sommets d'angle et les sommets de la frontière Sud sont mémorisés.



o <u>la classe C</u> : les sommets d'angle et les sommets de la frontière Ouest sont mémorisés.



o <u>la classe D</u> : les sommets d'angle et les sommets des frontières Sud et Ouest sont mémorisés.



2.2.3.2. Détermination de la région pour laquelle le maximum est atteint

La figure 4-8 visualise l'évolution du nombre de sommets mémorisés pendant l'exécution du programme. Avant de calculer le nombre maximum de sommets et d'arêtes stockés, nous allons rechercher pour quelle sous-région ce maximum est atteint.

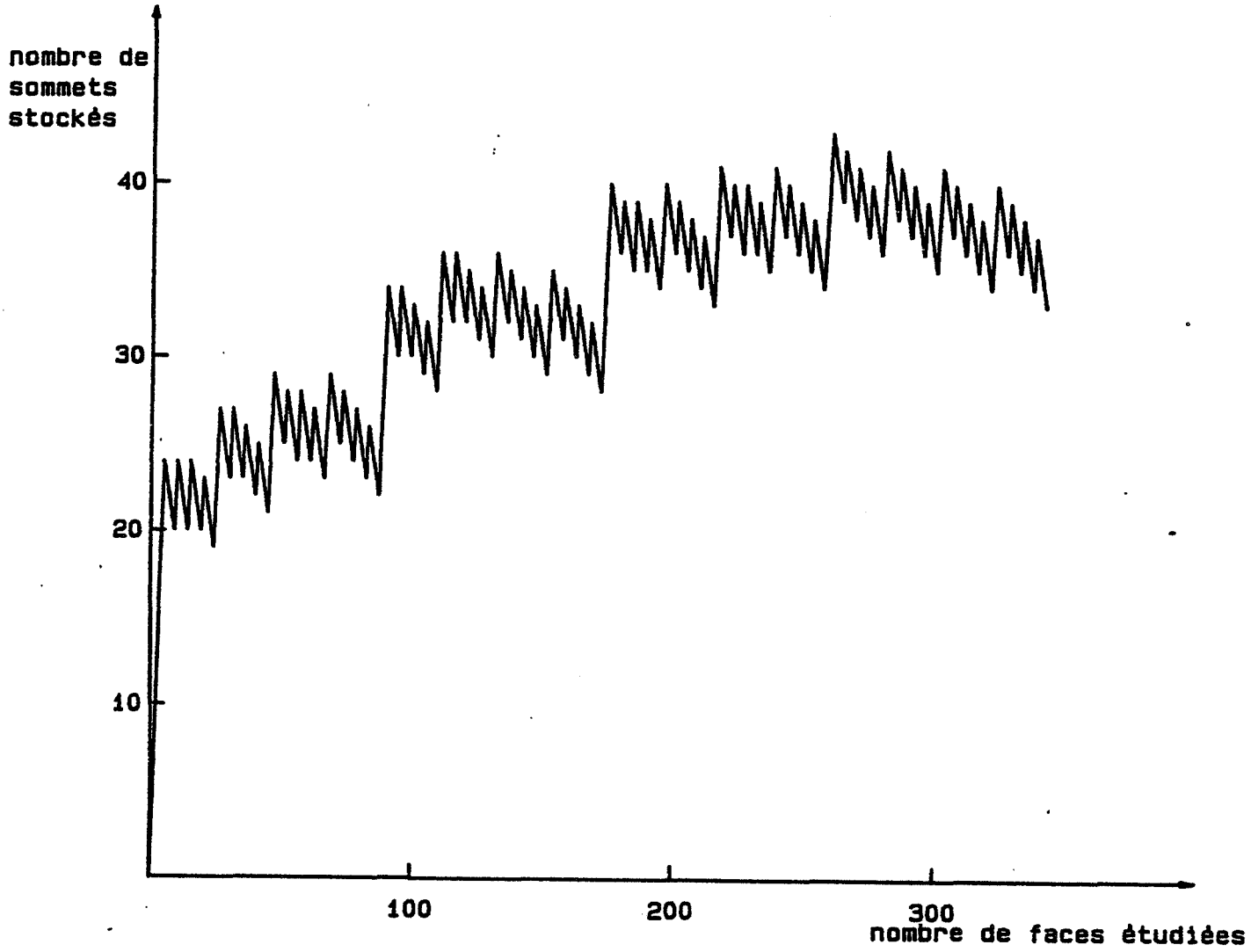


Figure 4-8 : Evolution du nombre de sommets mémorisés (n=4)

Propriété1 :

Le processus de subdivision d'une région est indépendant de la région.

Les sommets touchés par la subdivision d'une région sont les sommets internes et les sommets de frontière. Les sommets externes restent inchangés.

Propriété2 :

Soient R1 et R2 deux régions de même niveau de récursivité. Le nombre maximum de sommets mémorisés pendant l'étude de R1 est supérieur au nombre maximum de sommets mémorisés pendant l'étude de R2 si :

$$nb(SE(R1)) \geqslant nb(SE(R2))$$
$$\text{et}$$
$$SF(R1) \supseteq SF(R2)$$

C'est à dire:

Le nombre de sommets externes mémorisés lors de l'étude de R1 est supérieur au nombre de sommets externes mémorisés lors de l'étude de R2.

L'ensemble des sommets frontière de R2 mémorisés avant l'étude de R2 est inclus dans l'ensemble des sommets frontière de R1 mémorisés avant son étude.

Nous allons utiliser cette propriété pour rechercher pour quelle région le nombre de sommets stockés est maximum.

Evaluons tout d'abord le nombre de sommets externes et le nombre de sommets de frontière pour chacune des quatre sous-régions d'une région R. Ces valeurs sont dépendantes de la classe de R.

Soit n le niveau de récursivité de la région R. On considérera n=3 pour les figures.

[1] La région R appartient à la classe A

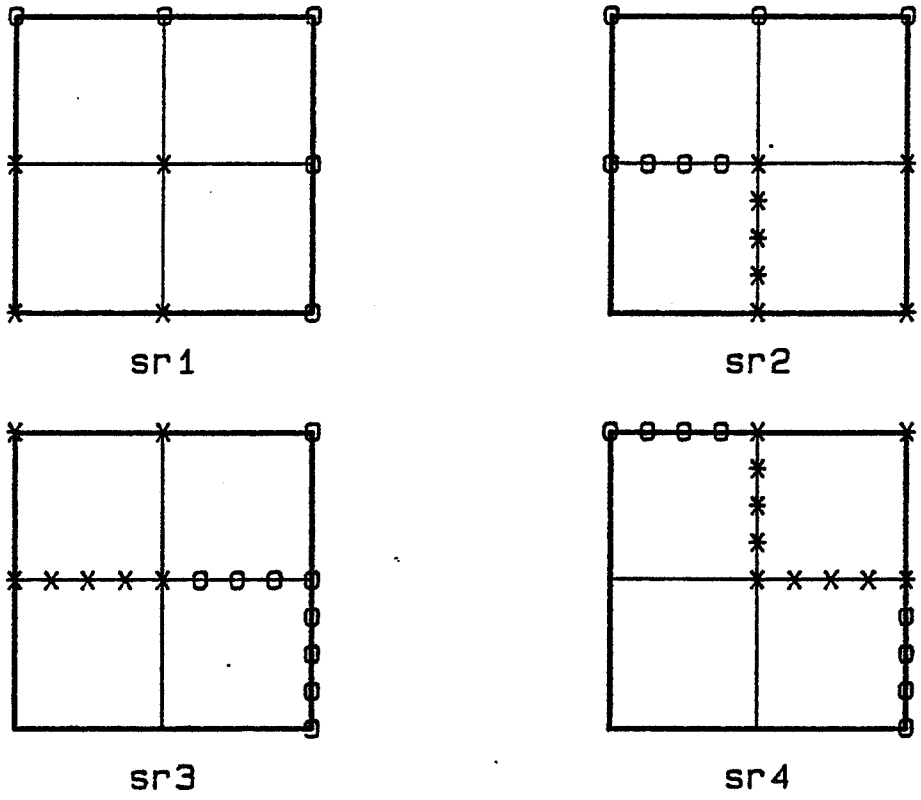


Figure 4-9 : Région de la classe A

	SF	SE
sr1	4	5
sr2	$2^{n-1}+3$	$2^{n-1}+3$
sr3	$2^{n-1}+3$	$2^{n}+1$
sr4	$2^{n}+2$	2^{n}

D'après la propriété 2, le nombre maximum de sommets ne peut être atteint pour la sous-région sr1. Par contre, il n'est pas possible de conclure entre les trois autres sous-régions.

[2] La région R appartient à la classe B

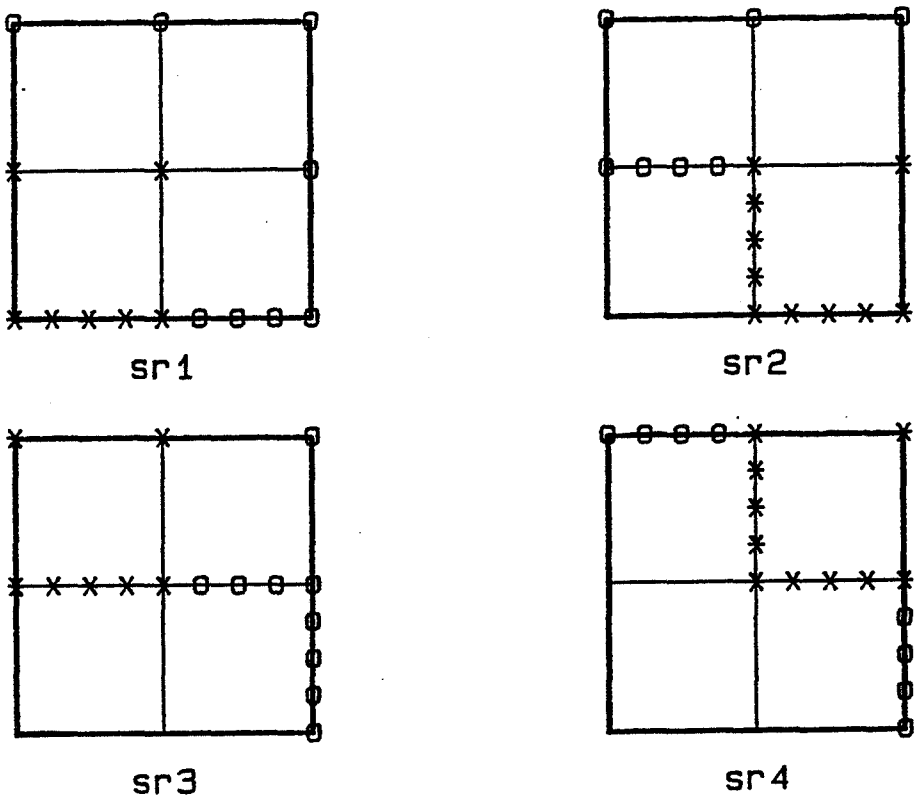


Figure 4-10 : Région de la classe B

	SF	SE
sr1	$2^{n-1}+3$	$2^{n-1}+4$
sr2	$2^{n}+2$	$2^{n-1}+3$
sr3	$2^{n-1}+3$	$2^{n}+1$
sr4	$2^{n}+2$	2^{n}

D'après la propriété 2, le nombre maximum de sommets est atteint soit pour la sous-région sr3 soit pour sr4.

