Etude et réalisation d'algorithmes pour la visualisation de scènes composées de facettes planes

Philippe Boulle

▶ **To cite this version:**

Philippe Boulle. Etude et réalisation d'algorithmes pour la visualisation de scènes composées de facettes planes. Modélisation et simulation. Institut National Polytechnique de Grenoble - INPG, 1980. Français. NNT : . tel-00293758

HAL Id: tel-00293758

https://theses.hal.science/tel-00293758

Submitted on 7 Jul 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THESE

présentée à

l'Institut National Polytechnique de Grenoble

pour obtenir le grade de
DOCTEUR INGENIEUR

par

Philippe BOULLE

ETUDE ET REALISATION D'ALGORITHMES POUR LA VISUALISATION DE SCENES COMPOSEES DE FACETTES PLANES.

Thèse soutenue le 9 septembre 1980 devant la Commission d'Examen :

Monsieur	G. VEILLON	Président
Messieurs	R.A. GUEDJ	Examinateurs
	M. LUCAS	
	J. MERMET	

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1979-1980

Président : M. Philippe TRAYNARD

Vice-Présidents : M. Georges LESPINARD
M. René PAUTHENET

PROFESSEURS DES UNIVERSITES

MM.	ANCEAU François	Informatique fondamentale et appliquée
	BENOIT Jean	Radioélectricité
	BESSON Jean	Chimie Minérale
	BLIMAN Samuel	Electronique
	BLOCH Daniel	Physique du Solide - Cristallographie
	BOIS Philippe	Mécanique
	BONNETAIN Lucien	Génie Chimique
	BONNIER Etienne	Métallurgie
	BOUVARD Maurice	Génie Mécanique
	BRISSONNEAU Pierre	Physique des Matériaux
	BUYLE-BODIN Maurice	Electronique
	CHARTIER Germain	Electronique
	CHERADAME Hervé	Chimie Physique Macromoléculaires
Mme	CHERUY Arlette	Automatique
MM.	CHIAVERINA Jean	Biologie, Biochimie, Agronomie
	COHEN Joseph	Electronique
	COUMES André	Electronique
	DURAND Francis	Métallurgie
	DURAND Jean-Louis	Physique Nucléaire et Corpusculaire
	FELICI Noël	Electrotechnique
	FOULARD Claude	Automatique
	GUYOT Pierre	Métallurgie Physique
	IVANES Marcel	Electrotechnique
	JOUBERT Jean-Claude	Physique du Solide - Cristallographie
	LACOUME Jean-Louis	Géographie - Traitement du Signal
	LANCIA Roland	Electronique - Automatique
	LESIEUR Marcel	Mécanique
	LESPINARD Georges	Mécanique
	LONGEQUEUE Jean-Pierre	Physique Nucléaire Corpusculaire
	MOREAU René	Mécanique
	MORET Roger	Physique Nucléaire Corpusculaire
	PARIAUD Jean-Charles	Chimie - Physique
	PAUTHENET René	Physique du Solide - Cristallographie
	PERRET René	Automatique

.../...

MM.	PERRET Robert	Electrotechnique
	PIAU Jean-Michel	Mécanique
	PIERRARD Jean-Marie	Mécanique
	POLOUJADOFF Michel	Electrotechnique
	POUPOT Christian	Electronique - Automatique
	RAMEAU Jean-Jacques	Chimie
	ROBERT André	Chimie Appliquée et des matériaux
	ROBERT François	Analyse numérique
	SABONNADIERE Jean-Claude	Electrotechnique
Mme	SAUCIER Gabrielle	Informatique fondamentale et appliquée
M.	SOHM Jean-Claude	Chimie - Physique
Mme	SCHLENKER Claire	Physique du Solide - Cristallographie
MM.	TRAYNARD Philippe	Chimie - Physique
	VEILLON Gérard	Informatique fondamentale et appliquée
	ZADWORNY François	Electronique

CHERCHEURS DU C.N.R.S. (Directeur et Maître de Recherche)

M.	FRUCHART Robert	Directeur de Recherche
MM.	ANSARA Ibrahïm	Maître de Recherche
	BRONOEL Guy	Maître de Recherche
	CARRE René	Maître de Recherche
	DAVID René	Maître de Recherche
	DRIOLE Jean	Maître de Recherche
	KAMARINOS Georges	Maître de Recherche
	KLEITZ Michel	Maître de Recherche
	LANDAU Ioan-Doré	Maître de Recherche
	MERMET Jean	Maître de Recherche
	MUNIER Jacques	Maître de Recherche

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

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

MM.	ALLIBERT Michel	
	BERNARD Claude	
	CAILLET Marcel	
Mme	CHATILLON Catherine	
MM.	COULON Michel	
	HAMMOU Abdelkader	
	JOUD Jean-Charles	
	RAVAINE Denis	
	SAINFORT	C.E.N.G.

.../...

MM. SARRAZIN Pierre
SOUQUET Jean-Louis
TOUZAIN Philippe
URBAIN Georges — Laboratoire des Ultra-Réfractaires ODEILLO

E.N.S.M.E.E.

MM. BISCONDI Michel
BOOS Jean-Yves
GUILHOT Bernard
KOBILANSKI André
LALAUZE René
LANCELOT Françis
LE COZE Jean
LESBATS Pierre
SOUSTELLE Michel
THEVENOT François
THOMAS Gérard
TRAN MINH Canh
DRIVER Julian
RIEU Jean

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

MM. BOREL Joseph
CHEHIKIAN Alain
VIKTOROVITCH Pierre

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

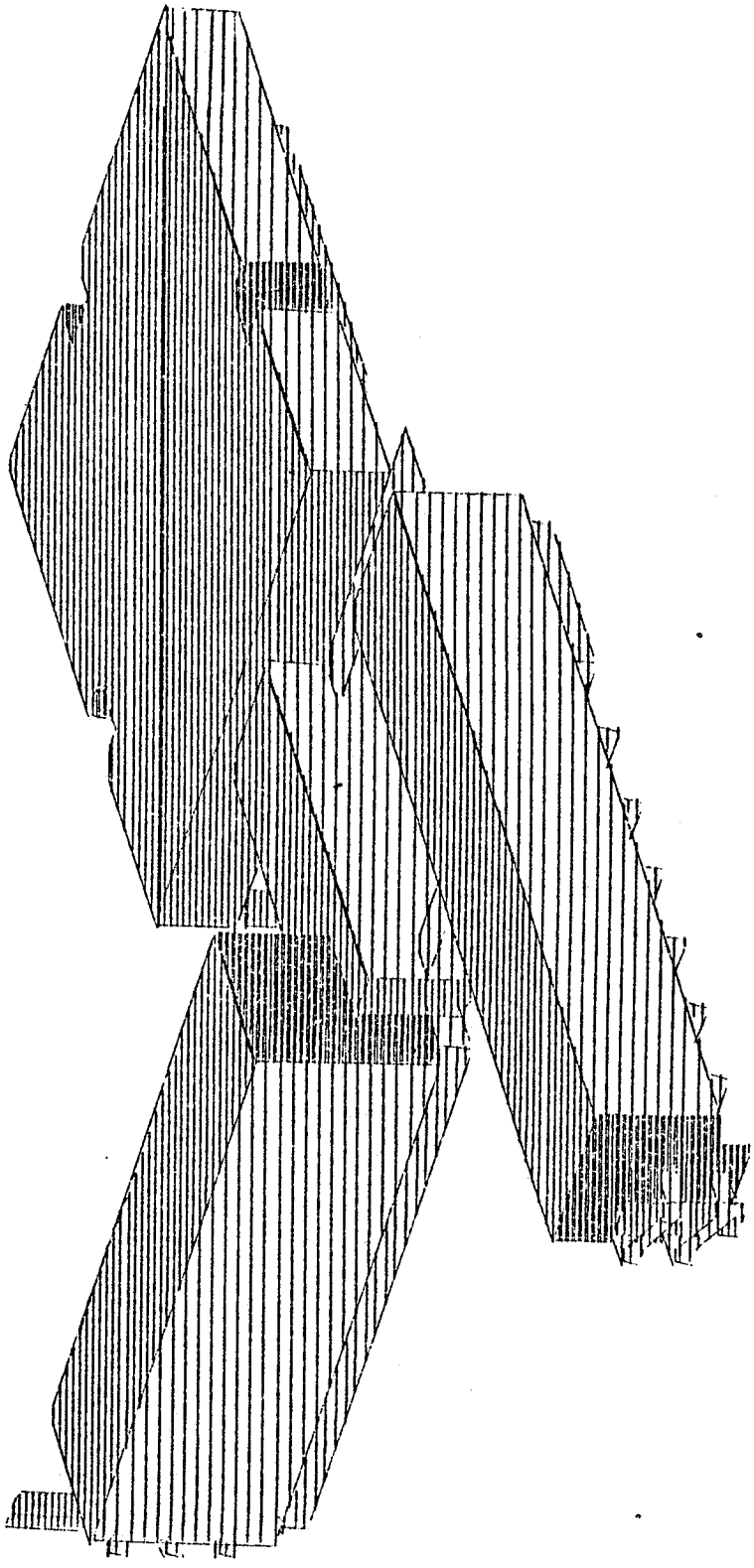
MM. BORNARD Guy
DESCHIZEAUX Pierre
GLANGEAUD François
JAUSSAUD Pierre
Mme JOURDAIN Geneviève
MM. LEJEUNE Gérard
PERARD Jacques

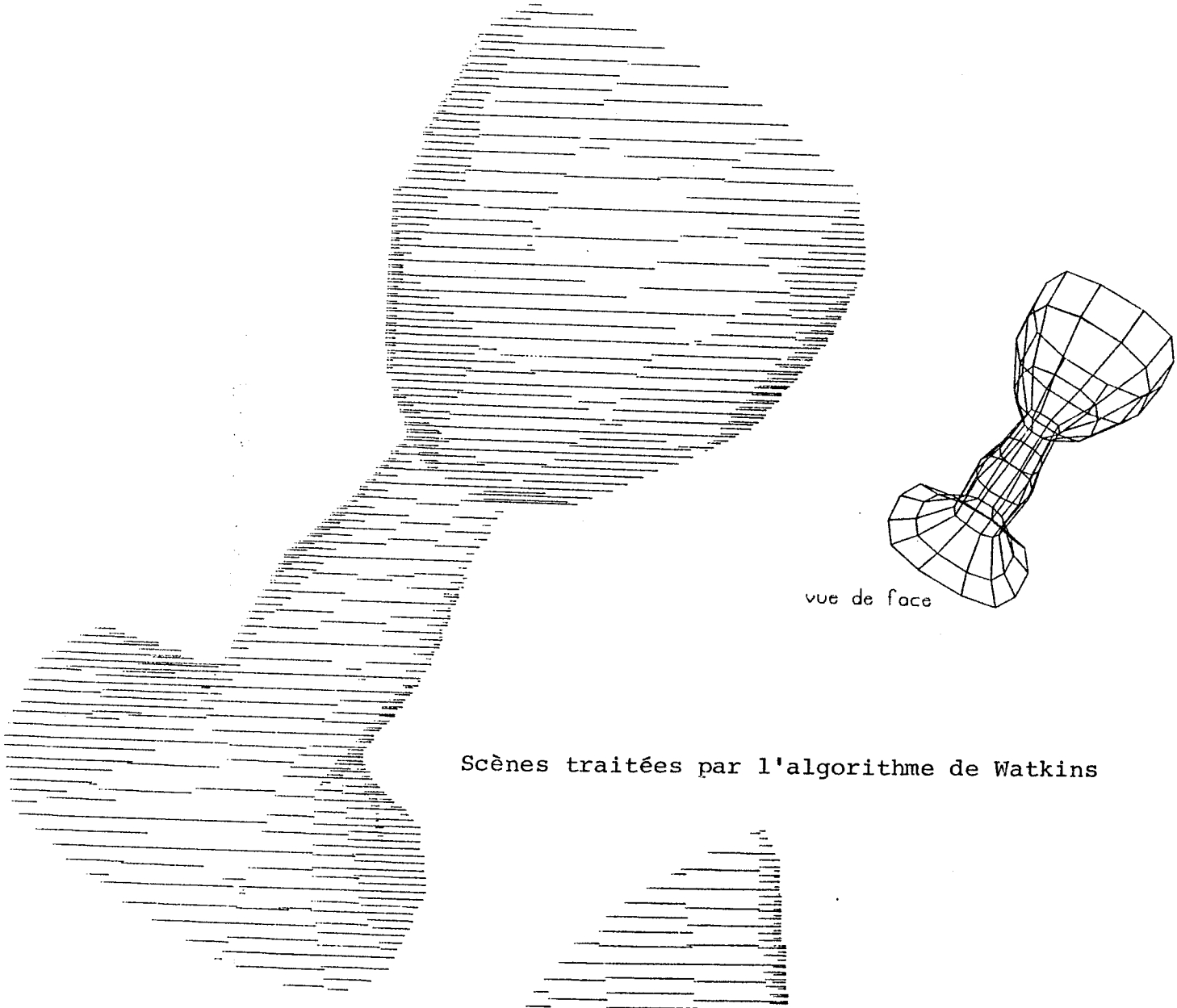
E.N.S.H.G.

M. DELHAYE Jean-Marc

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

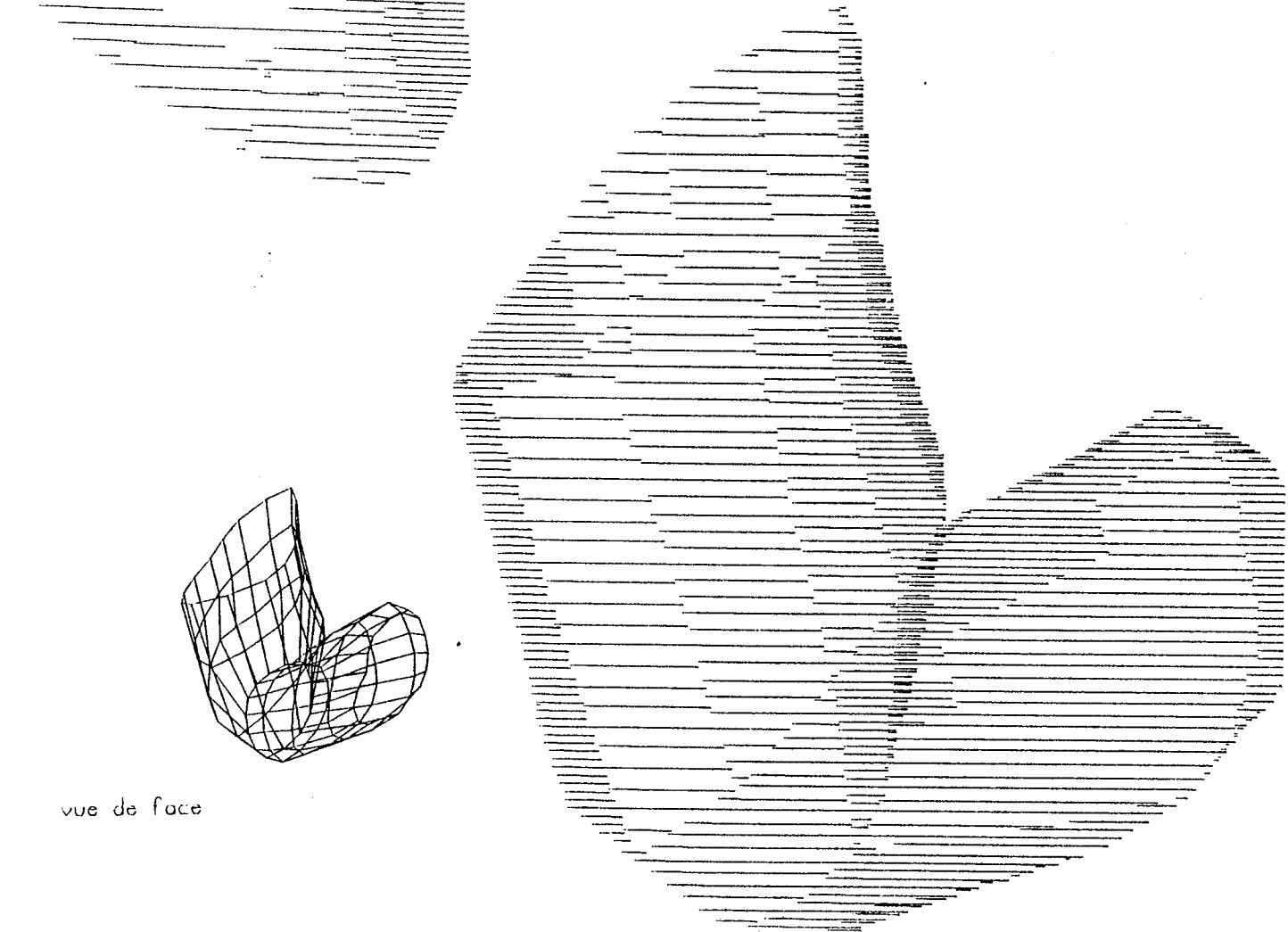
MM. COURTIN Jacques
LATOMBE Jean-Claude
LUCAS Michel
VERDILLON André





vue de face

Scènes traitées par l'algorithme de Watkins



vue de face

Ce travail a été effectué en partie sous contrat DRET n° 783 4408.

Je tiens à remercier,

Monsieur le Professeur G. VEILLON, qui a bien voulu me faire l'honneur de présider le jury de cette thèse,

Monsieur R.A. GUEDJ du Laboratoire Thomson L.C.R., qu'il dirige, pour l'intérêt qu'il a bien voulu porter à mon travail,

Monsieur M. LUCAS, Maître de Conférence à l'Université de Nantes, qui, par sa compétence et sa participation à cette étude, m'a apporté l'aide sans laquelle ce travail n'aurait pu aboutir,

Monsieur J. MERMET, Maître de Recherche au CNRS, pour l'attention qu'il a manifestée en acceptant de siéger à ce jury.

Je tiens également à remercier les membres de l'Equipe Graphique : F. FERREIRA, G. GRAS, A. LEDRU et F. MARTINEZ, dont les critiques constructives m'ont été très utiles.

Je n'oublierai pas les services de dactylographie et de reprographie pour la qualité du travail qu'ils ont effectué.

TABLE DES MATIERES

INTRODUCTION 3

CHAPITRE 1 : PRINCIPES GENERAUX 9

1.1. Classification 10

1.1.1. Nature des éléments composant la scène 10
1.1.2. Nature de la présentation 11
1.1.3. Nature des calculs 15
1.1.4. Différentes méthodes de base 17
1.1.5. Utilisation des propriétés topologiques de la scène 17
1.1.6. Conclusion 18

1.2. Etude des différents composants d'un algorithme de visibilité 18

1.2.1. Formalisation du processus 18
1.2.2. Les modèles de description 24
1.2.3. Fonctions de transition 28
1.2.4. Utilisation des tris et fonctions stratégie 29
1.2.5. Modèles de visibilité 29
1.2.6. Conclusion 31

1.3. Conclusion 31

CHAPITRE II : REALISATION 33

2.1. Présentation des algorithmes choisis 34
2.2. L'algorithme de Warnock 34

2.2.1. Description 34
2.2.2. Etude des différents éléments de l'algorithme 37
2.2.3. Etude des différents modules 42
2.2.4. Conclusion 62

2.3. L'algorithme de Watkins 64

2.3.1. Description 64
2.3.2. Etude des différents éléments de l'algorithme 65
2.3.3. Etude de visibilité 80
2.3.4. Production de dessin 102
2.3.5. Utilisation de la cohérence de la scène 105
2.3.6. Parallélisme 108
2.3.7. Conclusion 110

2.4. L'algorithme de Atherton et Weiler 113

2.4.1. Description 113
2.4.2. Etude des différents éléments de l'algorithme 113
2.4.3. Conclusion 126

2.5. Algorithme de Newell, Newell et Sancha 129

2.5.1. Description 129
2.5.2. Etude des différents éléments de l'algorithme 130
2.5.3. Conclusion 150

2.6. Conclusion 151

CHAPITRE III : LES TRAITEMENTS ELEMENTAIRES 154

3.1. Introduction 156
3.2. Etude de quelques opérations géométriques 161

3.2.1. Comparaison de segments 161
3.2.2. Comparaison point-contour 168
3.2.3. Algorithmes de découpage 182
3.2.4. Conclusion 204

3.3. Les tris 204

3.3.1. Exemples 204
3.3.2. Quelques critères de choix 205
3.3.3. Conclusion 206

3.4. Conclusion 209

CONCLUSION 211

BIBLIOGRAPHIE 215

INTRODUCTION

Ce travail se place dans le cadre d'un projet développé par l'équipe "Communication graphique" de l'E.N.S.I.M.A.G., la synthèse d'images réalistes.

Les différentes étapes de la synthèse d'images réalistes sont (cf. figure 1, [Mar 79])

- la description de la scène à présenter,
- l'analyse et la modélisation de la visibilité,
- l'analyse et la modélisation de l'aspect,
- la synthèse de l'image à partir des modèles de visibilité et d'aspect

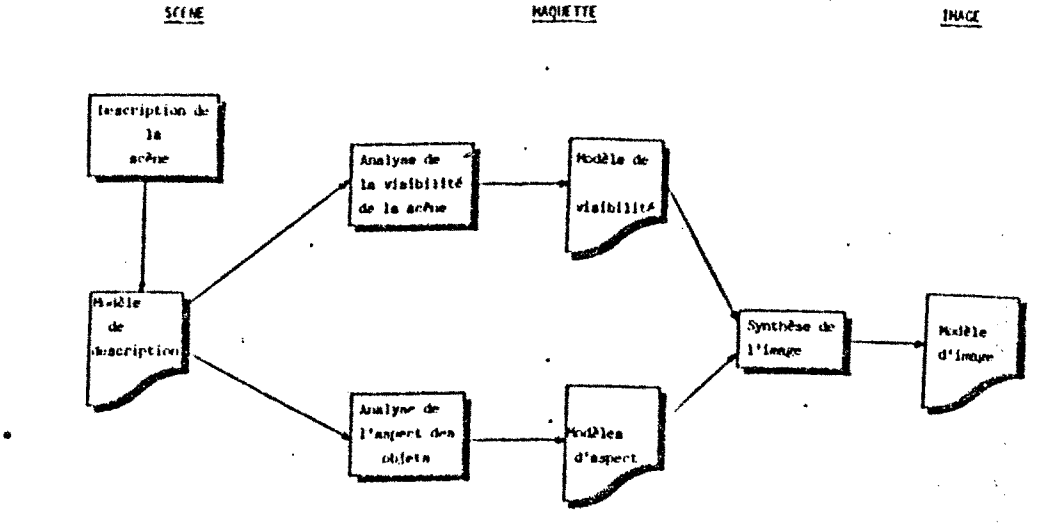


Fig. 1. Synoptique de la synthèse d'une image à partir de la description d'une scène.

La synthèse d'image est obtenue à partir de la description d'une scène tridimensionnelle qui peut revêtir les formes suivantes :

a) Objets composés de surfaces planes

L'ensemble d'objets à représenter est approché ou défini à l'aide d'un ensemble de faces polygonales planes :

- solides,

- polyèdres,

- approximation de surfaces gauches à l'aide de "facettes" planes.

C'est à ce type de représentation que nous nous sommes intéressés, en étudiant différentes solutions apportées au problème d'élimination de parties cachées.

b) Objets composés de surfaces gauches.

Chaque objet est décrit à l'aide d'un ensemble d'équations approchant différents "morceaux" de sa surface.

c) Objets "Fil de fer" (wire-frame en anglais)

Ce sont des graphes définis dans l'espace. La représentation graphique est composée de lignes représentant les arêtes joignant les sommets.

d) Représentations particulières.

1. Une scène peut être décrite à l'aide de volumes élémentaires qui ont été combinés à l'aide d'opérations.

2. Une scène peut être formée de solides particuliers :

Exemple : - solides à sections constantes

- sphères...

L'analyse et la modélisation de la visibilité d'une scène se font lors d'une seconde étape au cours de laquelle sont déterminés les objets ou portions d'objets visibles. Les nombreuses études faites à ce jour ont permis d'obtenir des algorithmes performants tant du point de vue qualité de l'image que rapidité d'exécution. Mais plusieurs problèmes importants restent posés :

- les algorithmes proposés ne traitent qu'un type donné de description et non des descriptions hybrides.

- les algorithmes proposés ont des performances variables en fonction des scènes traitées : déterminer les caractères de complexité d'une scène permettra de choisir l'algorithme le mieux adapté en ce qui concerne la qualité de l'image ou le temps d'exécution.

Le résultat d'analyse de la visibilité d'une scène sera un modèle de visibilité qui sera utilisé lors de la synthèse effective de l'image.

L'analyse et la modélisation de l'aspect d'une scène se font lors d'une troisième étape qui fournit l'aspect "réaliste" de l'image finale.

Il s'agit de "peindre" la maquette visible produite par l'analyse de visibilité c'est à dire :

- coloration intrinsèque des objets d'une scène ;
- effets de l'éclairage (luminance, reflets) ;
- notions de transparence, translucidité ;
- ombres portées.

Pour mieux faire comprendre ces deux notions, prenons deux exemples de scènes décrites à l'aide de polyèdres.

L'exemple de la figure 2 donne deux modèles de visibilité issus d'une même figure initiale.

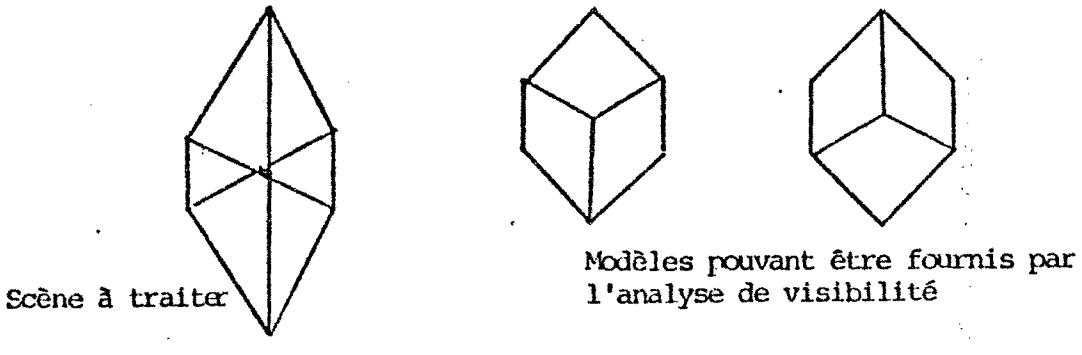


Fig. 2.

Si les modèles fournis par l'étude de visibilité sont plus facilement compréhensibles ils gardent cependant une certaine ambiguité du fait de l'abscence de sensation de profondeur qui devra être restituée par l'analyse de l'aspect de la scène puis par la synthèse de l'image. Ceci est illustré par l'exemple de la figure 3.

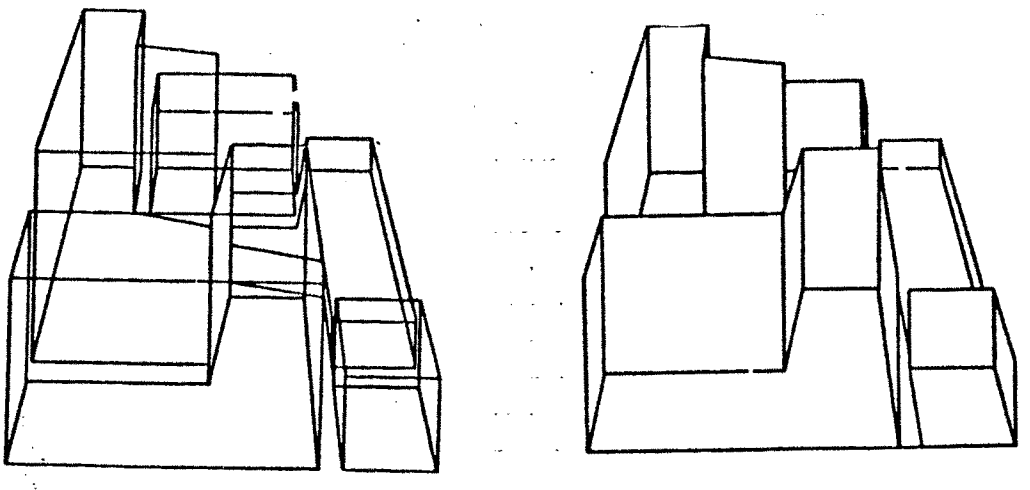
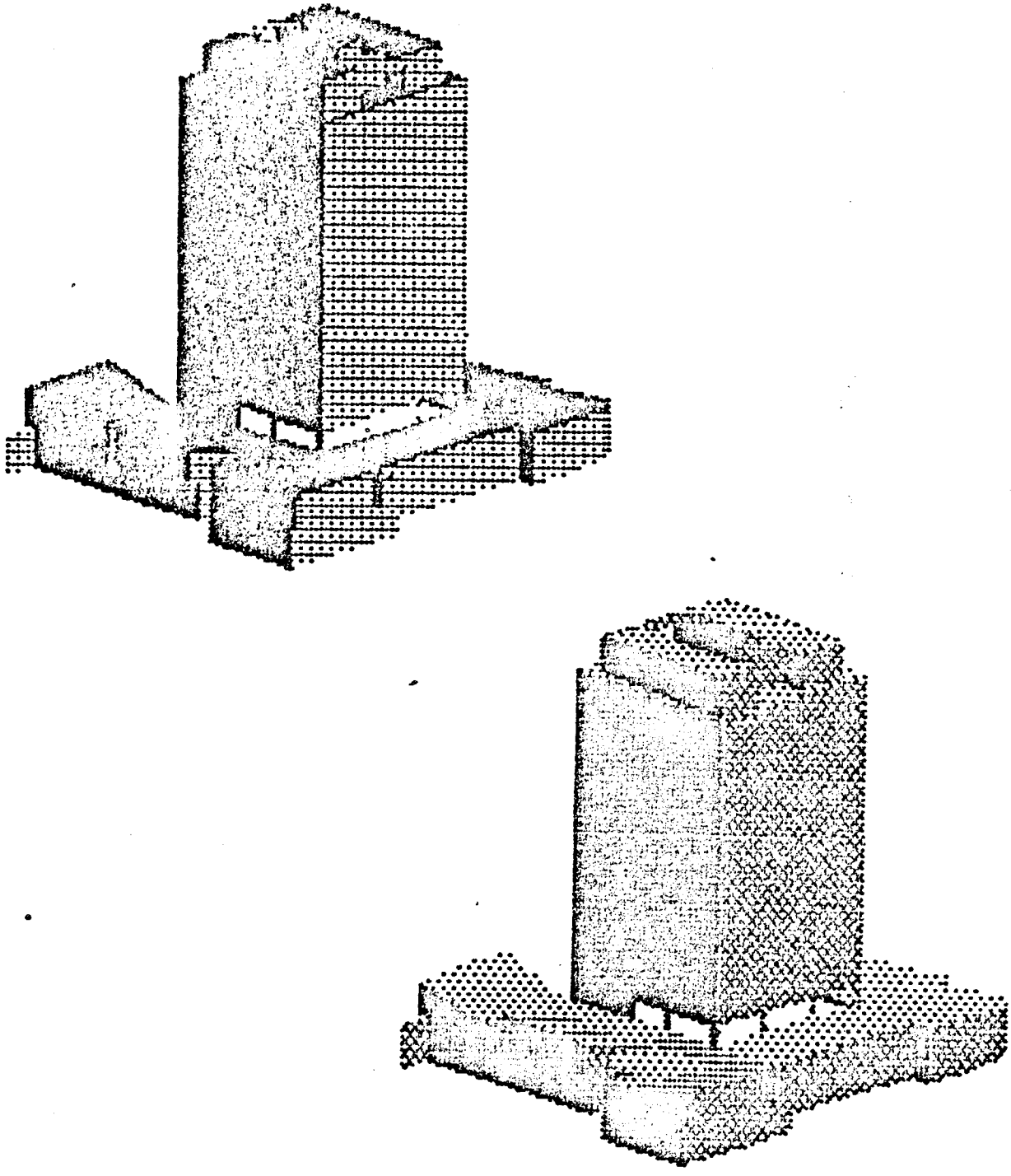


Fig. 3.

Ces deux exemples montrent en fait que l'élimination des parties cachées et l'analyse de l'aspect de la scène sont deux étapes nécessaires afin d'obtenir une image réaliste.

D'autres contextes d'utilisation des algorithmes d'étude de visibilité non exclusifs avec la synthèse d'images sont l'emploi en temps réel de ces algorithmes et dans les logiciels interactifs.

L'étude que nous avons menée mettra en valeur différents critères de choix d'algorithmes de visibilité en fonction des exigences des contextes d'utilisations.



Tour de l'I.R.M.A.

/ CHAPITRE I /

PRINCIPES GENERAUX

1. Principes généraux

1.1. Classification

Depuis 1963 date de parution de la première solution au problème de l'étude de visibilité ([Rob 63]) de nombreux travaux ont été effectués dans ce domaine, qui, par leur grand nombre, peuvent faire penser que le sujet a été épuisé. Cependant, chaque année de nouvelles solutions apparaissent. L'ensemble des travaux publiés peut être classé suivant divers critères :

- nature des éléments composant la scène ;
- nature de la présentation ;
- nature des calculs ;
- méthodes de base ;
- utilisation de propriétés topologiques.

1.1.1. Nature des éléments composant la scène

On peut considérer cinq grandes familles parmi les differentes scènes pouvant être traitées par ces algorithmes.

- scènes composées de faces planes : C'est dans ce domaine que le plus grand travail a été effectué, car tout élément géométrique peut être approché à l'aide de celles-ci et ce type de modélisation de scène est le mieux compris. Parmi toutes les solutions apportées, nous n'en citerons que quelques unes qui diffèrent par leur principe d'étude de visibilité et par les éléments produits :
[Rob 63] , [App 67] , [Sch 69] , [Gal 69] , [Lou 69] ,
[War 69] , [Wat 70] , [New 72] , [Cat 74] , [AtW 78]

- scènes composées de quadriques : La première solution (Wei66)date de 1966 et est due à R. Weiss. Les solutions suivantes diffèrent par les éléments produits (surfaces visibles, ombrage de ces surfaces) et donnent des solutions au problème de l'intersection des deux quadriques ([Mah 72], [Lev 76]).

- scènes composées de surfaces définies à l'aide de fonctions de deux variables : Ces méthodes qui fournissent toutes des dessins au trait diffèrent par leur principe de visibilité [KSG 68], [Wil 72], [Wri 73] partant du même principe proposent des solutions dont le comportement et la qualité du dessin vont en s'améliorant. [Enc 75] donne un algorithme dont le principe est similaire à celui de Warnock pour les polygones.

- scènes composées de surfaces paramétriques : La première solution est celle de Catmull ([Cat 74]). Griffiths ([Gri 75]) propose une méthode dont le principe est celui de Warnock et fournit un dessin au trait Blinn, Lane et Carpenter ([BLC 80]) proposent des méthodes dont le principe est celui de Watkins et qui permettent d'obtenir des images de grande qualité.

- scènes composées d'objets particuliers : Certains algorithmes tel [Fra 79] ne traitent que des objets ayant des propriétés géométriques particulières (objets de révolution, objets à section constante, sphères, prismes) et proposent des méthodes qui essaient d'accélérer le temps de traitement en utilisant au mieux ces propriétés.

1.1.2. Nature de la présentation

De ce point de vue on distingue deux familles d'algorithmes:

- les algorithmes de production d'images ;
- les algorithmes de production de dessins.

Algorithmes de production d'images :

Ils produisent des surfaces ou portions de surfaces visibles

Dans cette catégorie les principales solutions sont : [War 69], [Wat 70], [New 72], [Cat 74], [AtW 78].

Ils permettent d'obtenir des images d'un grand réalisme car chaque élément visible peut être affiché avec les informations d'aspect

qui lui sont attachées et de plus ils permettent de rendre les effets de transparence et de tenir compte de la présence d'une ou plusieurs sources lumineuses.

Algorithmes de production de dessins :

Ces algorithmes permettent d'afficher des contours ou portions de contour visibles. Parmi ces algorithmes on peut citer : [Rob 63], [App 67],[Gal 69], [Wil 72], [Wri 73].
Si ces algorithmes sont les premières solutions à l'étude de visibilité, ils semblent , du fait du nombre restreint d'application où ils peuvent être utilisés, perdre de leur intérêt par rapport à l'autre famille d'algorithmes.

Le tableau qui suit permet de réaliser une classification fonction de la nature de la scène et de la présentation. On peut faire les remarques suivantes :

. ce sont les scènes composées de faces planes, de surfaces paramétriques, qui sont les plus traitées. Par contre, celles composées de fonctions de deux variables semblent avoir été laissées de côté. Une des raisons est peut être que la modélisation de scènes composées de tels éléments est moins bien maitrisée.

On peut remarquer d'autre part, que si il existe une solution spécifique aux objets convexes, aucune solution n'a été trouvée pour les objets non convexes.

Parmi les différentes présentations on distingue quatre types qui sont :

- les portions de contour visibles (de 1563 à 1569) et les segments horizontaux (à partir de 1969) pour lesquels le plus grand nombre d'algorithmes existent. On constate que pour des scènes composées d'éléments de natures différentes, on peut obtenir des présentations identiques.

On remarque que chaque algorithme de visibilité est spécifique à un type de scène, ce qui fait que des scènes mixtes ne peuvent être directement traitées par un algorithme unique. Mais la remarque précédente permet de donner une solution à ce problème.