[3] La région R appartient à la classe C

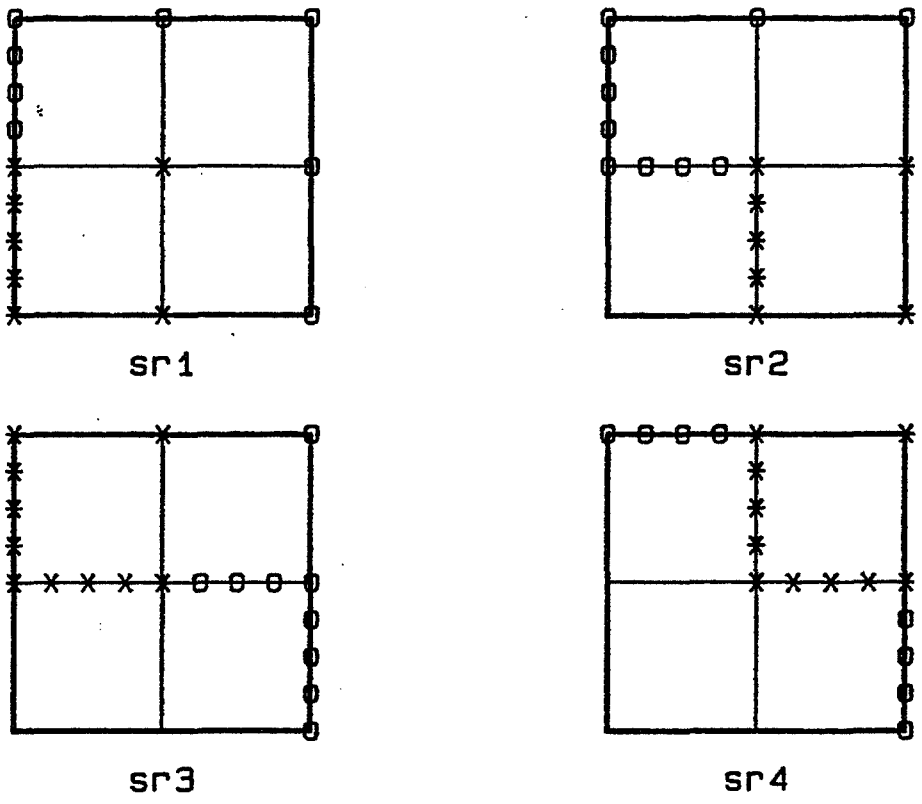


Figure 4-11 : Région de la classe C

	SF	SE
sr1	$2^{n-1}+3$	$2^{n-1}+4$
sr2	$2^{n-1}+3$	$2^{n}+2$
sr3	$2^{n}+2$	$2^{n}+1$
sr4	$2^{n}+2$	2^{n}

D'après la propriété 2, le nombre maximum de sommets est atteint soit pour sr2 soit pour sr3.

[4] La région R appartient à la classe D.

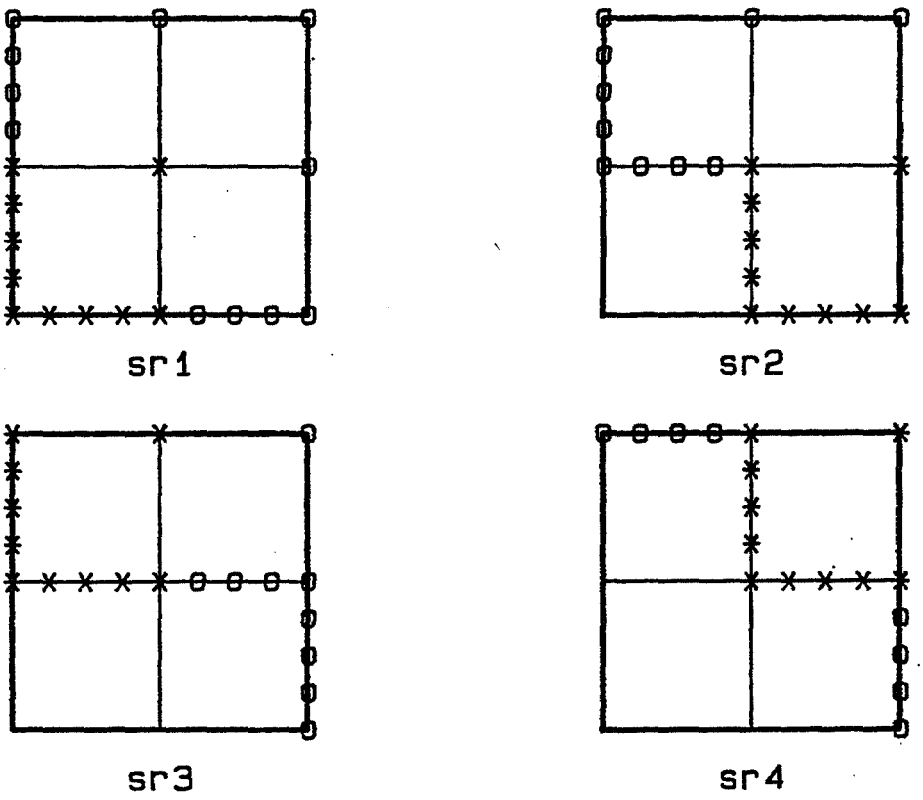


Figure 4-12 : Région de la classe D

	SF	SE
sr1	2^n+2	2^n+3
sr2	2^n+2	2^n+2
sr3	2^n+2	2^n+1
sr4	2^n+2	2^n

D'après la propriété 2, le nombre maximum de sommets est atteint pour la sous-région sr1.

Soit R la région initiale à étudier, on supposera que son niveau de récursivité est supérieur ou égal à 3. Cette région appartient à la classe A. Le nombre maximum de sommets est donc atteint pour l'une des sous-régions sr2, sr3 ou sr4.

La sous-région sr2 appartient à la classe C
La sous-région sr3 appartient à la classe B
La sous-région sr4 appartient à la classe D

En notant ssrij la sous-région j de la sous-région i de la région R, on peut affirmer que le maximum est atteint pour l'une des régions suivantes: ssr22, ssr23, ssr33, ssr34 ou ssr41.

Evaluons pour chacune de ces sous-régions le nombre de sommets externes et le nombre de sommets de frontière.

	SF	SE
ssr22	$2^{n-1}+3$	$2^{n}+7$
ssr23	$2^{n}+2$	$2^{n}+6$
ssr33	$2^{n-1}+3$	$2^{n+1}+2$
ssr34	$2^{n}+2$	$2^{n+1}+1$
ssr41	$2^{n}+2$	$2^{n+1}+3$

D'après la propriété 2, on peut affirmer que le nombre maximum de sommets est atteint pour la région ssr41.

Hypothèse d'induction:

Supposons qu'au niveau n le maximum soit atteint pour une région de classe D, alors au niveau n-1, le maximum est atteint pour la sous-région 1 de cette dernière qui est aussi de la classe D.

Si n est le niveau de récursivité de la région initiale R, on a montré qu'au niveau n-2, le maximum est atteint pour la région ssr41 qui appartient à la classe D. Donc d'après l'hypothèse d'induction précédente, le maximum est atteint lors de l'étude de la région de niveau 1 dont la suite des indices des régions mères est: 4,1,1...,1,1. On notera cette région Rmax.

Un raisonnement analogue permet de montrer le résultat pour les arêtes.

2.2.3.3. Calcul du maximum

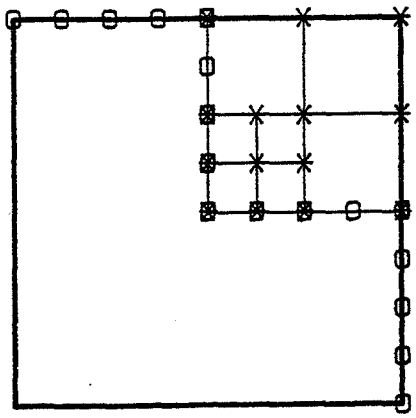


Figure 4-13 : Sommets stockés lors de l'étude de la région Rmax (n=3)

Comme le montre la figure 4-13, les sommets à prendre en compte sont de deux types: les sommets de frontière, appartenant aux régions étudiées antérieurement (o) et les sommets de subdivision créés pour atteindre la région courante (*).

Nombre de sommets de frontière : $2^{n+1}+1$
Nombre de sommets de subdivision : $9+5(n-2)$
Nombre de sommets de subdivision et de frontière : $3+2(n-1)$

Le nombre maximum de sommets est donc : $2^{n+1} + 3n - 1$.

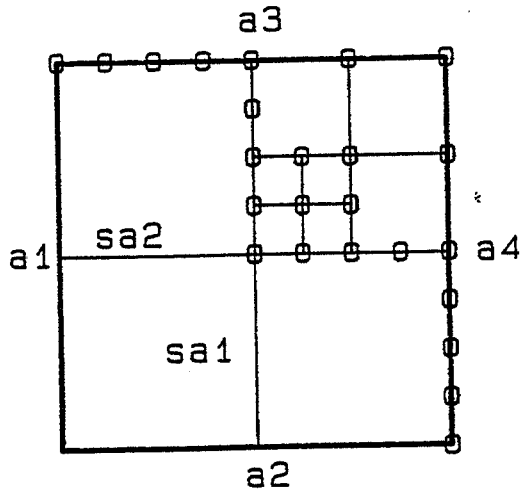


Figure 4-14 : Nombre maximum d'arêtes stockées

Le nombre maximum d'arête est atteint pour la configuration décrite figure 4-14.

Nombre d'arêtes mémorisées :

a1	: 1
a2	: 1
a3	: $2^n + 3$
a4	: $2^n + 3$
sa1	: 2^n
sa2	: 2^n
région N-E	: $10(n-2) + 6$

Le nombre maximum d'arêtes est donc égal à:

$$2 + 2(2^n+3) + 2(2^n) + 10(n-2) + 6$$
$$= 2^{n+2} + 10n - 6$$

Cette évaluation montre que la place augmente exponentiellement avec le nombre de niveaux de récursivité; ce résultat était prévisible pour un algorithme récursif. Cependant, si on considère qu'il est inutile de subdiviser à une résolution plus fine que le pixel, n est inférieur ou égal à 9. Il faut alors stocker au plus 1050 sommets et 2132 arêtes.

2.2.4. Evaluation en temps

Le temps CPU est aussi exponentiel avec le niveau de récursivité; la complexité de l'algorithme est évidemment $O(4^n)$. Cependant, pour les mêmes raisons que précédemment, les temps de calcul sont raisonnables. On constate que sur notre configuration, l'exécution est en général freinée par l'affichage.

Le temps d'exécution sur HP 9000, sans affichage, est environ d'une minute et peut sans doute être amélioré.

3. APPLICATION A LA SYNTHESE DE TEXTURES ET RESULTATS

3.1. Affichage

L'algorithme décrit précédemment est basé sur une méthode de subdivision de faces. Une région est subdivisée tant que le niveau de récursivité maximum n'est pas atteint. Une fois ce niveau atteint, la région est affichée. L'affichage suppose la connaissance de la couleur en chaque sommet. Cette couleur est calculée en fonction du coefficient du sommet de la grille, le choix des couleurs est décrit au paragraphe suivant. La région est ensuite remplie par un lissage de Gouraud [GOU71] afin de dissimuler la structure régulière de la grille de génération.

3.2. Les différents paramètres du modèle

Nous avons cité, lors de la présentation de l'algorithme, un certain nombre de paramètres du modèle. Nous allons, dans ce paragraphe, exposer avec plus de détails leur rôle et leur influence sur l'image finale.

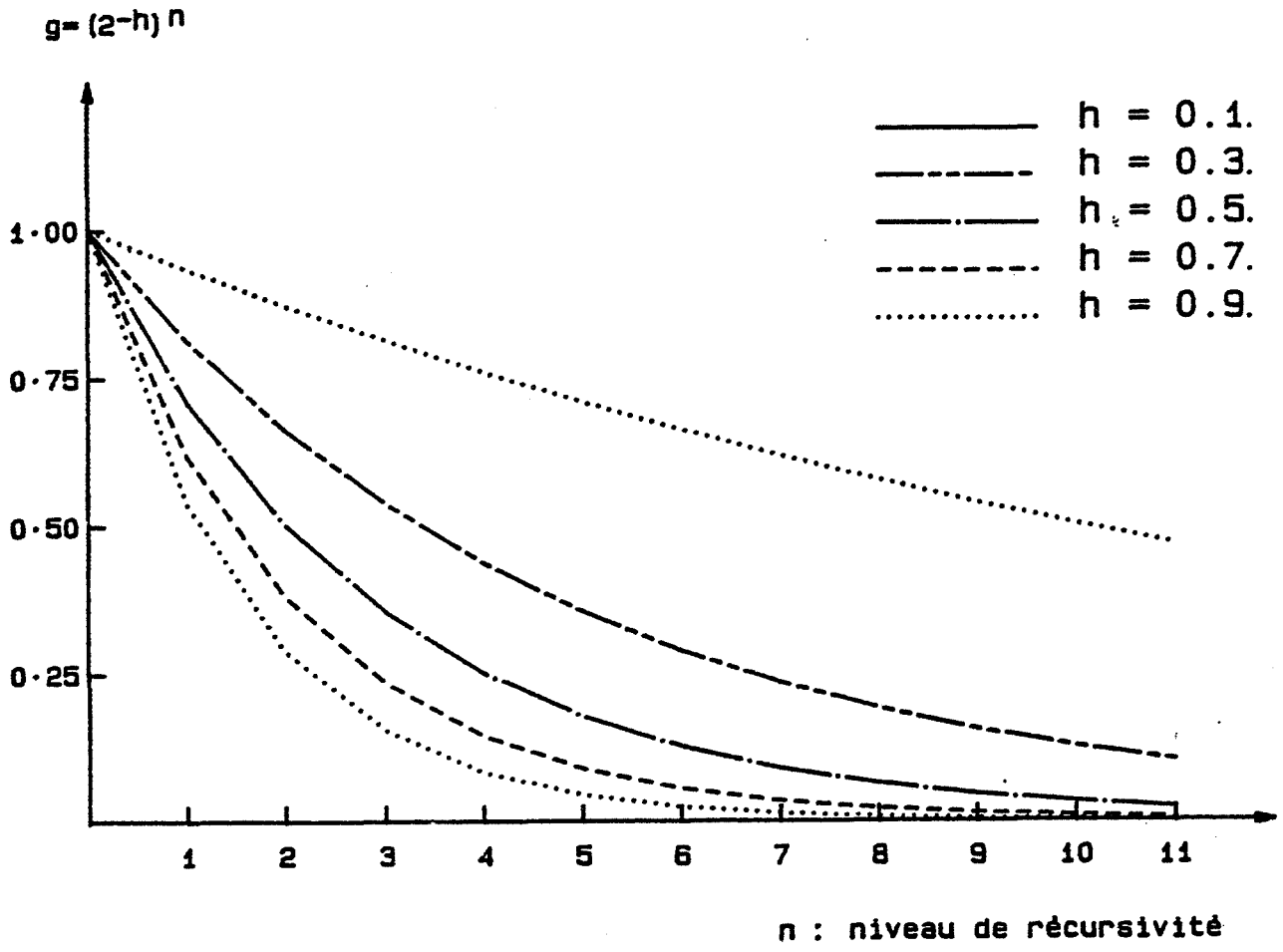


Figure 4-15 : $g = (2^{-h})^n$

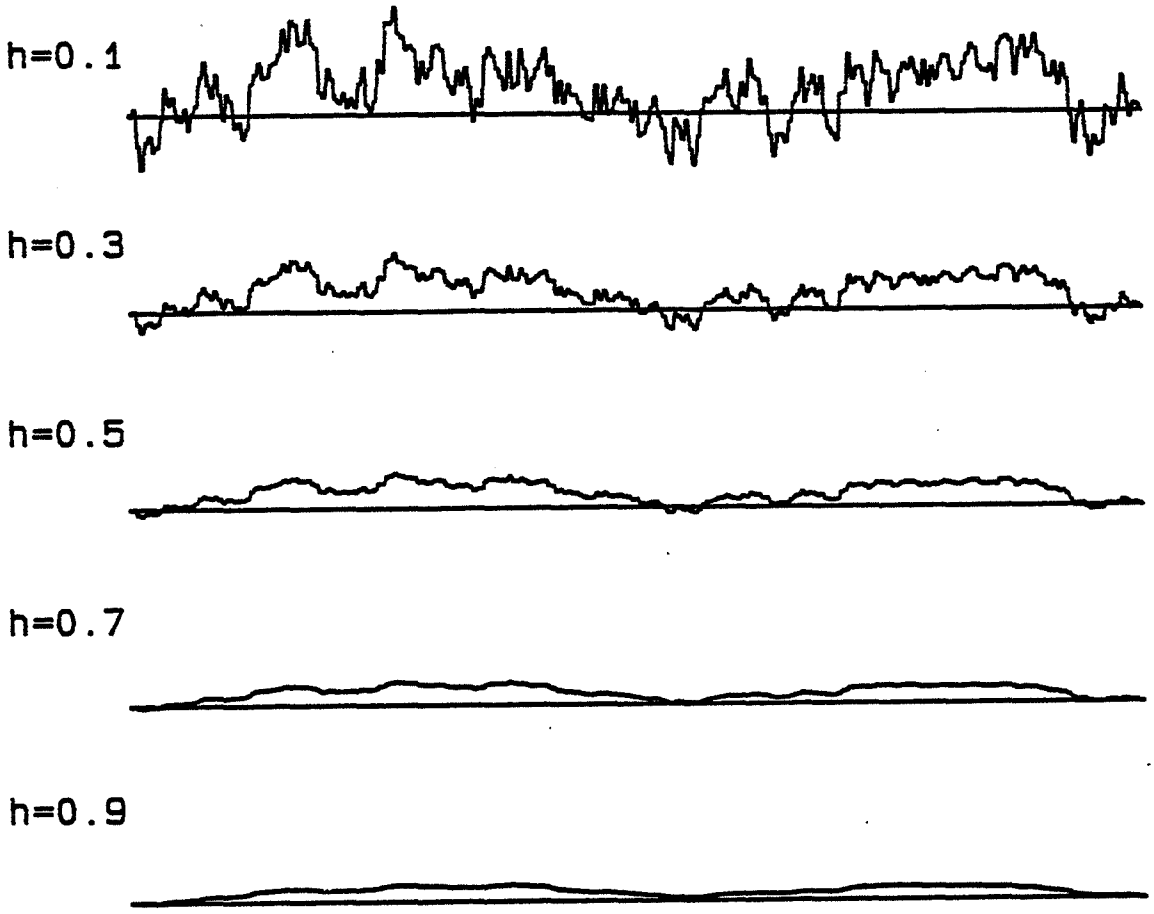


Figure 4-16 : Influence de h sur la subdivision d'un segment

- **Le paramètre h**
  h est un paramètre compris entre 0 et 1, qui régle, selon les propres termes de B. Mandelbrot [MAN75], "l'intensité de l'adoucissement et donc celle de la persistance des accroissements". Pour expliciter l'effet du paramètre h dans le cas monodimensionnel, la figure 4-15 visualise la trace de la fonction $g(n) = (2^{-h})^n$ pour différentes valeurs de h et la figure 4-16 représente le rôle de h dans la subdivision d'un segment après 5 niveaux de récursivité. Dans le cas bi-dimensionnel, la photo 7 visualise plusieurs occurences d'une texture de ciel avec différentes valeurs de h. On constate qu'une valeur de h faible contribue à augmenter le contraste de l'image, tandis qu'une valeur forte adoucit les contrastes.

- **Les couleurs**
  Les couleurs sont décrites dans une table de conversion adressée par les valeurs des sommets de la grille. A chaque adresse correspondent les trois composantes rouge, vert et bleu de la couleur à affecter au point. La table de

conversion est initialisée en associant à chacune des trois composantes une rampe continue de valeurs. Les trois rampes sont indépendantes et construites par interpolation linéaire entre un certain nombre (variable) de valeurs fixées interactivement à l'aide de la table à numériser.

Le choix des rampes permet quelques investigations colorimétriques intéressantes. Cependant, il serait fastidieux de devoir renouveler l'opération trop souvent. Nous avons par conséquent défini une table de conversion standard permettant de représenter des ciels variés. Cette table est construite avec un dégradé de couleurs allant du bleu au gris en passant par des teintes plus claires, blanchâtres. En fonction de la zone de la table adressée, on peut obtenir:

. un ciel de beau temps bleu uni,

. un ciel bleu avec quelques nuages blancs,

. un ciel blanchâtre avec quelques zones bleues et quelques nuages plus sombres,

. un ciel de mauvais temps dans les gris.

- **Les niveaux de récursivité**
  Le nombre maximum de niveaux de récursivité est un paramètre qui règle la taille des régions élémentaires affichées. Afin d'augmenter la souplesse de l'algorithme, il a paru intéressant de fixer le niveau de récursivité indépendamment suivant chacun des deux axes de subdivision. Comme le montre la photo 8, le nombre de niveaux de récursivité agit sur la taille des nuages de la texture.

3.3. Extensions

3.3.1. Mise en perspective

Jusqu'à présent, la texture est générée vue de face. Le problème de mise en perspective des textures planes à suscité différentes recherches: On trouve une bibliographie dans [WIL83] ou [GAN84]. Plusieurs méthodes utilisent la transformation inverse associant un point de la texture plane à chaque pixel. Cependant, l'affichage par cette technique est ralenti par le calcul de la transformation et par les filtrages nécessaires pour résoudre les problèmes d'aliassage. Pour des textures aléatoires générées par subdivision récursive, on peut exploiter des résultats de géométrie projective élémentaire pour construire directement la texture en perspective.

Considérons les droites X1,X2 et Y1,Y2, définies dans le plan de la texture et leurs images dans le repère écran X'1,X'2 et Y'1,Y'2 (cf. figure 4-17). X'1,X'2 et Y'1,Y'2 se coupent en X'_∞ et Y'_∞ qui sont les images des points à l'infini des directions $^\infty$X et Y du plan de la texture. La transformation perspective et son inverse étant toutes deux des transformations projectives, on peut appliquer le résultat suivant: le birapport de quatre points alignés est un invariant projectif [BER79].

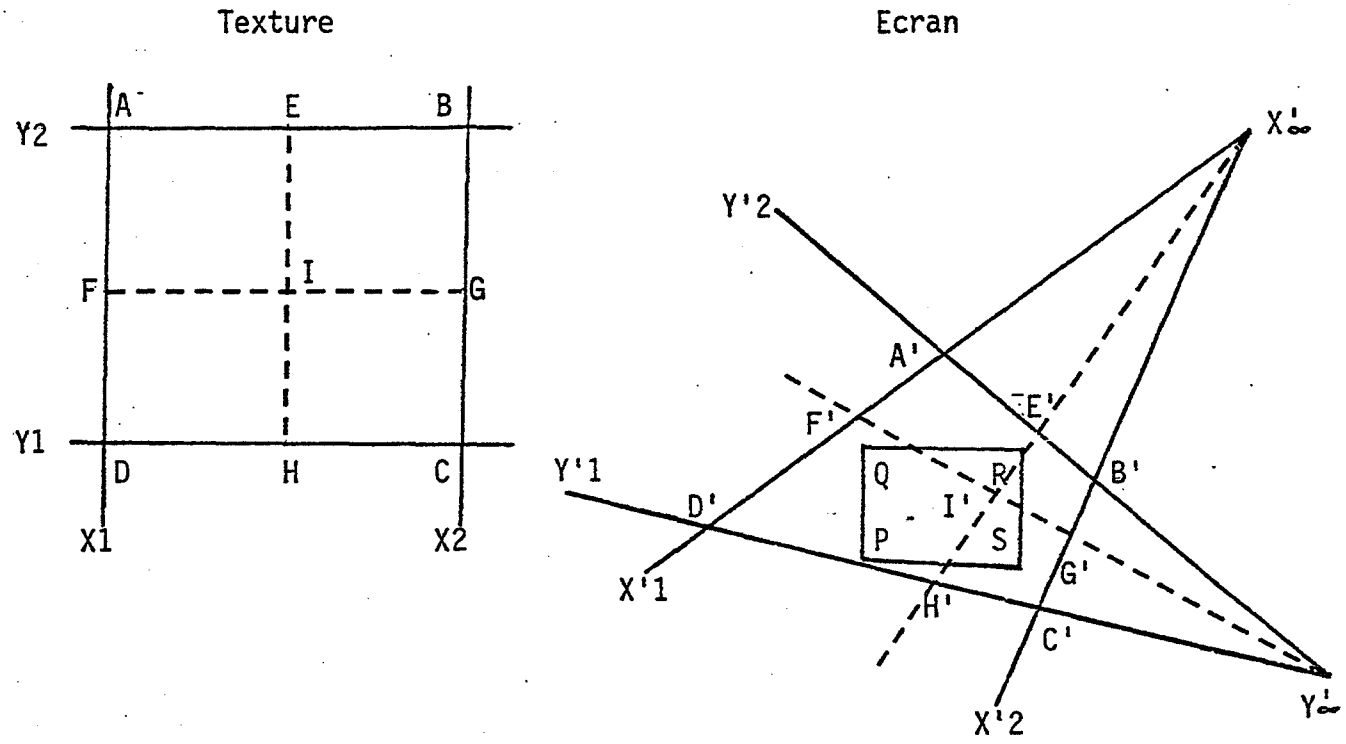


Figure 4-17 : Transformation perspective

De plus si U,V et M sont alignés sur L, droite d'un espace affine, alors U,V,M et le point à l'infini L_∞ de L forment une division harmonique si et seulement si M est le milieu de U V.