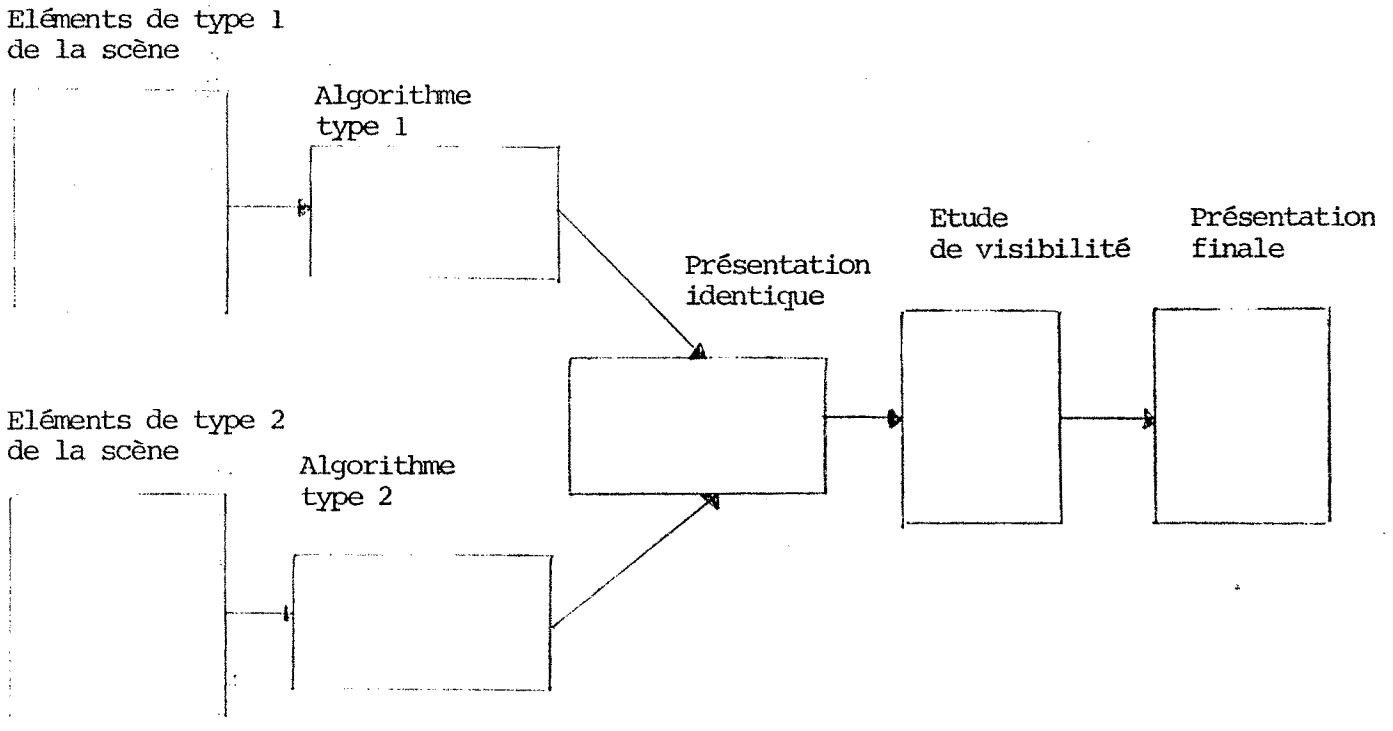


Figure 1.1. Traitement de scènes mixtes

	Portions de Contours visibles	Liste de Polygones ordonnée	Polygones visibles	Segments horizontaux	Points	Portions de polygones visibles	Segments visibles	portions de quadriques
Objets convexes composés de faces planes	Roberts (63)							
Faces Polygonales	Appel (67) Loutrel (67) Galimberti (65)	Newell (72)	Atherton et Weiler (78)	Bouknight (69)	Warnock (68)	Warnock (68)		
Objets composés de faces convexes et linéairement séparables				Schumacker (69)				
Quadriques	Weiss (66)			Mahl (72)				Mahl (72)
Surfaces Paramétriques	Griffiths (75)			Blinn, Lane Carpenter (80)	Catmull (74)		Williamson (72) Wright (73)	
Fonction de deux variables.							Encarnacao (75)	

Cette solution à ce problème n'est pas unique. Par exemple, l'algorithme de Catmull peut traiter des scènes mixtes car au cours des calculs, il se ramène à la comparaison d'éléments de même type qui est le point.

- Une liste d'éléments visibles de même format que les éléments de la scène ;

Il existe un seul algorithme de ce type, celui de Atherton et Weiler, qui traite des scènes composées de faces polygonales et qui produit des faces polygonales visibles.

- Une liste ordonnée d'éléments de même format que ceux de la scène : il existe un seul algorithme de ce type : celui de Newell pour des faces polygonales planes. Le fait qu'aucune solution n'a été fournie, pour les autres types de scène vient en grande part du fait qu'un élément peut en partie se cacher lui-même.

Cette classification montre la distinction entre algorithmes produisant un dessin au trait et ceux produisant une image. Après avoir montré que chaque algorithme ne pouvait traiter qu'un type de scène, nous avons indiqué des solutions pour des exemples mixtes.

Nous citerons en dernier lieu des méthodes qui afin d'augmenter le réalisme des images produites apportent des solutions au problème des ombres portées ([Bou 70], [App 67]) pour des scènes composées de polygones, ([Wil 78]) pour les quadriques, ([Cat 74]) pour des scènes mixtes On peut citer l'algorithme de Catmull ([Cat 78]) qui permet de supprimer des défauts de l'image mais qui est plutôt un traitement de l'image obtenu qu'une nouvelle solution à l'étude de visibilité.

1.1.3. Nature des calculs

Cette classification ([SSS 74]) des algorithmes de visibilité propose trois catégories :

Algorithmes opérant dans l'espace objet :

Ils ont pour caractéristique essentielle d'effectuer les calculs avec la précision permise par le calculateur. Les présentations obtenues ont donc une grande précision.

La majorité de ces algorithmes fournissent des dessins au trait :

Appel [App 67], Loutrel [Lou 70], Galimberti [Gam 69] Williamson [Wil 72], Roberts [Rob 63]

Seul l'algorithme de Atherton et Weiler [AtW 78] donne la possiblité d'avoir un dessin au trait ou une image.

Algorithmes opérant dans l'espace image :

Ces algorithmes tiennent compte au cours de l'étude de visibilité de l'unité d'affichage qui sera utilisée. La résolution limitée de celle-ci permet d'avoir une précision moindre au cours des calculs.

Ces algorithmes dont les buts principaux sont le temps réel et le réalisme de la présentation sont ceux de :

Romney [Rwe 69], Bouknight [Bou 70], Watkins [Wat 70]
Warnock [War 69], Catmull [Cat 74] , Wright [Wri 73]

Algorithmes à liste de priorité :

Les algorithmes de cette classe, la moins riche en éléments, travaillent en partie dans l'espace objet et en partie dans l'espace image. Le résultat du traitement d'une scène est une liste où à chaque élément a été affectée une priorité. Dans l'algorithme de Newell, elle correspond à un ordre d'affichage. Un intérêt de ces algorithmes est qu'une présentation peut être utilisée pour différentes unités d'affichage sans qu'aucune étude de visibilité soit à refaire, et ceci à l'inverse des algorithmes précédemment cités.

Dans cette famille, on peut citer l'algorithme de Schumacker et celui de Newell [NNS 72]

1.1.4. Différentes méthodes de base

En étudiant les travaux effectués, on constate que différents algorithmes utilisent le même principe de base, ce qui permet d'obtenir la classification suivante :

Méthode par parcours de contour :

Les éléments de la scène sont définis par leur contour et chacun de ceux-ci est comparé aux autres éléments [Wei 66][Gal 69] [Lou 69] utilisent ce principe.

Méthode par divisions successives de la scène :

Le principe de ces algorithmes est de diviser la scène, en projection afin de ne traiter que des zones où le nombre d'éléments à comparer est peu élevé. Les algorithmes qui utilisent cette méthode sont : [War 69] , [Gri 75], [Enc 75].

Méthode par balayage ligne par ligne :

L'utilisation de cette méthode est liée à l'apparition des terminaux graphiques à balayage télévision. Ces algorithmes tels [Wat 70], [Bou 70] ,[Mah 72], [BLC 80] découpent la scène par des plans horizontaux où l'analyse de visibilité est faite. Il est à noter que ces algorithmes permettent de fournir des images d'un grand réalisme.

Méthode par résolution point par point :

A cette classe appartient bien sûr l'algorithme "Z-buffer" ([Cat 74]) mais on peut y adjoindre l'algorithme de Newell où l'entité élémentaire traitée n'est plus le point mais une face.

1.1.5. Utilisation de propriétés topologiques de la scène

Nous décrivons ici différents travaux, qui, s'ils ne présentent pas des méthodes effectuant une étude de visibilité complète, permettent de réduire, par l'utilisation de propriétés topologiques de la scène,

le nombre d'éléments qui seront à traiter par la suite.

Le premier travail effectué dans ce domaine a été celui de Schumacker,[Sch 69], qui, à chaque face d'un même objet, affecte des priorités de façon indépendante du point de vue.

\ Jones, [Jon 70], représente la scène par un graphe dont les noeuds sont des cellules et les arcs les possiblités de vision d'une cellule à une autre. Clark [Cla 76], représente la scène à l'aide d'une structure hiérarchisée qui permet de minimiser les comparaisons entre éléments.

Levine [Lev 76] décompose la scène à l'aide de plans séparant les objets. A l'aide d'une structure représentant cette partition de l'espace, il obtient une solution rapide à l'élimination de parties cachées

1.1.6. Conclusion

De nombreuses solutions ont donc été trouvées à l'étude de visibilité pour des scènes composées d'éléments de natures diverses (polygones, quadriques, ...). On peut distinguer deux tendances principales qui sont :

- une meilleure utilisation de renseignements d'ordre topologique sur la scène et une structuration de celle-ci afin d'améliorer les performances des algorithmes d'étude de visibilité.

- l'obtention d'images d'un plus grand réalisme, ce qui implique l'utilisation d'algorithmes d'étude de visibilité adaptés au traitement ultérieur qui permet d'approcher le réalisme désiré.

1.2. Etude des différents composants d'un algorithme de visibilité

1.2.1. Formalisation du processus d'élimination de parties cachées

1.2.1.1. Le modèle de Klos :

Ce modèle a été présenté dans [Klo 75] et [Gil 78].

Un algorithme de visibilité est formé de différentes étapes, chacune de celles-ci conduit à une représentation intermédiaire de la

scène traitée pour aboutir au modèle de visibilité.

Il est possible de donner une définition formelle de tels algorithmes soit :

Définition : un algorithme de visibilité est un 5-uplet

(MD, Ψ , σ, RI, MV)

dont les éléments sont définis ci-après.

MD : Modèle de description

Le premier élément qui différencie ces algorithmes est cet ensemble qui indique quels sont les éléments traités. Les algorithmes publiés jusqu'à ce jour ne traitent qu'une classe de modèles de description alors que certaines applications contiennent différents types de modèles dont la visibilité doit être étudiée.

Dans l'étude qui suit MD est constitué d'un ensemble de faces polygonales planes auxquelles sont associées des informations telles que :

- convexité : faces convexes, concaves,
- propriété physiques : transparence, réflectance, luminance, couleur...

Ψ : Ensemble des fonctions de transition :

Ce sont les opérations géométriques utilisées afin d'obtenir le modèle de visibilité. Une opération géométrique est la comparaison de deux éléments appartenant soit au modèle de description soit à une représentation intermédiaire. L'algorithme de Watkins utilise, par exemple, l'opéraiton d'intersection entre un plan et un polygone, l'opération d'intersection entre segments.

σ : Fonction stratégie :

La fonction stratégie employée par l'algorithme de visibilité décrit l'ordre d'application des fonctions de transition.

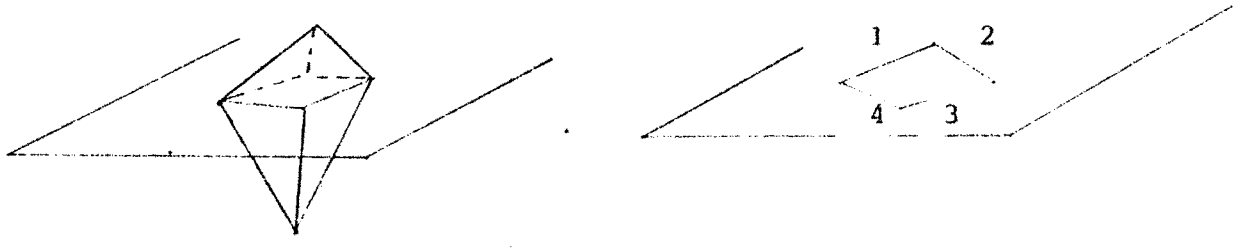
RI : Ensemble de représentations intermédiaires : RI

Soit la fonction stratégie σ :

$= (f_1, \ldots f_i, f_j, \ldots f_n)$

La fonction de transition f_i est appliquée à l'ensemble d'éléments RI_i et donne en résultat RI_j, en sachant que f_1 est appliquée à MD le modèle de description et que f_n produit le modèle de visibilité MV.

Figure 1.2. : Application à l'algorithme de Watkins. Cet algorithme etudie la visibilité d'un ensemble de segments appartenant à un plan de balayage donné :



Pour cette scène une représentation intermédiaire sera l'ensemble des segments (1-2-3-4).

MV : modèle de visibilité : les éléments de cet ensemble sont de trois types différents :

- point (Warnock),
- polygones (Atherton et Weiler)
- segments (Warnock, Watkins).

Exemple : Application à l'algorithme de Watkins (ISSS 74]).

MD : l'ensemble MD est constitué de faces polygonales planes. Une couleur est attachée à chaque face.

Ψ : PER : projection orthogonale

INT : calcule pour chaque face l'intersection avec le plan de balayage en cours d'étude.

CONT : réalisée en fait par l'étude d'une zone dans le plan où des segments sont en conflit

PRO : cf. d. 3

boucle de traitement des plans de balayage

σ : fonction stratégie PER → IS → CONT → PRO → affichage

RI : il est constitué d'un ensemble de segments provenant de l'application du test IS par rapport à un plan de balayage.

MV : modèle de visibilité formé des segments visibles pour chaque plan de balayage. A chaque segment est associée la couleur de la face à laquelle il appartient.

Cet algorithme travaille dans l'espace écran. En effet, la première fonction de transition appliquée est PER. On constate que la donnée de la fonction stratégie permet de faire une classification des différents algorithmes :

- algorithmes travaillant dans l'espace écran : PER est la première fonction appliquée ;
- algorithmes travaillant dans l'espace objet : PER appliquée avant l'affichage.

Cette schématisation des algorithmes de visibilité met en valeur différents éléments qui permettent de comparer, de différencier les solutions apportées au problème d'élimination de parties cachées :

- MD permet de comparer les algorithmes qui traitent effectivement les mêmes éléments (polyèdres convexes, polyèdres quelconques, surfaces gauches, ...).

- le résultat obtenu, l'ensemble MV, permet de choisir l'algorithme qui donne l'ensemble MV désiré pour l'application envisagée.

Cette description met aussi en valeur le fait que ces

algorithmes ont en commun l'utilisation de fonctions de base (les éléments de Ψ) et que leurs performances sont étroitement liées aux performances de ces fonctions.

Par contre, cette description ne fait pas apparaître un élément important que sont les tris appliqués soit à l'ensemble MD soit à des éléments de RI. Ces tris ont pour but d'accélérer le traitement et sont liés à l'ordre d'application des fonctions de transition c'est-à-dire à la fonction stratégie. La formalisation par un quintuplet omet donc un élément essentiel de ces algorithmes dont une meilleure connaissance serait une voie d'optimisation.

1.2.1.2. Un nouveau modèle

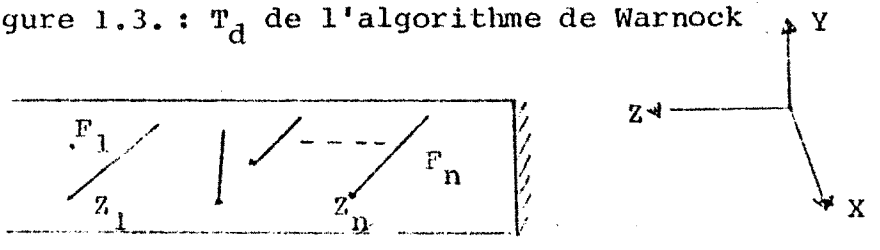
Définition : Un algorithme d'élimination de parties cachées est un 7-uplet :

$(MD, T_d, \Psi, \sigma, T_i, RI, MV)$.

T_d : tri appliqué au modèle de description :

Ce tri est le premier traitement appliqué au modèle de description MD avant toute application d'une fonction de transition. Il se caractérise aussi par le fait qu'on n'a aucune information à priori sur l'état des données à trier.

Figure 1.3. : T_d de l'algorithme de Warnock



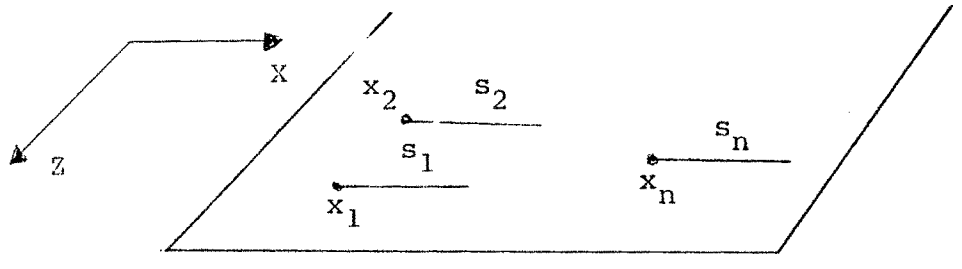
les faces $(f_1, \ldots f_n)$ sont triées par valeurs de z_i croissantes c'est-à-dire en fonction de leur éloignement par rapport à l'observateur.

T_i : tri appliqué aux représentations intermédiaires :

ce tri se distingue du précédent par deux points :

- il fait partie en général d'une boucle de traitement alors que T_d n'est appliqué qu'une fois ;
- les données à trier peuvent être considérées comme ayant déjà un certain état ordonné.

Figure 1.4. : T_i dans l'algorithme de Watkins.



Pour l'étude de la visibilité dans chaque plan de balayage les segments s_i sont triés par valeur de x_i croissante.

1.2.1.3. Conclusion

On possède donc un schéma donnant une représentation exhaustive des algorithmes de visibilité. Ceci permet :

- de décrire ces algorithmes en suivant une démarche identique pour tous ;
- d'effectuer une comparaison de ceux-ci à partir de leurs éléments constitutifs.

Cette formalisation montre aussi comment ces algorithmes de visibilité s'insèrent dans le processus de synthèse d'image.

L'élément d'entrée MD est fourni par le module description de la scène et l'élément de sortie MV sera traité avec le modèle d'aspect pour fournir le modèle d'image.

1.2.2. Les modèles de description

1.2.2.1. Informations attachées à un modèle de description

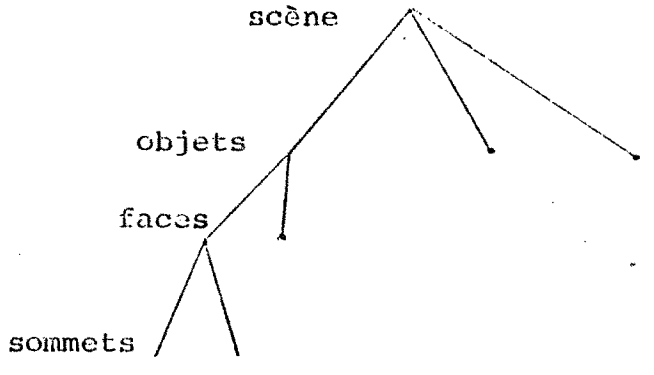
On distingue trois types d'informations qui sont :

Des informations géométriques, topologiques et d'aspect.

- Les informations géométriques sont, par exemple, les coordonnées des sommets d'une scène composée de polygones, les valeurs en y pour une fonction définie sur un maillage dans le plan xoz.

- Les informations topologiques dont l'utilisation permet une accélération du traitement des scènes pour les algorithmes d'étude de visibilité et qui peuvent être :

. la décomposition de la scène en objets, chaque objet étant défini par un ensemble de faces. Chaque face est définie par une suite de sommets.



Si la scène est composée de faces polygonales, les informations pouvant être associées à celles-ci sont convexité, orientation à priori, etc...

Pour chaque objet, si cette notion existe, on peut posséder un graphe d'adjacence entre faces, une matrice d'état pour chaque couple de face.

D'autres informations peuvent être l'indication de pénétration entre éléments de la scène ou la possibilité de décomposer l'ensemble des objets de la scène, par des plans séparateurs.

L'intérêt de ces informations est qu'elles ne dépendent

pas de la position de l'observateur, et qu'elles peuvent donc être acquises une fois pour toutes.

- Les informations d'aspect, qui ne sont pas nécessaires pour tous les algorithmes, indiquent la couleur, la transparence et plus généralement le comportement du matériau composant les éléments de la scène vis-à-vis de la lumière.

1.2.2.2. Structuration de la scène

Cette structuration est transmise par la phase de modélisation de la scène et n'est pas acquise pour chaque vue traitée par les algorithmes.

On entend par là essentiellement la décomposition de la scène en objets, et la décomposition des objets en entités élémentaires qui peuvent être par exemple des faces polygonales. Dans certains cas, cette structuration indiquera plusieurs occurences d'un même objet, ou la décomposition de chaque objet en objets élémentaires.

On constate que ces informations sont uniquement liées à la phase de modélisation de la scène, où là seulement elles peuvent être acquises.

1.2.2.3. Préparation à l'élimination

Une vue de la scène est définie par trois paramètres:

- la position de l'observateur relative à un repère lié à la scène ;

- un axe de vue ;

- un angle de vue.

Ils définissent ainsi un cône de vision, tout élément ou portion d'élément extérieur à celui-ci ne devra pas être pris en compte lors de l'étude de visibilité. Le premier traitement appliqué à la scène

sera un algorithme de découpage par rapport au cône de vision.

Pour des commodités de calcul au cours de l'étude de visibilité l'observateur sera supposé situé sur l'axe oz du repère lié à la scène.(Figure 1.5.).

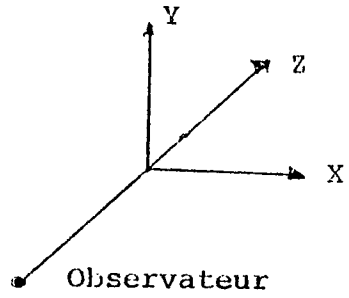


Figure 1.5. Position de l'observateur

L'observateur peut être situé à une distance finie, la scène en trois dimensions devra être projetée sur une surface bi-dimensionnelle. La projection utilisée est la projection perspective.

La scène subira cette transformation avant l'élimination des parties cachées, au cours de laquelle l'observateur peut être supposé alors situé à une distance infinie de la scène.

Suivant les calculs réalisés après la prise en compte de l'observateur, on distingue deux types de prétraitement :

- le prétraitement avec modification de la scène ;
- le prétraitement équivalent à une structuration de la scène.

. Modification de la scène

a) élimination des faces arrières : pour un objet ne possédant pas de faces avant transparentes, il est possible d'éliminer les faces arrières qui ne peuvent évidemment influer sur la visibilité des autres éléments de la scène.

b) élimination des faces parallèles à l'axe de vue.

Ces deux techniques ont pour principal avantage de diminuer le nombre d'éléments qui seront à traiter ultérieurement mais il ne faut pas négliger le temps passé à ce type de prétraitement dans le cas où le nombre de faces est élevé (pour chaque face un calcul de normale est nécessaire.).

c) consolidation des objets convexes :

Cette technique consiste à éliminer les faces arrières d'un objet et à en extraire la silhouette. Les faces avant n'interviendront pas au cours de l'étude de visibilité. A chaque objet convexe correspondra alors une face convexe dans la suite du traitement. On verra par la suite que cette technique n'est applicable que pour certains algorithmes.

. Structuration de la scène :

Les deux techniques qui vont être décrites ont pour caractéristique commune de n'éliminer aucun élément de la scène, mais d'effectuer une restructuration de celle-ci.

a) découpage de la scène en zones et en boîtes :

. une zone est un ensemble d'éléments de la scène dont il se peut que la visibilité soit liée ;

. une zone est composée d'un ensemble de boîtes. Deux boîtes, ensemble d'éléments, ont la propriété de ne pas se recouvrir en profondeur.

L'intérêt de cette restructuration de la scène est que les zones sont traitées indépendamment et le découpage en boîte peut être utilisé dans certains algorithmes afin de réduire le nombre de comparaisons dans une même zone.

La faiblesse de cette méthode vient essentiellement du fait que zones et boîtes sont déterminées par comparaisons des éléments à l'aide de tests minimaux qui ne donnent que des conditions suffisantes. L'existence même de zones peut être très faible. Dans le cas d'un paysage où un terrain fait la jonction entre les autres éléments de la scène, aucune zone ne pourra être détectée.

b) Partition de l'ensemble des faces d'un objet :

Cette technique consiste à partitionner l'ensemble des faces d'un objet en faces arrières et faces avant. Si elle n'est utilisable que pour certains algorithmes, elle est par contre intéressante dans le cas d'une étude de visibilité dynamique où la partition pour une vue est déduite de celle faite pour la vue précédente.

1.2.2.4. Conclusion

Nous avons montré la nature des modèles de description, quelles sont les informations nécessaires et celles dont on peut tirer profit dans la suite du traitement. Les prétraitements indiqués utilisent des informations topologiques sur la scène. Dans certains cas, l'étude de visibilité est terminée (élimination des faces arrières pour un objet convexe quelconque) sinon lorsque ces informations ne peuvent apporter aucune facilité de traitement on utilise alors un algorithme d'étude de visibilité.

1.2.3. Fonctions de transition

Les fonctions de transition sont en fait des opérateurs de comparaison entre éléments géométriques qui appartiennent soit au modèle de description soit à une représentation intermédiaire.

La diversité des éléments pouvant être comparés et des critères de comparaison nous a conduit à faire une étude détaillée (cf chapitre 3) de ces problèmes auxquels nous avons été confrontés lors de la réalisation de plusieurs algorithmes d'étude de visibilité.

1.2.4. Utilisation de tris et fonctions stratégie

Tout algorithme d'étude de visibilité utilise le tri ([SSS 74]) Les éléments triés peuvent être des faces polygonales, des arêtes, des segments ... La clé d'un tri peut être le point le plus proche de l'observateur pour chaque face, le sommet le plus bas d'une arête...

L'emploi d'un tri, son choix et le choix de la clé sont en fait liés à la fonction stratégie utilisée. Par exemple, dans l'algorithme de Watkins, où l'étude se fait par plans de balayage successifs, il est intéressant d'effectuer un tri initial des arêtes en fonction du plan où elles apparaissent plutôt que de chercher pour chaque plan les arêtes qui entrent en jeu. La fonction stratégie qui est l'ordre d'application de tris et de fonctions de transition tend à réduire par ces fonctions le nombre d'éléments à comparer. Cette réduction peut se faire en faisant des tris ou comparaisons intervenant dans les trois directions de l'espaces x, y et z. Une fonction stratégie peut donc se caractériser dans l'ordre de considération de ces trois directions.

L'algorithme de Newell utilise l'ordre zyx, celui de Watkins l'ordre yxz, celui de Warnock l'ordre xyz et yxz.

Si l'on considère que les éléments sont répartis de façon équilibrée suivant ces trois directions, on ne peut faire le choix d'un algorithme en fonction de l'ordre qu'il utilise. Au contraire, si une scène présente une direction de répartition privilègiée, ce caractère particulier pourra guider le choix d'un algorithme.

1.2.5. Modèles de visibilité

Un modèle de visibilité est l'ensemble des informations qui traité par le module de synthèse avec les informations d'aspects, fournit l'image ou dessin final.

1.2.5.1. Modèles ligne visible et surface visible.

Un modèle ligne visible permet d'obtenir un dessin au trait ([Gal 69], [Lou 69],[War 69]). La principale différence entre les modèles fournis par ces algorithmes sera la qualité du dessin et la quantité d'informations contenue. Une portion d'arête visible pourra être transmise en une seule fois par un algorithme ([Gal 69]) ou en plusieurs fois ([War 69]).

Un modèle surface visible permet de produire des surfaces ou portions de surfaces visibles ([NNS], [Wat 70],[AtW 78]). Dans le cas de la synthèse d'images réalistes ce sont ces algorithmes qui sont retenus puisqu'ils permettent de rendre l'aspect des matériaux constituant les éléments de la scène. Il est à noter que tout algorithme de surface visible permet d'obtenir une présentation de type ligne visible.

1.2.5.2. Modèles finis et non finis

Un modèle fini fournit les éléments visibles de la scène Un modèle non fini ne donne pas ces éléments mais un ordre de traitement de ceux-ci par le modèle de synthèse qui donnera lui l'image ou dessin final.

Si un modèle non fini, du type de celui fourni par l'algorithme de Newell, nécessite plus de traitement de la part du module de synthèse, il présente l'avantage de pouvoir être utilisé plusieurs fois en faisant varier l'aspect de certains éléments de la scène.

1.2.5.3. Liens avec la synthèse

Le premier facteur sera le nombre d'éléments transmis pour la synthèse.

Dans le cas d'un dessin, le temps de traitement sera proportionnel au nombre d'éléments visibles et aussi à l'ordre dans lequel ils sont transmis.

Dans le cas d'une image, plusieurs critères sont à retenir :

. le comportement vis à vis du remplissage, qui tient compte du nombre d'éléments à remplir, de leur adaptation à ce traitement et du type de terminal utilisé ;

. le modèle de visibilité permet-il de rendre l'effet de transparence ?

. - dans le cas d'un ombrage continu, certaines informations de contiguité ou non contiguité sont nécessaires et devront donc être attachées au modèle de visibilité.

1.2.6. Conclusion

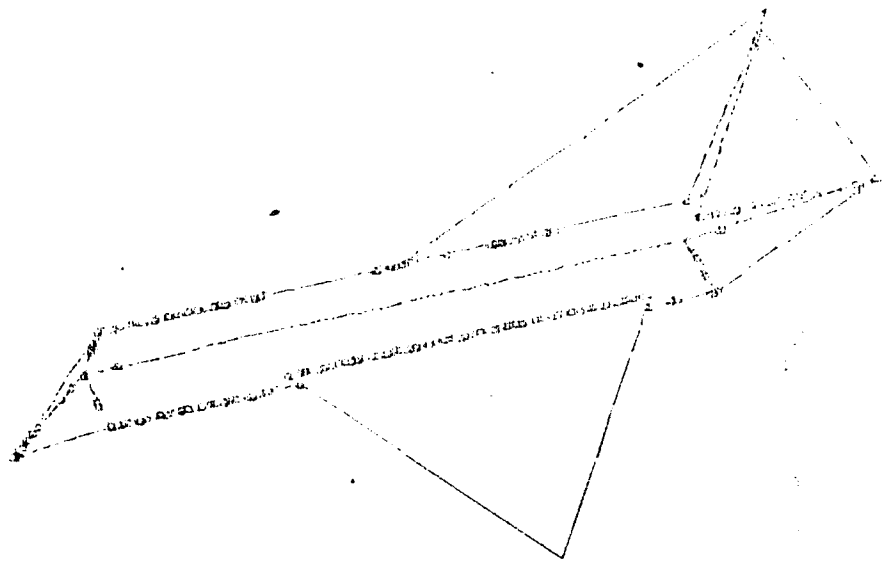
Nous avons par la formalisation introduite décomposé tout algorithme d'étude de visibilité. Celle-ci met en valeur les éléments constitutifs de chacun dont nous avons détaillé la nature.

Il est apparu que la phase de modélisation devra être productrice d'informations attachées à la scène qui seront utilisées dans une phase de préparation à l'élimination et aussi fournit une scène structurée.

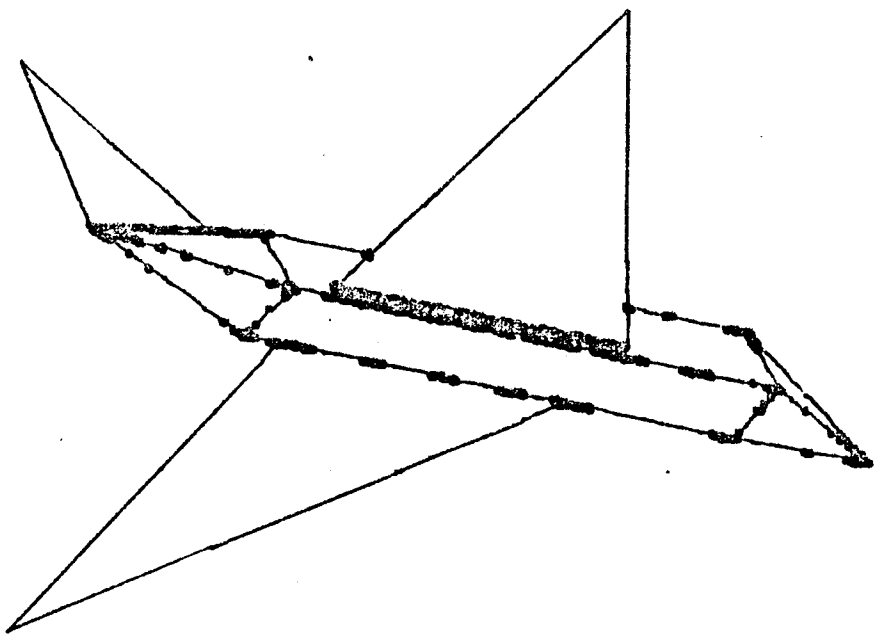
1.3. Conclusion

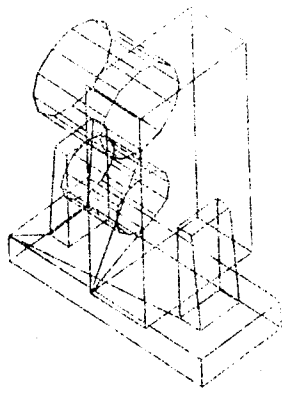
Les différents critères de classification ont montré la diversité des problèmes et des solutions apportées.

La formalisation introduite permet de faire une description exhaustive de chaque algorithme et nous avons montré quels pouvaient être les différents éléments d'un de ceux-ci. Ceci permet à un utilisateur de faire le choix de l'algorithme adapté à son application en fonction des différents critères énoncés.

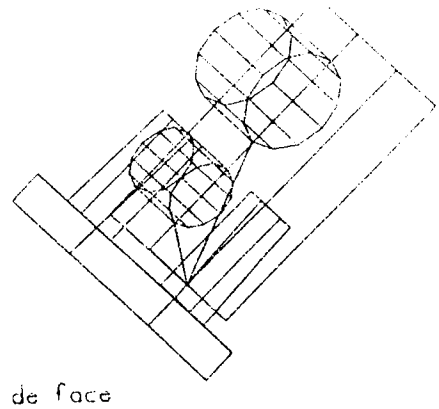


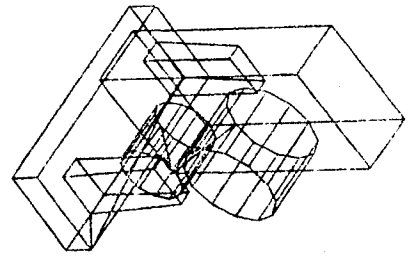
Deux vues d'une même scène traitées par l'algorithme de Warnock



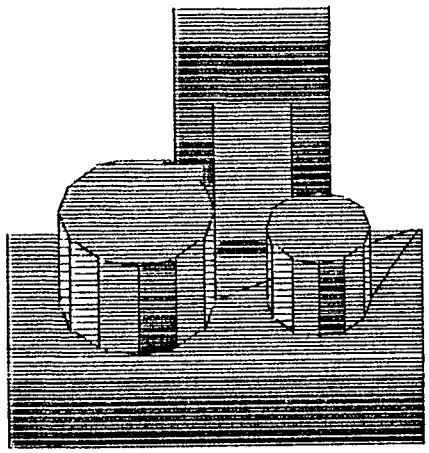


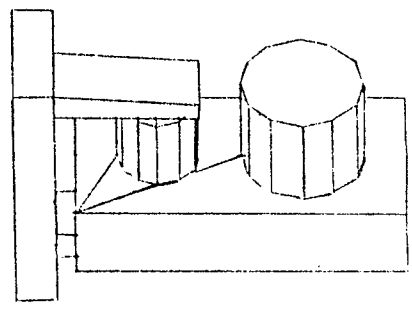
vue de face

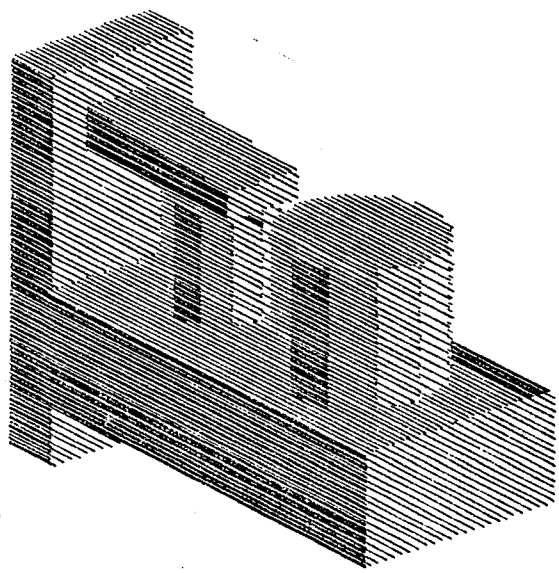




vue de haut







CHAPITRE II

REALISATION

2.1. Présentation des algorithmes choisis

Quatre algorithmes ont été retenus, le choix s'est essentiellement fait en fonction des modèles de visibilité qu'ils produisent.
Ces algorithmes traitent des modèles de description identiques, constitués de faces polygonales. Les seules distinctions entre ces modèles sont de nature topologique sur les faces. Cette restriction sur les scènes traitées qui semble importante, nous permet d'envisager un grand nombre d'applications puisque toute scène d'une autre nature pourra être approchée par un ensemble de faces polygonales. Les quatre algorithmes produisent des modèles qui sont :

- pour l'algorithme de Atherton et Weiler, travaillant dans l'espace objet, un ensemble de faces polygonales visibles ;
- pour l'algorithme de Newell, Newell et Sancha, dit à liste de priorité une liste ordonnée de polygones ;
- pour l'algorithme de Watkins, travaillant dans l'espace écran, un ensemble de segments horizontaux ;
- pour l'algorithme de Warnock, travaillant dans l'espace image, un ensemble d'arêtes ou portions d'arêtes visibles.

Nous possédons donc 4 algorithmes, appartenant aux différentes branches de la classification donnée en 1.1.2., et produisant des modèles de visibilité différents.

Nous étudierons l'intérêt de ces algorithmes au point de vue de leur comportement, de leur modèle de visibilité, et mettrons en valeur les différents problèmes rencontrés au cours de leur réalisation.

2.2. L'algorithme de Warnock

2.2.1. Description

C'est un des premiers algorithmes publiés (cf [War69]) et

il se range dans la catégorie des algorithmes travaillant dans l'espace écran suivant la classification présentée en 1.1.3.

Le principe de l'étude de visibilité est d'essayer de "comprendre" une portion de scène contenue dans un domaine. Si la scène est comprise, on l'affiche; sinon le domaine est divisé en quatre sous-domaines qui sont étudiés successivement. Le processus d'étude d'un domaine s'arrête lorsque la limite de résolution de l'écran est atteinte, ou lorsque tous les sous-domaines ont été compris. Le domaine initial est un parallélépipède de section égale à l'écran. L'observateur est supposé situé à une distance infinie de la scène (voir figure 2.1.).

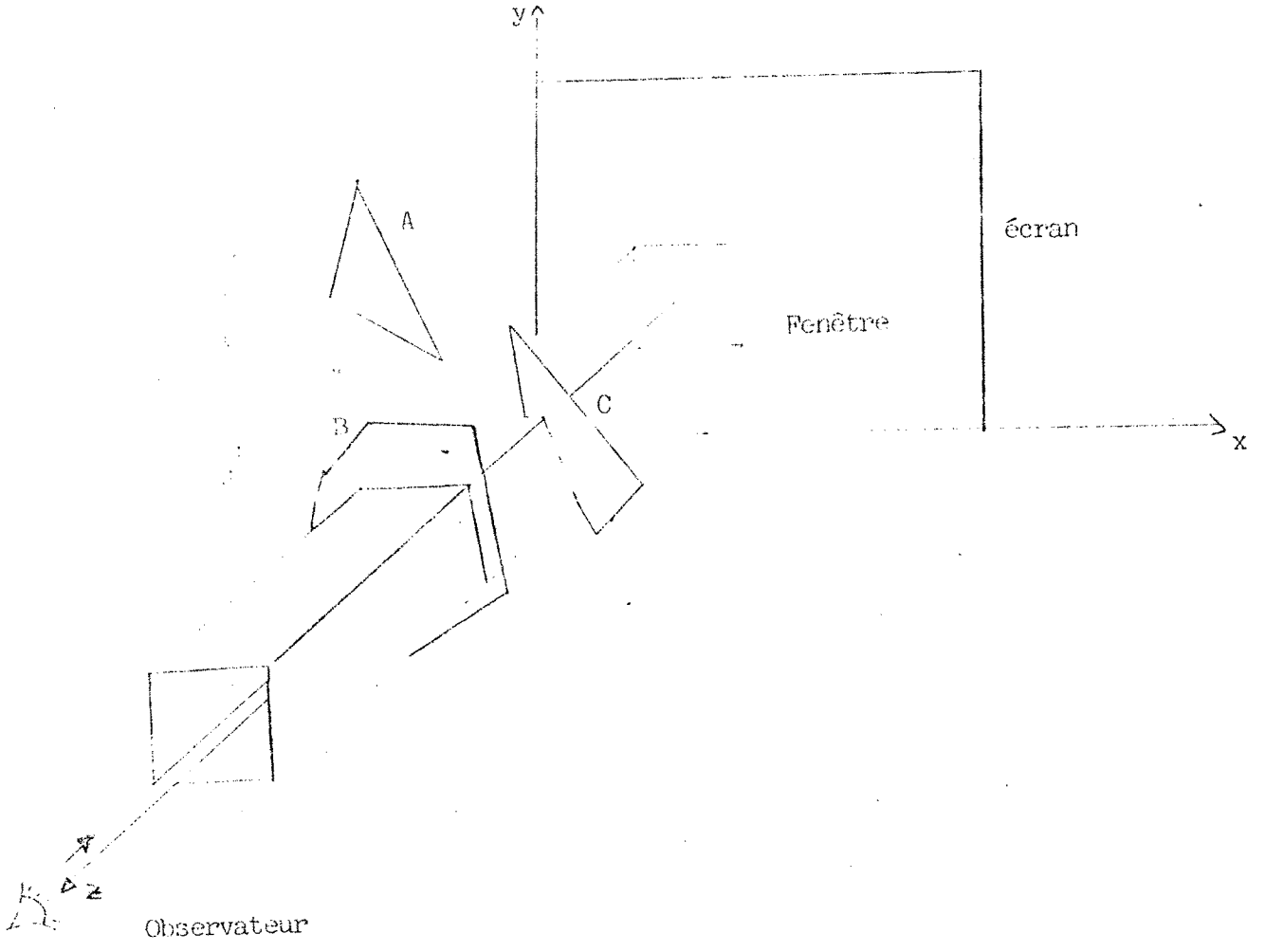
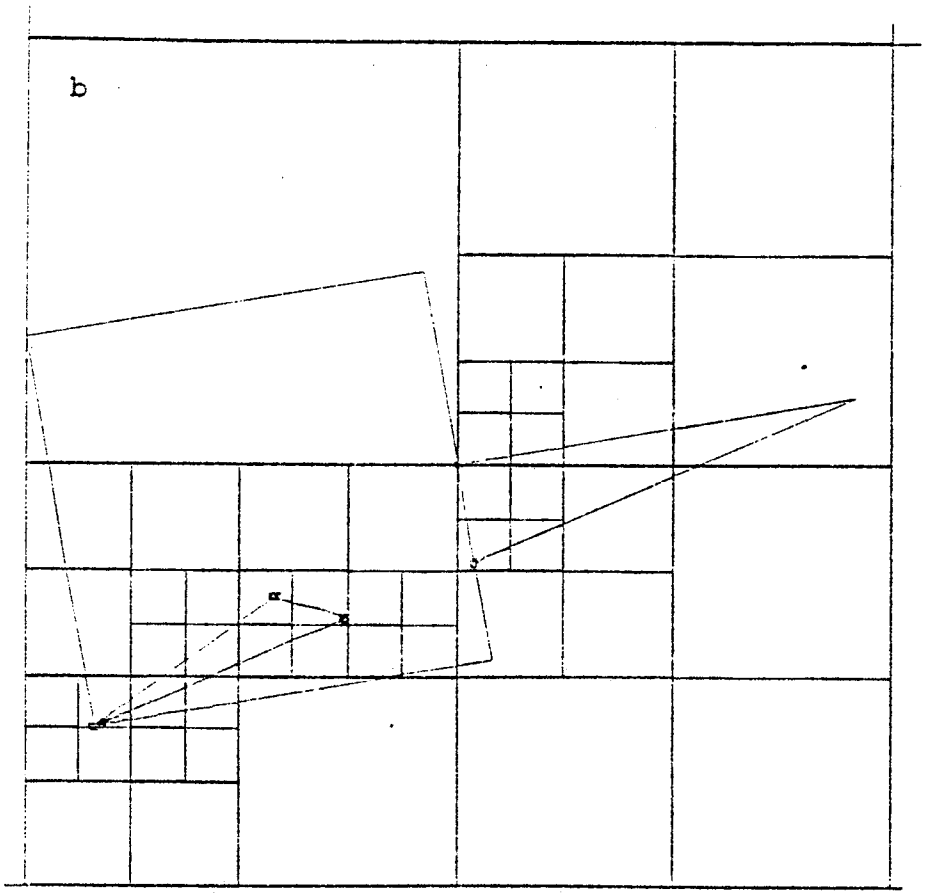
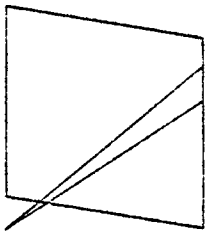


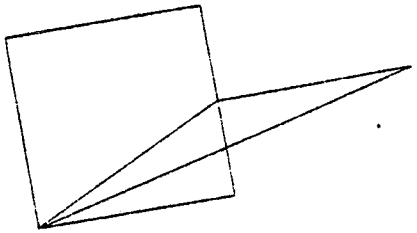
Fig. 2.1. Principe de l'algorithme de Warnock.



a



VUE DE COTE



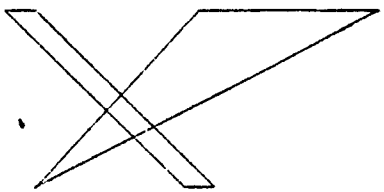
VUE DE FACE

AMPLIX = 1000.00
AMPLIY = 2143.26
AMPLIZ = 1332.10
XC = 50.00
YC = 100.00
ZC = 50.0000

NOMBRE DE SOMMETS : 9

NOMBRE DE FACES : 2

NOMBRE DE SECTIONS: 1



VUE DE HAUT

c

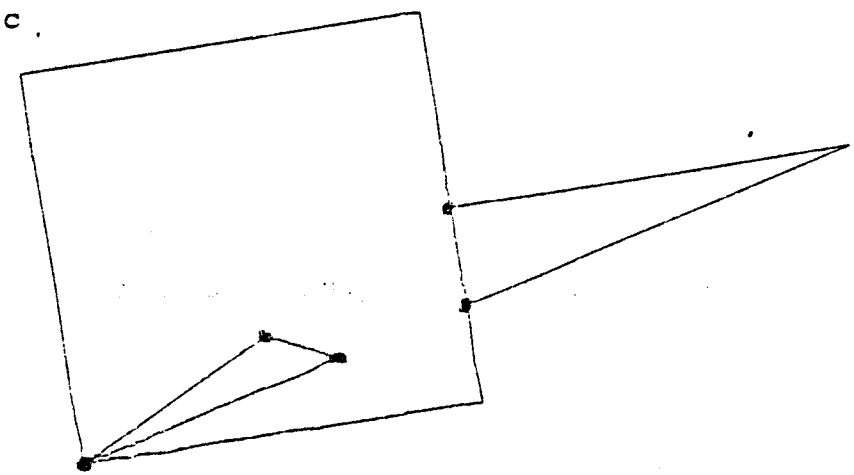


Figure 2.2.

La figure 2.2. présente un exemple de scène traitée par l'algorithme de Warnock :

a) 3 vues de la scène à traiter

b) découpage de la vue de face

c) vue de face traitée

L'algorithme initial (War 69) est un algorithme de surfaces cachées ; ceci signifie que les éléments affichés sont des portions de faces. Nous étudierons par contre une version de cet algorithme, où les éléments affichés sont des arêtes ou portions d'arêtes, et nous verrons quelles sont les modifications à apporter afin de réaliser un algorithme de surfaces cachées.

2.2.2. Etude des différents éléments de l'algorithme :

Cet algorithme sera étudié en se conformant à la formalisation décrite auparavant, qui définit un algorithme de visibilité par un 7-uplet: (MD, Td, ϕ, σ, T_i, RI, MV).

2.2.2.1. Modèle de description : MD

L'ensemble MD est constitué de faces polygonales planes, convexes ou non. Aucune face ne peut être trouée mais les faces peuvent se pénétrer. A chacune est associée une information d'aspect. (voir figure 2.3.).

Représentation de l'ensemble MD

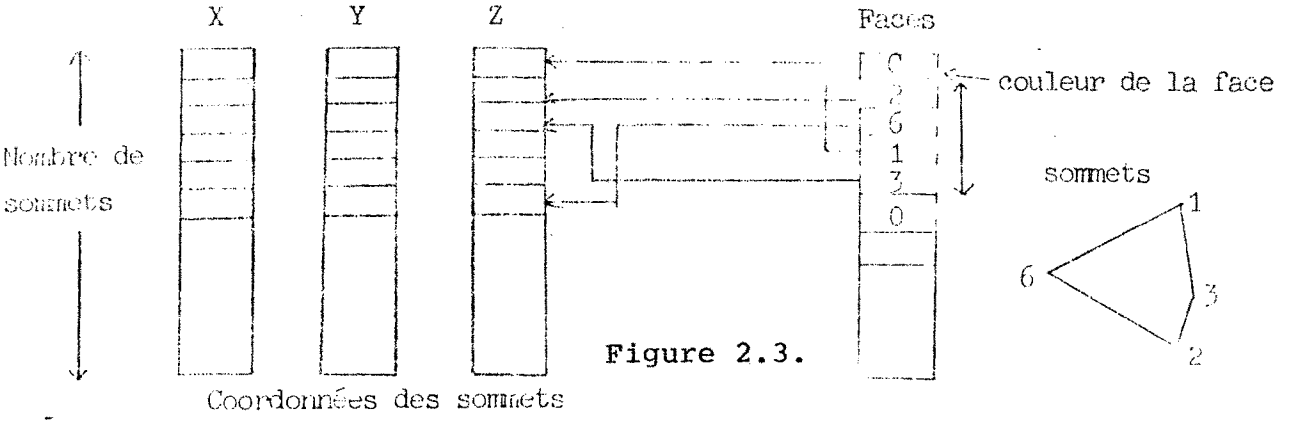


Figure 2.3.

2.2.2.2. Tri initial des données : Td

Les éléments de MD sont triés initialement de telle sorte qu'ils soient traités par la suite, par ordre d'éloignement par rapport à l'observateur : $F_i <= F_j <=> Z_i < = Z_j$ Z_i étant un des points les plus proches de l'observateur de la face F_i (voir figure 2.4.).

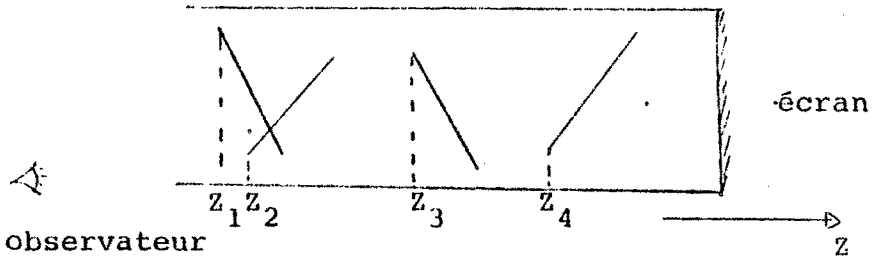


Figure 2.4.

Le tri choisi est le tri rapide "Quick sort", car l'ensemble MD n'a a priori aucun ordre. La complexité maximale du tri est $O(n^2)$, elle est atteinte lorsque les données sont triées ou en ordre inverse ; la complexité moyenne de ce tri est O(nlog(n)).

2.2.2.3. Ensemble des fonctions de transition

ϕ = (PER, POS, PRO, VT).

Fonction produisant la vue perspective : PER

L'observateur est supposé situé sur l'axe Oz à l'infini (cf. Figure 1). La fonction employée est la projection orthogonale

$$\text{soit } M \in \mathbb{R}^3 \quad M = \begin{vmatrix} x \\ y \\ z \end{vmatrix} \quad PER(M) = \begin{vmatrix} x \\ y \end{vmatrix}$$

Fonction déterminant les relations spatiales entre un polygone et une fenêtre : POS

Cette fonction permet de déterminer si un polygone est intersectant, disjoint ou englobant (cf. Figure 1.1.) par rapport à la fenêtre d'étude. Cette opération est réalisée par une comparaison C_c^c qui est décrite au CHAPITRE 3.

Test de profondeur : PRØ

Ce test permet de comparer la position de deux polygones par rapport à l'observateur, les deux éléments pouvant être en conflit puisqu'ils ont une intersection non vide avec la même fenêtre. Ce test s'effectue en comparant les valeurs en z des plans contenant les faces par rapport aux plans qui délimitent la fenêtre (voir figure 2.5.).

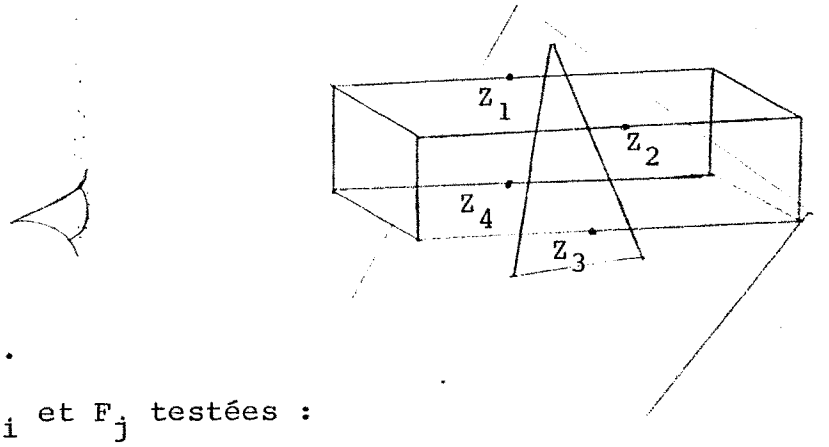


Fig. 2.5.

Soient F_i et F_j testées :

1) Si $\forall k \in 1, 4$ $\quad Z_{i_k} \leq Z_{j_k}$ $\quad F_i$ devant F_j

2) si $\forall k \in \{1, 4\}$ $\quad Z_{j_k} \leq Z_{i_k}$ $\quad F_j$ devant F_i

3) si $\exists k,k'$ $\quad Z_{i_k} < Z_{j_k}$ et $Z_{i_{k'}} > Z_{j_{k'}}$

alors intersection des deux faces.

On remarque que ce test est entâché d'erreur, puisque la conclusion dans le cas 3 peut être fausse.

Test de visibilité : VT

La fonction VT qui consiste en l'élimination des faces arrères d'un objet n'est pas appliquée. En effet, les éléments de MD étant des polygones, la notion de volume a été perdue. Cette fonction pourrait être utilisée si les polygones étaient décrits par une suite ordonnée

de sommets qui permettent de calculer une normale extérieure. Ceci permettrait de réduire considérablement le nombre de faces étudiées, mais suppose que toutes les faces sont opaques.

2.2.2.4. Fonction stratégie

La fonction stratégie de l'algorithme de Warnock est schématisée figure 2.6.

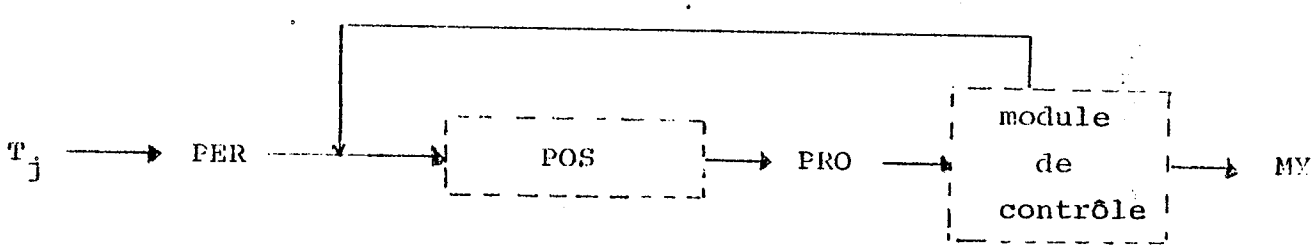


Fig. 2.6. Fonction stratégie.

Module de contrôle:

Il crée les divisions de fenêtre en cas d'échec et gère l'ensemble des fenêtres à étudier.

Module d'analyse :

Il étudie les relations spatiales des éléments de MD par rapport à la fenêtre en cours d'étude et classe ces éléments en trois sous-ensembles (englobants, disjoints, intersectants). (voir fig. 2.7.)

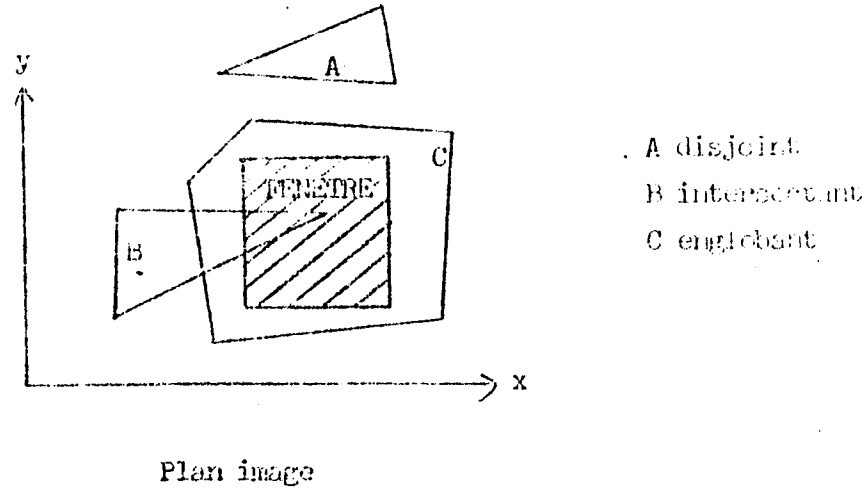


Fig. 2.7. Relations spatiales faces-fenêtre.

Module de compréhension :

Ce module qui ne comprend que la fonction profondeur PRØ, utilise les données préparées par le module d'analyse, afin de déterminer si l'affichage est possible.

2.2.2.5. Ensemble des représentations intermédiaires : RI

Les différentes représentations intermédiaires sont en fait indiquées par la fonction stratégie, schématisée figure 2.8.

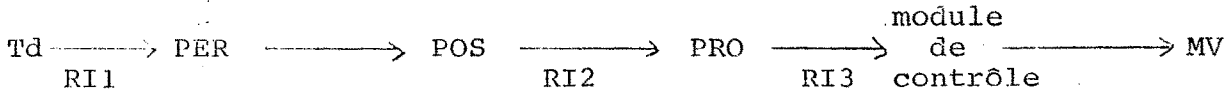


Fig. 2.8. Représentations intermédiaires

RI1 : ensemble des faces triées par valeur de Zmin croissante

RI2 : ensemble des faces partitionné en éléments :

- englobants,
- intersectants,
- disjoints.

RI3 : ensemble vide ou ensemble des éléments affichables à cette étape de l'étude.

2.2.2.6. Modèle de visibilité : MV

Le modèle de visibilité peut être de deux types différents :

- Type 1 : on désire obtenir une image, les éléments de MV seront des polygones fermés auxquels sont associés des informations d'aspect.
- Type 2 : on désire obtenir un dessin au trait, les éléments de MV seront des segments ou des points dans le cas où l'étude aura été faite jusqu'à limite de résolution de l'écran. Dans le cas où une intersection entre deux faces est visible, elle sera affichée.

Il est à noter que le phénomène de transparence n'est pas rendu car les

faces transparentes sont ôtées du modèle de description avant l'étude de visibilité.

2.2.3. Etude des différents modules

2.2.3.1. Module d'analyse :

Le rôle du module d'analyse est de partitionner l'ensemble MD en trois sous-ensembles : intersectants, englobants, disjoints (cf. fig 1). A chaque fenêtre, l'ensemble MD devrait être étudié, mais la façon même d'étudier ces fenêtres permet de réduire le travail à effectuer pour chacune de celles-ci (voir figure 2.9).

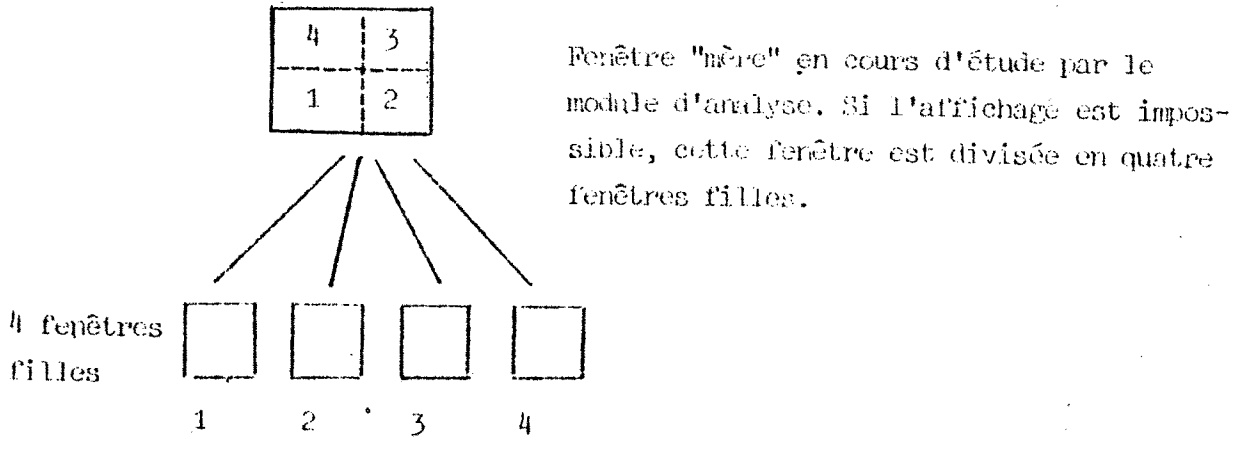


Fig. 2.9. Découpage d'une fenêtre.

D'où le schéma de la figure 2.10 pour l'étude d'une fenêtre fille :

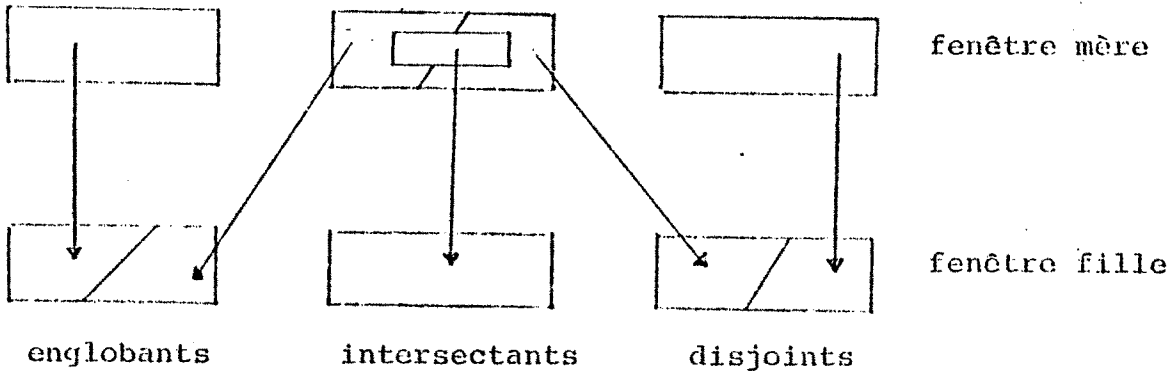


Fig. 2.10

La propriété pour un polygone d'être englobant ou disjoint, se conservant au passage mère-fille, seuls les intersectants de la fenêtre mère seront à traiter par le module d'analyse lors de l'étude des fenêtres filles.

Le module d'analyse travaille donc sur une représentation intermédiaire qui est une partition de l'ensemble MD en trois sous-ensembles.

On peut améliorer ce traitement à la fois au plan individuel (1 polygone) et au plan général (ensemble des polygones).

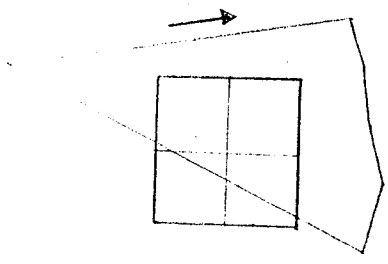
Traitement d'un polygone

La présence d'intersectants implique donc le plus de travail de la part du module d'analyse. En effet, ce sont les polygones intersectants de la fenêtre mère qui entraînent une étude de polygones pour les fenêtres filles (si il y a eu échec).

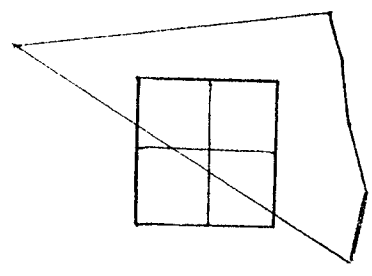
Or, les solutions proposées n'utilisent aucune information sur les intersectants, qui pourrait être conservée au passage mère-fille, comme c'est le cas pour les englobants et disjoints.

Tout polygone intersectant est réétudié par un parcours séquentiel de ses arêtes, alors qu'on peut remarquer qu'une arête intersectante dans une fenêtre mère le sera dans au moins une des fenêtres filles.

La solution proposée sera d'appliquer les tests POS à cette arête, lors de l'étude des fenêtres filles, ce qui permet d'accélérer le processus de classification des polygones (voir figure 2.11.).



L'étude de ce polygone par rapport à chaque fenêtre fille, nécessite 13 applications de POS.



Dans 3 fenêtres filles, POS ne sera appliqué qu'une fois.

Fig. 2.11.

Traitement de l'ensemble des polygones

L'ensemble des polygones a été initialement ordonné par le tri T_d (fig. 2.4.). Ceci permet d'effectuer le traitement de ceux-ci en commençant par les éléments les plus proches de l'observateur (supposé situé sur l'axe O z à moins l'infini). (Voir figure 2.12.).

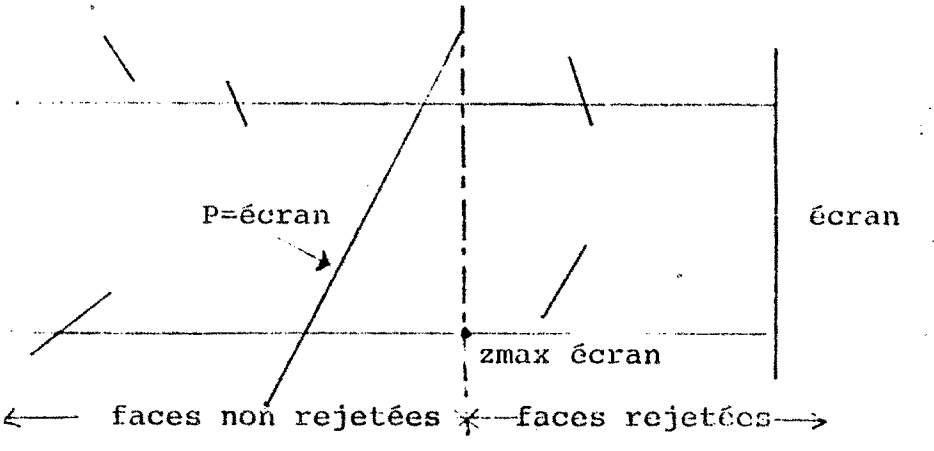


Fig. 2.12.

Soit P un polygone écran, les faces ayant un zmin supérieur au zmax écran, ne seront pas étudiées, car elles ne peuvent pas être visibles dans la fenêtre étudiée.

La découverte d'un élément englobant permet, grâce au tri initial T_d de MD, de n'étudier qu'une partie des éléments (englobants et intersectants) présents dans une fenêtre.

La présence de faces pouvant être englobantes par rapport à une fenêtre, permet donc d'accélérer la tache accomplie par le module d'analyse. Ceci dépend de la taille, de la forme, de l'orientation et de l'éloignement des faces traitées.

Conclusion

Le résultat du traitement de la scène par ce module, se traduit par le passage au module de compréhension de deux listes ordonnées, celle des englobants et celles des intersectants. Les facteurs influençant le temps de traitement dans ce module seront résumés ultérieurement.

2.2.3.2. Module de compréhension

Choix d'un module de compréhension

C'est par cette partie que se distinguent les différentes versions de cet algorithme. En effet, ce module, à partir des informations fournies par l'analyse, décide en fonction de certains critères si l'affichage est possible.

Quelques solutions seraient :

- ne jamais comprendre le contenu de la fenêtre. Le travail du module de compréhension serait nul. Mais ceci conduirait d'une part à un tracé point par point de l'image finale, et d'autre part les temps de calcul seraient prohibitifs.

- comprendre les cas suivants :

. un englobant en avant de tout autre élément : il n'y a rien à afficher.

. un seul intersectant devant le cache le plus en avant.

Si cette solution semble à priori plus complète que la précédente, elle conduit tout de même à un nombre important de divisions, le plus souvent jusqu'à la limite de résolution. Ceci provient du fait que les faces traitées appartiennent en général à des solides, et donc qu'une arête appartient presque toujours à deux faces. Dans ce cas, le module de compréhension décèle un échec (voir figure 2.13.).

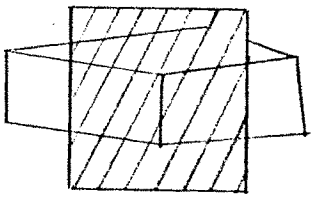
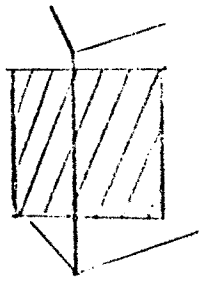


Fig. 2.13. Faces adjacentes.

Une solution pour ce problème est de reconnaître également le cas suivant :

plusieurs intersectants visibles dans la fenêtre, les arêtes vues ont même projection.



La solution adoptée permet de reconnaître les cas suivants :

1. Un englobant et aucun élément le pénétrant ou situé devant (voir figure 2.14.a).
2. Deux englobants se pénétrant en avant de tout autre élément (voir figure 2.14.b).
3. Des intersectants se situent devant un cache et les arêtes vues ont même projection (voir figure 2.14.c).
4. Un seul intersectant se situe dans la fenêtre ou devant un cache. les portions d'arêtes vues sont affichées (le cas ne sera jamais rencontré si les objets traités sont des solides).

   Cette solution est en fait un compromis entre une certaine complexité du module de compréhension pour la reconnaissance de ces cas et le nombre de divisions nécessitées par son insuffisance. Le choix d'un module de compréhension dépend en fait de la probabilité de présence de certaines configurations dans la scène à étudier.

   La solution choisie ne permet, par exemple, pas de reconnaître les cas décrits sur la figure 2.14.d.

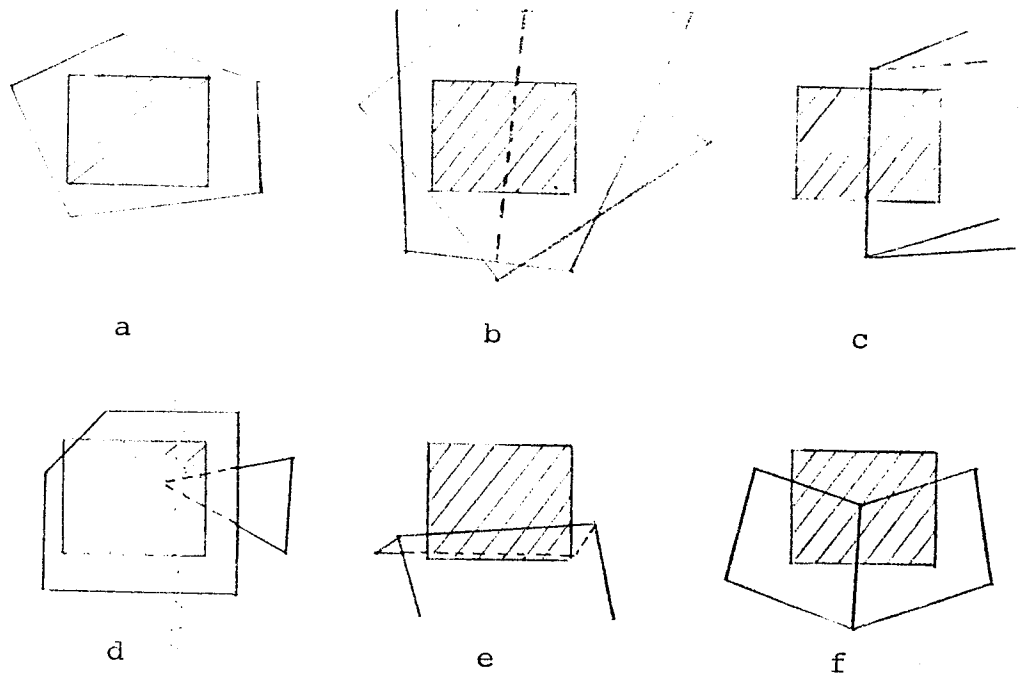


Fig. 2.14.

Remarque :

Les cas d'échec sont causés par la présence d'intersectants dans la fenêtre, les arêtes vues n'ayant pas même projection. Si par exemple un sommet d'une face est présent dans la fenêtre, ceci conduira presque certainement à un échec.

Recherche des cas de compréhension

Le module de compréhension procède en étudiant les deux listes ordonnées fournies par l'analyse (liste des englobants, liste des intersectants). Les étapes de cette étude sont :

- l'étude des englobants,
- la comparaison des intersectants aux englobants.

Etude des englobants :

Description :

Le but de cette étude est de déterminer un des deux cas suivants (voir figure 2.15.)