On a donc:

$$(U,V,M,L_\infty) = (U',V',M',L'_\infty) = -1$$

où les primes représentent l'image du point dans le repère de l'écran.

D'où:

$$\frac{\overline{L'_\infty U'}}{\overline{L'_\infty V'}} = -\frac{\overline{M'U'}}{\overline{M'V'}}$$

Les coordonnées de M' peuvent donc être évaluées avec deux divisions.

3.3.2. Antialiassage

La mise en perspective de textures entraîne généralement des problèmes d'aliassage. Ce phénomène se manifeste sous la forme de moirés dans les zones fortement comprimées de l'image. Le théorème d'échantillonnage montre que ces moirés sont dûs au repliement du spectre lorsque l'image n'est plus suffisamment échantillonnée. Pour résoudre ce problème, un grand nombre de méthodes ont été développées [GAN84]; elles utilisent un filtrage de l'image, dans les zones les plus comprimées. Cependant, ces techniques, appliquées à la texture lors de l'exécution de la transformation perspective, sont généralement couteuses en temps de calcul.

Dans notre application, le problème de l'aliassage peut être résolu simplement en stoppant le mécanisme de subdivision lorsque la taille de l'arête de la région est inférieure à 1 pixel.

3.3.3. Fenêtrage

La technique de mise en perspective décrite précédemment exige une définition de la région à texturer très contraignante: quadrilatère image par la transformation perspective d'un rectangle. Il est donc nécessaire de disposer de procédures de fenêtrage efficaces. On peut distinguer:

* **le fenêtrage des régions non élémentaires**
  Une région non élémentaire est fenêtrée afin de définir si elle est

  + entièrement externe
    Dans ce cas, elle n'est pas étudiée (pas subdivisée).

  + entièrement interne
    Dans ce cas elle est étudiée mais ses régions filles ne sont pas fenêtrées.

  + coupée
    Dans ce cas, elle est étudiée et ses régions filles doivent être fenêtrées.

* **le fenêtrage des régions élémentaires**
  Avant d'être affichée, une région élémentaire, dont aucune des régions mères n'est entièrement interne, est fenêtrée.

Cette méthode est compatible avec différentes formes de fenêtres (rectangulaire, convexe, concave). Seule la procédure de fenêtrage d'une région doit être modifiée.

3.3.4. Représentation du soleil

L'objectif de la recherche étant de représenter des ciels réalistes,l'étude n'aurait pas été complète sans essayer de visualiser le soleil.

Avant d'étudier les différentes situations qui peuvent se présenter, il faut décrire plus précisément ce que l'on souhaite représenter. On peut distinguer deux aspects: le soleil et le halo qui l'entoure.

- Les problèmes liés à la visualisation du soleil sont dépendants du matériel. Il n'est pas possible de visualiser sur un écran un point (ou une surface) suffisamment lumineux pour réellement représenter le soleil. Cependant, une tâche blanchâtre constitue une première approximation intéressante.

- Le halo est beaucoup plus facile à représenter. Il s'agit d'une tâche blanchâtre entourant le soleil. La teinte de cette tâche décroit en fonction de la distance. Le modèle standard de décroissance lumineuse en fonction du carré de la distance paraît satisfaisant.

La visualisation du soleil (et du halo), dépend du temps (du type de ciel) représenté. L'effet sur un ciel très nuageux est insignifiant. Ce cas est identifiable par la couleur. On fixe dans la table de conversion une borne correspondant au seuil jusqu'au quel l'action du soleil doit être visualisée. Ainsi, l'évaluation de la couleur de chaque point de la grille incorporera une estimation de l'action du soleil.

Malgré les problèmes matériels interdisant certains effets, l'usage d'un modèle simple comme celui décrit précédemment, contribue à améliorer le rendu de l'image.

3.4. Résultats et conclusions

La méthode décrite dans ce chapitre, est une méthode spécifique. Nous allons résumer les caractéristiques des images générées, nous envisagerons ensuite quelques extensions du modèle.

Nous pouvons tout d'abord noter le caractère aléatoire et irrégulier des résultats. Cet aspect convient très bien à la simulation de phénomènes naturels. Par ailleurs, les textures crées jusqu'à présent sont assez basses fréquences. Cette caractéristique est liée à un certain nombre de choix qui sont discutés ci-dessous. L'homothétie interne, créée par la subdivision récursive, favorise une structure d'imbrication des

formes qui est bien adaptée à la visualisation de phénomènes naturels tels que les nuages, les veines du bois (cf. photo 11).

Cette étude à été fortement orientée par l'objectif que nous nous sommes fixé: visualiser des textures représentant des ciels nuageux. On peut cependant envisager un certain nombre de développements qui devraient probablement permettre de découvrir de nouvelles applications du modèle.

- L'interpolation linéaire de la couleur sur les éléments de grille crée un lissage de l'image. L'ajout de bruit blanc à l'interpolation linéaire devrait permettre un rendu plus irrégulier.

- L'interpolation linéaire lors de la construction de la table de conversion des couleurs crée aussi un lissage de l'image. Il peut être intéressant de créer des discontinuités dans cette table pour obtenir, dans l'image, des zones de couleur différentes, bien délimitées.

- Le choix d'une fonction g(n) différente de $(2^{-h})^n$ ou d'un tirage d'une variable aléatoire non gaussienne créerait de nouveaux domaines d'investigation. Cette méthode est bien adaptée à la génération de textures naturelles assez basses fréquences. Les textures hautes fréquences sont plus difficiles à obtenir: elles nécessitent d'utiliser un nombre de niveaux de récursivité plus grand.

CHAPITRE 5

INITIALISATION DES TABLES DE COULEUR

INITIALISATION DES TABLES DE COULEUR

Nous exposons ici un problème rencontré pour visualiser les images présentées dans les chapitres précédents. La mémoire d'image utilisée dispose de trois tables de couleur, R, V et B de 4096 valeurs chacune. L'affichage d'une image suppose donc d'initialiser correctement la table de couleur.

Après un exposé bref des atouts et des problèmes de la table de couleur, nous présentons l'algorithme utilisé pour déterminer une table de couleur permettant de représenter les couleurs de l'image à afficher avec une précision suffisante.

On emploiera le terme couleur pour indiquer un triplet de valeurs RVB (Rouge, Vert, Bleu) qui produit une sensation colorée. Par abus de langage, teinte sera souvent employé avec le même sens.

1. DESCRIPTION MATERIELLE

Une mémoire d'image est une mémoire RAM rapide à double accès, structurée en p plans de nxm points. Elle est lue au rythme du balayage vidéo par le processeur d'entretien qui communique l'information au dispositif d'affichage. Les nxm points représentent les pixels de l'image et pour chaque pixel, on peut stocker une information codée sur p bits. En général les p plans sont utilisés pour coder la couleur du pixel. On peut distinguer les mémoires 8 bits, utilisées pour représenter des niveaux de gris ou 256 couleurs fixes, des mémoires 12, 15, 18, 21, ou 24 bits qui permettent de représenter une couleur par ses trois composantes rouge, verte et bleue sur 4, 5, 6, 7 ou 8 bits. Il existe des mémoires d'image qui fournissent un nombre de plans différent des valeurs précitées mais elles sont plus rares.

Les mémoires d'image permettent donc de représenter au plus 2^p couleurs. Avec des valeurs de p faibles, exemple 8 ou 12, le nombre de couleurs représentable est tout à fait suffisant pour la production d'images en enseignement assisté par ordinateur ou en graphique d'affaire ("business graphics"). Par contre,il est insuffisant pour la synthèse d'images réalistes: ombrage, éclairement, antialiassage, textures, où les zones colorées ne sont plus unies mais utilisent des dégradés de teintes qui supposent un nombre de couleurs simultanément présentes dans l'image beaucoup plus élevé. Dans ce cas, la faible taille de la palette de couleur se traduit par un zonage de l'image (cf. figure 12). Pour pallier cet inconvénient sans augmenter énormément la taille de la mémoire d'image, les constructeurs ajoutent souvent un dispositif supplémentaire appelé table de

couleur. Les valeurs présentes dans la mémoire d'image ne sont plus des codes de couleur mais des valeurs adressant la table de couleur. Le nombre d'éléments de la table de couleur est évidemment égal à 2^p, p étant le nombre de plans de la mémoire d'image. Chaque élément de la table contient une information codée sur b bits (b>p). Cette information correspond souvent à la valeur de rouge, de vert et de bleu qui sera fournie au convertisseur numérique/analogique du dispositif d'affichage; dans ce cas, chaque composante est codée sur q bits, avec b=3q. La table de couleur dont nous disposons admet 12 bits en entrée et 3×8 en sortie. Il est donc possible de représenter 4096 couleurs choisies parmi 16 millions.

2. INTERET DES TABLES DE COULEUR

Comme nous venons de le voir le premier rôle de la table de couleur est de permettre une définition par l'utilisateur, ou le programme d'application, des couleurs qu'il souhaite manipuler sans augmentation du nombre de plans de la mémoire d'image. Mais ce n'est pas son seul intérêt, en effet, la souplesse fournie par les tables de couleur a été largement exploitée:

* Une utilisation très naturelle consiste à utiliser la table pour corriger des distorsions, ainsi la gamma correction rectifiant les défauts de linéarité des moniteurs vidéo [CAT79]. Les modifications dues à ces distorsions sont introduites directement par la table de couleur ce qui permet de s'affranchir lors des calculs de ces problèmes liés au matériel.

* Un grand nombre d'effets sont réalisés grace à l'utilisaton d'une table de couleur. Parmi ces derniers, on peut citer:

  - La désignation.

    En associant une adresse à chaque objet présent dans l'image, on obtient une relation bijective entre les objets et les adresses qui facilite la désignation.

  - L'animation.

    La modification de quelques éléments dans la table de couleur est très rapide. On peut donc en modifiant rapidement la couleur des objets, simuler leur déplacement, [SHO79].

  - Le calcul d'éclairement.

    D. Warm [WAR83] montre qu'il est possible de déplacer une source lumineuse dans une scène 3D par simple modification de la table de couleur. Ceci suppose cependant de n'utiliser que très peu de teintes dans la scène.

La présence d'une table de couleur permet donc de générer des effets qu'il serait impossible d'obtenir autrement. Cependant, la gestion de cette table peut être complexe et très contraignante pour certaines applications.

Nous étudions maintenant le problème suivant. Supposons que l'on dispose d'une image originale dont la couleur des pixels est définie sur r bits. On souhaite afficher cette image sur une mémoire d'image de p plans disposant d'une table de couleur avec p<r.

3. PROBLEMES DE GESTION DE LA TABLE DE COULEUR

On peut distinguer deux modes de gestion de la table de couleur:

* **table fixe**

  Dans ce cas, les valeurs de la table sont prédéfinies et non modifiables. La table fournit des valeurs fixes, réparties uniformément sur tout le spectre des triplets RVB représentables. L'affichage d'une image est réalisé en affectant à chaque point de couleur C de l'image originale la couleur de la table qui est visuellement la plus proche de C.

  La table de couleur fixe permet une gestion simple. Par contre, la définition des couleurs de l'image finale peut être médiocre car de nombreuses couleurs de la table de couleur sont inutilisées. L'image finale est donc représentée avec beaucoup moins de 2^p couleurs différentes.