- un cache en avant de tout autre englobant,
- deux englobants se coupent

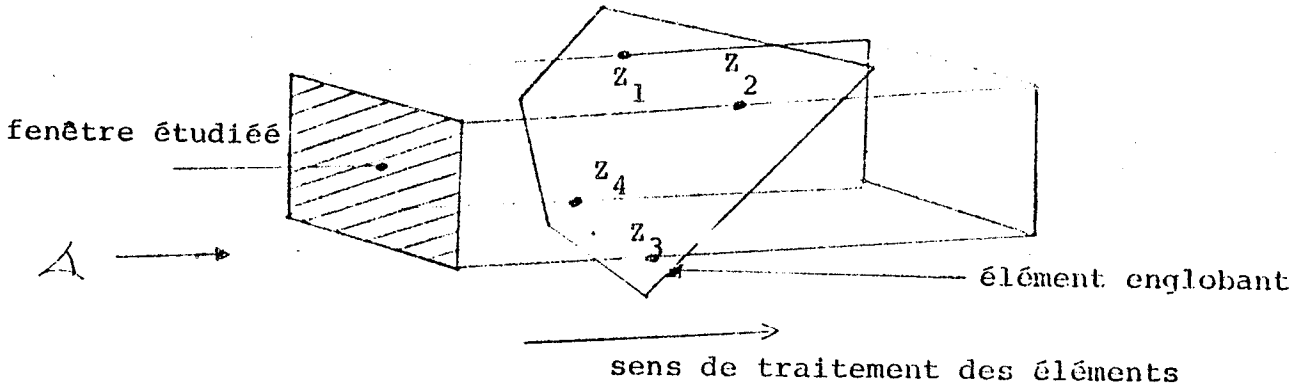


Fig. 2.15. Face englobante.

Les valeurs Z_i (i=1,4) sont calculées pour chaque englobant et comparées à celles de l'englobant précédent, afin de déterminer si l'élément :

1. cache le précédent,
2. est caché par le précédent,
3. coupe le précédent.

Certains renseignements seront conservés afin d'être utilisés lors de l'étude des fenêtres filles, si un échec a été trouvé :
dans les cas 1 et 2, l'élément caché ne sera pas étudié puisqu'il ne pourra influer sur la visibilité des éléments.
La seule cause d'échec à cette étape sera d'avoir rencontré plus de deux éléments se pénétrant.

Evaluation :

E étant le nombre d'englobants traités, cette étude nécessite au plus :

- 4 * E calculs de profondeur (Z_i)
- 4 * (E-1) comparaisons entre réels.

En fait, ce nombre important d'opérations peut être réduit par 4, si les éléments de l'ensemble MD ne peuvent se pénétrer. En effet, il suffira alors de tester les englobants, suivant un axe de la fenêtre seulement. D'autre part, la recherche de deux englobants intersectants ne sera plus nécessaire dans ce cas.

Comparaison des intersectants aux englobants :

Cette étude ne sera faite que si, à l'étape précédente, un des deux cas de succès a été reconnu, et elle le sera en fonction de ce dernier.

. premier cas : un cache a été déterminé (figure 2.16)

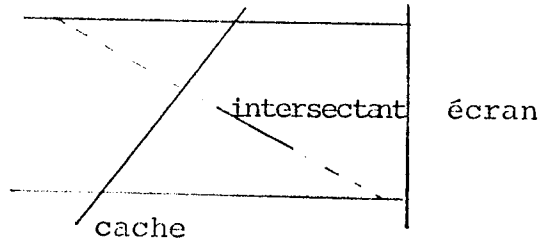


Fig. 2.16.

Les différentes positions d'un intersectant par rapport au cache sont :

- l'intersectant se situe derrière le cache ;
- l'intersectant se situe devant le cache ;
- l'intersectant coupe le cache.

La solution adoptée a été de déterminer la trace du plan contenant l'intersectant, sur les quatre plans qui délimitent la fenêtre traitée.

Mais cette solution a conduit à des résultats faux (cf. figure 2.16.). Afin d'éviter ceci, les calculs sont faits par rapport à la fenêtre englobant l'intersectant. (figure 2.17.).

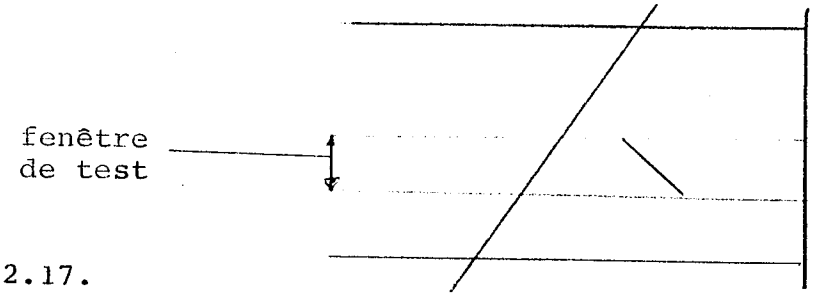


Fig. 2.17.

. deuxième cas : deux englobants intersectants (figure 2.18.).

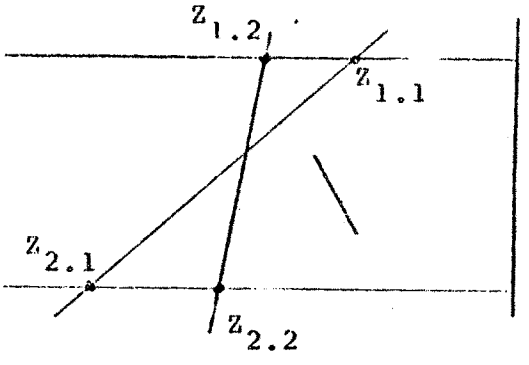


Fig. 2.18.

Le même processus est appliqué. Les comparaisons de profondeur se font ici par rapport aux $Z_{i,j}$ maximaux. Les mêmes résultats erronés se produisant, la solution citée plus haut est utilisée.

Evaluation

La comparaison d'un élément intersectant nécessite :

cas a) 8 calculs de profondeur ;

cas b) 12 calculs de profondeur.

Le coût du module de compréhension dépend donc en grande partie, du nombre d'intersectants présents dans la fenêtre. Par contre, supprimer l'hypothèse d'avoir des faces pénétrantes, permet de réduire le nombre de tests de la façon suivante :

seul le cas a) est rencontré, et il faut effectuer deux calculs de profondeur.

Décision à l'issue de ces deux étapes :

	ECHEC		SUCCES
cache	intersectant pénétrant	intersectants en avant et arêtes vues, n'ont pas même projection	aucun élément vu, ou arêtes vues, ont même projection
deux englobants se pénétrant	un intersectant en avant	intersectant pénétrant	aucun élément en avant

Conclusion

Deux remarques peuvent être faites sur les faces intersectantes, par rapport à une fenêtre :

- ce sont sur elles que le module de compréhension effectue le plus d'opérations élémentaires (calculs de profondeur, tests).

- la majorité des cas d'échecs viendra de leur présence dans la fenêtre étudiée. En effet, les cas de compréhension pour une situation comprenant des intersectants vus, sont très peu fréquents :

. un seul intersectant vu. Or, les objets qui composent la scène, sont en majorité des solides ;

. Les arêtes vues doivent avoir même projection, ce qui exclut en grande partie le cas où un sommet d'une face est vu. Cette dernière remarque conduira à l'étude d'une solution au contrôleur, qui permet d'effectuer le découpage en fenêtres filles, en fonction des sommets de faces intersectantes ayant conduit à un échec.

2.2.3.3. Le module de contrôle

Description

Le rôle du contrôleur est de gérer l'ensemble des fenêtres à étudier :

- si l'étude faite par le module de compréhension conduit à un échec,

il divise la fenêtre mère (si la limite de résolution de l'écran n'est pas atteinte) en 4 fenêtres filles qui seront alors étudiées (voir figure 2.19)

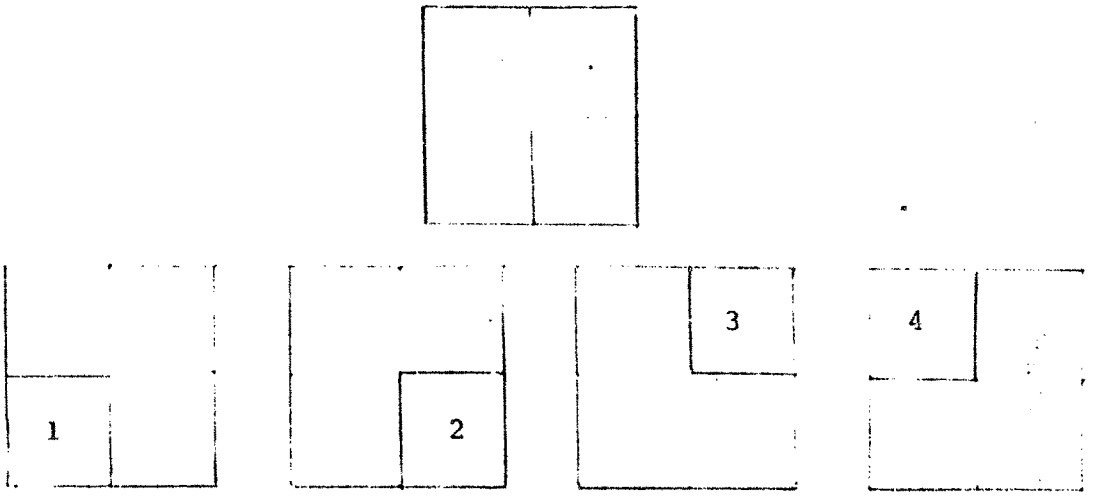


Fig. 2.19. Ordre d'étude des fenêtres filles

Lors d'un succès le contrôleur fournit au module d'analyse la fenêtre suivante à étudier.

Plusieurs remarques peuvent être faites sur la manière de choisir d'une part la fenêtre initiale, et d'autre part les fenêtres filles à partir d'une fenêtre mère.

Choix de la fenêtre initiale, découpage

Différentes solutions peuvent être envisagées :

1. La fenêtre initiale est l'écran, ce qui représente la fenêtre maximale englobant la scène ;

2. La fenêtre initiale est le plus petit rectangle (ou carré) englobant la scène

3. Orienter les axes de la fenêtre initiale en fonction de la "direction principale" de la scène ;

4. Appuyer le découpage sur des éléments particuliers de la scène.

Inconvénients de la solution 1

Soit la scène suivante, composée d'un carré cachant un triangle (figure 2.20.)

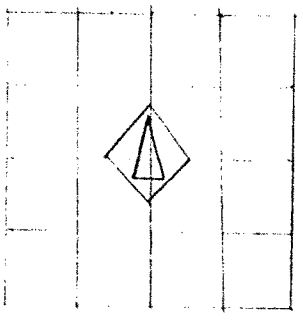


Fig. 2.20.

Le nombre de fenêtres étudiées pour déterminer la visibilité de cette scène, sera égal à peu près à 60.

Ce résultat s'explique par un découpage qui ne tient compte d'aucune caractéristique de la scène. Si par cette solution la tache du contrôleur est simplifiée au maximum, cette simplicité entraîne par contre un grand nombre de divisions, d'où des temps de calcul prohibitifs pour des scènes peu complexes.

Apports de la solution 4

L'idée est d'appuyer le découpage en fenêtres filles, sur des éléments ayant provoqué un échec : sommets, portions d'arêtes,élément qui son mémorisés lors de l'étude par le module de compréhension (voir figure 2.21.).

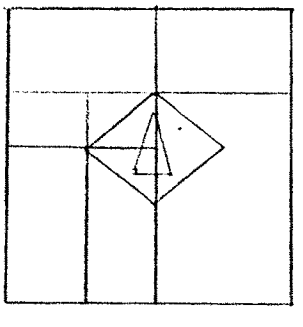


Fig. 2.21.

Cette solution ne semble pas plus performante que la précédente. Le choix des fenêtres ne tient en fait compte que des éléments visibles, mais non des problèmes posés par les éléments cachés (ici le triangle). D'autre part, cette solution crée deux types de problèmes :

- gestion plus complexe des fenêtres par le contrôleur
- nécessité d'effectuer un traitement particulier aux fenêtres, dont soit la longueur, soit la hauteur, est égale à un.

Solution 2

La fenêtre initiale est le plus petit carré englobant la scène (figure 2.22.).

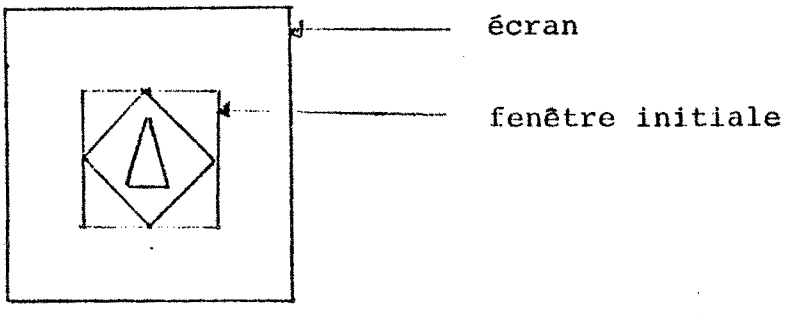


Fig. 2.22.

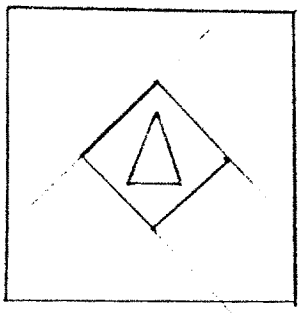
Dans cet exemple, 20 fenêtres environ seront étudiées, c'est-à-dire trois fois moins qu'en appliquant la première solution.

Cette solution exige par contre un temps de préparation des données plus important, ceci afin de déterminer quel est le plus petit carré englobant la scène à étudier.

D'autre part, on peut remarquer la fragilité des solutions présentées jusqu'ici. Il suffit en effet de déplacer légèrement le triangle pour que le nombre de fenêtres examinées augmente considérablement.

Combinaison des solutions 2 et 3

La fenêtre initiale sera ici choisie en fonction d'une direction principale de la scène, et en fonction de l'étendue sur l'écran de cette scène (figure 2.23.).



Dans cet exemple, une seule fenêtre sera étudiée.

Fig. 2.23.

Ce résultat dépend des caractéristiques suivantes de la scène :

- un seul objet, ou un seul élément cachant tous les autres éléments,

- forme, orientation des éléments visibles. Ici, la face visible est un carré qui peut donc coïncider avec une fenêtre d'étude. Ceci est à rapprocher d'un algorithme (ATW 77), où en fait les fenêtres d'étude sont constituées de polygones ou portion de polygones les plus proches de l'observateur.

Conclusion

L'étude du contrôleur a permis de mettre en évidence plusieurs facteurs qui influent sur l'efficacité de cet algorithme de visibilité :

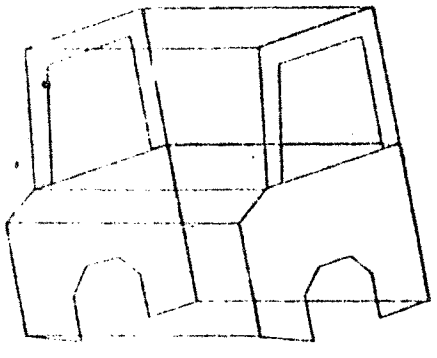
- présence de un ou plusieurs objets dans la scène. Par le terme objet, on entend un ensemble de faces, dont la visibilité est liée.

- forme, surface, orientation des éléments visibles de la scène,

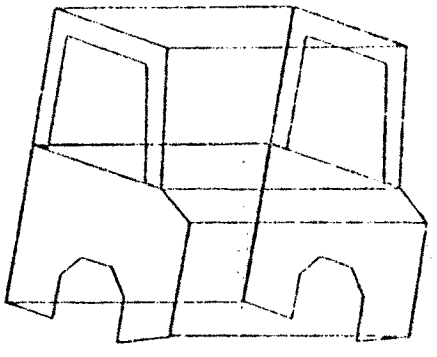
- influence des éléments cachés si la face la plus en avant est un intersectant.

2.2.3.4. Réalisation de l'algorithme

Les différents tests effectués ont montré l'importance à attacher à la précision à laquelle doivent être faits les calculs réalisés par les différentes fonctions de transition. En effet, cet algorithme dit à résolution limitée, travaille dans l'espace écran, où les points adressables ont des coordonnées entières. Il est donc tout à fait incohérent d'effectuer les calculs avec une précision trop importante. Les figures ci-dessous montrent des tests effectués sur des données identiques avec des précisions différentes

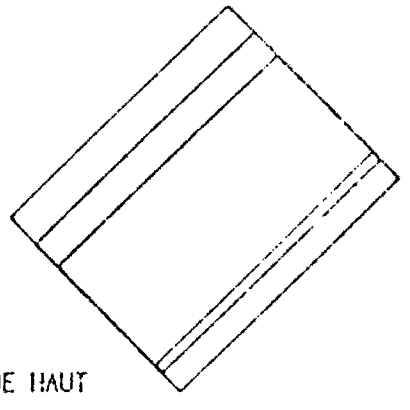


VUE DE COTE



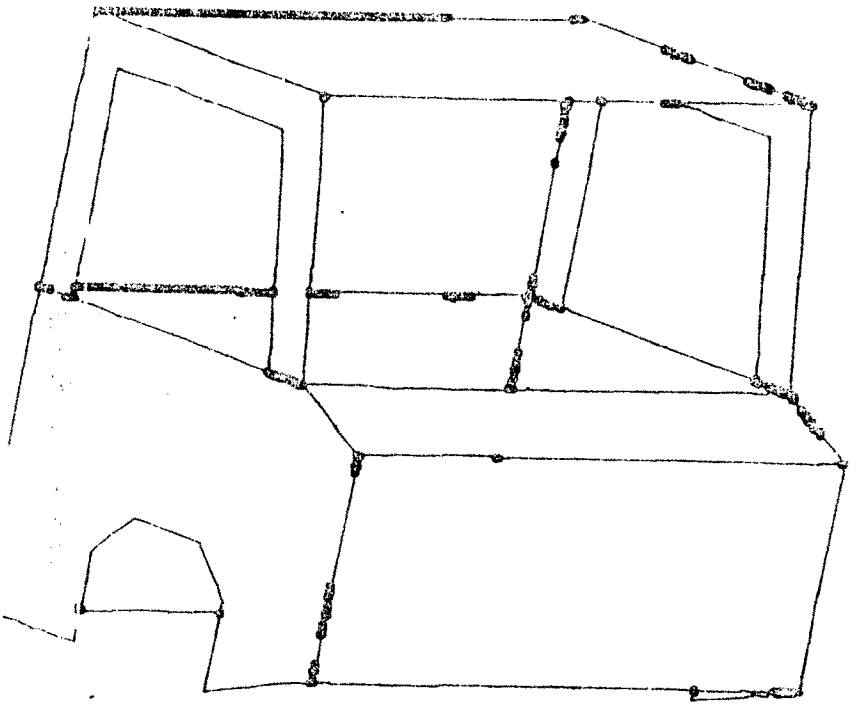
VUE DE FACE

NOMBRE DE SOMMETS : 34
NOMBRE DE FACES : 9
NOMBRE DE SECTIONS: 1



VUE DE HAUT

Fig. 2.24.



E = 0.01

Précision des tests à E

E = 1.0

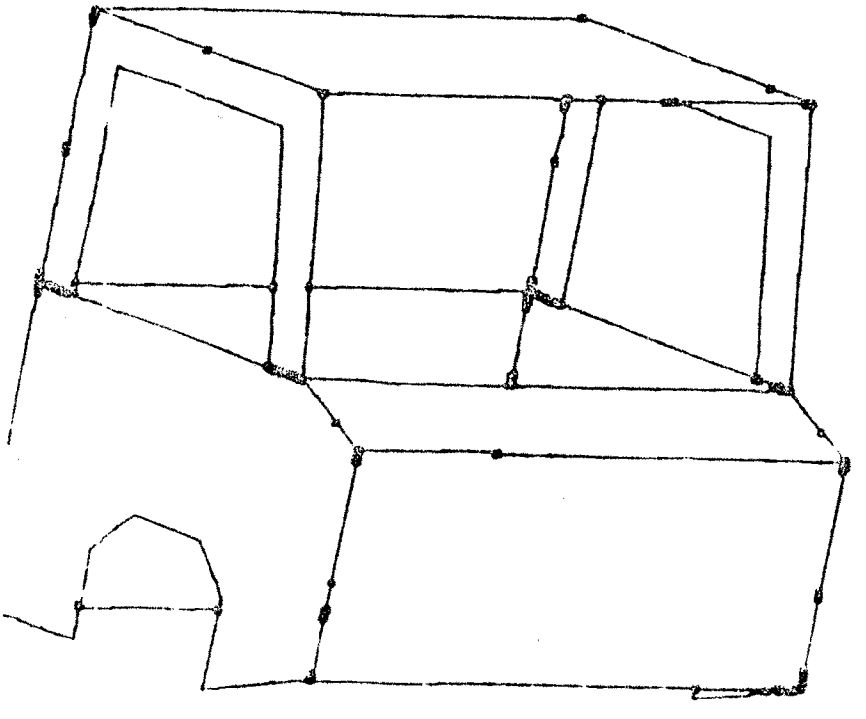


Fig. 2.25.

Cette remarque a permis d'obtenir deux résultats importants :

- diminution du temps de calcul, le nombre de fenêtres examinées étant moindre ;

- meilleure qualité du dessin.

En fait, il est envisageable d'effectuer les calculs sur des variables entières, ce qui, d'une part, diminue les temps de calcul (opérations sur des entiers par rapport aux opérations sur des réels), et d'autre part, ne modifie pas la qualité du dessin. Ce problème est en fait commun à tous les algorithmes de visibilité qui travaillent dans l'espace écran.

Comparaison du module d'analyse et du module de compréhension :

Les différents tests effectués ont montré la répartition suivante des temps d'exécution pour les deux modules :

- module d'analyse 60 % du temps d'exécution de l'algorithme ;

- module de compréhension 6 %.

Cette disproportion s'explique par plusieurs raisons :

1. Le module d'analyse qui étudie les relations spatiales dans une fenêtre, opère sur tous les éléments, alors qu'en début de traitement les informations accumulées sont peut-être suffisantes pour que le module de compréhension décide d'un échec.

2. Le module de compréhension peut donc déceler un échec en examinant seulement les éléments les plus proches de l'observateur, et n'utilise qu'une faible proportion des informations qui lui sont fournies.

D'autre part, on peut incriminer la "faiblesse" du module de compréhension, qui ne décide d'un affichage que dans un nombre limité de configurations.

Ces résultats s'expliquent donc par l'indépendance totale des deux modules, et de l'insuffisance du module de compréhension.

Evaluation des différentes fonctions de transition

1. Fonction T_d :

La complexité moyenne de cet algorithme est, nbfaces étant le nombre de faces qui composent la scène : k x nbfaces x log(nbfaces)

2. Fonction POS :

L'étude n'est ici faite que pour les éléments intersectants de la fenêtre mère. Ces éléments sont indiqués grâce à un historique des faces, dont la consultation est proportionnelle à nbfaces. I_m sera le nombre de faces restantes à étudier.

Soit T_i le temps d'exécution de C_c^c appliqué à une arête

Soit T_c le temps d'exécution de C_c^p appliqué à une arête

I_e le nombre d'éléments englobants par rapport à la fenêtre

I_d le nombre d'éléments disjoints par rapport à la fenêtre

a_i le nombre d'arêtes d'un polygone P_i

T le temps d'exécution du module d'analyse, sera :

$$(T_i + T_c) \times \varepsilon\, a_i + T_i \times |I| <= T <= (T_i + T_e) \times \varepsilon\, a_i - T_c \times |I|$$
$$I_e \cup I_d$$

Cette formule indique que T dépend :

- du nombre d'arêtes par polygone ;
- du nombre d'intersectants présents dans la fenêtre mère ;
- de la proportion d'englobants et de disjoints déterminés dans la fenêtre fille.

En fait, il se peut que le nombre de faces étudiées soit inférieur à I_m et cela pour deux raisons :

- si dans la fenêtre mère un cache a été trouvé, tous les éléments situés derrière ne seront pas traités dans l'étude des fenêtres filles.

- si un cache est trouvé dans l'étude de la fenêtre fille, les éléments situés derrière ne seront pas traités, mais simplement marqués comme invisibles (ceci étant rendu possible, car les éléments de MD sont triés).

Ce facteur de simplification dépend de la présence d'un englobant et du numéro d'ordre de cet élément, c'est-à-dire de son éloignement par rapport à l'observateur.

3. Fonction PRØ

Ce test est appliqué aux éléments intersectants et englobants d'une fenêtre. Soit T_d le temps de calcul d'une intersection d'un plan et d'une droite, T_c le temps de comparaison de deux réels, T temps d'exécution de PRØ

$$T <= 4 . [|I_e| . T_d + (I_e - 1) \times T_c] + 4 . [2|I_p| \times T_d + |I_f| \times Tc)$$

Cette borne max proportionnelle au nombre d'intersectants et au nombre d'englobants, est en général très supérieure au temps d'exécution réel, car :

a) le nombre d'intersectants, d'englobants, est moindre, car le module d'analyse a éliminé ceux rendus invisibles par la présence d'un cache.
b) ces tests sont appliqués tant qu'un échec n'a pas été déterminé. Ceci dépend donc de l'éloignement de l'élément qui provoquera l'échec. Ce temps de traitement est donc proportionnel à la complexité de la scène visible, plutôt qu'à la complexité de la scène.

<u>Résumé des différentes caractéristiques de la scène qui influencent le temps de traitement</u> :

- nombre de faces à traiter (tri, module d'analyse)
- nombre d'arêtes par polygone (tests réalisant la fonction POS).
- surface globale de la scène projetée (choix d'une fenêtre initiale dans le module contrôleur).

- présence de un ou plusieurs groupes d'éléments dans la scène. Un groupe d'éléments est formé de faces dont la visibilité est liée (module contrôleur).

- forme, orientation des éléments les plus proches d l'observateur. Ceci pour exprimer la probabilité qu'a une face, d'être englobante par rapport à une fenêtre. L'algorithme sera plus performant si la surface moyenne projetée des faces visibles est plus grande. Ceci intervient dans la décision prise dans le module de compréhension.

La figure 2.26. présente plusieurs exemples :

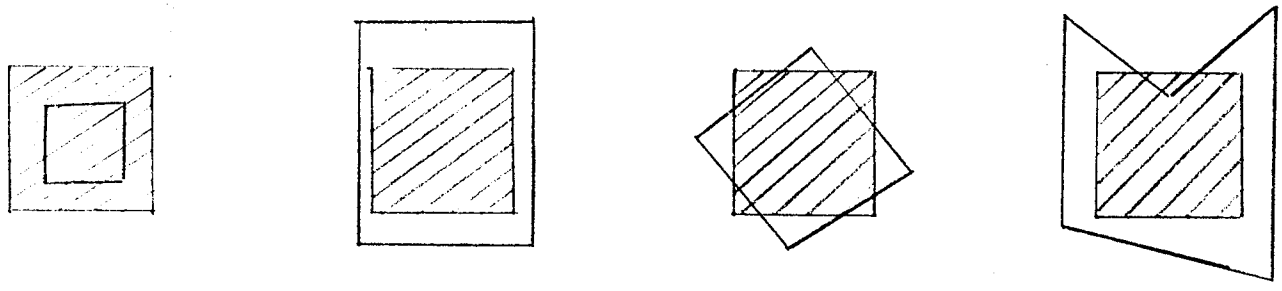


fig. 2.26.

- influence des éléments cachés si la face la plus en avant est un intersectant ;

- répartition des éléments en x et en y, qui influe sur le nombre de fenêtres à traiter ;

- répartition en z des éléments qui influe sur le traitement par le module d'analyse d'une fenêtre. Le temps de traitement dans ce module pouvant être réduit si un cache est trouvé parmi les premiers éléments traités.

- présence de faces qui se pénètrent (test PRØ).

2.2.4. Conclusion :

On peut porter au sujet de cet algorithme des jugements sur différents points :

- programmation : cet algorithme ne présente aucun problème de programmation ; Il peut être programmé aisément de façon récursive.

- il permet de traiter des faces qui se pénètrent et des polygones plan quelconques, mais ne possèdent pas de trou.

- on peut obtenir deux types de sortie :

  . dessin au trait, avec affichage des arêtes d'intersection ;

  . image.

Les désavantages de cet algorithme sont :

- une qualité du dessin au trait médiocre ;

- si le temps d'exécution a été amélioré en grande part du fait des améliorations apportées au contrôleur et des nouveaux cas de compréhension, il reste très élevé, ce qui ne permet pas d'envisager des applications en temps réel, dans le cas d'une réalisation par logiciel.

- dans le cas de la production d'une image, le phénomène de transparence n'est pas rendu.

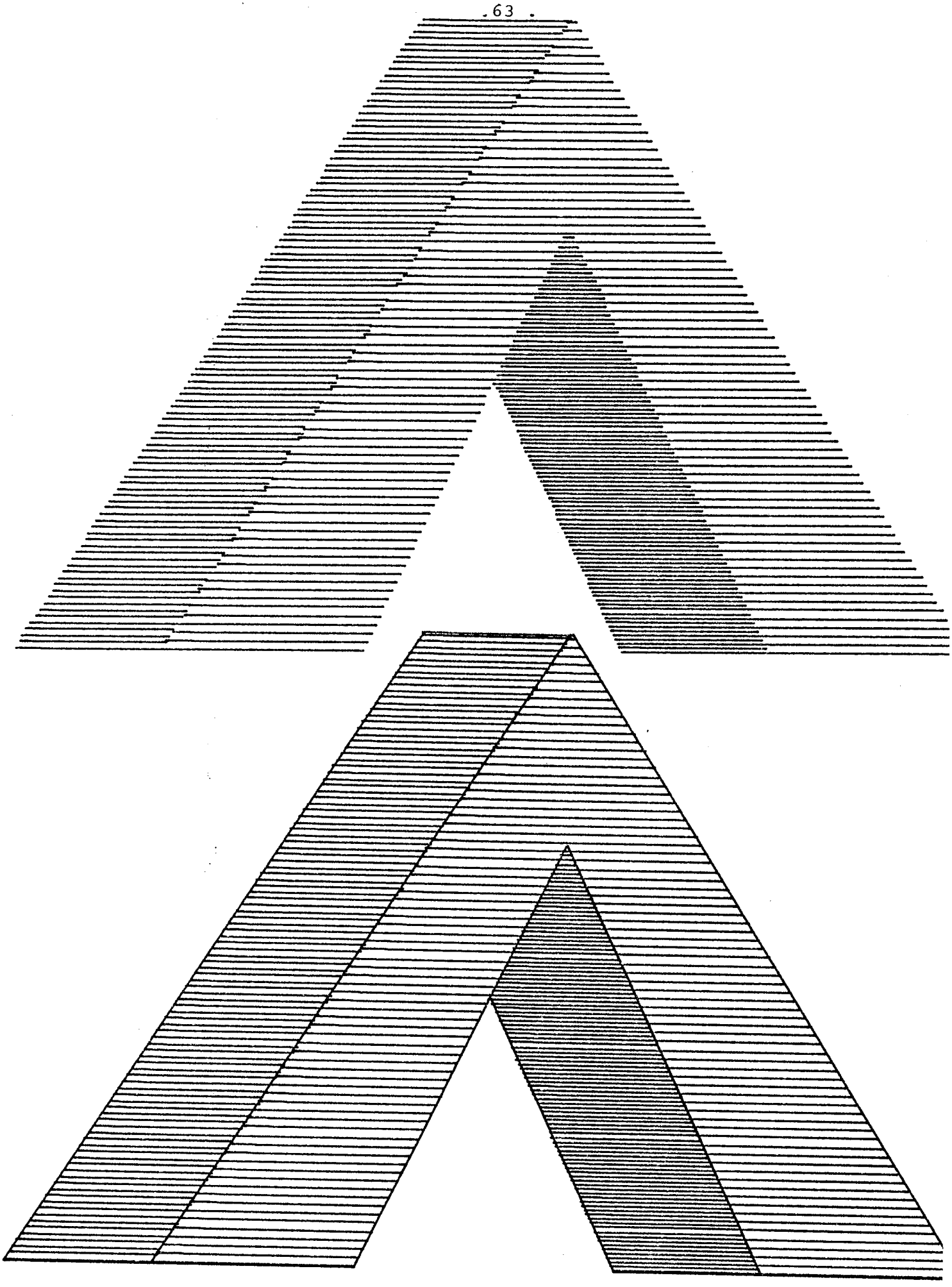


Fig. 2.27. Scènes traitées par l'algorithme de Watkins

2.3. L'algorithme de Watkins

2.3.1. Description

Cette famille d'algorithmes postérieure à l'algorithme de Warnock doit son origine au travail de Watkins, Romney et Evans ([RWE 69])qui publiaient en 1969 un article où la préoccupation de génération d'images en temps réel apparaissait pour la première fois. A partir de cet article d'autres furent publiés qui proposaient des améliorations, d'autres techniques, des méthodes de production de dessin au trait.

Pour étudier la visibilité dans le parallélépipède défini par l'écran et l'axe OZ on détermine pour chaque plan perpendiculaire à l'axe OY l'ensemble des segments ou portions de segments (voir fig. 2.28.).

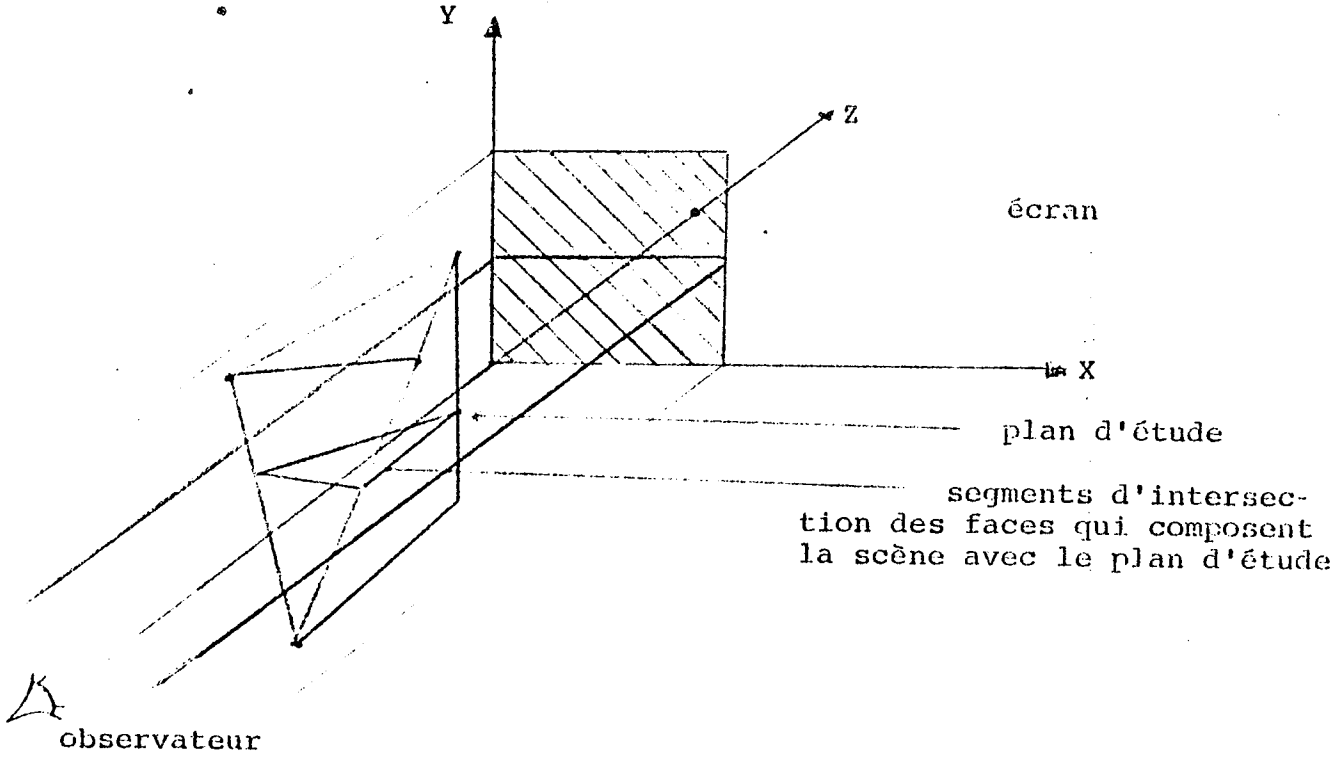


Fig. 2.28.

L'intérêt porté à ce type d'algorithme vient en grande part du modèle de visibilité qui est fourni car les éléments visibles sont produits dans un ordre qui est compatible avec un affichage de type balayage télévision

technologie qui a connu un grand essor.

2.3.2. Etude des différents éléments de l'algorithme

2.3.2.1. MD : Modèle de description

On présente ici les éléments d'entrée de l'algorithme ainsi que les informations mémorisées pour la suite du traitement.

Caractéristiques communes aux modèles des différents algorithmes :

Les éléments d'entrée sont des faces polygonales planes fermées, pouvant posséder des trous.

Soit F élément de MD dont le plan de support P, a pour équation : $A.x + B.y + C.z = d$.

Soit Si le segment obtenu par intersection de F et du plan $y = i$. Dans ce plan le segment a une pente $p = \Delta X/\Delta Z$ qui sera la même pour tous les plans où le segment sera présent. On ne calculera cette pente qu'une fois : $p = -C/A$.

La contrainte de face plane pourrait donc être supprimée si on calcule par contre à chaque plan d'étude et pour tout segment sa pente $\Delta X/\Delta Z$. Ceci permettrait de traiter des surfaces du type $y = f(x, z)$, un maillage étant défini dans le plan Oxz.

Il est par contre nécessaire de disposer de contours fermés. Cette hypothèse permet pour toute face de déterminer le segment ou les segments provenant de l'intersection avec le plan d'étude.

Distinction entre les différents modèles de description:

Des différences apparaissent au niveau des éléments (propriétés des faces) et au niveau de l'ensemble lui-même.

Certains algorithmes n'admettent en entrée que des faces convexes, d'autres n'imposent pas de restrictions de ce type. Il est à citer l'algorithme de Romney (cf. [RWE 69]) qui traite des faces triangulaires.

L'avantage de ce modèle de description est qu'il permet de minimiser les calculs dans la partie étude de visibilité. La distance R d'un point à une face est déterminée préalablement en fonction des sommets :

Soit (O_x, O_y, O_z) le point de vue

Soit (X, Y, Z) un point de l'espace appartenant à la face, on a la relation

$$a_1X + a_2Y + a_3Z = 1$$

Soit (I_x^i, I_y^i) la projection d'un sommet du triangle sur le plan de visualisation. On obtient : $i \in [1, 3]$

$$1/R_Z^i = a_1 I_x^i + a_2 I_y^i + a_3$$

$R_{z\bullet}^i$ étant la distance du sommet à l'observateur. On obtiendra pour tout point : $1/R_z = a_1 I_x + a_2 I_y + a_3$.

Les comparaisons de profondeur se feront à l'aide des coefficients (a_i) évitant ainsi les divisions, opérations coûteuses.

Les algorithmes qui seront présentés traiteront des faces au nombre de sommets quelconque car on évitera ainsi le problème de triangulation des faces qui est très coûteux en temps de calcul.

La seule distinction, en ce qui concerne les ensembles MD, est la possibilité d'avoir des éléments dans MD qui se pénètrent ou non. Cette possibilité joue un rôle important dans l'étude de visibilité, car la présence de faces pénétrantes nécessite des tests plus élaborés et entraîne donc des temps d'exécution plus importants.

Couleur associée aux faces

A chaque face est associée une couleur qui sera celle des segments visibles de cette face, s'il y en a.

Afin de rendre l'image plus réaliste on supposera qu'une source lumineuse est située à la position de l'observateur (ceci permet d'éviter le problème des ombres portées). L'intensité de la lumière incidente sur

une face est proportionnelle à $\cos^2\theta/R^2$. θ étant l'angle formé par une normale à la face et à la direction de vue. R est la distance de la face à la source lumineuse. (figure 2.29.).

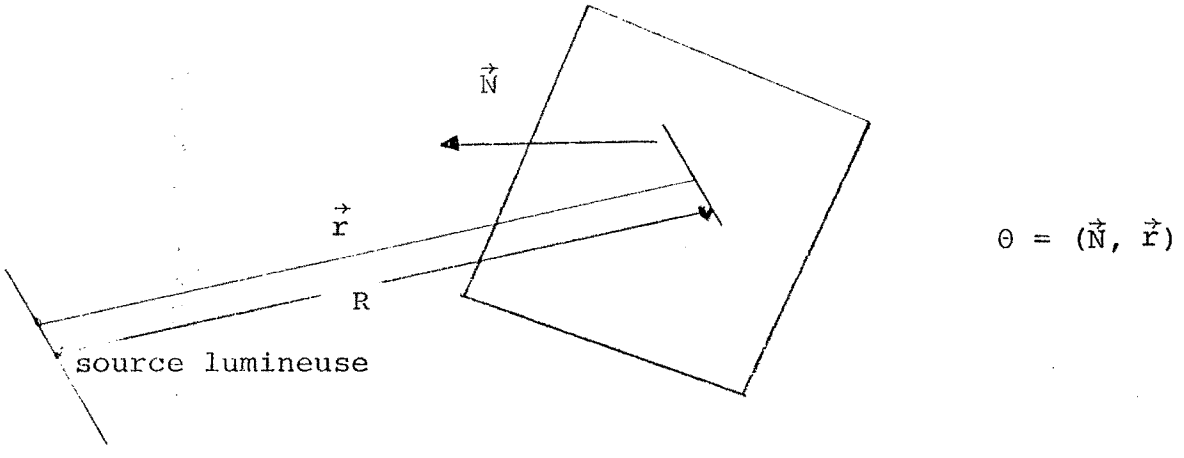


Fig. 2.29. Position de la source lumineuse.

Le vecteur normal à une face est calculé par le produit vectoriel de deux arêtes adjacentes.

L'application de cette méthode de modulation de l'intensité conduit à une meilleure compréhension de l'image qui sur un matériel possédant plusieurs niveaux d'intensité atteindrait un certain degré de réalisme. Les tests effectués ont été réalisés sur un matériel ayant un seul niveau d'intensité. Plusieurs niveaux de 1 à 4 ont été simulés en remplissant les surfaces visibles par des hachures plus ou moins espacées (figure 2.30.).

Fig. 2.30. Différentes vues d'un même objet

Structure de données

La structure de données utilisée pour la représentation des faces est la même que celle employée dans l'algorithme de Warnock (figure 2.31.).

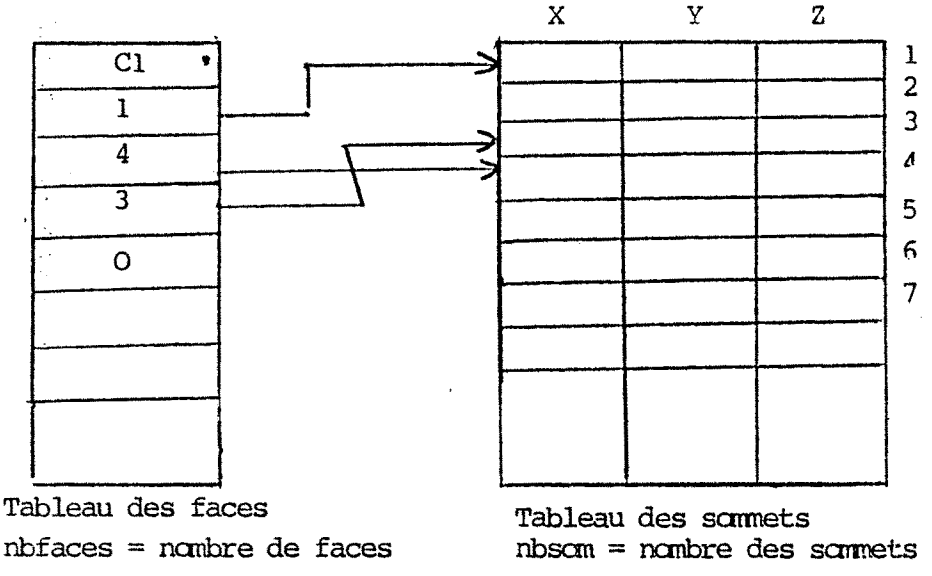


Fig. 2.31. Représentation du modèle de description.

2.3.2.2. TD : Tri initial des données

Un sens de balayage de l'écran étant choisi (de bas en haut pour les algorithmes décrits) on parcourt l'ensemble des arêtes afin de déterminer pour chacune le numéro de la ligne où elle apparaît pour la première fois.

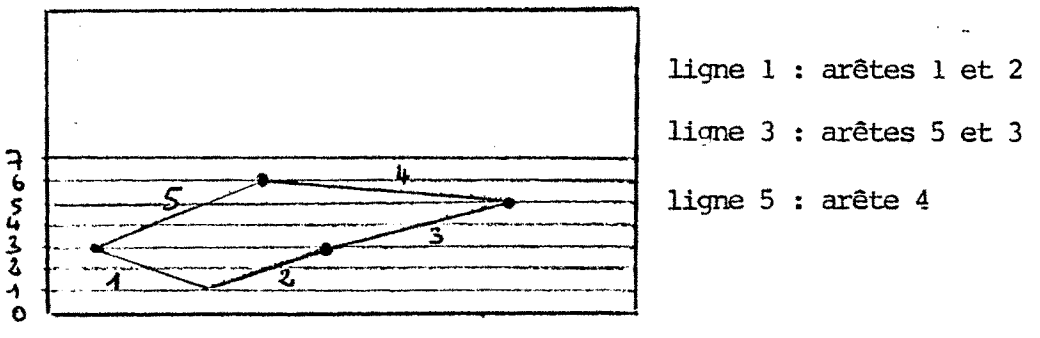


Fig. 2.32. Apparition des arêtes sur les lignes de balayage.

Pour chaque ligne on constitue la liste des arêtes qui sont présentes pour la première fois (figure 2.32.). Cette méthode permet d'éviter la consultation, à chaque étude, de l'ensemble des arêtes afin de déterminer les nouveaux éléments. On obtient donc un traitement en k x nbarêtes au lieu de 513 x k x nbarêtes, k étant le temps de traitement d'une arête.

Structure de données

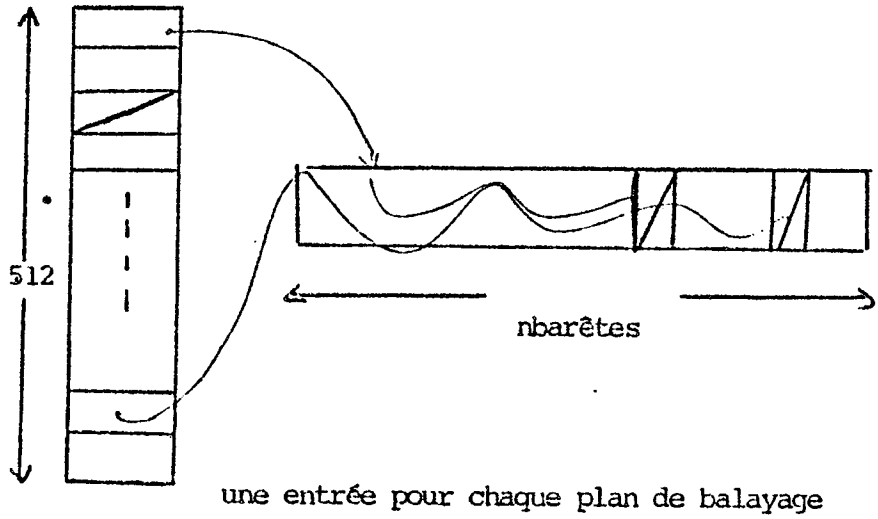


Fig. 2.33. Structure de données associée aux lignes de balayage.

2.3.2.3. Fonction stratégie

Cette fonction stratégie est présentée figure 2.34. :

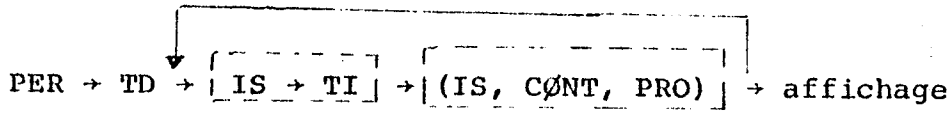


Fig. 2.34 Fonction stratégie

La première fonction de transition appliquée est PER, ce qui indique que l'algorithme travaille dans l'espace image.

Cette fonction se décompose en deux parties :

. calcul des éléments présents dans un plan (IS, TI) ;

. étude de visibilité dans un plan (IS, CØNT, PRØ).

La première partie est commune à tous les algorithmes qui diffèrent par contre par la seconde, l'étude de visibilité d'un ensemble de segments coplanaires.

2.3.2.4. IS : Calcul des éléments présents dans un plan d'étude

La méthode la plus évidente afin d'obtenir les segments contenus dans un plan d'étude serait de calculer pour chaque élément de MD son intersection avec ce plan. Le coût de cette méthode serait 512 x nbfaces et mettrait en oeuvre un algorithme de découpage coûteux.

La méthode utilisée par cette famille d'algorithmes tient compte d'une certaine cohérence de la scène lors du passage d'un plan d'étude au suivant (figure 2.35.).

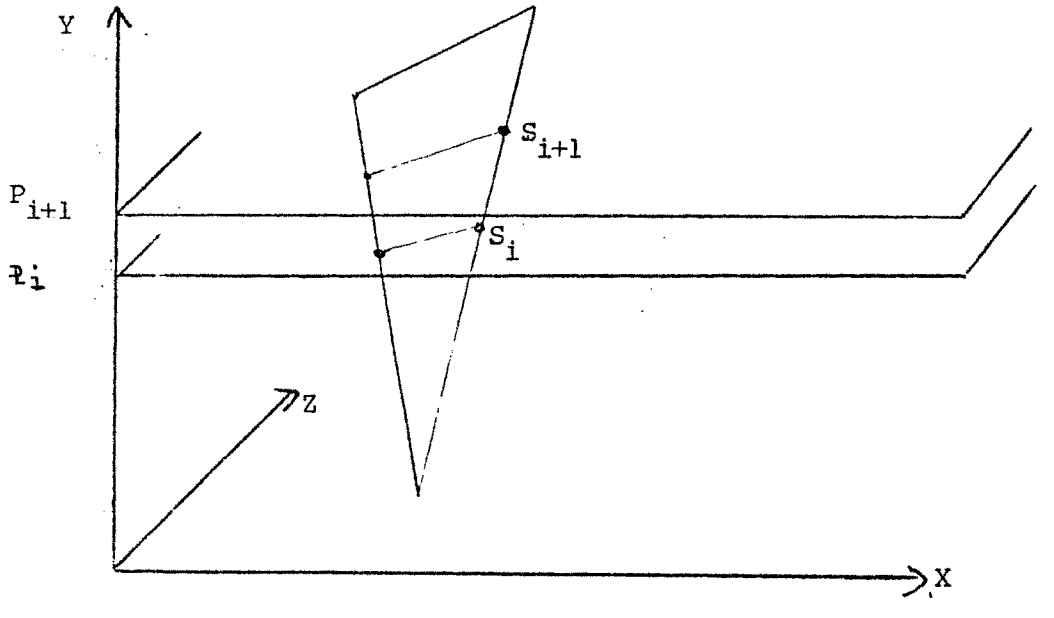


Fig. 2.35. Détermination des éléments présents dans un plan.

soit $S_i = (G_i, D_i)$ segment provenant de l'intersection de F avec P_i

on obtient $S_{i+1} = (G_{i+1}, D_{i+1})$

$$G_{i+1} = G_i + \frac{DX/DY}{DZ/DY}_G$$

$$D_{i+1} = D_i + \frac{DX/DY}{DZ/DY}_D$$

On obtient le nouveau segment à étudier à partir du segment précédent et des pentes des arêtes qui déterminent ce segment si ces dernières restent identiques au passage au plan suivant.

Les informations nécessaires pour la mise à jour des segments sont :

- extrémité gauche X, Z, DX/DY, DZ/DY, COMPT
- extrémité droite idem
- et le numéro de la face à laquelle le segment appartient : POLY.

La variable COMPT, négative dont la valeur absolue indique le nombre de lignes où l'extrémité aura les mêmes caractéristiques, c'est à-dire où elle appartient à la même arête.

Les actions effectuées avant l'étude de visibilité dans le plan sont :

1. mise à jour des segments présents dans le plan précédent.

   Pour chaque extrémité

   COMPT ← COMPT+1

   si COMPT = 0 alors POLY est un polygone changeant

   sinon X ← X + DX/DY

   Z ← Z + DZ/DY

2. insertion des nouveaux segments à partir de la liste construite en 1D.

Les polygones en jeu deviennent des polygones changeants. Les informations attachées aux nouveaux segments sont initialisés (figure 2.36.).

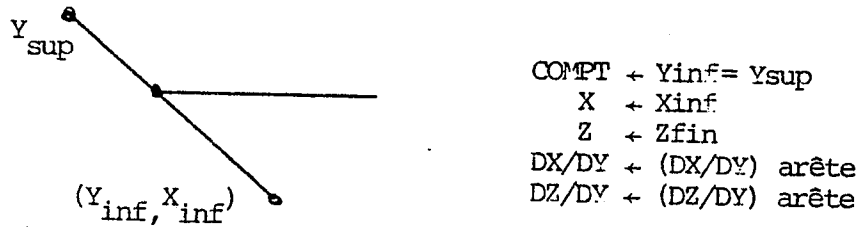


Fig. 2.36.

3. pour chaque polygone changeant, mise à jour de la liste des segments qu'ils forment du fait de leur intersection avec le plan d'étude.

La structure de données utilisée est schématisée figure 2.37.

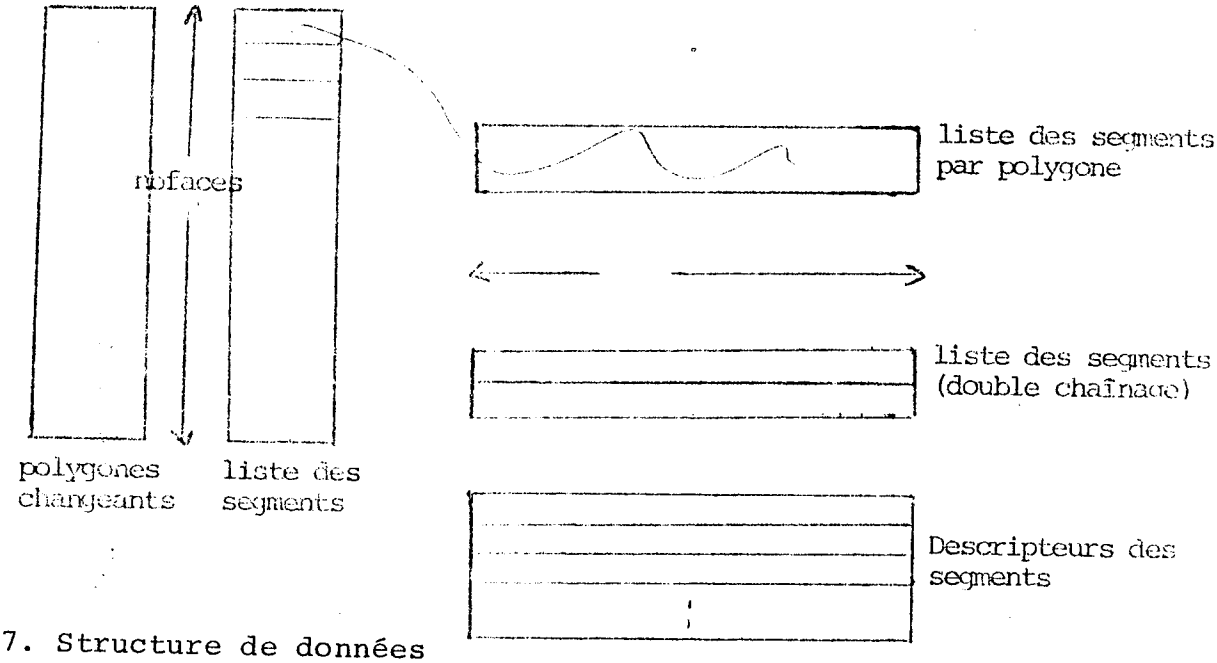


Fig. 2.37. Structure de données

Remarque :

Ces algorithmes travaillent dans l'espace écran, les points adressables ont donc des coordonnées entières. Les segments étudiés dans un plan peuvent donc être représentés par des variables entières. Ceci nécessite en fait une précaution lors de la mise à jour citée plus haut. (figure 2.38.).

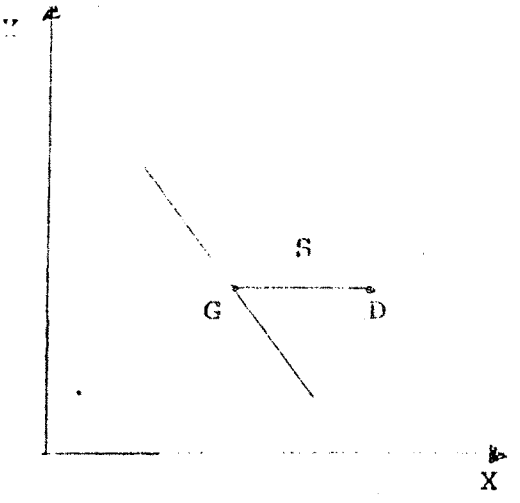


Fig. 2.38. Mise à jour d'un segment.

Soit le segment S. La variable X_G est un entier, lors de la mise à jour on aura : $X_G \leftarrow \text{Round } (X_G + (DX/DY)_G)$.
si la pente est inférieure à 0.5 X_G ne changera donc jamais de valeur ce qui entraînera une erreur dans le dessin obtenu, si cet élément est visible. Pour éviter ceci, on mémorise pour chaque extrémité une variable X^R, réelle telle que :

$$X^R \leftarrow X^R + DX/DY$$
$$X \leftarrow [X^R]$$

Conclusion

Pour chaque plan d'étude le nombre d'opérations est proportionnel au nombre de segments présents. De plus les opérations en jeu sont additions et tests à l'exclusion des divisions et multiplications coûteuses en temps d'exécution.

Ce procédé, caractéristique de ces algorithmes, explique en grande partie les performances obtenues, notamment en évitant le problème des méthodes de découpage (cf. Warnock).

2.3.2.5. TI : Tri des éléments présents dans un plan d'étude

Eléments à trier

A cette étape de l'algorithme on possède donc tous les éléments nécessaires pour réaliser l'étude de visiblité. La nécessité de cette étape supplémentaire vient de la manière d'étudier la visibilité dans un plan. En effet, tous ces algorithmes, à l'exception de l'algorithme "Fusion" procèdent par un balayage de gauche à droite du plan (figure 2.39.).

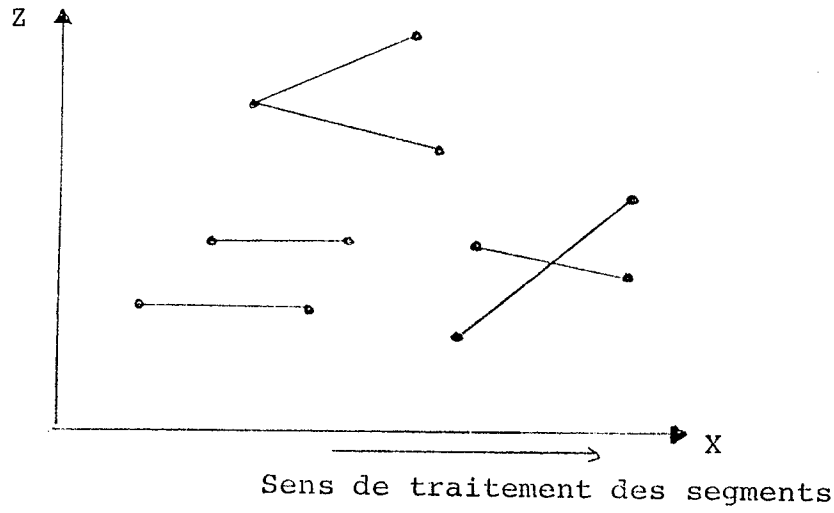


Fig. 2.39.

Les éléments à traiter, présents dans une liste double peuvent être de deux types suivant l'algorithme de visibilité utilisé par la suite :

ler cas : un élément est un segment qui sera représenté par le descripteur suivant :

X gauche
Z gauche
DX/DY gauche
DZ/DY gauche
X droit
Z droit
DX/DY droit
DZ/DY droit
Compteur gauche
Compteur droit
Polygone
Lien gauche
Lien droit

cf. algorithmes Boite, Fusion.

2ème cas : un élément est une extrémité de segment dont les descripteur est :

X
Z
DX/DY
COMPTEUR Cf. Algorithme Pile
Numéro segment
Polygone
Lien gauche
Lien droit
DZ/DY

La clé du tri sera :

- dans le premier cas : l'abscisse de l'extrémité gauche de l'élément ;
- dans le second cas : l'abscisse de l'élément.

Etat des données avant l'application du tri :

On peut distinguer deux types de données dans la liste des éléments à trier :

T1 : les éléments qui apparaissent pour la première fois dans ce plan d'étude

T2 : les éléments qui proviennent de l'étude du plan précédent, quasi ordonnés.

Les éléments de type T1 ont été ajoutés en tête de liste par la partie IS, précédente, de l'algorithme.

La présence des éléments de type T2 a conduit au choix du tri bulle dont on peut considérer que le comportement dans ce cas est en O(N).

Une autre raison de ce choix est que l'on considère que le nombre d'éléments du type T1 est faible devant le nombre total d'éléments. En fait cette considération n'est évidemment pas justifiée pour toutes les scènes Dans le cas d'approximation de surfaces gauches par des polygones, du fait du nombre important et de la taille réduite des faces, le nombre d'éléments nouveaux à chaque plan n'est pas négligeable. On pourrait envisager de trier les éléments de type T1 par un tri tel Quicksort, les éléments de type T2 par un tri bulle et de faire ensuite l'insertion des éléments en nombre moindre dans l'ensemble restant.

Remarque sur la clé associée au tri

Dans les algorithmes publiés aucune action de tri n'est prévue lorsque deux éléments ont la même clé. Or ce cas est très fréquent et on verra par la suite l'importance de ce fait dans les algorithmes de visibilité qui seront décrits.

2.3.2.6. Etude de visibilité

Nous donnerons ici le principe de différentes méthodes, dont l'étude et la réalisation seront décrites dans le paragraphe suivant.

Algorithme Fusion ([Luc 77]).

On dispose à un moment donné de la liste des éléments visibles et on compare un segment non encore étudié à cette liste. Initialement cette liste est constituée du segment de fond. En fin de traitement elle contient l'ensemble des segments ou portions de segments visibles (fig. 2.40)

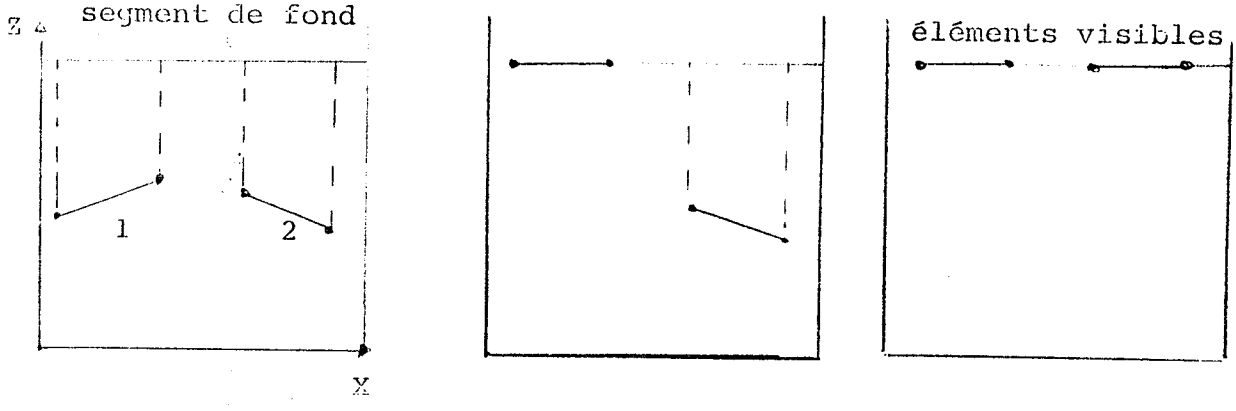


Fig. 2.40. Etude de visibilité par la méthode fusion.

Algorithme Pile (cf. [HaG 77])

L'étude de visibilité se fait par un balayage du plan de gauche à droite. Les éléments traités sont les extrémités des segments triés par valeur d'abscisse croissante. Le principe de cet algorithme repose sur la remarque que l'élément visible courant peut changer dans deux cas de figure:

1. on atteint la borne droite de l'élément visible. On doit choisir le nouvel élément à traiter (voir figure 2.41a).

2. le segment visible courant est caché par un nouvel élément. Ceci se produit lorsqu'on rencontre la borne gauche d'un nouveau segment (voir figure 2.41b).

L'algorithme décrit dans [HaG 77] permet de traiter des polygones convexes qui ne se pénètrent pas.

En fait, il est facilement modifiable pour admettre des polygones quelconques et on décrira les modifications à apporter pour tolérer dans MD des éléments qui se pénètrent.

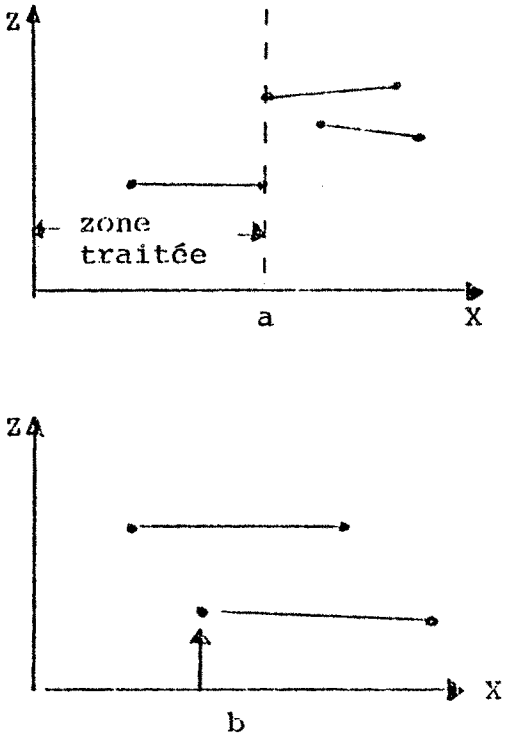


Fig. 2.41. Etude de visibilité par la méthode Pile.

Algorithme boite ([RWE 69], [NeS 73])

Le principe de cette méthode, qui présente beaucoup d'analogies avec l'algorithme de Warnock, est le suivant : l'ensemble de segments est étudié, si la configuration n'est pas comprise il est divisé en deux portions qui sont étudiées à leur tour, sinon les éléments visibles sont affichés. L'étude de visibilité se fait par rapport à la boite englobant les segments déjà étudiés, boite qui est comparée à un nouveau segment non encore pris en compte. La situation est comprise si le segment et la boite sont dans une des positions reconnues pour l'algorithme.

Algorithme point par point ([Luc 77]).

Le principe de cet algorithme est de posséder pour chaque point de la ligne de balayage les informations suivantes :

- couleur du point ;
- profondeur du point.

Cet algorithme, ayant l'avantage d'être une programmation très simple, est d'un coût élevé, qui provient de l'étude point par point des segments. Les algorithmes qui seront présentés étudient la visibilité du plan segment par segment réduisant ainsi le nombre de comparaisons à effectuer.

Nous donnons ici l'algorithme réalisant cette étude de visibilité, car il ne sera pas étudié dans les paragraphes suivants étant donné sa pauvreté du point de vue du comportement.

L'algorithme sera donc :

Pour I = 1 à nombre de segments faire

```
début
XDeb := Xd (i) ;
   XFIN := Xf(i) ; Couleur := Couleur(I) ;
   Pour X = Xdeb jusqu'à XFIN faire
   début
        Calcul (Z) ;
        Ranger (X, Z, Couleur)
   fin
fin ;
Ranger(X, Z, Couleur)
début
   si Z < Couleur (X) alors
   début
        Crête(X) := Z ;
        Couleur(X) := Couleur
   fin
fin ;
```

2.3.3. Etude de visibilité

2.3.3.1. Algorithme Fusion

Les éléments traités (segments contenus dans le plan d'étude) sont représentés dans une liste double ordonnée telle que :

$$S_i < S_j \qquad x_i^1 < x_j^1$$

Ils sont comparés à la liste des éléments visibles (liste double). Cette liste est telle que les projections des éléments sont jointives : une droite x = cte coupe un élément ou deux (figure 2.42.).

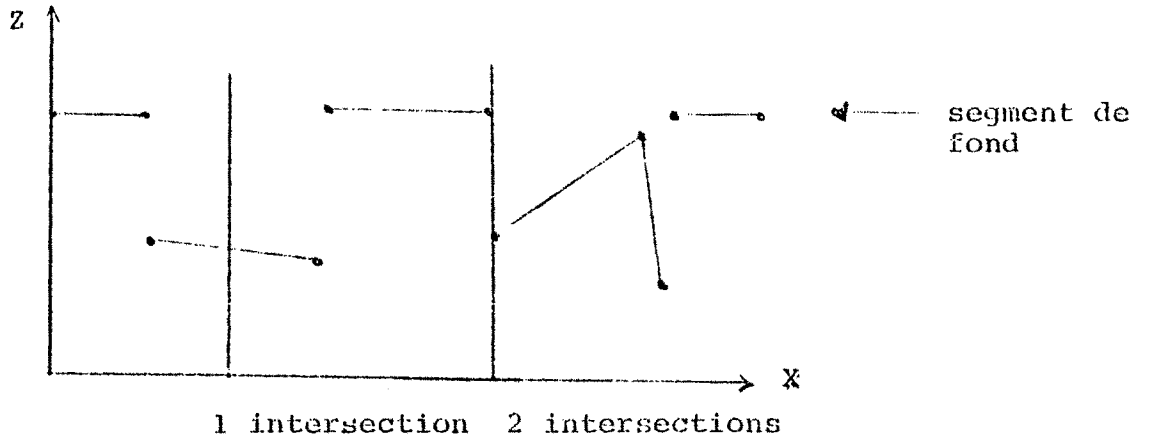


Figure 2.42.

La comparaison d'un segment à la ligne de crête (liste des éléments visibles) se fait par l'application des deux tests IS et PRO.

Test_IS : soit S1 le segment à comparer

$S_1 = (x_1, z_1)\ (x'_1, Z'_1)$ et S_2 le premier segment de ligne de crête en conflit avec S_1 $\quad S_2 = (x_2, z_2)\ (x'_2, z'_2)$ et $x'_2 \geq x_1$.

Les quatre configurations possibles sont données figure 2.43.

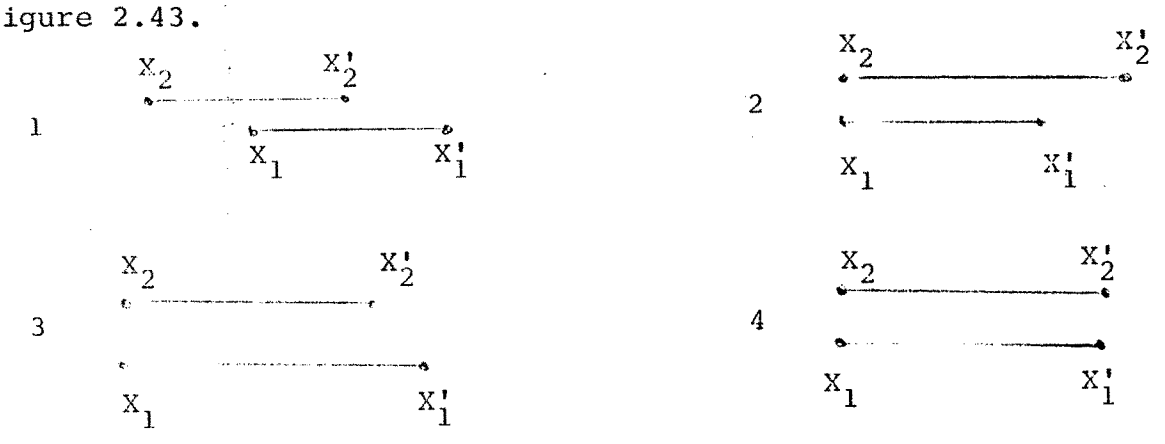


Fig. 2.43. Configuration pour deux segments.

Le test IS est appliqué aux deux segments afin de se ramener à la configuration 4.

Test_PRO : On applique alors le test PRO afin de déterminer quel segment ou portions de segments sont visibles. Quatre cas se présentent qui sont répertoriés figure 2.44.

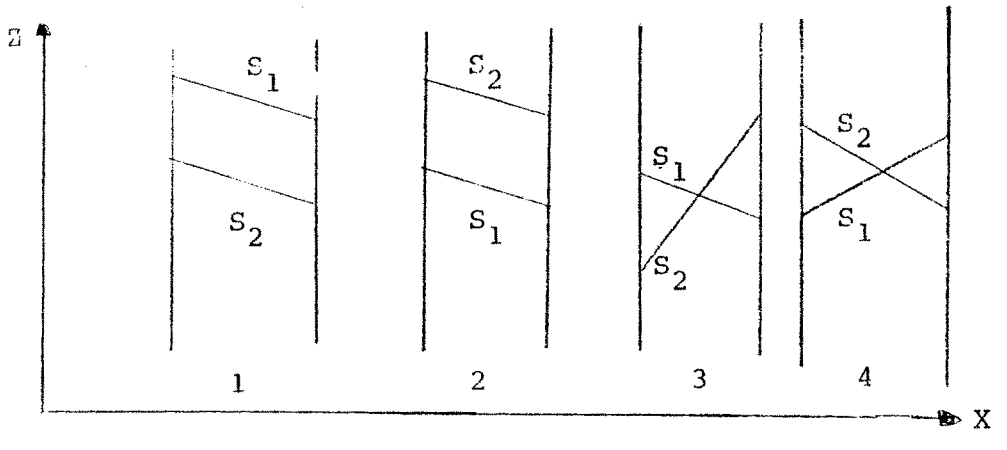


Fig. 2.44. Influence entre segments.

en 1 : S_1 est invisible

en 2 : S_1 est visible

en 3 et 4 : intersection des deux segments. On détermine le point d'intersection et on insère les portions de segments dans la ligne de crête, l'ordre dépendant du cas.

Le testPRO nécessite deux tests de profondeur (test sur des variables réelles) car on admet les faces qui se pénètrent.

Lorsque tous les éléments de la liste ont été traités la ligne de crête contient les éléments visibles qui peuvent être alors affichés.

Tri des données

Les segments à étudier sont triés sur leur extrémité gauche par valeur croissante de l'abscisse, mais on remarque que le principe de l'algorithme ne nécessite pas ce tri.

Nous avons donc fait différents essais pour les mêmes données mais traitées dans les différents ordres possibles.

L'ensemble de segments de la figure 2.45. étudiés dans les 6 ordres possibles a donné des écarts dans les temps d'exécution très importants.

ordre 1-2-3 : t_1 ; 2-1-3 : t_2

avec $t_2 = 1,35.t_1$

On constate une différence pouvant aller jusqu'à 30 % entre les temps d'exécution.

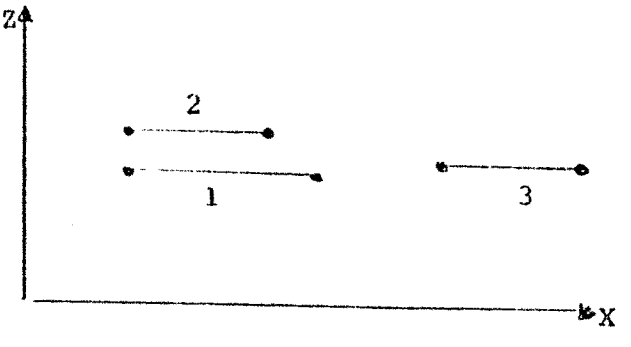


Fig. 2.45. Ordre de traitement des segments.