* **table variable**

  Pour parvenir à une meilleure utilisation des 2^p couleurs de la table de couleur, il est nécessaire d'associer une table de couleur différente à chaque image.

  Les problèmes apparaissent dès que l'on réfléchit aux procédés de construction de l'image. On se rend vite compte que la gestion de la table alourdit sensiblement les traitements. Il est difficile d'insérer dans une image existante une sous-image ayant sa propre table de couleur. Ou problème inverse; après avoir supprimé certaines couleurs, il est impossible de répartir les couleurs restantes équitablement sur les 2^p valeurs disponibles car l'information nécessaire est perdue.

  On verra que la gestion de la table de couleur variable impose de connaître toutes les couleurs de l'image originale avant le traitement. Il n'est pas possible d'insérer le passage en table variable dans le processus de production de l'image.

A titre d'exemple, les photos 12 et 2 ont été réalisées dans les même conditions. L'utilisation d'une table fixe pour la photo 12 explique le zonage de l'image tandis que l'application de l'algorithme présenté dans ce chapitre fourni une image visuellement plus correcte, (cf. photo 2).

4. TRAVAUX ANTERIEURS

Heckbert [HEC82] décrit des algorithmes de quantification adaptative d'images. Ces algorithmes se décomposent en trois étapes:

- Echantillonnage de l'image originale pour en déduire des statistiques. Ceci consiste à déterminer l'histogramme de l'image initiale.

- Choix de la table de couleur. Plusieurs critères peuvent être appliqués:

  . choix des 2^p premières couleurs triées dans l'ordre des fréquences décroissantes.

  . choix de 2^p couleurs telles que chaque couleur choisie représente le même nombre de pixels de l'image originale.

- La couleur de chaque pixel de l'image originale est alors remplacée par une des 2^p couleurs de la table de couleur.

5. PRESENTATION DE LA METHODE

L'objectif de cette étude est d'écrire un algorithme permettant d'afficher convenablement, sur une mémoire d'image de p plans, une image, calculée sur r bits. Nous avons choisi une solution simple pour laquelle aucune statistique de l'image n'est réalisée. C'est à dire que la seule information prise en compte est l'existence de la couleur. Le nombre d'occurence d'une couleur ou la couleur des points voisins ne sont pas considérés.

5.1. Description générale

Comme la méthode de Heckbert, cette méthode est décomposable en trois phases:

. Etude de l'image à traiter

Le choix des couleurs à insérer dans la table de couleur suppose de connaître toutes les couleurs présentes dans l'image originale. L'étude de l'image est donc une phase préliminaire de lecture de l'image originale et d'insertion de toute nouvelle couleur dans une structure de données. Cette structure, décrite ci-dessous, est destinée à stocker l'ensemble des

couleurs de l'image originale. En fait, on opérera quelques regroupements de couleurs afin que la structure ne devienne pas trop importante. La règle étant de ne pas représenter plus de 2^p couleurs dans la structure.

. **Initialisation des tables de couleurs**

La table de couleurs est initialisée avec les couleurs présentes dans la structure de données.

. **Modification des valeurs de l'image**

A chaque pixel de l'image est affecté la couleur de la table de couleur qui est la plus proche de la couleur du pixel de l'image originale. Le terme proche sera explicité ultérieurement.

Avant d'étudier plus précisément chacune de ces trois étapes, nous allons décrire la structure de données utilisée.

5.2. Structure de données

La structure de données va permettre de mémoriser les différentes couleurs présentes dans l'image. Chaque couleur C_i de l'image originale est définie par ses trois composantes rouge, verte et bleue: R_i, V_i et B_i. Chacune des composantes est représentée par n bits (3n=r), r étant le nombre de bits nécessaires pour représenter la couleur des points de l'image:

$$C_i = R_i, V_i, B_i$$
$$R_i = r_i^{n-1}2^{n-1} + r_i^{n-2}2^{n-2} + \ldots + 2r_i^1 + r_i^0$$
$$V_i = v_i^{n-1}2^{n-1} + v_i^{n-2}2^{n-2} + \ldots + 2v_i^1 + v_i^0$$
$$B_i = b_i^{n-1}2^{n-1} + b_i^{n-2}2^{n-2} + \ldots + 2b_i^1 + b_i^0$$

On définit C_i par les n valeurs :

$$C_i^j = 4r_i^j + 2v_i^j + b_i^j \qquad j \in [0,n-1]$$

on a donc

$$0 \leq C_i^j \leq 7$$

Le calcul des C_i^j ne nécessite que des additions et des décalages. Ce sont les valeurs C_i^j qui vont être stockées dans la structure de données. Cette structure, mémorisant l'ensemble des couleurs est un arbre binaire. Cet arbre est le résultat d'une

binarisation de l'arbre suivant qui associe jusqu'à 8 fils à chaque noeud. Ces fils correspondent aux 8 valeurs de C_i^j.

exemple: n=3

$C_1 = 5\ 2\ 1 \quad R_1 = 4 \quad V_1 = 2 \quad B_1 = 5$

$C_2 = 5\ 2\ 3 \quad R_2 = 4 \quad V_2 = 3 \quad B_2 = 5$

$C_3 = 1\ 4\ 6 \quad R_3 = 3 \quad V_3 = 1 \quad B_3 = 4$

$C_4 = 1\ 5\ 7 \quad R_4 = 3 \quad V_4 = 1 \quad B_4 = 7$

$C_5 = 2\ 2\ 3 \quad R_5 = 0 \quad V_5 = 7 \quad B_5 = 1$

$C_6 = 1\ 5\ 4 \quad R_6 = 3 \quad V_6 = 0 \quad B_6 = 6$

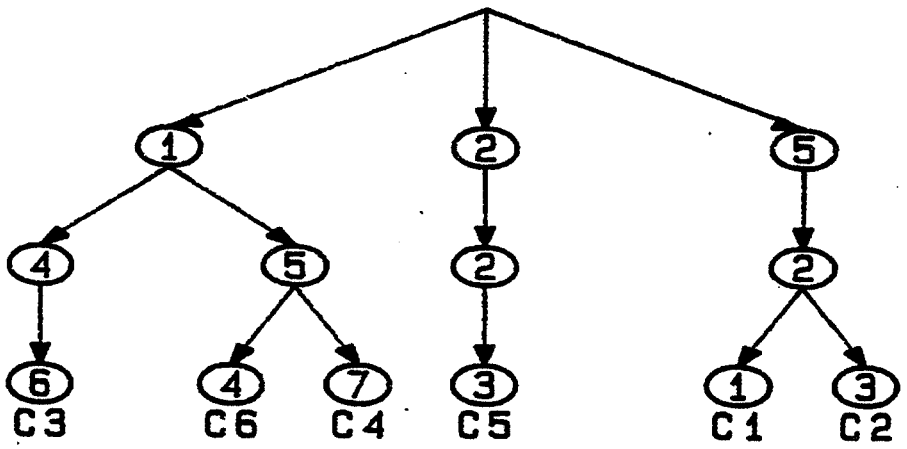


Figure 5-1 : Arbre de représentation des couleurs

Après binarisation, chaque noeud a deux champs du type pointeur; le champ fils pointe vers la liste des valeurs de niveau inférieur et le pointeur suivant chaîne les différentes valeurs d'un même niveau.

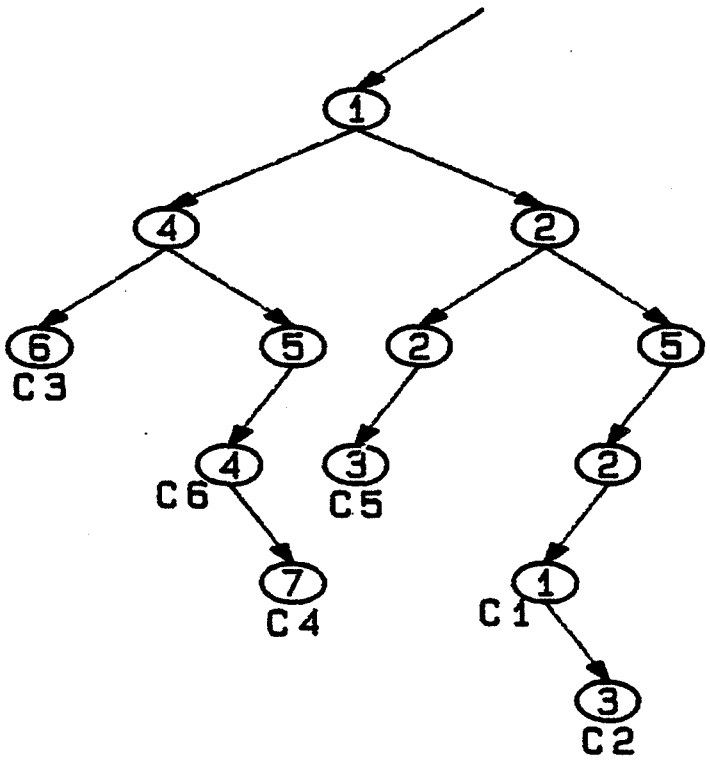


Figure 5-2 : Arbre binarisé

Description Pascal des types utilisés pour les tables de couleurs

```
t_pt_arbre = ^t_arbre;
t_arbre    = RECORD
                valeur  : integer;
                fils    : t_pt_arbre;
                suivant : t_pt_arbre;
             END;
```

On appelera feuille colorée de l'arbre tout enregistrement dont le champ fils est à nil. On distinguera différents niveaux de l'arbre; par définition, le $j^{ème}$ niveau contient les C_i^j pour tout i.

Chaque enregistrement E_i dont le champ fils est à nil représente une couleur. Soit CH_i le chemin allant de la racine à E_i; les valeurs de la couleur C_i représentée par E_i sont les valeurs des enregistrements de CH_i pour lesquelles le pointeur appartenant au chemin est du type fils. (cf. figure 5-2).

5.3. Première phase: étude de l'image à traiter

Cette phase est un balayage de l'image. La couleur de chaque point est lue puis insérée si elle n'appartient pas encore à la structure. Il est évident qu'une telle structure va très vite devenir importante. La première optimisation consiste à ne mémoriser dans la structure qu'au plus 2^p valeurs, nombre des couleurs différentes représentables. Donc, si le nombre de couleurs présentes dans la structure est supérieur à 2^p, l'insertion est suivie d'une destruction de certaines couleurs de l'arbre jusqu'à ce que la condition précédente devienne fausse.

Une solution correcte pour diminuer le nombre de couleurs présentes dans la structure consisterait à:

- rechercher dans la structure les deux couleurs C_1 et C_2 les plus proches
- déterminer la couleur C_m moyenne des deux précédentes
- supprimer C_1 et C_2 de la structure
- insérer C_m dans la structure

Ces règles supposent la définition de "plus proche" et "moyenne". Toujours pour une solution idéale, les deux couleurs les plus proches doivent être les deux couleurs qui visuellement sont le moins discernables. La moyenne de deux couleurs est la couleur qui est visuellement la plus proche des deux couleurs précédentes.

La recherche des couleurs C_1 et C_2 est assez complexe. Elle suppose

- de formaliser les critères visuels exprimés ci-dessus
- de balayer la structure pour rechercher ces deux couleurs.

Nous avons choisi une technique de destruction plus simple. Nous détruisons les feuilles colorées de l'arbre. Cette destruction des feuilles colorées revient en fait à représenter la couleur avec un bit de moins pour chaque composante, c'est à dire à tronquer la valeur. L'ordre dans lequel sont détruites les feuilles colorées est déduit d'un balayage de l'arbre en préordre, (cf. figure 5-3).