On peut distinguer, en fait, deux types d'opérations dans l'étude de visibilité :

- la comparaison des segments avec les éléments visibles à un moment donné ;
- les opérations d'insertion dans la ligne de crête.

Le tri permet de n'insérer dans la ligne de crête que des segments ou portions de segments qui auront une forte probabilité d'être les éléments visibles de la scène. Ceci minimise donc les deux types d'opérations cités plus haut.

La clé du tri qui est effectué est : (voir figure 2.46)

$$S_i = (x_i^1, z_i^1)\ (x_i^2, z_i^2)$$

$$S_j = (x_j^1, z_j^1)\ (x_j^2, z_j^2)$$

$$S_i < S_j \longleftrightarrow x_i^1 < x_j^1$$

ou

$$(x_i^1 = x_j^1) \text{ et } ((z_i^1 < z_j^2) \text{ ou } ((z_i^1 = z_i^2) \text{ et } (p_i < p_j)))$$

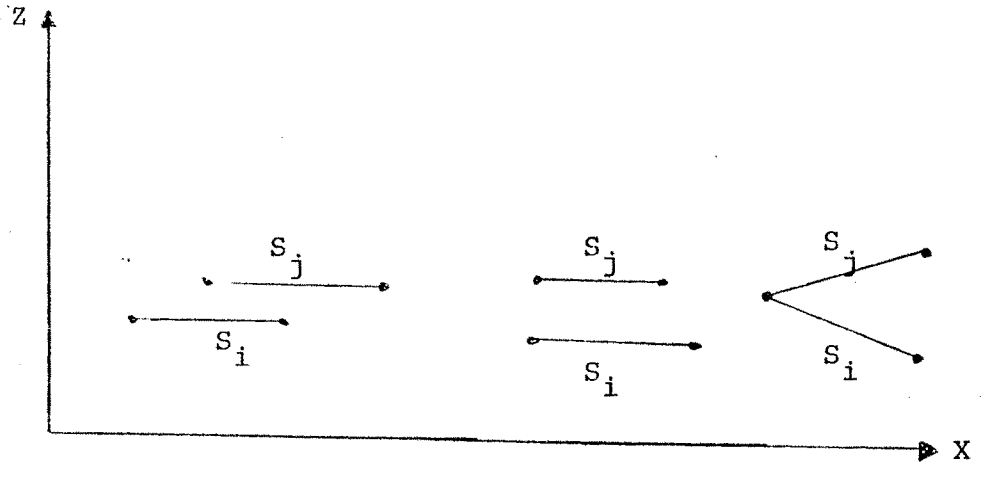


Fig. 2.46. Tri des segments.

Conclusion :

Malgré l'influence importante du tri des données sur le temps d'exécution cet algorithme a des performances médiocres comparé aux autres algcrithmes étudiés.

Ceci s'explique par le fait qu'à un moment donné subsistent dans la ligne de crêtes des éléments qui ne seront pas des éléments visibles. Ceci ayant impliqué des opérations d'insertion dans la liste et l'application du test IS qui sont coûteuses.

2.3.3.2. Algorithme Pile

Cette méthode d'étude de visibilité présente à priori un avantage par rapport à la méthode boite qui est sa simplicité de programmation mise en évidence par la description de l'algorithme ci-dessous. Nous verrons par la suite quelles sont les caractérisitiques de cette méthode.

Algorithme

Initialisations ;

/* segcour indique le segment courant, segsuiv le segment suivant à traiter, gauche logique qui indique quelle extrémité de ce segment est traitée.Initialement segcour : numéro du ler segment */

/* Xdroit donne l'abscisse de la zone traitée de 0 à Xdroit */ while change do

begin

/* recherche de l'élément suivant, mise à jour de segsuiv, gauche */

Elément suivant ;

If il y a élément then

begin

Afficher segcour de Xdroit à extrémité elemsuiv ;

Xdroit := extrémité elem suiv ;

end ;

```
      if segcour = 0 then segcour := segsuiv ;
      if ¬ gauche then
      begin
        /* Fin du traitement du segment numéro segsuiv */
           Marque (segsuiv) := true ;
           if segcour = segsuiv then
           begin
        /* le segment visible est traité */
           Depiler jusqu'à trouver un élément non marqué ;
        /* si il n'y en a pas segcour = 0 c'est-à-dire segment de fond */
        end ;
      end
      else
      begin
          /* nouveau segment à traiter, on le compare au segment courant */
      if segsuiv ¬= segcour then
      begin
           if segsuiv plus près que segcour then
           begin
              empiler (segcour) ;
              segcour := segsuiv
           end
           else empiler (segsuiv)
      end
      end
   end
   else change := false
end ;
/* fin de l'algorithme */
```

Test de profondeur

Ce test est effectué chaque fois que l'on traite une extrémité gauche, ceci afin de la comparer à l'élément visible courant (segcour dans l'algorithme).

Dans la version proposée par [HaG 77] le test effectué se fait par calcul de la profondeur du segment courant pour la valeur de l'abscisse de l'extrémité traitée. En fait on peut distinguer les trois cas de la figure 2.47.

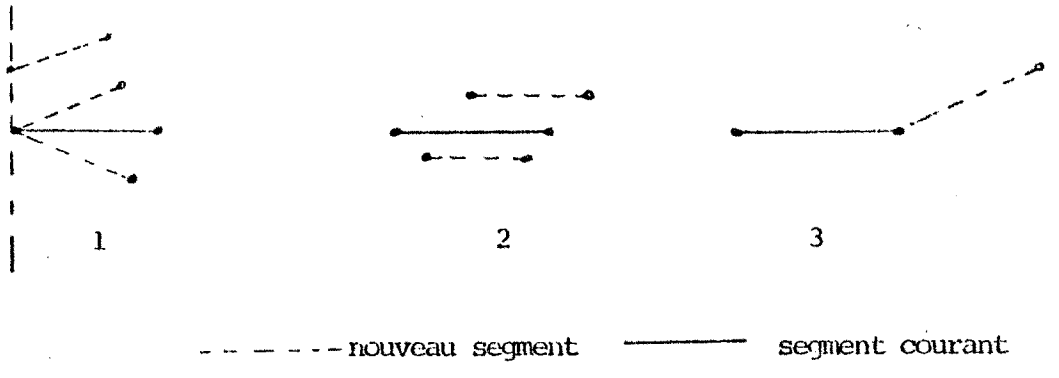


Fig. 2.47. Positions segment courant et segment suivant

Dans le cas 1 le nouveau segment ne peut être que caché du fait du tri qui est effectué sur les données (cf. description du tri dans l'algorithme 'FUSION).

Dans le cas 2 on doit effectuer un test de profondeur afin de déterminer quel est l'élément visible.

Dans le cas 3 le segment courant est visible jusqu'à son extrémité droite et le segment suivant devient le segment courant visible. Aucun empilement n'est effectué et le tri permet d'assurer que le nouvel élément est bien le segment courant visible.

Il est à remarquer l'importance du tri qui permet de distinguer les différents cas et par là de simplifier et d'accélérer le traitement, entre autres en évitant empilements et tests de profondeur inutiles

comme c'était le cas dans la version initiale (gain de temps d'environ 30 %).

Evaluation

Les différentes actions pouvant être réalisées dans cet algorithme sont :

- affichage d'un segment ou portion de segment ;
- recherche d'une extrémité ;
- comparaison de profondeur ;
- empilement, dépilement d'un segment.

Soit nbseg le nombre de segments présents dans le plan d'étude

- recherche d'une extrémité : $(2 * nbseg-1) * C_1$
- affichage : un affichage est réalisé pour chaque nouvelle extrémité traitée sauf lorsqu'il y a égalité d'abscisse avec la précédente. nbaff = 2 * nbseg-1 (nombre d'égalité entre extrémités).
- test de profondeur : un tel test est effectué pour toute extrémité gauche de segment sauf pour la première : $(nbseg-1) * C_3$
- empilement : lors du traitement de l'élément courant tous les éléments en conflit sont empilés. Un segment

  $S_1 = (x_1, z_1)\ (x_1', z_1')\ x_1 < x_1'$ est en conflit avec

  $S_2 = (x_2, z_2)\ (x_2', z_2')\ x_2 < x_2'$ si

  $$x_2 < x_1 < x_2'$$

soit Ω l'ensemble des segments ou portions de segments visibles

$S_i \in \Omega$ α_i empilements

d'où le temps $(\sum_{S_i \in \Omega} \alpha) * C_4 \leq (nbseg-1) * C_4$

de même pour les dépilements

d'où évaluation : $(2 * nbseg-1)) * C_1 + nbaff * C_2 + (nbseg-1) * C_3 + (\sum_{S_i \in \Omega} \alpha_i) * (C_4 + C_5)$

Le facteur déterminant pour le temps d'exécution sera donc le nombre de conflits entre segments, qui dépend de la disposition en x des extrémités.

Cas des faces pénétrantes

Les modifications à apporter à la version actuelle de l'algorithme se situent dans la partie qui traite une extrémité gauche pour les cas 1 et 2 (figure 2.48.).

Cas 1 Cas 2

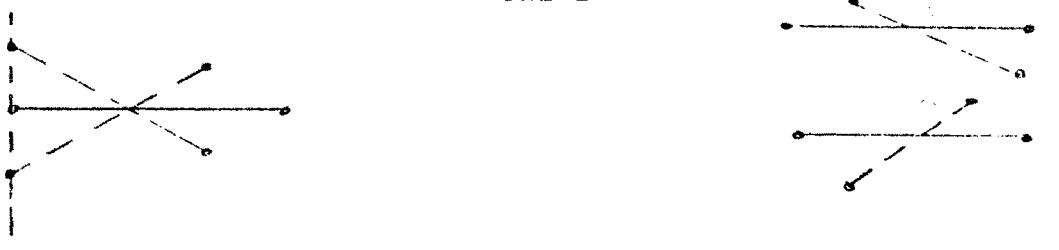


Fig. 2.48. Cas d'intersection de segments.

On affichera l'élément visible jusqu'au point d'intersection et si l'élément courant est caché on l'empile et l'élément nouveau devient courant sinon on empile le nouvel élément.

Si ces modifications ne posent aucun problème pour leur réalisation elles pénalisent l'algorithme du fait des tests supplémentaires à effectuer. Le coût de cet algorithme, calculé à partir de l'évaluation précédente sera :

$$(2*nbseg-1)*C_1+nbaff'*C_2+\underline{((nbseg-1)*2C_3+n_2*C_6)}+(\sum_{S_i \in \Omega} \alpha)*(C_4+C_5)$$

n_2 : est le nombre d'intersections rencontrées

C_6 : temps de calcul de l'intersection de deux segments (6 multiplications, 3 soustractions, 2 divisions).

nbaff' = nbaff + n_2.

On remarque donc l'importance du cas des faces pénétrantes pour les performances d'un algorithme d'étude de visibilité.

Conclusion

Le premier avantage de cet algorithme est sa programmation très simple et son peu d'encombrement comparé aux deux autres algorithmes cités. Des trois algorithmes il s'est révélé être le plus performant. Un avantage de cet algorithme est qu'il n'effectue aucun calcul sur les éléments affichés. Les extrémités des éléments visibles sont celles de la structure de données et ne sont donc pas entachées d'erreur de calcul ce qui donne une meilleure qualité du dessin. D'autre part cet algorithme peut être facilement modifié afin de traiter les faces qui se pénètrent, mais dans ce cas il n'affichera pas l'arête d'intersection des faces en jeu.

2.3.3.3. Algorithme Boîte

De manière analogue à l'algorithme de Warnock, on peut distinguer dans l'analyse de visibilité du plan, trois modules :

- module d'analyse ;
- module de décision ;
- module de contrôle.

Module d'analyse

Il détermine quels sont les segments dans l'empan étudié (test CONT) ou portions de segments (tests IS, CONT). Tout nouveau segment est comparé aux autres segments déjà présents et qui caractérisent une boîte définie par trois variables :

- compte : nombre de segments contenus dans la boîte ;
- coordonnées des sommets de la boîte ;
- type : indique la présence de 2 segments écrans.

L'étude du nouveau segment sera faite en fonction de la variable compte :

- compte = 0 C'est le premier segment ; on ajuste la boîte sur

ce segment et compte est mis à 1 ;

- compte = 1 Plusieurs cas sont distingués :

. le nouveau segment est devant la boîte (figure 2.49a)

Ceci nécessite donc deux tests de profondeur. Dans ce cas, l'ancien segment est oublié et on ajuste la boîte sur ce nouveau segment.

. le nouveau segment est caché par la boîte (figure 2.49b)

. les deux segments traversent l'empan. On mémorise le point d'intersection et type est mis à 1. Compte est mis à 2 et la boîte est ajustée sur les deux segments.

. Autres cas : type est mis à zéro et compte à 2.

La boîte est ajustée (figure 2.49c).

- compte est supérieur à 1

Deux cas sont détectés :

1. Le nouveau segment cache la boîte. On détermine pour ceci que le segment s'étend de part et d'autre de la boîte et on fait un test de profondeur aux extrémités gauche et droite de la boîte. On oublie alors les anciens segments, la boîte est ajustée sur le nouveau et compte est mis à 1.

2. Autres cas : (figure 2.49d).

On remarque que la comparaison segment-boîte met en jeu des tests suffisants et des erreurs sont donc commises (voir figure 2.49e).

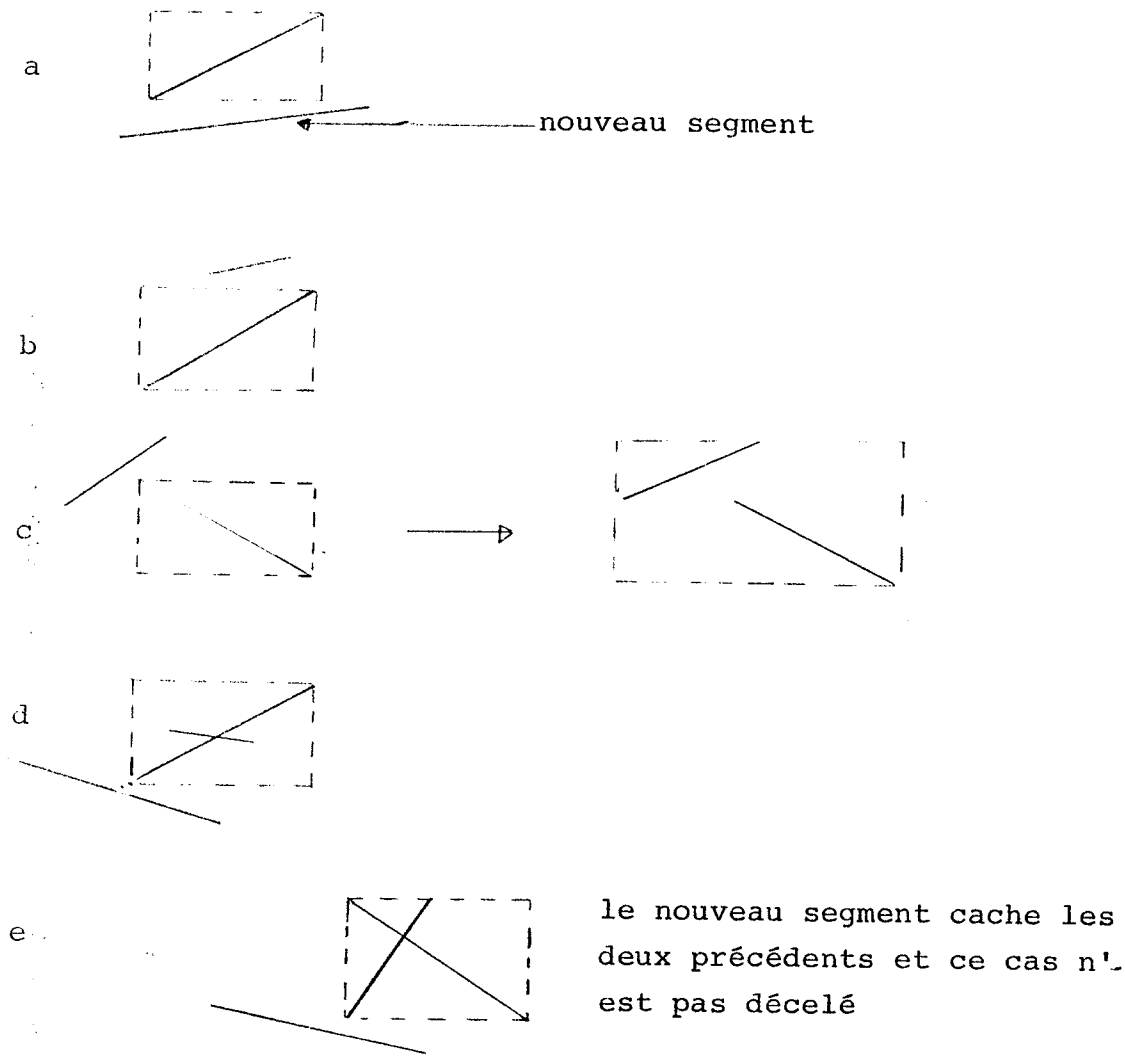


Fig. 2.49 Configurations segment-boîte.

Module de décision

A partir des informations préparées par le module d'analyse celui-ci va décider si l'affichage est possible ou si l'empan doit être divisé.

Quatre cas sont reconnus :

- compte = 0 il n'y a rien à afficher
- compte = 1 un segment seulement est visible. Il est affiché.

- type = 1 une intersection a été décelée. On affiche les deux portions de segments visibles.
- compte > 1 et type = 0 on fait appel au module contrôleur pour diviser l'empan.

Module de contrôle

Le module de décision a déterminé un échec. D'autre part le module d'analyse a mémorisé, si il y en a, une extrémité de segment ou un point d'intersection. La division de l'empan sera faite en ce point ou au milieu si il n'y en a pas.

La cohérence de la scène lors du passage d'un plan au suivant est utilisée en mémorisant les points de divisions qui ont conduit à un succès. Ceux-ci seront mis à jour au plan suivant et utilisés pour l'étude de visibilité.

Précision des calculs

Cet algorithme travaille dans l'espace écran où les points adressables ont des coordonnées entières (512 x 512). Or les sommets décrivant les faces ont des coordonnées réelles, sur lesquelles l'étude de visibilité est faite. La précision des calculs est donc disproportionnée par rapport au résultat. Les sommets sont donc approchés par des points aux coordonnées (X, Y) entières. Ceci a permis de diviser les temps de calcul par trois et ceci pour deux raisons :

- les opérations sur des variables entières sont moins coûteuses (rapport de 1 à 3 avec les opérations sur des réels).
- le nombre d'empans étudiés dans un plan est réduit. En effet certains recouvrements entre segments sont éliminés du fait du passage en coordonnées entières (voir figure 2.50).

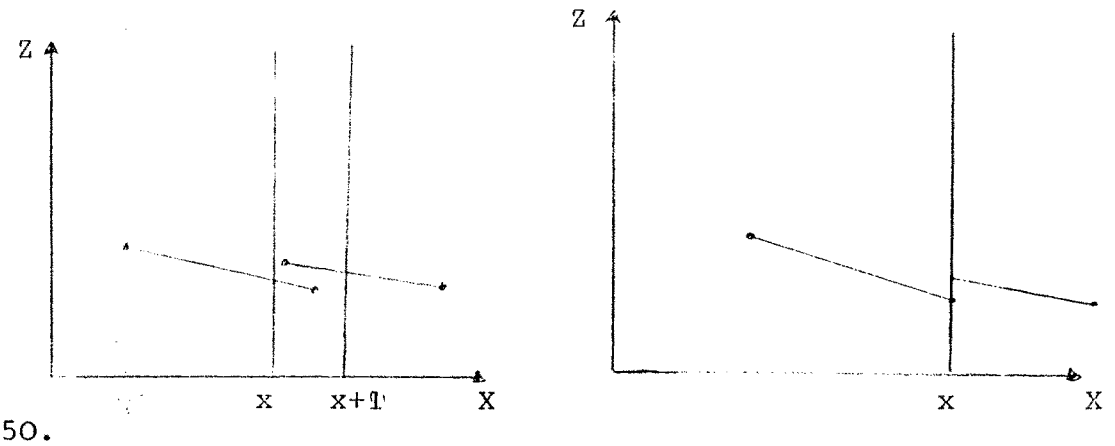


Fig. 2.50.

Si une évaluation du comportement de cet algorithme est difficile (une majoration en O ($NBSEG^2$)) à obtenir il permet de traiter les scènes dans des temps appréciables notamment si on compare les temps d'exécution de l'algorithme de Warnock pour des mêmes scènes (6 à 7 fois plus importants).

Dans l'exemple de la figure 2.51. en faisant une rotation de la scène autour de l'axe vertical on obtient des temps suivants :

- algorithme Watkins initial de 6 à 8 secondes ;
- algorithme Watkins travaillant sur des variables entières : de 3 à 3.7 secondes ;
- l'algorithme pile donne des temps de 1 à 1.4 secondes.

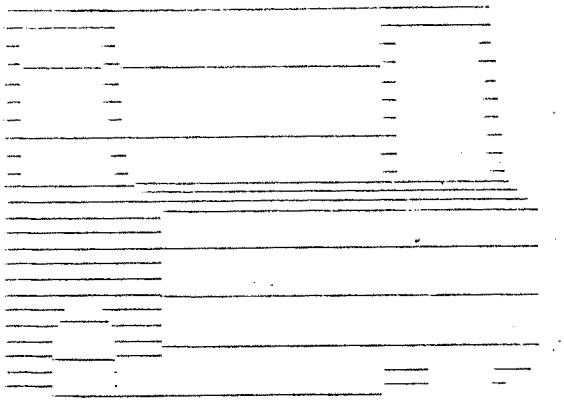


Fig. 2.51. Exemple de scène.

2.3.3.4. Comparaison des algorithmes Pile et Boîte

Un ensemble de scènes a été traité par ces deux algorithmes ce qui a permis d'obtenir un ensemble de mesures qui sera étudié dans ce paragraphe.

Description des scènes traitées .

Les scènes sont composées de n carrés (dans l'exemple n varie de 1 à 5) initialement aux côtés parallèles aux axes du repère défini dans le plan. Les scènes initiales subissent une rotation d'un angle x (x prendra les valeurs 15 30 45 60) ce qui permettra d'étudier le comportement des algorithmes vis à vis de deux points :

. répétition d'un même motif (le carré)

. transformation géométrique de la scène (rotation).

Ceci afin de dégager déjà certains facteurs liés à la scène traitée qui influencent le comportement des deux algorithmes.

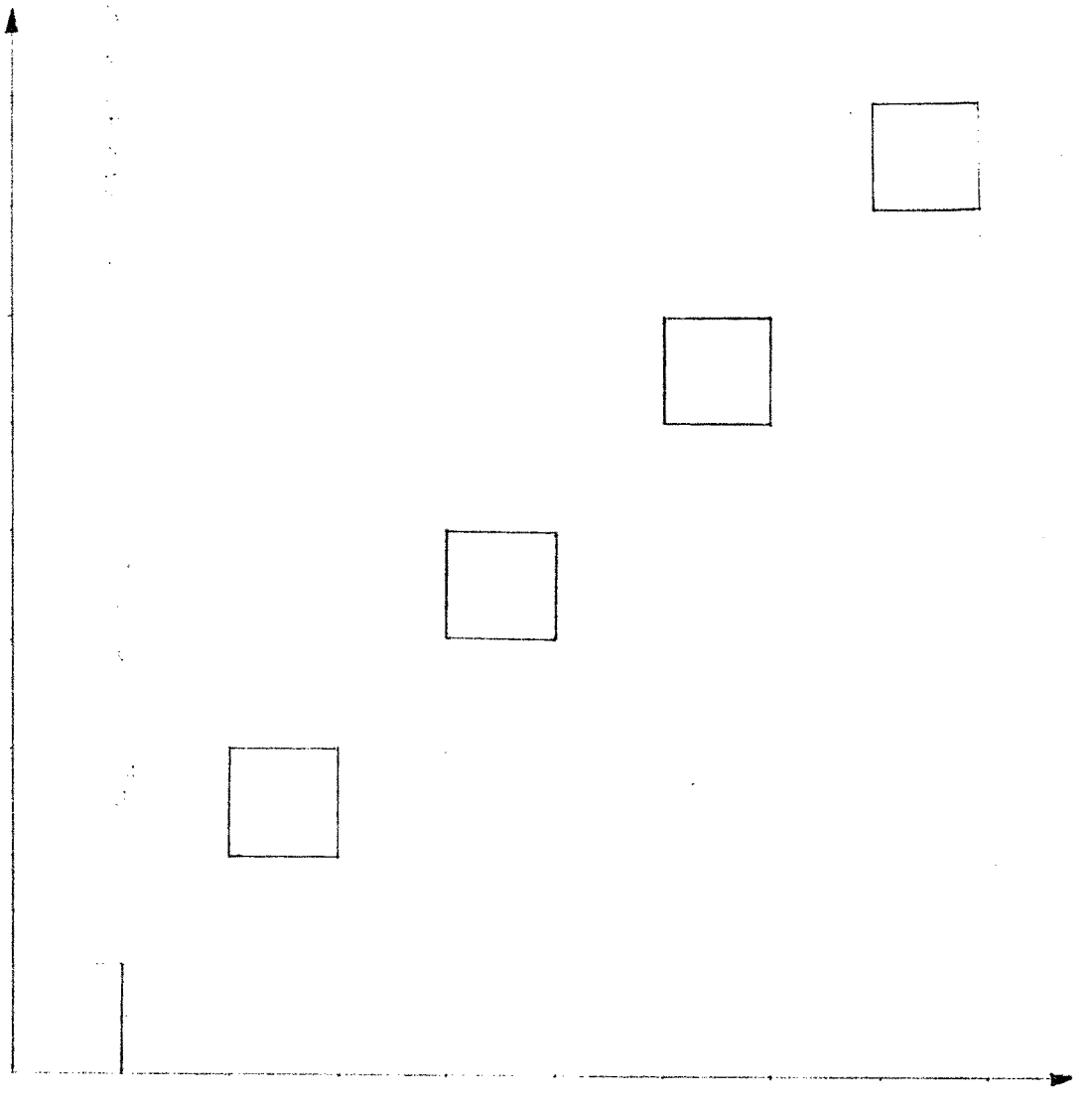
Etude des mesures

Les mesures effectuées ont consisté à démontrer les opérations nécessaires à l'analyse de visibilité des scènes. Les opérations relevées sont : addition, multiplication, soustraction, division, accès tableau, test, affectation qui ont des coûts distincts, le nombre total d'opérations donnera un ordre de grandeur du coût d'une étude de visibilité bien que la sommation devrait être pondérée en fonction du coût de chaque opération.

La première observation à faire en étudiant les mesures est que l'algorithme boîte est plus coûteux que l'algorithme Pile et ceci pour chaque type d'opérations, les coûts totaux allant du simple au double.

Scènes : carrés aux côtés parallèles aux axes

Les scènes sont composées d'éléments qui n'ont aucune influence mutuelle. Chaque carré possède seulement 1 côté visible (voir figure 2.52a).



a)

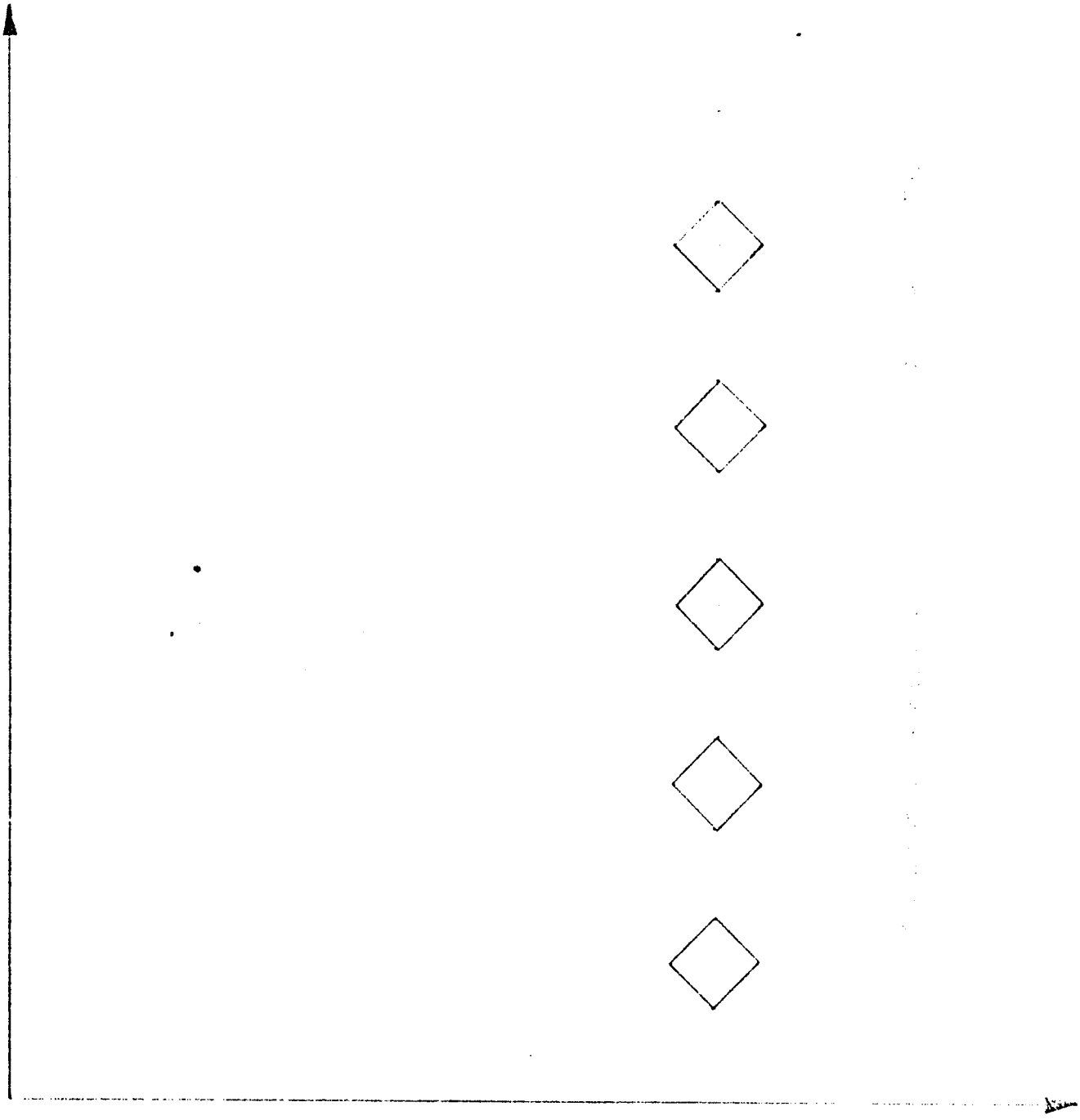


Fig. 2.52. Scènes traitées.

Le comportement de l'algorithme Pile s'explique par le fait que l'étude de visibilité se déroule par balayage de gauche à droite en étudiant les segments en conflit. On aura donc n études de conflit d'où le comportement en O(n) de cet algorithme pour ce type de scènes (voir figure 2.53.).

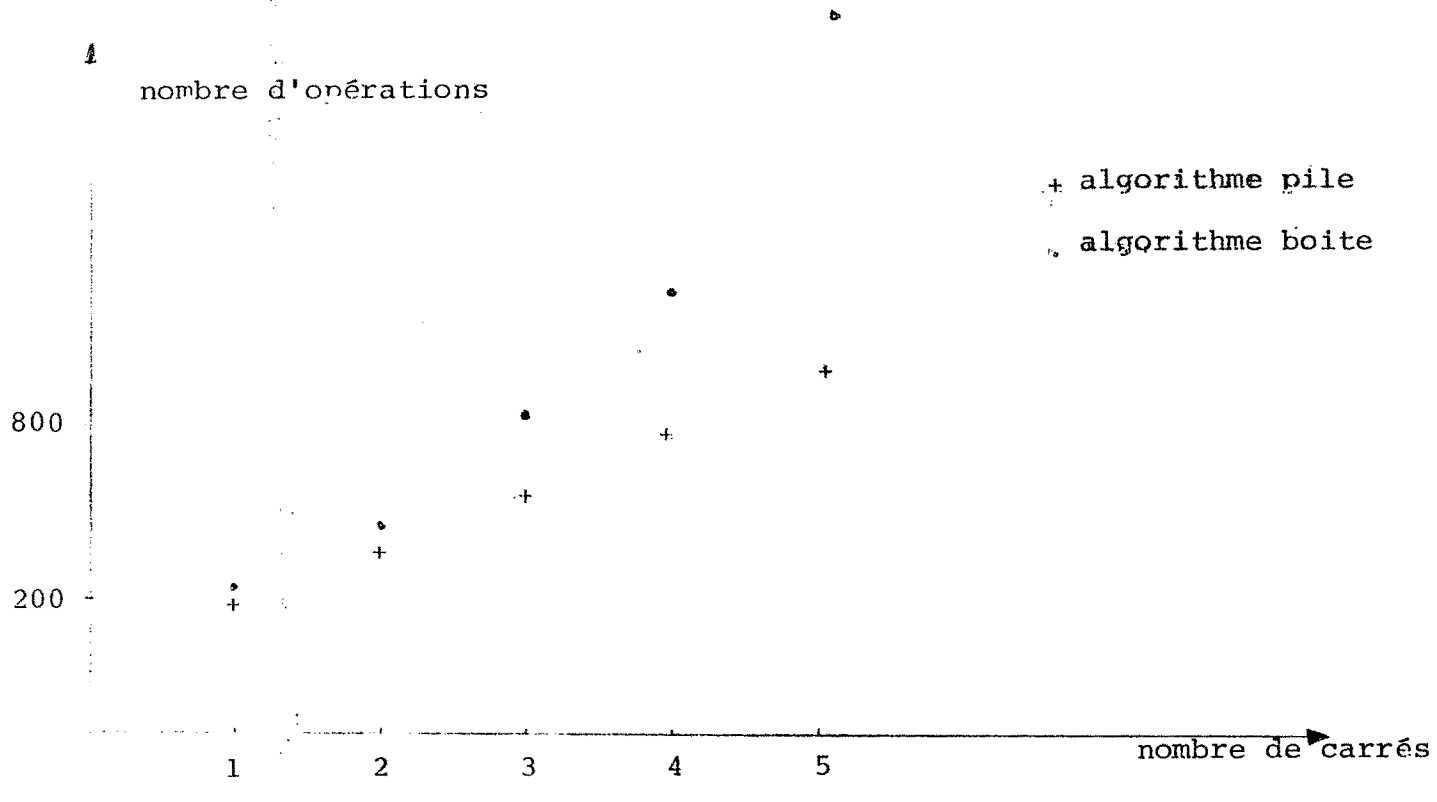


Fig. 2.53 Mesures du comportement des 2 algorithmes.

La méthode Boîte, d'un coût plus élevé, possède un comportement exponentiel qui la défavorise par rapport de l'autre algorithme.

Ce comportement vient des divisions successives de la scène afin de se ramener à un cas simple qui sera ici celui de deux segments horizontaux de même taille en conflit. Le processus de division conduit à un comportement en $O(n \log_2 n)$ où n est le nombre de carrés (voir fig. 2.53.)

Scènes obtenues par rotation de 45° par rapport à la scène initiale (voir figure 2.52c).

Les coûts de chaque algorithme sont pour cet ensemble de scènes supérieurs à ceux enregistrés pour les scènes précédentes. De même le coût de l'algorithme Pile est inférieur à celui de la méthode Watkins pour toute scène de cet ensemble (voir figure 2.54).

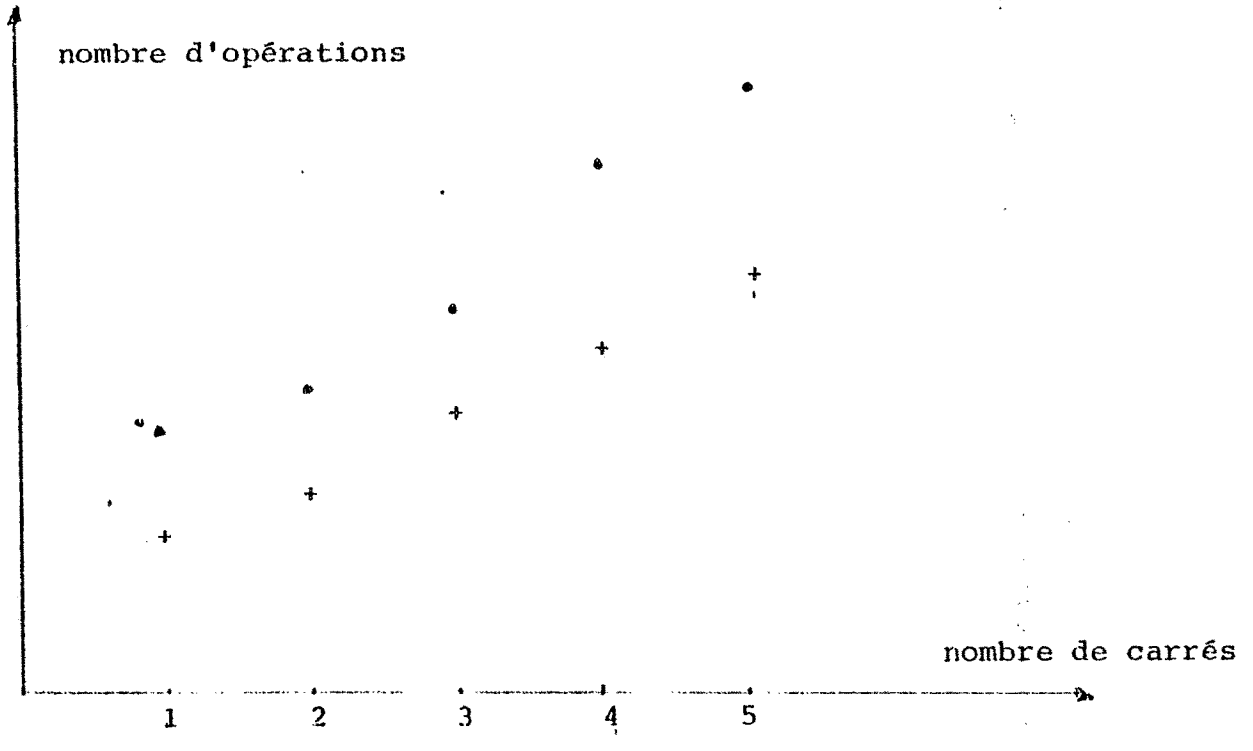


Fig. 2.54.

L'algorithme boîte a pour cet ensemble de scènes un comportement linéaire en fonction du nombre de carrés alors que son comportement était logarithmique pour l'ensemble précédent.

Les scènes de cet ensemble ont la particularité de former une seule zone, ensemble de segments en conflit, facteur indépendant dans ce cas du nombre de carrés. Seul le nombre de segments en conflit est ici dépendant du nombre de carrés, à l'inverse des scènes précédentes.

L'algorithme Pile conserve un comportement linéaire mais les nombres d'opérations pour les scènes sont supérieurs à ceux enregistrés pour les scènes initiales. Ceci à nombre de carrés égal. Ceci provient du fait que le nombre de segments à chercher est plus important. Il passe de 2xn à 4xn car on ne prend pas en considération les segments parallèles à l'axe des ordonnées.

. Scènes obtenues par rotation de 30 ou 60° à partir des scènes initiales (figure 2.52d).

On traitera ces deux types de scènes car elles sont symétriques par rapport à l'axe des ordonnées ce qui donne des mesures sensiblement identiques lors du traitement par les deux algorithmes

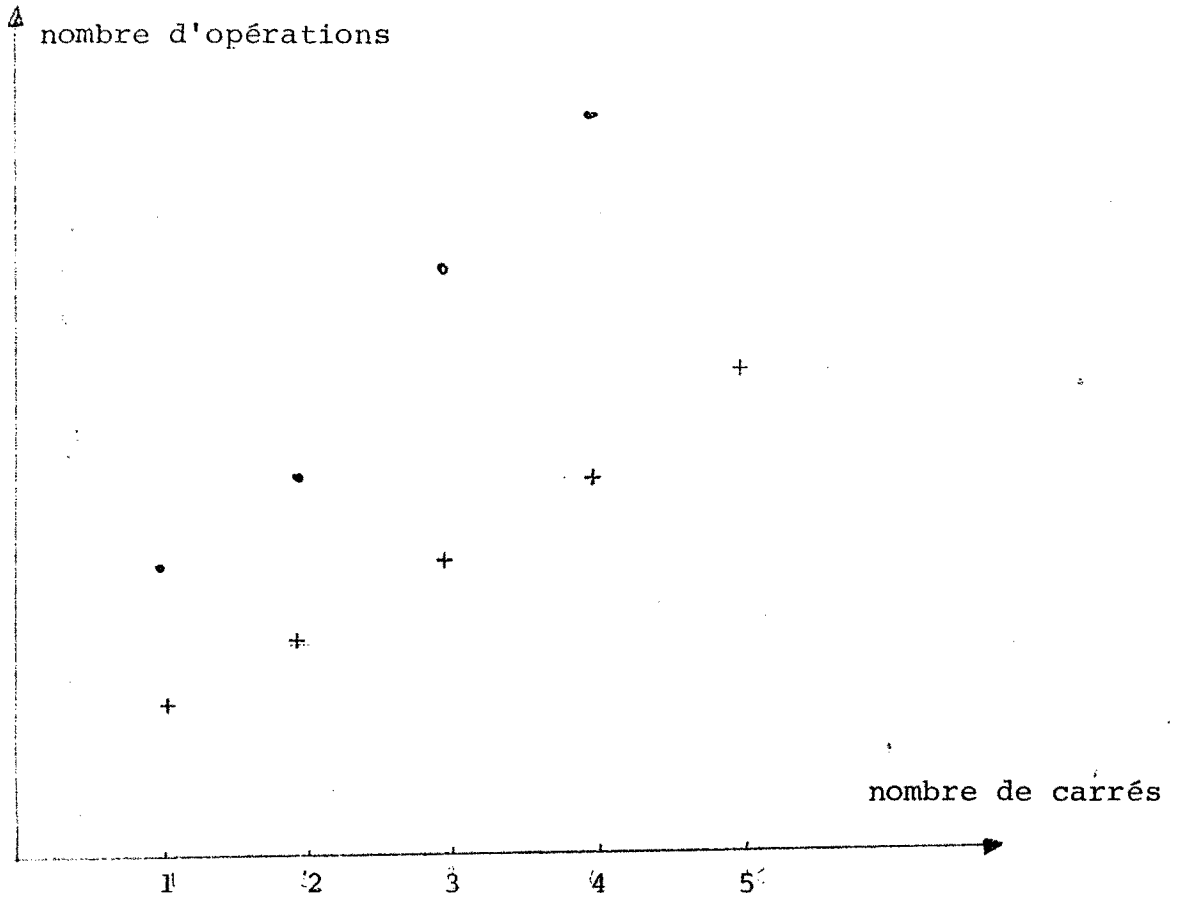


Fig. 2.55

Une première remarque est que pour un même nombre de carrés le nombre d'opérations est ici le plus élevé de tous et ceci pour les deux algorithmes. Les différences entre les deux sont ici les plus accentuées et d'autant plus que le nombre de carrés est plus important.

L'algorithme Pile présente un nombre d'opérations plus élevé que précedemment car le nombre de segments à empiler est ici plus important (de 2 à 4), le nombre total de segments à étudier étant le même.

L'algorithme Boîte présente lui un comportement exponentiel car dans ce cas de figure le nombre de segments ou portions de segments visibles est dépendant de n.

Conclusion

. Comportement de l'algorithme Boîte :

du principe même de cette étude de visibilité, on peut déduire un comportement qu'un jeu d'essai plus important permettrait de vérifier.

soit une scène composée de n segments, le modèle de visibilité étant formé de n_vis segments ou portions de segments visibles.

Cette étude aura été faite par $1,2,\ldots 2^k$ études de portions de plans qui contiendront chacune $n, \frac{1}{2}, \ldots \frac{n}{2^k}$ segments ($k = \log n\_vis$) d'où un comportement en n x log n_vis que l'ensemble des scènes étudiées auparavant corrobore, mais qui ne peut bien sûr servir de preuve et est plutôt présenté dans un but d'illustration.

. Comportement de l'algorithme Pile :

Cet algorithme a lui un comportement linéaire fonction du nombre de segments qui composent la scène, une évaluation probabiliste permettrait de donner un comportement moyen. Ici interviendrait la loi de répartition des segments dans le plan qu'il est difficile de donner puisqu'elle dépend des types d'application traités.

- Comparaison des deux méthodes :

. encombrement mémoire : l'algorithme Watkins utilise une place mémoire constante tandis que l'autre méthode utilise une pile dont la taille varie entre 1 et le nombre de segments qui composent la scène.

. les différents tests effectués ont montré la supériorité de la méthode Pile sur la méthode Boîte et cette étude a mis en valeur différents éléments qui influent sur le comportement de ces algorithmes :

- le nombre de zones qui composent la scène. Deux zones sont des ensembles de segments n'ayant aucune influence réciproque. L'algorithme Pile de par son principe utilise au mieux cette propriété des scènes puisque l'étude des zones sera indépendante. L'algorithme Boîte du fait de ce découpage arbitraire n'utilise pas cette décomposition de la scène et son comportement dépendra de la répartition des zones sur l'axe des abscisses (voir figure 2.56.).

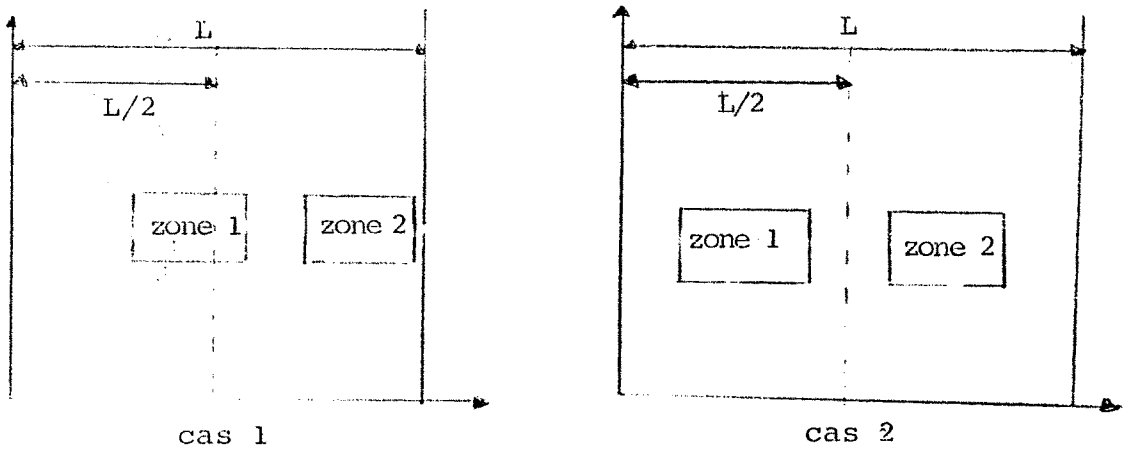


Fig. 2.56 Translation d'une scène.

Dans le deuxième cas le traitement des zones sera indépendant tandis que dans le premier la division arbitraire en x = L/2 rend dépendant le traitement des deux zones.

Une scène translatée d'une autre suivant l'axe des x n'aura donc pas forcément le même temps de traitement que la scène initiale

ce qui n'est pas le cas par la méthode Pile.

- le nombre de segments en conflit dont est fonction la profondeur de la pile dans la méthode "Pile" et le nombre de comparaisons segment-boîte dans la méthode "Boîte". La première méthode n'effectue des comparaisons de profondeur que lorsque deux segments sont en conflit alors que la deuxième par son nombre excessif de calcul de profondeur du fait de la complexité de l'opération de comparaison segment-boîte.

La méthode Pile semble donc la meilleure des deux, bien que dans certains cas la méthode Boîte puisse s'avérer meilleure :

Exemple : une scène comportant un segment écran cachant tous les segments sera traitée rapidement par l'algorithme Boîte alors que la méthode Pile étudie tous les segments de la scène. Ceci provient du module de décision qui est apte à reconnaître de telles configurations alors que l'algorithme Pile procède à une étude exhaustive de la scène.

2.3.4. Production de dessin

Cette famille d'algorithmes, intialement créée pour produire des images, peut grâce à quelques modifications permettre de sortir des dessins ([Arc 72]).

2.3.4.1. Principe

L'étude de chaque plan produit l'ensemble des éléments visibles. Ceux-ci sont délimités par une succession de points appartenant à des arêtes. Ceci permet donc pour chaque plan d'étude de déterminer l'état, visible ou invisible, des arêtes.

Les variables qui caractériseront l'état des arêtes sont:

Xdeb

Ydeb

Etat

(Xdeb, Ydeb) sont les coordonnées de la projection du point où l'arête devient visible.

Etat est une variable qui peut prendre trois valeurs :

0 arête invisible

1 arête visible au plan précédent

2 arête visible pour le plan actuel.

Les différentes actions effectuées sont :

si l'état d'une arête est 0 ou 1 on affiche la portion visible, si état est égal à 1 il est mis à 0.

Au début d'étude du plan suivant si Etat est à 2, ilest mis à 1. Si une arête se termine et état = 2 on affiche.

2.3.4.2. Remarques

L'avantage de cette méthode est qu'elle est applicable à toutes les versions étudiées, ce qui a permis de comparer les résultats obtenus, vis-à-vis de la qualité du dessin, pour les trois algorithmes.

Arêtes d'intersection

Ces différents algorithmes ne traitent pas les arêtes d'intersections dans le cas des faces pénétrantes. Deux solutions se présentent :

. à chaque pas on affiche un point. Le résultat du point de vue qualité du dessin est très pauvre ;

. dès qu'une intersection est rencontrée, on crée un descripteur pour l'arête d'intersection identique aux descripteurs des arêtes de la scène.

Précision des calculs

Les sommets des faces sont décrits, dans l'espace utilisateur, par des coordonnées réelles ; ils sont projetés dans un plan où les points adressables ont des coordonnées entières d'où différentes imprécisions dans le dessin obtenu (voir figure 2.57.).

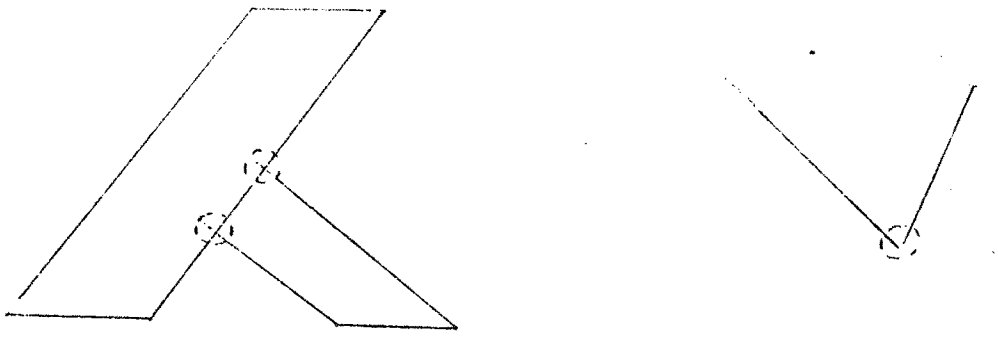


Fig. 2.57. Imprécision dans le dessin.

Deux arêtes visibles au plan précédent dont une sera cachée au plan suivant, donnent un dessin où l'arête invisible croise l'autre arête.

Un sommet partageant deux arêtes n'est pas tracé.Ceci est en fait dû aux algorithmes qui n'étudient pas la visibilité d'un segment de taille nulle. On y remédie en étudiant de tels segments ce qui a pour conséquence d'augmenter les temps de calcul.

On peut remédier à ces imprécisions en effectuant l'étude de visibilité dans un domaine de plus grande résolution (ex : 1024 x 1024 au lieu de 512 x 512). Si cette solution augmente les temps de calcul elle ne le fera pas dans des proportions importantes si l'on se rapporte aux résultats précédents (cf. algorithme pile) qui indiquent que le temps d'exécution peut être rendu indépendant de l'étendue de la scène dans le plan de projection.

Arêtes horizontales

Les arêtes horizontales ne sont pas traitées dans ces algorithmes. Il faudra donc créer une liste de ces arêtes, (liste triée par valeur de Ycroissante) et les insérer dans la liste des segments présents dans les plans d'étude en jeu.

2.3.4.3. Conclusion

Le tracé des arêtes apporte une meilleure compréhension du dessin obtenu et permet d'apprécier plus facilement la disposition relative

des faces visibles sans que le temps d'exécution soit augmenté de façon appréciable.

2.3.5. Utilisation de la cohérence de la scène

2.3.5.1. Changements des éléments lors du passage d'un plan au plan suivant

On peut distinguer trois types de modification

1.- Apparition d'un élément (figure 2.58a) ;

2.- Disparition d'un élément (figure 2.58b) ;

3.- Modification de la disposition relative des faces en conflit (figure 2.58c).

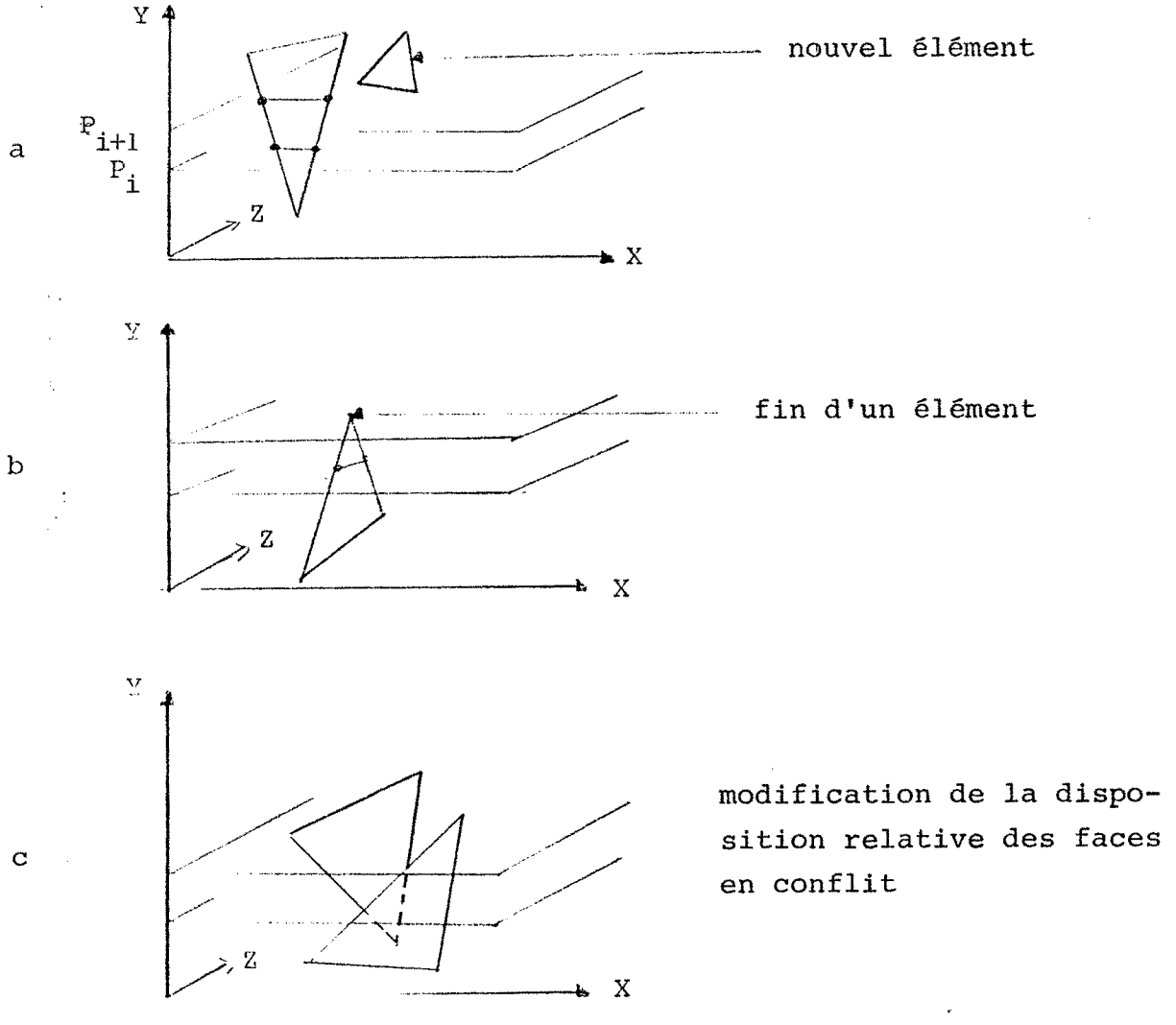


Fig. 2.58 Modifications de la scène.

Les deux premiers cas peuvent être décelés grâce aux informations présentes dans la structure de données :

- on possède pour chaque arête en cours de traitement le nombre de lignes où elle est présente ;

- on possède pour chaque plan de balayage la liste des arêtes qui entrent en jeu à ce plan.

Dans le troisième cas les informations nécessaires ne sont pas présentes dans tous les algorithmes étudiés.

Dans un plan supposons que nous possédions l'ensemble des extrémités des segments ordonnés par valeur de x croissante.

Un changement du 3ème type se traduira par un changement de l'ordre des extrémités lors du passage au plan suivant.

L'information nécessaire est donc une liste ordonnée des extrémités des segments par valeur croissante des abscisses. Dans les trois algorithmes étudiés seul l'algorithme pile travaille sur une telle liste. Afin d'appliquer les modifications qui seront citées aux deux autres algorithmes il faudra ajouter une telle liste à leur structure de données.

2.3.5.2. Utilisation de ces renseignements

Si, lorsqu'on passe au plan d'étude suivant, aucun changement des différents types cités n'est apparu, l'analyse de la visibilité n'est pas nécessaire pour ce plan. L'étude précédente a fourni la liste des éléments visibles. Connaissant la pente des arêtes qui délimitent les éléments visibles, il suffira d'ajouter à chaque élément la pente de l'arête correspondante afin d'obtenir les segments ou portions de segments visibles pour ce plan.

2.3.5.3. Améliorations dues à cette méthode : comportement de l'algorithme, temps d'exécution

L'application de cette méthode à l'algorithme pile permet d'obtenir un résultat important : le traitement d'une scène est fonction des

différents changements (types 1, 2, 3) et non de l'amplitude en Y de cette scène.

En effet, l'étude de visibilité n'est faite que pour les plans où un changement au moins apparaît, les autres plans étant traités de façon incrémentielle à partir des précédents. Ceci a permis de diminuer des temps de traitement par trois et pour la majorité des scènes par deux au moins.

2.3.5.4. Restrictions

Cette méthode ne peut être appliquée lorsque des faces se pénètrent car ne connaissant pas l'équation, la longueur de l'arête d'intersection, on ne peut déterminer à priori les changements qu'elle implique

2.3.5.5. Critique

Si l'étude d'un plan de balayage ne se produit que lorsqu'un changement dans la configuration a été détecté il peut provenir en fait d'éléments invisibles ou d'éléments dont l'influence sur la visibilité peut être déduite sans que l'étude du plan soit à refaire (voir figure 2.59)

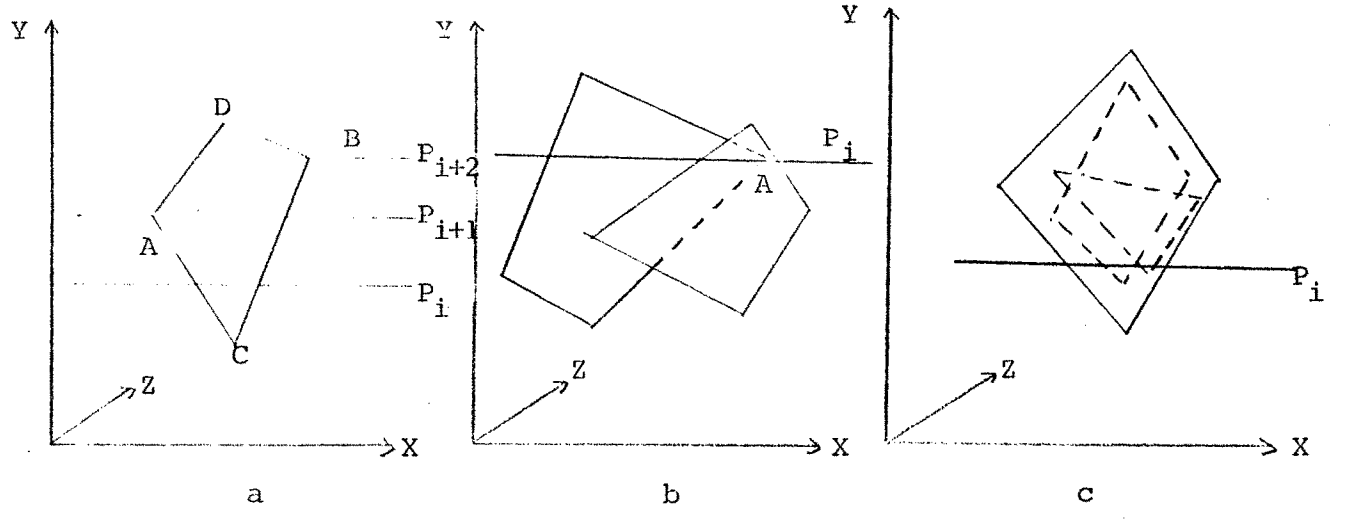


Fig. 2.59. Changements de la scène sans modification de la visibilité.

Dans le cas de la figure a l'étude de cette scène exigerait 3 études de plan en C, A, B alors qu'une seule étude est nécessaire celle en C. On remarque que les points A et B sont sommets, chacun, de deux arêtes. Lorsqu'on est à l'étude P_i le point A est déjà étudié du fait de l'arête AC. Dans l'ensemble des éléments visibles seule l'arête délimitant est à changer. Le segment subira alors les modifications dues à l'arête AD. Il est de même dans la figure 2 où on peut grâce aux traitements précédents affirmer que le point A est invisible. Dans la figure 3 le changement de type 3 (cf. §1) n'a aucun effet sur la visibilité de la scène puisque les deux éléments en jeu sont cachés.

En résumé, une étude de visibilité d'un plan sera faite:

- changement de type 1 : si on ne peut rien prédire sur la visibilité du sommet, on dira que l'on a un sommet pendant (figure 2.60.).

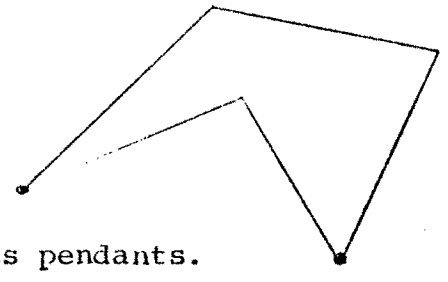


Fig. 2.60. Sommets pendants.

- changement de type 1 : si le sommet est un sommet terminal : aucune arête n'aura ce sommet comme extrémité la plus basse.

- changement de type 2 : au moins un des deux éléments en jeu est visible.

Cette amélioration, non encore testée, devrait diminuer considérablement les temps d'exécution des algorithmes de visibilité, le nombre de plans où l'étude de visibilité sera faite étant diminuée de façon importante.

2.3.6. Parallélisme

Différentes options peuvent être prises pour une réalisation en tâches parallèles de ces algorithmes de visibilité.

2.3.6.1. Technique de Kaplan et Greenberg ([KGr 79])

Cette méthode consiste à diviser l'ensemble de plans à étudier en groupes, chaque groupe étant analysé de façon indépendante.

Cette solution présente diviers avantages :

. chaque processeur utilise le même code ;

. la gestion des processeurs et l'intercommunication sont minimisées.

Mais cette solution présente l'inconvénient de ne pas utiliser la cohérence de la scène (cf. 2.3.5.) entre plans successifs.

Cette méthode présente bien sûr un grand intérêt, pour une réalisation cablée, vu les coûts décroissants des processeurs et mémoires.

2.3.6.3. Une autre méthode

Cette méthode consiste à utiliser la décomposition qui est apparue dans l'étude de ces algorithmes. On distingue en effet deux tâches qui sont :

- l'obtention des segments présents dans un plan d'étude.
- l'étude de visibilité dans un plan.

On pourra envisager une implémentation au type :

. un processeur est chargé de l'étude de visibilité ;

. un processeur prépare l'ensemble des segments pour le plan suivant et indiquera au processeur d'étude de visibilité si une étude complète est nécessaire ou si une simple mise à jour est suffisante.

Cette méthode présente l'avantage d'utiliser la cohérence permise par les algorithmes étudiés mais nécessite une intercommunication plus complexe entre processeurs.

2.3.6.3. Conclusion

Dans le cadre d'une réalisation en parallèle, ces deux solutions devront être étudiées, notamment en faisant varier le nombre de

de lignes par groupe dans la première méthode. On peut envisager un mixage des deux méthodes, c'est-à-dire d'appliquer la décomposition citée au paragraphe précédent à l'étude de visibilité pour un groupe de lignes.

2.3.7. Conclusion

Les principaux résultats obtenus concernent :

1.- Une meilleure compréhension de l'image obtenue en affichant les arêtes ou portions d'arêtes visibles. Le manque de réalisme des résultats est dû aux limitations importantes imposées par le matériel sur lequel ont été effectués les tests.

2.- L'amélioration des temps d'exécution de l'algorithme Watkins en faisant l'étude de visibilité de segments aux abscisses à valeurs entières (temps d'exécution divisé par trois).

3.- L'importance du tri des segments à traiter pour l'algorithme pile où un gain de 30 % a été obtenu.

4.- On peut rendre le comportement de ces algorithmes fonction des changements de la scène lors du passage à un autre plan d'étude, et non plus fonction de l'étendue de la scène. Ceci entraîne bien sûr une diminution importante des temps d'exécution. Ces améliorations ne sont possibles que lorsque l'ensemble MD ne contient pas de faces pénétrantes.

Pour mesurer le comportement de ces algorithmes en fonctiond'une scène on peut considérer deux points essentiels :

- nombre de plans à étudier ;
- étude d'un plan.

Nombre de plans à étudier

a) La scène comporte des faces pénétrantes :

Soit [Ymin, Ymax] l'étendue de la scène en Y le nombre de plans étudiés sera :

$$Ymax - Ymin + 1.$$

b) La scène ne comporte pas de faces pénétrantes :

L'étude d'un plan sera faite si et seulement si un sommet pendant au terminal est présent. Joignons dans le plan de projection (OXY) tous les sommets de ce type. Soit nbhoriz le nombre de segments horizontaux et nbsom' le nombre de ces sommets, le nombre de plans étudiés sera :

p =(nbsom' - nbhoriz) ⊕ plans où apparaissent des conflits entre faces, une au moins étant visibles

nbsom' dépend essentiellement de la convexité ou non convexité des faces (une face convexe a un sommet pendant et un sommet terminal).

Etude dans un plan (point par point).

Le coût est ici proportionnel à la somme des longueurs des segments présents dans le plan et indépendant des positions relatives de ceux-ci.

Etude dans un plan (autres algorithmes).

Il s'agit des algorithmes étudiés en 2.3.3.4. Leur comportement est fonction des dispositions relatives des segments. Les différents éléments que l'on peut retenir sont :

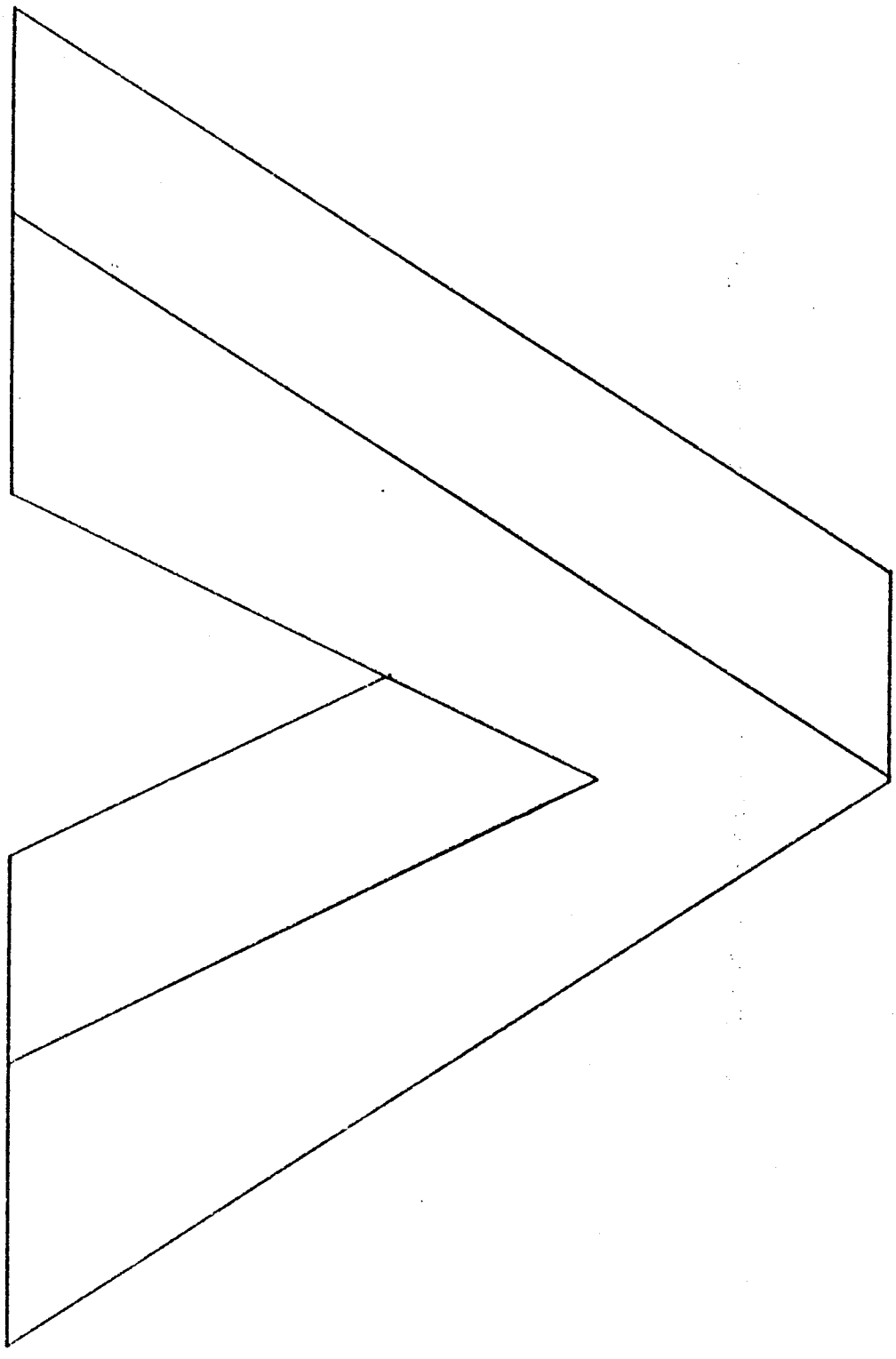
- nombre de segments qui dépend du nombre de faces présentes et de leur convexité ou non convexité.

- pour un segment donné le nombre de segments en conflit avec lui. Ceci fait intervenir la disposition relative des faces en x.

- présence de segments qui s'intersectent qui implique des tests plus coûteux.

Si la programmation de la partie commune de ces algorithmes est complexe on peut par contre classer par ordre de complexité décroissante les différentes méthodes d'étude de visibilité : Watkins, Fusion, Pile, algorithme point par point.

Cette famille d'algorithmes a aussi l'avantage de procéder à l'étude de visibilité de façon cohérente avec un balayage télévision. Une version câblée de l'algorithme Watkins a permis de produire des images en temps réel.



Scène traitée par l'algorithme de Atherton et Weiler

2.4. L'algorithme de Atherton et Weiler ([ATW 78])

2.4.1. Description

Cette solution à l'étude de visibilité d'une scène composée de polygones, qui fait partie de la famille des algorithmes travaillant dans l'espace objet, fournit un modèle de visibilité dont les éléments sont des faces polygonales (fig. 2.61.). C'est le seul algorithme opérant dans l'espace objet et permettant d'obtenir une image.

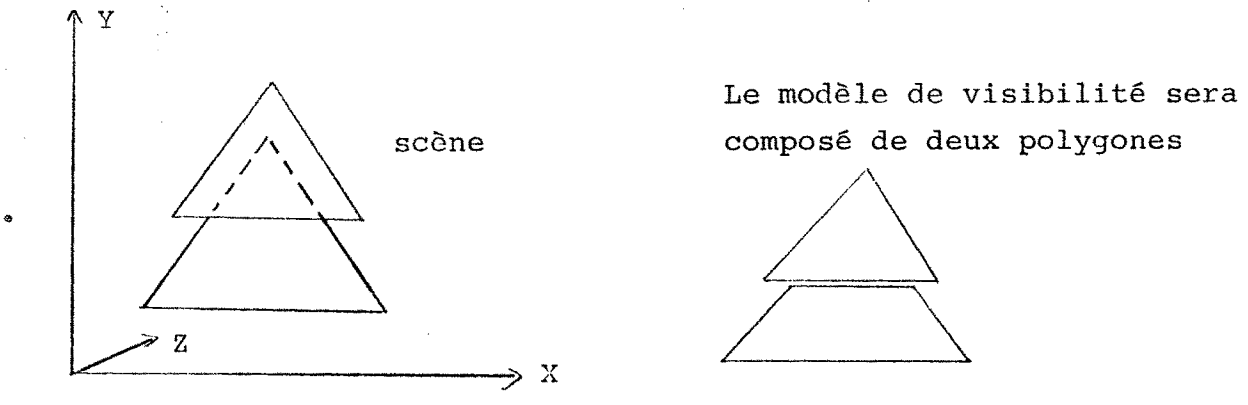


Fig. 2.61. Scène et modèle de visibilité.

L'intérêt de cet algorithme est essentiellement le type du modèle de visibilité ; nous verrons les contraintes qu'il impose et les possibilités qu'il offre dans le cadre de la synthèse d'images.

2.4.2. Etude des différents éléments de l'algorithme

2.4.2.1. Le modèle de description

Cet algorithme étudie la visibilité d'un ensemble de faces polygonales décrites par les éléments suivants :

- un contour extérieur orienté (matière à droite lors du parcours, par convention) ;
- un ensemble de contours intérieurs qui définissent des trous (orientés en sens inverse du contour extérieur).

- une information d'aspect qui indique l'opacité ou la transparence de cette face et un ensemble de détails qui sont des contours polygonaux définissant des surfaces ayant leur couleur propre (figure 2.62.).

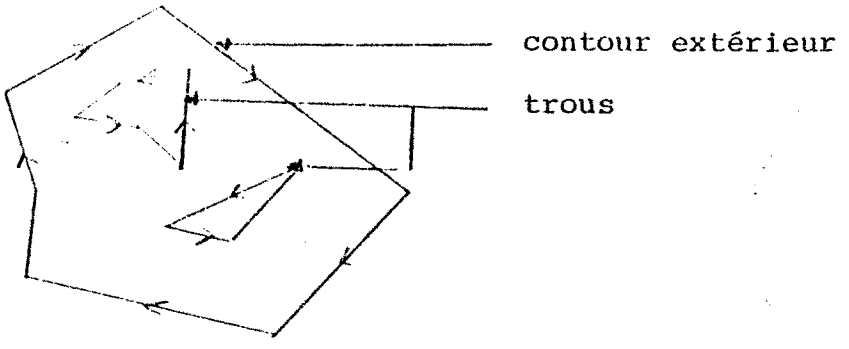


Fig. 2.62. Exemple de face traitée.