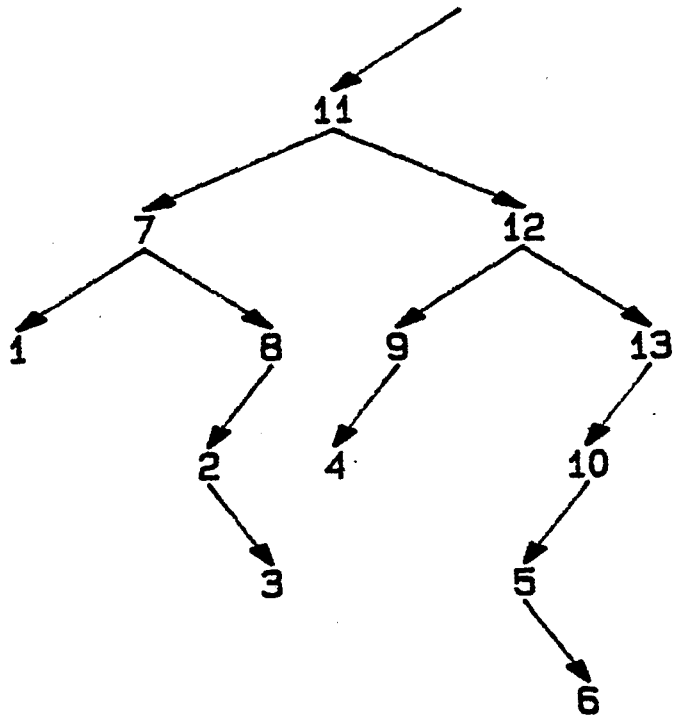


Figure 5-3 : Ordre de destruction

On constate que

- la destruction des feuilles colorées ne supprime pas nécessairement une couleur
- la distance entre deux couleurs est représentée par le maximum des différences entre chacune des composantes:

$$d(C_i, C_j) = \max(|R_i - R_j|, |V_i - V_j|, |B_i - B_j|)$$

Algorithme 5-1 : Etude de l'image à traiter

```
DEBUT
   POUR chaque point de l'image FAIRE
      SI la couleur du point n'existe pas dans la structure ALORS
         l'insérer
      TANTQUE le nombre de feuilles colorées est
               supérieur à 2^p FAIRE
         supprimer une feuille
FIN
```

Les destructions sont exécutées de façon ordonnée. Par conséquent, toute nouvelle couleur est insérée avec une précision compatible avec l'état courant de l'arbre. Dans l'exemple précédent, après les 6 premières destructions, au lieu d'insérer la couleur C_7 = 2 3 5, on insérera la couleur tronquée 2 3. Pour faciliter la destruction des feuilles colorées, il est nécessaire de mémoriser un pointeur par niveau pour rendre possible le balayage de l'arbre en préordre.

5.4. Deuxième phase: initialisation de la table de couleurs

La liste des couleurs qui doivent être présentes dans la table est directement déductible de la structure précédente. Les chemins reliant la racine à chaque feuille colorée correspondent aux valeurs qui doivent figurer dans la table de couleur. Pour la phase suivante, il est nécessaire de stocker pour chaque feuille colorée, l'adresse correspondante de la table. Le champ valeur de l'enregistrement étant du type entier, il est possible de coder cette adresse dans le champ valeur de la feuille colorée associée à la couleur.

Algorithme 5-2 : Procédure récursive d'initialisation de la table de couleurs

```
PROCEDURE ini_table(couleur:t_couleur;pointeur:t_pt_arbre;
                    niveau:integer);
BEGIN
   mise à jour de la couleur en fonction de la valeur de
    l'enregistrement pointé par pointeur;
   IF pointeur^.fils = nil THEN
   BEGIN
      rajouter couleur dans la table de couleur
      mémoriser l'adresse de la table de couleur dans la
      structure
   END
   ELSE
   BEGIN
      temp := pointeur^.fils
      WHILE temp ≠ nil DO
      BEGIN
         ini_table(couleur,temp,niveau+1);
         temp := temp^.suivant;
      END
   END
END
```

5.5. Troisième phase: modification des valeurs de l'image

Cette phase est un balayage de l'image originale. La couleur C_i de chaque pixel est recherchée dans la structure. Cette couleur risque de ne pas être représentée par q bits dans la structure car les bits de poids faible peuvent avoir été détruits. Dans ce cas, l'adresse à affecter au point se trouve dans le premier enregistrement du type feuille colorée rencontré.

```
DEBUT
  POUR chaque point de l'image originale FAIRE
  DEBUT
    rechercher l'adresse correspondant à la couleur originale
    affecter cette adresse au point de l'image finale
  FIN
FIN
```

5.6. Evaluation

En temps

Le temps mis par les phases 1 et 3 est proportionnel au nombre de points de l'image.

Le temps mis par la phase 2 est proportionnel au nombre de valeurs disponibles soit 2^p.

En place

Une borne supérieure du nombre d'éléments de la structure est égale à :

$$2^{p(3q-p+4)/3}$$

Le nombre d'éléments des (p+2)/3 premiers niveaux est inférieur à 2^{p+1}, et le nombre d'éléments des niveaux supérieurs est inférieur à 2^p donc la somme est égale à

$$2^{p+1} + 2^{p(q-(p+2)/3)} = 2^{p(2+q-(p+2)/3)}$$
$$2^{p(3q-p+4)/3}$$

ce qui pour p=12 et q=8 donne: 21 846 éléments. Le nombre de mots mémoire est donc borné par 63K.

5.7. Optimisations

Il est possible d'accélérer l'exécution de l'algorithme en tenant compte de la cohérence de l'image: c'est à dire en exploitant le fait que deux points voisins ont souvent la même couleur. Pour cela il suffit de mémoriser les dernières valeurs

étudiées; en cas d'égalité de l'une de ces valeurs avec la valeur courante, la phase de recherche est évitée.

En regardant la structure en cours de construction, on constate que les premiers niveaux, correspondants aux bits de poids fort, sont les mieux remplis. Par conséquent, une structure dynamique pour les représenter est lourde et volumineuse. On choisira donc de représenter les premiers niveaux sous forme d'un tableau. Il s'agit d'un tableau de pointeurs vers des enregistrements de type t_pt_arbre. Le premier niveau de l'arbre de l'exemple précédent peut être remplacé par un tableau de la façon suivante (cf. figure 5-4).

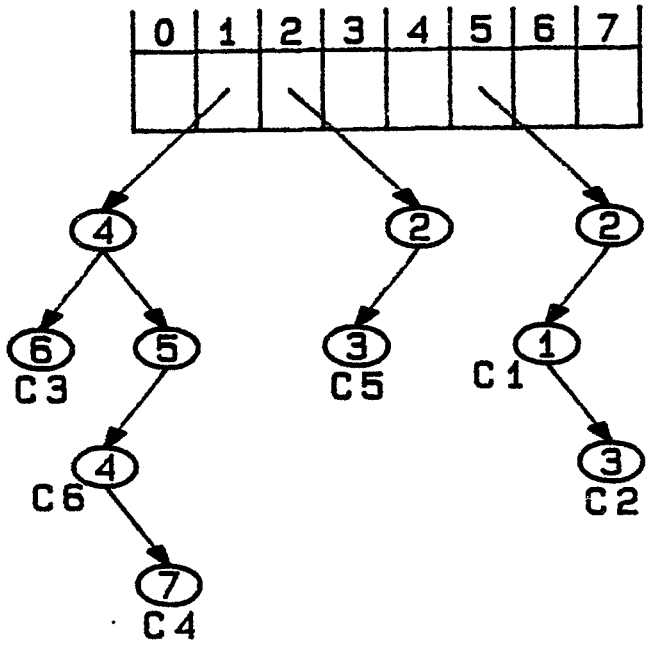


Figure 5-4 : Remplacement du premier niveau de la structure par un tableau

Le choix du nombre de niveaux à représenter sous forme de tableau dépend de l'image. Avec un peu d'expérience, on peut trouver un compromis intéressant. Il faut remarquer que si cette modification de la structure permet une économie de place, elle implique aussi un gain de temps. Pour les applications présentées dans ce rapport, nous avons représenté les quatre premiers

niveaux par un tableau de 4096 valeurs.

On constate qu'il est dans certains cas illusoire d'étudier les q bits d'une couleur si q est grand. On s'efforcera donc de ne pas étudier trop de bits inutilement, ce qui influera fortement sur la place mémoire occupée et le temps d'exécution.

6. CONCLUSION

L'algorithme que nous venons d'exposer est très simple; il ne fait appel à aucune statistique de l'image ce qui permet de minimiser les calculs à exécuter. Les résultats obtenus avec cette méthode paraissent suffisants compte tenu du matériel dont nous disposons: 4096 couleurs disponibles. On remarque cependant qu'on commence à apercevoir du zonage dans certaines régions, par conséquent, la visualisation des même images sur un matériel offrant moins de couleurs nécessiterait probablement d'avoir recours à des critères plus complexes de sélection des couleurs à insérer dans la table de couleurs.

ANNEXE

ANNEXE

Les travaux présentés dans ce rapport sont réalisés sur le matériel informatique de l'Ecole des Mines de Saint-Etienne.

* l'ordinateur hôte est un HP9000 multiutilisateur avec:

  . 1 processeur 32 bits

  . 1,5 méga octects de mémoire centrale

  . 1 disque de 120 méga octets

  Tous les logiciels sont développés en Pascal sous HP-UX, système d'exploitation de la famille UNIX[1].

* la mémoire d'image est un LEXIDATA solidview[2]. Ces caractéristiques principales sont:

  . une résolution de 640 x 512 points,

  . une table de couleur de 12 bits en entrée et 3x8 en sortie soit 4096 couleurs représentables,

  . 12 plans mémoire adressants la table de couleur,

  . 12 plans mémoire de profondeur,

  . 4 plans réservés à la superposition (overlay),

  . diverses opérations microprogrammées.

Les problèmes de table de couleur (cf. cinquième chapitre) nous ont conduits à n'utiliser la mémoire d'image qu'en fin de processus. Toutes les images présentées sont calculées sur 24 bits dans la mémoire virtuelle du HP9000. L'image 24 bits fournie par le processus de synthèse est ensuite traitée par l'algorithme présenté au chapitre 5. Une image 12 bits et la table de couleurs associée sont alors disponibles pour l'affichage sur la mémoire d'image.

---

1 UNIX est une marque commerciale déposée par A.T. & T.

2 SOLIDVIEW est une marque commerciale déposée par LEXIDATA.

PHOTOS

Photo 1 : Vue de toute la région

Photo 2 : Insertion de prismes sur les terrains

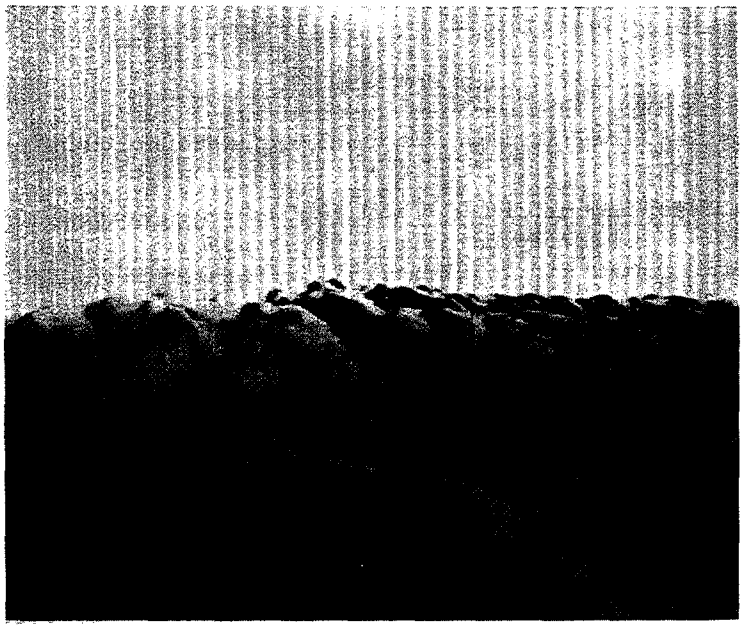


Photo 3 : Vue de dessus
Cette photo a été présentée au
"Computer Art Compétition" d'EUROGRAPHICS'84
et a obtenu le "Award of Best Technical Achievement".

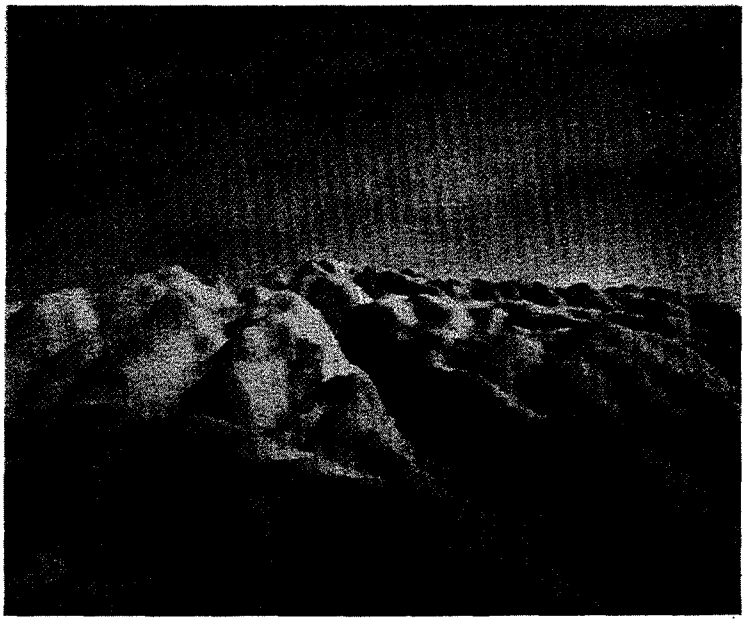


Photo 4 : Vue de dessus avec du bleu atmosphérique et un filtrage en fonction de la distance

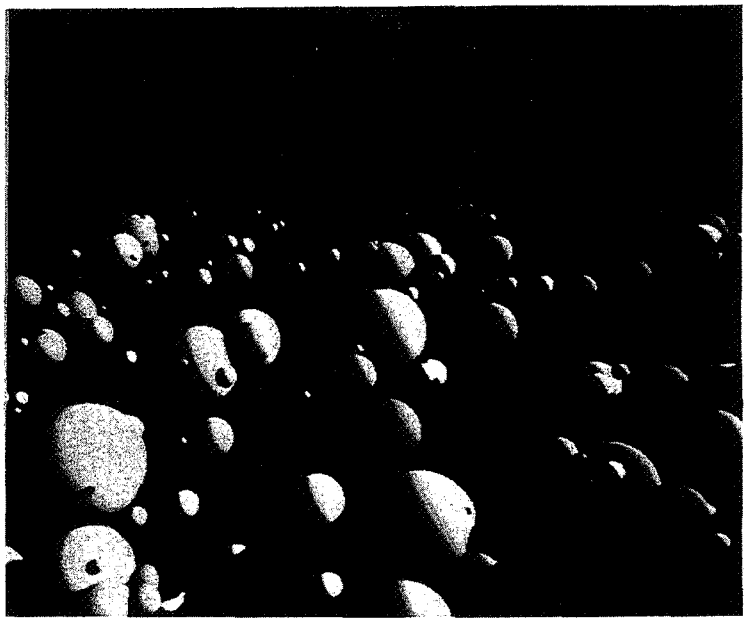


Photo 5 : Une scène de sphères

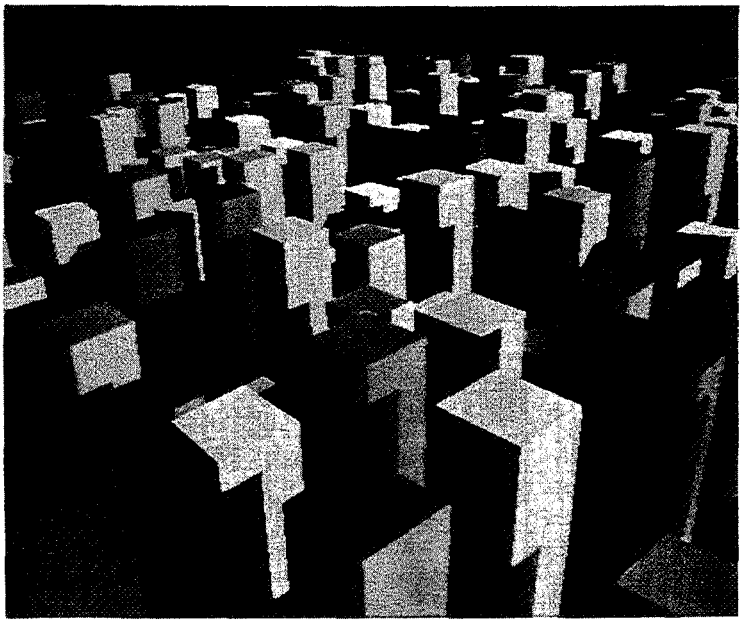


Photo 6 : Une scène de prismes

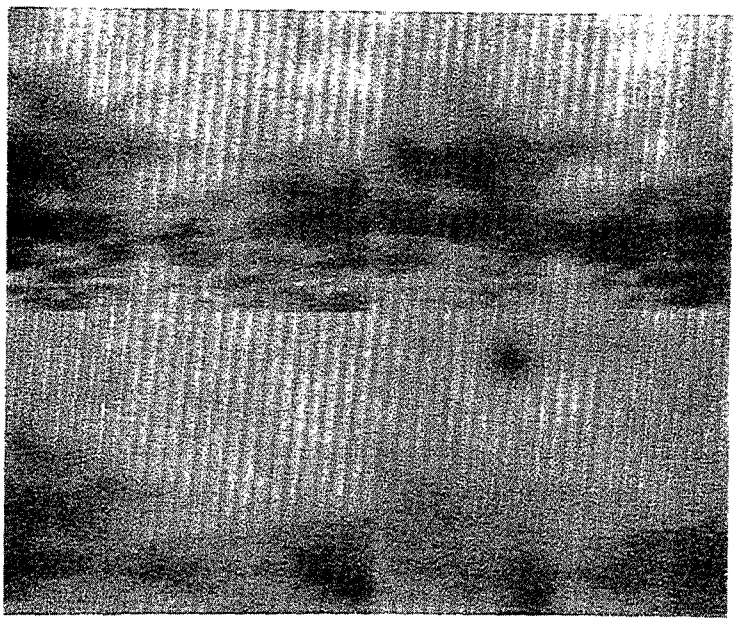


Photo 7 : Différentes valeurs de h

$$h = \begin{vmatrix} 0.1 & 0.3 \\ 0.5 & 0.7 \end{vmatrix}$$

Photo 8 : Différentes valeurs de niveaux de récursivité

$$nh;nv = \begin{vmatrix} 4;7 & 4;5 \\ 5;5 & 6;7 \end{vmatrix}$$

Photo 9 : Différentes couleurs

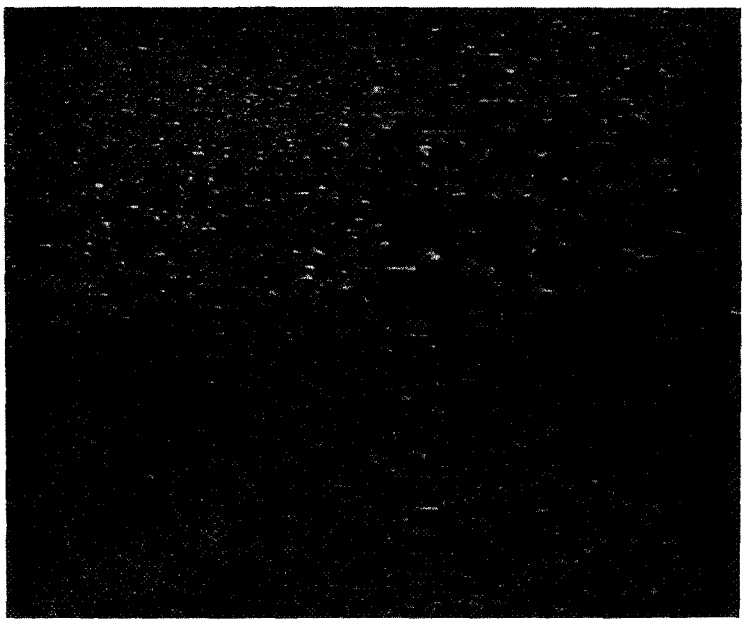


Photo 10 : h fonction de n
(nmax=nh=nv, h=0 si 0≤n≤ni, h=0.3 si ni<n≤nmax
nmax;ni = |8;7;perspective 7;6;perspective|
|8;7 8;6 |)

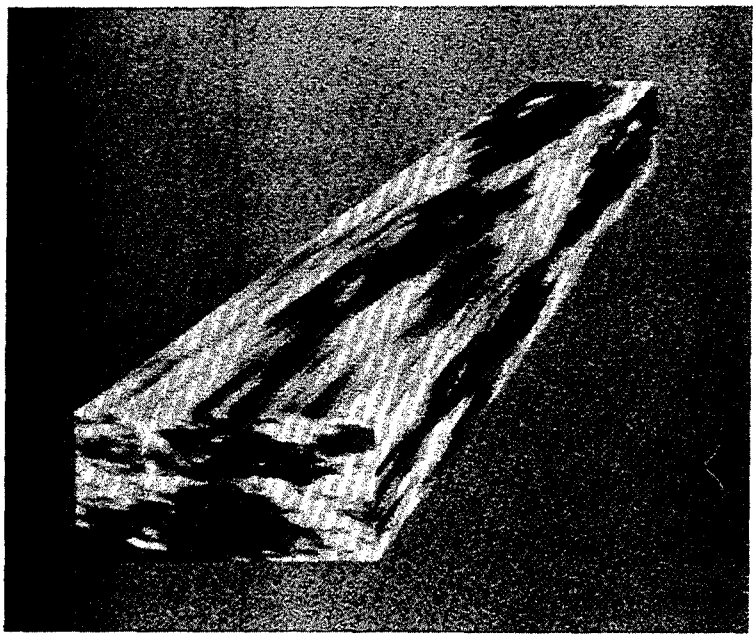


Photo 11 : Exemple de texture générée

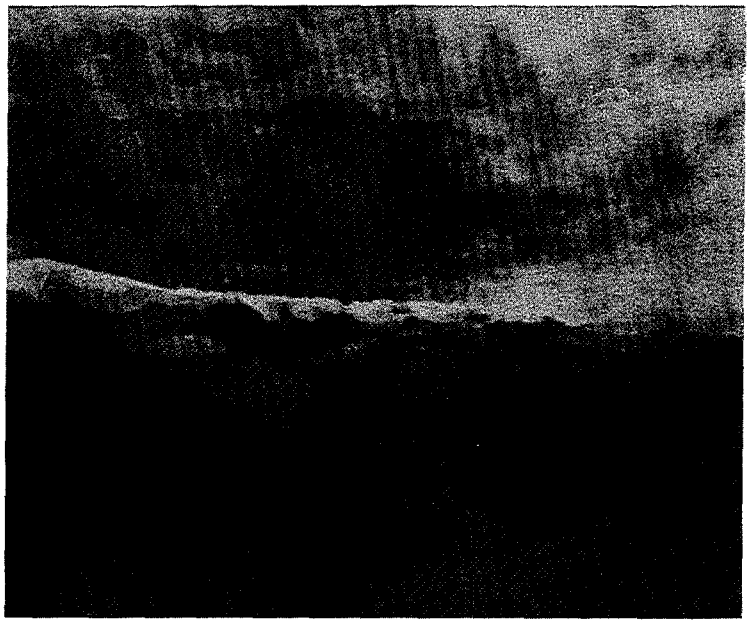


Photo 12 : Vue de la vallée avec une table de couleur fixe

REFERENCES

REFERENCES

[ACM82] "Comment on Computer Rendering of Fractal Stochastic Models", letters by B.B. MANDELBROT and A. FOURNIER, A. FUSSEL, L. CARPENTER, CACM, 25-8, Aug. 1982.

[ALL83] G.ALLAIN, "Images de Synthèse dans les Simulateurs de vol", Bulletin INRIA, 88, 1983.

[AND82] D.P. ANDERSON, "Hidden Line Elimination in Projected Grid Surfaces", ACM Trans. on Graph., 1-4, Oct. 1982.

[AON84] M. AONO, T.L. KUNII, "Botanical Tree Image Generation", IEEE CG&A, 4-6, Jun. 1984.

[APP68] A. APPEL, "Some Techniques for Shading Machine Renderings of Solids", SJCC, 32, 1968.

[BER77] J. BERTIN, La Graphique et le Traitement Graphique de l'Information, Flammarion, Nouvelle Bibliothèque Scientifique, Paris, 1979.

[BER79] M. BERGER, Géométrie 1/Action de Groupes, Espaces Affines et Projectifs, Cedic/Fernand Nathan, 1979.

[BLI78] J.F. BLINN, "Simulation of Wrinkled Surfaces", Comput. Graph., 12-3, Aug. 1978.

[BOU84] CH. BOUVILLE, R. BRUSQ, J.L. DUBOIS, I. MARCHAL "Synthèse d'Images par Lancer de Rayon: Algorithmes et Architecture", Premier Colloque Image, CESTA-GRETSI, Biarritz, Mai 1984.

[BRO84] W.F. BRONSVOORT, J.J. VAN WIJK, F.W. JANSEN, "Two Methods for Improving the Efficiency of Ray Casting in Solid Modeling", CAD, 16-1, Jan. 1984.

[CAT78] E. CATMULL, J. CLARK,
"Recursively Generated B-Spline Surfaces on Arbitrary Topological Meshes",
CAD, 10-6, Nov. 1978.

[CAT79] E. CATMULL,
"A Tutorial on Compensation Tables",
Comput. Graph., 13-2, Aug. 1979.

[CHR83] A. CHRISTENSEN,
"Uniform Grids from Raster Data",
IEEE CG&A, 3-6, Sep. 1983.

[CLA76] J.H. CLARK,
"Hierarchical Geometric Models for Visible Surface Algorithms",
CACM, 19-10, Oct. 1976.

[COQ84a] S. COQUILLART, M. GANGNET,
"Shaded Display of Digital Maps",
IEEE CG&A, 4-7, Jul. 1984.

[COQ84b] S. COQUILLART,
"Displaying Random Fields",
Admis pour publication dans Computer Graphics Forum.

[CSU83] C.A. CSURI, J.F. BLINN, J. GOMEZ, N.L. MAX, W.T. REEVES,
panel: "The Simulation of Natural Phenomena",
Comput. Graph.,17-3, Jul. 1983.

[DUN78] W. DUNGAN, A. STENGER, G. SUTTY,
"Texture Tile Considerations for Raster Graphics",
Comput. Graph., 12-3, Aug. 1978.

[DUN79] W. DUNGAN,
"A Terrain and Cloud Computer Image Generation Model",
Comput. Graph., 13-2, Aug. 1979.

[FIS80] B. FISHMAN, B.J. SCHACHTER,
"Computer Display of Height Fields",
Comput. and Graph., 5, 1980.

[FOL82] J.D. FOLEY, A. VAN DAM,
Fundamentals of Interactive Computer Graphics,
Addison Wesley, 1982.

[FOU82] A. FOURNIER, D. FUSSEL, L. CARPENTER,
"Computer Rendering of Stochastic Models",
CACM, 25-6, Jun. 1982.

[FUC79] H. FUCHS,
"Predetermining Visibility Priority in 3D Scenes",
Comput. Graph., 13-2, Aug. 79.

[GAG83] A. GAGALOWICZ,
Vers un Modèle de Textures,
Thèse d'état, PARIS VI, 1983.

[GAN83] M. GANGNET,
"Depth Sorting by Windowing",
Proc. EUROGRAPHICS'83 North-Holland, 1983.

[GAN84] M. GANGNET, D. GHAZANFARPOUR,
"Comparaison de Techniques de Mise en Perspectives de Textures Planes",
Premier Colloque Image, CESTA-GRETSI, Biarritz, Mai 1984.

[GHE68] A. GHEORGHIU, V. DRAGOMIR,
La Représentation des Structures Constructives,
Eyrolles, Paris, 1968.

[GOL71] E. GOLDSTEIN, R. NAGLE,
"3D Visual Simulation",
Simulation, 16, Jan. 1971.

[GOU71] H. GOURAUD,
"Computer Display of Curved Surfaces",
Ph. D. Thesis, University of Utah, Jun. 1971.

[HAL83] R.A. HALL, D.P. GREENBERG,
"A Testbed for Realistic Image Synthesis",
IEEE CG&A, 1983.

[HAN83] P. HANRAHAN,
"Ray Tracing Algebraic Surfaces",
Comput. Graph., 17-3, Jul. 1983.

[HAR79] R.M. HARALICK,
"Statistical and Structural Approaches to Texture",
Proc. IEEE, 67-5, May 1979.

[HEC82] P. HECKBERT,
"Color Image Quantization for Frame Buffer Display"
Comput. Graph., 16-3 Jul. 1982.

[JAN83] F.W.JANSEN, J.J. VAN WIJK,
"Fast Previewing Techniques in Raster Graphics",
Proc. EUROGRAPHICS'83 North-Holland, 1983.

[JON71] C.B. JONES,
"A New Approach to the 'Hidden Line'Problem",
The Computer Journal, 14-3, Aug. 1971.

[KAJ82] J.T. KAJIYA,
"Ray Tracing Parametric Patches",
Comput. Graph., 16-3, Jul. 1982.

[KAJ83a] J.T. KAJIYA,
"New Techniques for Ray Tracing Procedurally Defined Objects",
Comput. Graph., 17-3, Jul. 1983.

[KAJ83b] J.T. KAJIYA,
SIGGRAPH83 Tutorial on Ray Tracing.

[LAN80] J. LANE, L. CARPENTER, J. BLINN, T. WHITTED,
"Scan Line Methods for Displaying Parametrically Defined Surfaces",
CACM, 23-1, Jan. 1980.

[LU 79] S.Y. LU, K.S. FU,
"Stochastic Tree Grammar Inference for Texture Synthesis and Discrimination",
CGIP, 9-3, 1979.

[MAN75] B.B. MANDELBROT,
Les Objets Fractals: Forme, Hasard et Dimension,
Flammarion, 1975.

[MAN82] B.B. MANDELBROT,
The Fractal Geometry of Nature,
Freeman, 1982.

[MAR79] F. MARTINEZ,
"An Approach to the Modelling and Display of Landscapes",
Proc. EUROGRAPHICS'79, Oct. 1979.

[MAR82] F. MARTINEZ,
Vers une Approche Systématique de la Synthèse d'Image, Aspects Logiciel et Matériel,
Thèse d'état, INPG Grenoble, Nov 1982.

[MAS82] D. MASSALOUX,
Modèle Stochastique de Synthèse de Textures,
Thèse Docteur-Ingénieur, ENST, 1982.

[MAX81] N.L. MAX,
"Vectorized Procedural Models for Natural Terrain: Waves and Islands in the Sunset",
Comput. Graph., 15-3, Aug. 1981.

[MIC84] D. MICHELUCCI, M. GANGNET,
"Saisie de Plans à Partir de Tracés à Main-Levée",
Proc. MICAD'84, Mar. 1984.

[MOR76] P. MORVAN, M. LUCAS,
Images et Ordinateur, Introduction à l'Infographie Interactive
Larousse, Série Informatique, 1976.

[NEW79] W.M. NEWMAN, R.F. SPROUL,
Principles of Interactive Computer Graphics,
McGraw-Hill, 1979.

[NOR82] A. NORTON, A.P. ROCKWOOD, P.T. SKOLMOSKI,
"Clamping : A Method of Antialiasing Textured Surfaces by Bandwidth Limiting in Object Space",
Comput. Graph., 16-3, Jul. 1982.

[REE83] W.T. REEVES,
"Particule Systems - A Technique for Modeling a Class of Fuzzy Objects",
ACM Trans. on Graph., 2-2, Apr. 1983.

[REQ80] A. REQUICHA,
"Representations of Rigid Solids: Theory, Methods and Systems"
Comput. Surveys, 12-4, Dec. 1980.

[RIB84] M. RIBEYRON,
"Synthèse d'éléments de végétation",
Rapport de stage, EMSE, 1984.

[RUB80] S.M. RUBIN, T. WHITTED,
"A 3-Dimensionnal Representation for Fast Rendering of Complex Scenes",
Comput. Graph., 14-3, Jul. 1980.

[ROT82] S.D. ROTH,
"Ray Casting for Modeling Solids",
CGIP, 18-2, Feb. 1982.

[SCH80] B.J. SCHACHTER,
"Long Crested Waves Models",
CGIP, 12, 1980.

[SCH81] B.J. SCHACHTER,
"Computer Image Generation for Flight Simulation"
IEEE CG&A, 1-10, Oct. 1981.

[SCH83] B.J. SCHACHTER,
Computer Image Generation,
John Wiley&Sons, 1983.

[SHO79] R.G. SHOUP,
"Color Table Animation",
Comput. Graph. 13-2, Aug. 1979.

[SON83] MA SONGDE,
Synthèse de Textures,
Thèse de 3eme cycle, PARIS VI, 1983.

[SUT74] I. SUTHERLAND, R. SPROULL, R. SCHUMACHER,
"A Characterization of Ten Hidden Surface Elimination Algorithms",
ACM Computing Surveys, 6-1, Mar. 1974.

[WAR83] D.R. WARN,
"Lighting Controls for Synthetic Images",
Comput. Graph. 17-3 Jul. 1983.

[WHI80] T.J. WHITTED,
"An Improved Illumination Model for Shaded Display",
CACM, 23-6, Jun. 1980.

[WHI82] T.J. WHITTED, D.M. WEIMER,
"A Software Testbed for the Development of 3D Raster Graphics Systèms",
ACM Trans. on Graph., 1-1, Jan. 1982.

[WIL72] H. WILLIAMSON,
"Algorithm 420: Hidden-Line Plotting Program",
CACM, 15-2, Feb. 1972,

[WIL83] L. WILLIAMS,
"Pyramidal Parametrics",
Comput. Graph. 17-3, Jul. 1983.

[WOO82] J.R. WOODWARK, K.M. QUINLAN,
"Reducing the Effect of Complexity on Volume Model Evaluation",
CAD, 14-2 Mar. 1982.

[WRI73] T.J. WRIGHT,
"A Two-Space Solution to the Hidden Line Problem for Plotting Functions of Two Variables",
IEEE Trans. Comput. 22-1, Jan. 1973.

[ZUC76] S.W. ZUCKER,
"Towards a Model of Texture",
CGIP, 5-2 1976.

AUTORISATION de SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 avril 1974,

VU les rapports de présentation de Messieurs GANGNET et MARTINEZ,

Mademoiselle Sabine COQUILLART

est autorisée à présenter une thèse en soutenance pour l'obtention du titre de DOCTEUR de TROISIEME CYCLE, spécialité "Informatique".

Fait à Saint Etienne, le 14 novembre 1984

Le Président de l'I.N.P.-G

D. BLOCH
Président
de l'Institut National Polytechnique
de Grenoble

P.O. le Vice-Président.

Le Président de l'EMSE

M. MERMET
INGÉNIEUR GÉNÉRAL DES MINES
DIRECTEUR
ÉCOLE NATIONALE SUPÉRIEURE DES
MINES DE SAINT-ETIENNE

REPRESENTATION DE PAYSAGES ET TRACE DE RAYON

Sabine COQUILLART

n° d'ordre: 43 IS

Mots-clés:
Synthèse d'Images
Représentation de Paysages
Tracé de Rayon
Génération de Textures
Manipulation de Tables de Couleur

RESUME

La synthèse d'images de paysages est l'axe directeur de cette thèse. Ce thème permet d'aborder différents problèmes ouverts en synthèse d'image tri-dimensionnelle.

Plusieurs techniques d'optimisation de la méthode du tracé de rayon sont exposées à travers deux applications: la visualisation de surfaces topographiques définies sur une grille rectangulaire et la représentation de scènes composées d'objets variés.

Pour accroître le réalisme des images générées, un algorithme de synthèse de textures adapté à la représentation de textures naturelles et plus particulièrement de ciels nuageux est présenté.

Enfin une discussion des problèmes liés à l'utilisation d'une table de couleur est fournie accompagnée d'un exposé de la solution adoptée pour produire les images présentées dans ce rapport.