Il n'y a aucune restriction sur la nature (convexité ou non convexité des contours).

Représentation des contours

Afin de simplifier l'algorithme de découpage de deux faces chaque contour est représenté par deux listes circulaires, qui permettent les parcours directs et inverses de chaque contour. L'occupation mémoire pour la représentation du modèle de description peut être considérée comme une fois et demi plus importante que celle nécessitée par les algorithmes de Watkins et de Warnock.

Prétraitement

Le modèle de description est décomposé en un ensemble de zones qui sont indépendantes vis-à-vis de la visibilité. Cette étape n'est pas une étude de visibilité mais une simple partition du modèle de description : $MD = Z_1\ UZ_2..UZ_n$.

Une zone est un ensemble de faces pouvant se recouvrir.

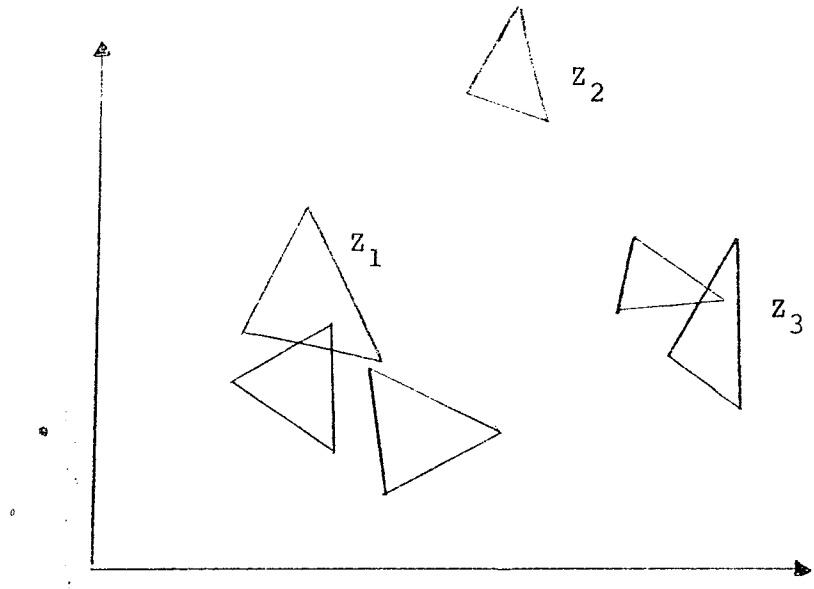


Fig. 2.63. Ensemble de zones.

La scène de la figure 2.63. sera décomposée en 3 zones dont l'étude de visibilité sera faite de façon indépendante.

Les zones sont déterminées par des tests suffisants de recouvrement qui utilisent les rectangles à bords parallèles aux axes Ox Oy qui englobent les faces.

2.4.2.2. Tri initial des données

L'entité traitée est une zone constituée d'un ensemble de faces pouvant être en conflit.

Ce tri se décompose en deux étapes qui sont :

Décomposition d'une zone en boîtes :

Une zone est un ensemble de boîtes ordonné par éloignement décroissant vis-à-vis de l'observateur.

Chaque face possède une valeur minimale et maximale en z. Cet ensemble de clés est trié par valeur croissante. Le tri utilisé est un tri rapide car on ne possède pas d'information à priori sur l'état des données. A partir de ces données les boîtes sont déterminées de façon simple (figure 2.64.).

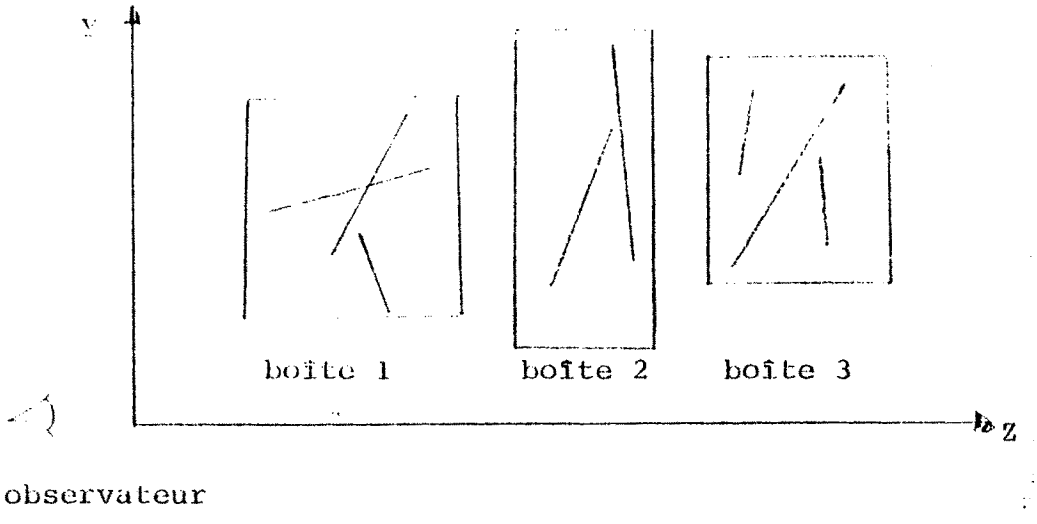


Fig. 2.64. Décomposition d'une zone en boîtes.

Tri des faces composant une boîte.

Une boîte est un ensemble de faces ordonnées par éloignement croissant vis-à-vis de l'observateur.

L'ensemble des faces regroupées en une boîte sera donc trié en fonction de leur éloignement croissant vis-à-vis de l'observateur. La clé du tri pour chaque face sera la valeur minimale en z.

Remarque :

L'entité élémentaire dont la visibilité est étudiée, est la boîte, et l'algorithme de visibilité est construit afin de traiter cette entité qui est fournie par le prétraitement et les tris initiaux.

2.4.2.3. Ensemble des opérations géométriques utilisées

Cet ensemble est composé uniquement de deux éléments :

. Détermination des positions relatives de deux faces vis-à-vis de leur profondeur, c'est-à-dire déterminer si une face est entièrement située dans le demi espace arrière défini par l'autre face.

Soient F1 et F2 les deux faces à comparer. $C_1(n)$ $C_2(p)$ leurs contours extérieurs. Le test utilisé est C_p^p qui détermine la position d'un point par rapport à un plan qui possède une normale orientée. Ce test sera appliqué au maximum nxp fois.

. Découpage de deux faces

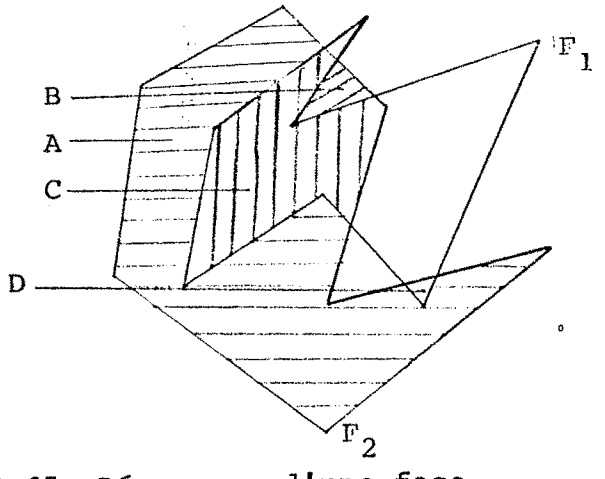
Cet algorithme est décrit en 3.2.3.3., nous rappelerons les éléments qu'il fournit en sortie.

Soient F1 la face de découpage et F2 la face découpée

La comparaison C_{F1}^{F2} fournit deux listes (voir figure 2.65.).

La liste des faces provenant de F2 et intérieures à F1

La liste des faces provenant de F2 et extérieures à F1.



liste des faces intérieures :C,D
liste des faces extérieures :A,B

Fig. 2.65. Découpage d'une face.

L'ensemble ϕ est donc composé de deux parties :

$\phi = (C_p^p, C_f^f)$.

2.4.2.4. Fonction stratégie

Le modèle de description est décomposé en zones dont la visibilité est étudiée de façon indépendante. Nous associons une fonction stratégie T_z pour cette étude et une fonction T_b pour l'étude de visibilité des boîtes qui composent les zones.

Fonction stratégie T_z

Une zone est un ensemble de boîtes ordonné par éloignement décroissant vis-à-vis de l'observation.

Soit MV_i le modèle de visibilité fourni par l'étude de la boîte B_i au pas suivant l'élément analysé sera Bi+1UMV$_i$. On ajoute les éléments visibles de la boîte B_i à la boîte Bi+1 qui à son tour est étudiée.

On peut en déduire l'algorithme d'étude de visibilité

```
d'une zone :
     boîte := première_boîte ; il_y_a_boîte := vrai ;
     tant_que il_y a boîte, faire
     début
          modèle := Etude_de_visibilité (boîte) ;
          si boîte = dernière_boîte alors
          début
               modèle_de_visiblité := modèle_de_visibilité U modèle :
               il_y_a_boîte := faux
          fin
          sinon
          début
               boîte := boîte_suivante ;
               boîte := boîte U modèle
          fin
     fin
fin algorithme.
```

L'avantage du découpage des zones en boîtes est de réduire le nombre de comparaisons entre faces en ne faisant intervenir que les éléments visibles d'une boîte précédente dans l'étude de la boîte suivante.

Mais l'ordre d'étude des boîtes, de la plus éloignée à la plus proche, peut s'avérer dans certains cas inefficace alors que l'ordre inverse d'étude permet de palier ces inconvénients en n'étudiant en fait que les éléments qui ont une forte probabilité d'être visibles (figure 2.66.).

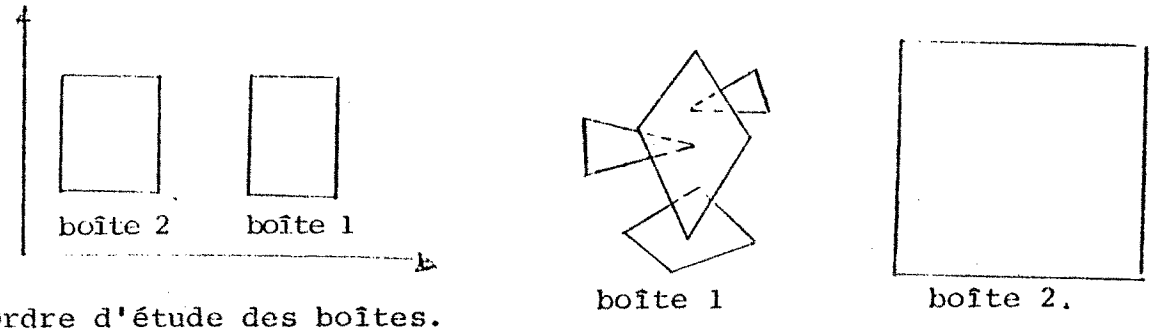


Fig. 2.66. Ordre d'étude des boîtes.

L'étude de visibilité dans l'ordre 1 - 2 va nécessiter la comparaison de 4 faces et la création de 3 nouvelles faces pour la boîte 1, alors que ces éléments sont cachés par la face écran qui constitue la boîte 2.

L'ordre inverse d'étude permet d'éliminer les éléments de la boîte 1 par un simple test.

C'est donc ce deuxième ordre d'étude des boîtes qui a été choisi car il permet de comparer les éléments des boîtes suivantes aux éléments qui seront visibles et d'éviter ainsi la création de portions de faces qui pourront être reconnues invisibles en fin de traitement seulement.

Fonction_stratégie T_B

Cette fonction stratégie décrit le principe d'étude de visibilité dans une boîte. C'est dans cette partie de l'algorithme de visibilité que sont utilisées les fonctions géométriques décrites en 2.4.2.3. car les tests minimaux utilisés n'ont pas permis un découpage plus important de cet ensemble de faces en zones et en boîtes.

L'algorithme correspondant est le suivant :

$c\phi$: l'algorithme de découpage C_f^f fournit deux listes :
la liste des éléments intérieurs à la face de découpage pointée par Plint
la liste des éléments extérieurs à la face de découpage pointée par Plext
Pcoupage est la face de découpage ;

Pcoupage := premier_élément_de_la_boîte ;

Plext := nil ;

Plint := suivant (Pcoupage) ;

il_y_a_élément := vrai ;

tant qu'il_y_a_élément faire :

$c\phi$: on coupe les éléments de la liste pointée par Plint par la face de coupage

si (plint = nil) alors

début

```
            mettre pcoupage dans le modèle de visibilité ou dans la boîte
suivante ;
            il y a élément := faux
      fin
      sinon
      début
            plint = nil
            découper tous les éléments intérieurs par pcoupage ;
            si (plint ¬= nil) alors
            début
                  cϕ : il y a des éléments intérieurs à Pcoupage situés
                       devant celui-ci ; inverser pcoupage et plint.
                  ordonner_les_éléments_intérieurs ;
                  plext = nil ;
            fin
            sinon
            début
                  mettre pcoupage dans le modèle de visibilité ou dans
                  la boîte suivante ;
                  Pcoupage = plext
                  plint = suivant (pcoupage) ;
                  plext = nil ;
            fin
      fin
   fin
fin algorithme ;
```

La description de cet algorithme met en évidence les deux points principaux qui influencent le comportement de celui-ci :

. le nombre d'éléments de la liste des faces extérieures

produites lors du découpage parmi lesquelles sera choisi le prochain polygone de découpage, nombre qui est lié à la proportion d'éléments qui peuvent être visibles.

. le deuxième point est la validité du tri initial. Un tri qui ne fournit pas un ordre correct implique le choix d'un nouveau polygone à partir duquel sera effectué un nouveau découpage.

Suivant certaines hypothèses, il est possible de donner le comportement de cet algorithme qui est la partie principale de cet algorithme d'étude de visibilité

. un seul élément est visible et le tri initial est correct : le comportement de l'algorithme sera linéaire en fonction du nombre de faces qui composent la boîte ;

. chaque élément de la scène comporte r éléments visibles (figure 2.67.).

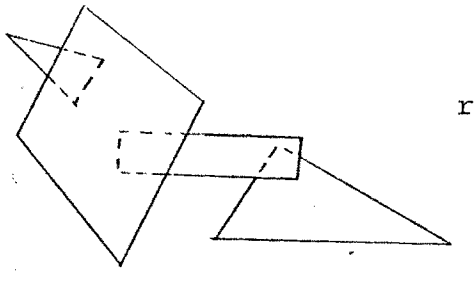


Fig. 2.67. Eléments visibles dans une scène.

Faisons l'hypothèse supplémentaire qu'une face est découpée en r + 1 faces par une face seulement.

Il n'y aura n x r éléments visibles déterminés par n x r découpages de la scène.

Au kième coupage k * r + n - k éléments sont découpés d'où un comportement en :

$$\sum_{k=0}^{n \times r - 1} (k * r + n - k) = (r - 1) \times \frac{(n \times r - 1)\ (n \times r)}{2} + \frac{n^2 * r}{2}$$

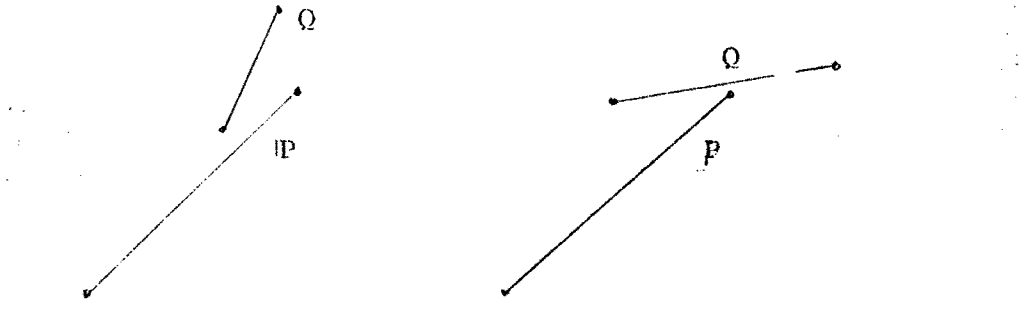
Le comportement sera fonction du carré du nombre d'éléments de la scène et du cube du nombre d'éléments visibles.

On constate donc la grande sensibilité de cet algorithme à la complexité de la scène visible (nombre d'éléments visibles) qui peut déterminer un comportement linéaire ou exponentiel.

Un autre point sensible est le tri des faces composant une boîte.

Il s'agit de déterminer la position des faces intérieures au polygone de découpage. Or cette étude n'est effectuée que lorsque toutes les faces ont été découpées. Une autre solution qui a été choisie est de déterminer la position d'une face intérieure dès qu'elle a été produite par l'algorithme de découpage. Ceci permet d'éviter un nombre de découpages inutiles d'autant plus que la probabilité que le tri soit faux est plus grande pour les faces les plus proches.

Le tri des faces se fait sur le point le plus proche de l'observateur de chaque face ce qui conduit bien sûr à des erreurs. Il est possible d'ordonner les faces en fonction d'autres critères (figure 2.68.), qui impliqueront des tests plus coûteux mais qui réduiront le nombre d'erreurs et les appels inutiles à l'algorithme de découpage.



Q P : Q est situé dans le demi espace avant de P

Q P : P est situé dans le demi espace arrière de Q

Fig. 2.68. Clés du tri.

Sous certaines restrictions sur le modèle de description il est possible de réduire le nombre de faces à comparer, nombre qui dans l'étude de visibilité d'une boîte est un élément prépondérant par l'influence qu'il a sur le comportement de cet algorithme.

Supposons que le modèle de description soit constitué d'objets qui ne se pénètrent pas. Il est alors possible pour chaque objet d'en extraire la silhouette qui sera considérée comme une face ayant pour détails l'ensemble des faces avant. Ceci implique de plus que les faces avant ne soient pas transparentes et que l'objet soit convexe, cas où il est facile d'extraire la silhouette et de déterminer les détails.

Une scène composée de n objets satisfaisant ces conditions sera réduite à un ensemble de n faces, ce qui permet de réduire considérablement le nombre d'éléments à étudier et ceci d'autant plus que le nombre de faces par objets sera élevé.

2.4.2.5. Ensemble des représentations intermédiaires

La première représentation intermédiaire contient l'ensemble des faces du modèle de description trié par le tri initial Td décrit en 2.4.2.2.

La représentation intermédiaire suivante est un ensemble de zones.

Chaque zone est décomposée en un ensemble de représentations intermédiaires que sont les boîtes.

. Au cours de l'étude de visibilité d'une boite chaque pas de l'algorithme donne :

- une représentation intermédiaire, fonction d'une face de découpage, composée des éléments intérieurs et extérieurs ;

- une représentation intermédiaire, contenant les éléments visibles.

. Au cours de l'étude de visibilité d'une zone à chaque boîte sont associés les éléments visibles de la boîte précédente ce qui conduit à une autre représentation intermédiaire et finalement au modèle de visibilité associé à cette zone.

La dernière représentation intermédiaire, le modèle de visibilité, est l'union des modèles de visibilité de chaque zone.

2.4.2.6. Modèle de visibilité

Nous étudierons différents points relatifs au modèle de visibilité qui sont premièrement la possibilité de traiter le problème des ombres portées et de la transparence et deuxièmement le format de ce modèle de visibilité.

Ombres portées

Les ombres portées sont déterminées en appliquant l'algorithme de visibilité, l'observateur étant à la position de la source lumineuse. Chaque face du modèle de visibilité ainsi obtenu est rattachée comme détail représentant une portion de la face éclairée à la face de la scène initiale dont elle provient (figure 2.69.).

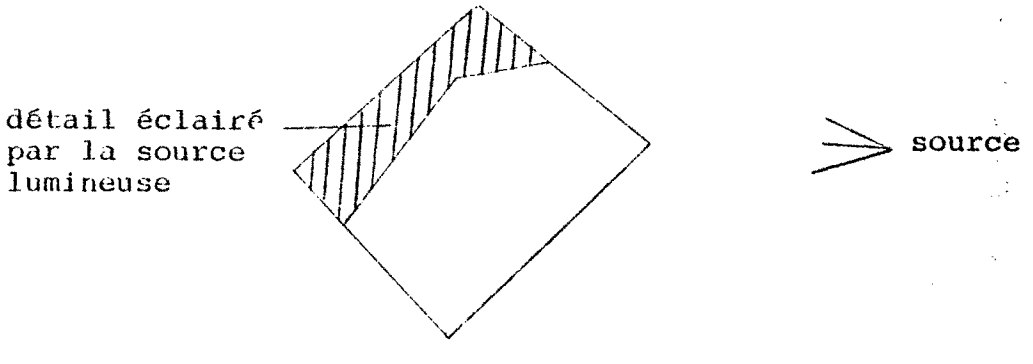


Fig. 2.69. Prise en compte d'une source lumineuse.

A ce détail est attachée une information qui traduit l'éclairement produit par la source lumineuse.

Il est possible de prendre en compte différentes sources lumineuses. Une face pourra comporter plusieurs détails d'éclairement qui

possèdent chacun l'information permettant de rendre cet éclairement (figure 2.70.).

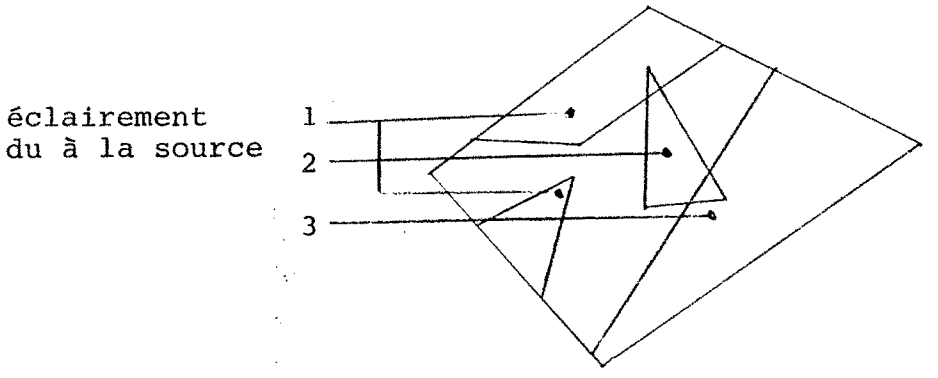


Fig. 2.70. Détails d'ombre.

Il est donc possible de prendre en compte différentes sources lumineuses. On peut considérer qu'obtenir un tel modèle de visibilité prendra le double du temps d'une élimination de parties cachées.

Il est préférable d'effectuer l'étude de visibilité proprement dite avant de prendre en compte les sources lumineuses car dans ces études de visibilité on ne prendra pas en compte les détails appartenant aux faces invisibles ce qui réduira l'espace mémoire nécessaire.

Prise en compte de la transparence.

Le problème est de déterminer, lorsqu'une face transparente est visible totalement ou en partie, quels sont les éléments situés derrière cette face. Soit f une portion visible de la face F transparente, on détermine par l'algorithme de découpage quels sont les éléments intérieurs à f. On effectue donc une étude de visibilité sur cet ensemble qui fournira les éléments visibles à travers f.

A f sera associée une liste de portions de faces visibles à travers elle.

Structure du modèle de visibilité

Il est composé de faces visibles qui peuvent posséder un

ensemble de détails de natures différentes :

. détails provenant du modèle de description ;

. détails traduisant les ombres portées.

De plus, si une face visible est transparente, une liste des éléments visibles situés derrière elle lui est attachée (figure 2.71.).

Soit F une face du modèle de description ; f une partie de F visible

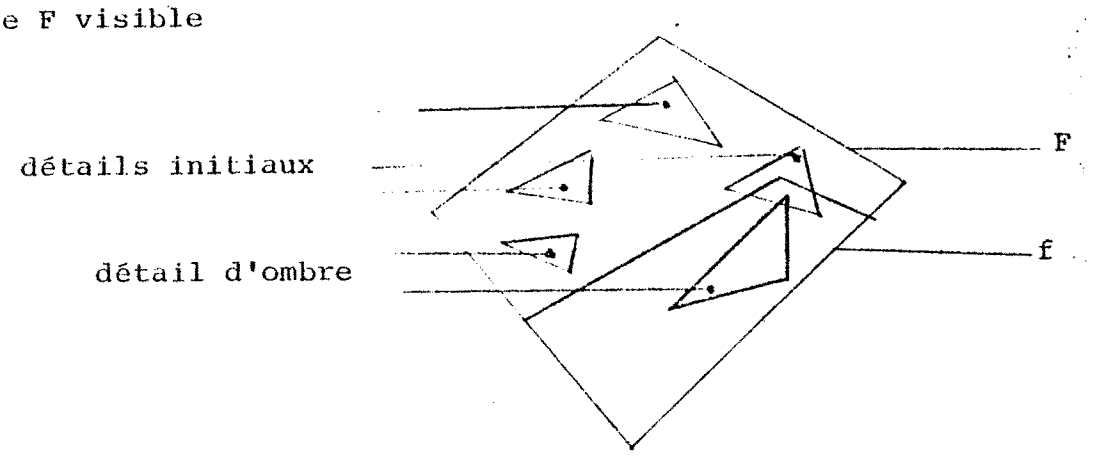


Fig. 2.71. Différents détails d'une face.

Il est nécessaire d'effectuer un traitement des détails de F :

. découpage des détails initiaux et d'ombre par rapport à f ;

. on obtient donc une face f et une liste de détails qui lui appartiennent. Il sera parfois nécessaire et ceci dépend de la réalisation de la synthèse, d'effectuer un découpage des détails initiaux par les détails d'ombre afin de connaître leur éclairement.

2.4.3. Conclusion

L'étude de cet algorithme de visibilité peut se résumer sur différents points, qui sont :

2.4.3.1. Modèle de description

Il présente l'intérêt d'une grande généralité ce qui le

rend adapté à un grand nombre d'applications. Il est possible de plus que des faces se pénètrent, ce cas n'impose qu'une modification très simple dans l'étude de visibilité d'une boîte.

2.4.3.2. Prétraitement

L'intérêt de ce prétraitement est d'une part la décomposition du modèle de description en zones dont la visibilité est étudiée de façon indépendante. Ceci autorise donc un traitement de chaque zone en parallèle. Le comportement de l'algorithme d'étude de visibilité qui dépend de deux facteurs, le nombre de faces visibles et le nombre de faces initiales d'une façon qui peut être exponentielle, montre tout l'intérêt d'une telle décomposition de la scène qui diminue les deux nombres dans chaque entité à traiter.

Les limitations viennent de la façon même de déterminer les zones et les boîtes. En effet, cette étude est faite à l'aide de tests suffisants qui ne donnent des résultats positifs qu'en un nombre limité

Du fait de l'apport du découpage en zones, il serait intéressant d'effectuer des tests plus coûteux que ceux utilisés jusqu'à maintenant, mais qui donnent de meilleurs résultats.

Il est à souligner que l'inconvénient majeur de cet algorithme est d'une part son comportement (cf. IV) et d'autre part la complexité de l'algorithme de découpage utilisé, ce qui montre l'importance des diverses méthodes qui peuvent être employées afin de réduire le nombre de comparaisons entre faces et de permettre un traitement en parallèle.

2.4.3.3. Modèle de visibilité

Le principal intérêt de cet algorithme est le modèle de visibilité qui est fourni et ceci, sur deux points :

Le format :

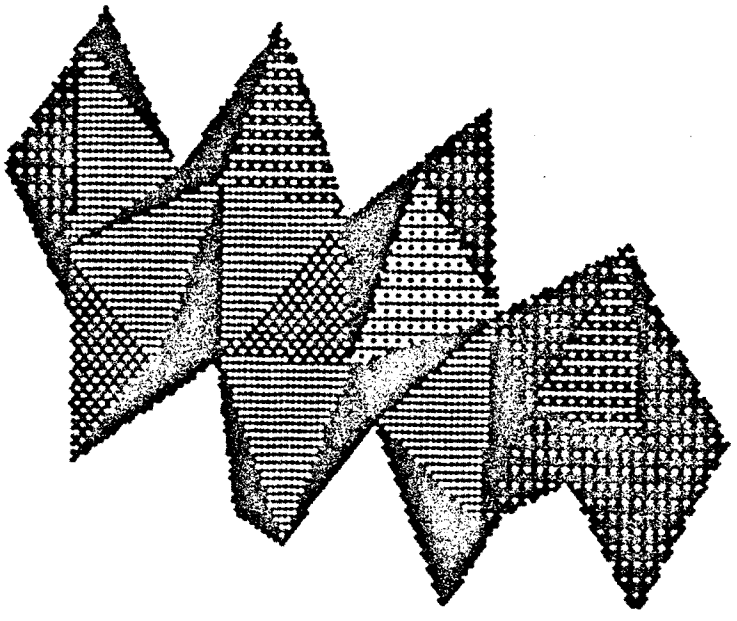
Le modèle de visibilité est constitué de faces décrites

par un contour extérieur et un ensemble de contours intérieurs. A chaque contour est associée une information d'aspect ; le modèle de visibilité permet d'employer dans la synthèse de l'image des algorithmes de remplissage performants. D'autre part, les calculs étant effectués dans l'espace objet, le modèle fourni est d'une grande précision, ce qui rend l'algorithme adapté à un grand nombre d'applications.

Transparence et ombres portées :

On a montré que le format même du modèle de visibilité permet de prendre en compte le phénomène de transparence et de produire les ombres portées dues à diverses sources lumineuses.

Si cette prise en compte augmente le temps de traitement d'une scène, elle permet par contre de produire des images d'un grand réalisme, ce qui se situe dans le cadre où nous utilisons les algorithmes d'étude de visibilité.



Isoèdres : algorithme de Newell, Newell et Sancha

2.5. Algorithme de Newell, Newell et Sancha ([NNS 72])

2.5.1. Description

Cet algorithme, dit à liste de priorité, est le premier permettant de rendre des effets tels que la transparence. Le modèle de visibilité n'est que le modèle de description réordonné. L'ordre des éléments permet d'obtenir l'image finale lors de l'affichage (voir figure 2.72.).

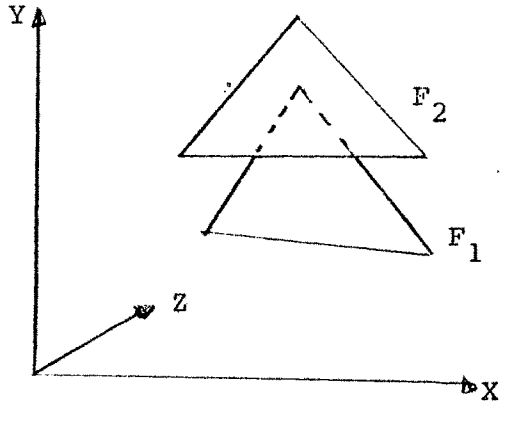


Fig. 2.72. Un modèle de visibilité.

Le modèle de visibilité sera constitué des faces F1 et F2 dans cet ordre.

Il se peut, qu'au cours du classement des faces, aucun ordre ne puisse être établi ; dans ce cas, une face sera découpée (figure 2.73.).

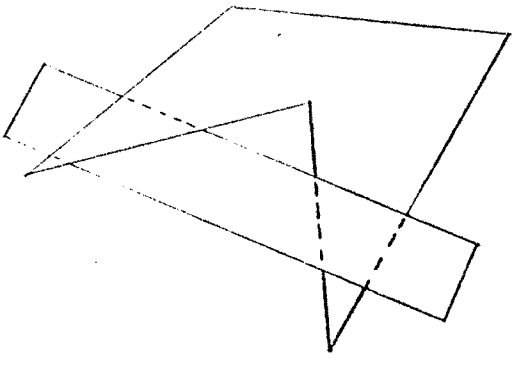


Fig. 2.73. Cas d'un découpage de face.

Ces deux faces se cachent mutuellement, une des deux sera découpée en fonction du plan de l'autre.

2.5.2. Etude des différents éléments de l'algorithme

2.5.2.1. Le modèle de description

Description des éléments traités

Cet algorithme étudie la visibilité d'un ensemble de faces polygonales planes.

Chaque face polygonale est décrite par un contour extérieur convexe ou non ainsi que par des contours intérieurs qui définissent des trous.

De plus, les éléments qui constituent la scène peuvent se pénétrer.

Il est à noter qu'à l'opposé des 3 autres algorithmes étudiés, celui-ci n'utilise aucune information d'aspect pour déterminer les éléments visibles, fait dont on étudiera les implications dans la description du modèle de visibilité.

Représentation des éléments traités

Chaque contour est représenté par la suite des sommets qui le définissent (figure 2.74.).

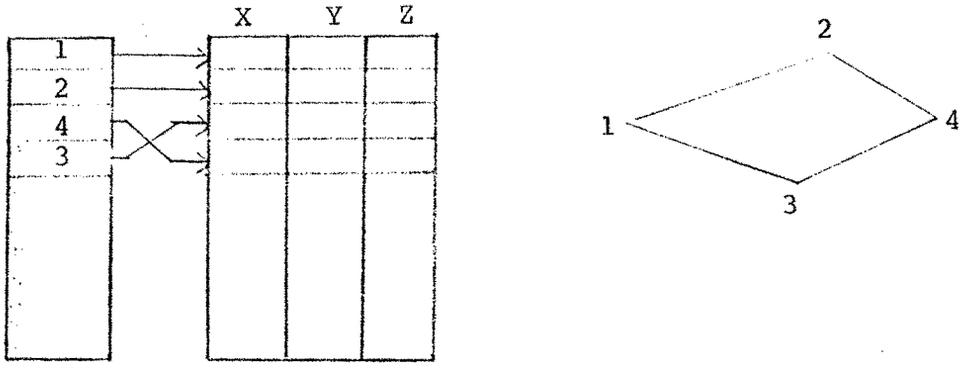


Fig. 2.74. Structure de données pour le modèle de description.

D'autres informations sont calculées avant l'étude de visibilité :

. les cotes minimales et maximales de chaque face, suivant chaque axe ;

. les coefficients de l'équation du plan de support de chaque face. Ils sont calculés de telle sorte que :

soit P l'équation d'une face

O l'observateur.

On aura : P (O) > o. Ceci permet de définir le demi espace avant et le demi espace arrière de chaque face.

2.5.2.2. Tri initial de données

Chaque face possède un ou plusieurs points les plus éloignés de l'observateur qui sont définis par une valeur maximum en z. (voir figure 2.75.).

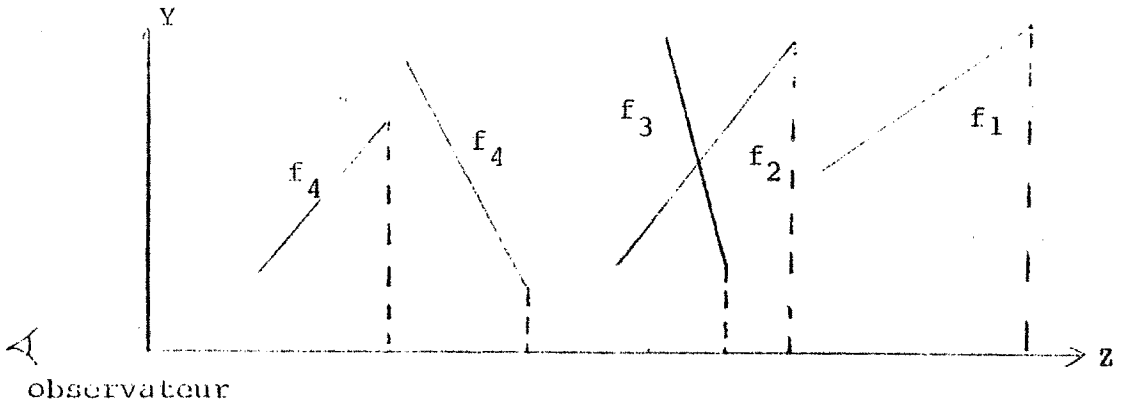


Fig. 2.75. Clés associées aux faces.

L'ensemble des faces constituant le modèle de description sera trié suivant la valeur maximale en z associée à chacune des faces et ceci par valeur décroissante.

La première face de la liste aura donc une forte probabilité de ne cacher aucune autre face.

Ne possédant aucune information à priori sur l'état des données à trier, le tri choisi a été le tri par segmentation.

2.5.2.3. ϕ : Ensemble des opérations géométriques utilisées

L'ensemble ϕ est composé de deux éléments, le calcul des positions relatives des faces et le découpage d'une face par une droite.

Détermination des positions relatives en profondeur de deux faces.

Chaque face définit un demi espace avant et un demi espace arrière (figure 2.76)

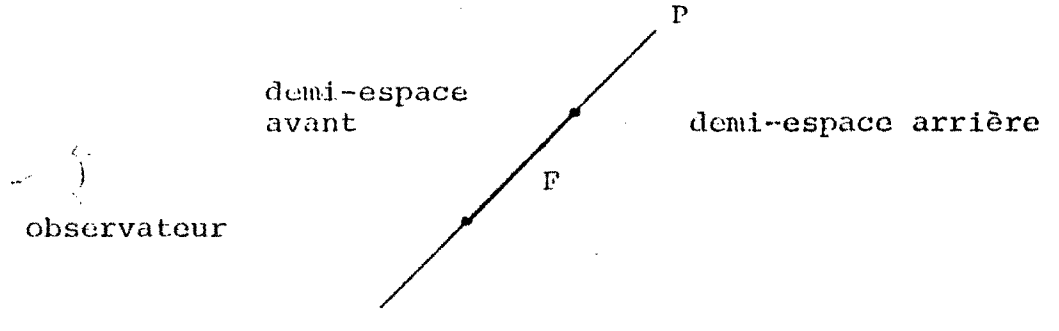


Fig. 2.76. Orientation d'une face.

On désire déterminer dans quel demi-plan se situe une face f.

f est définie par une suite (s_1, ... s_n) de sommets et on applique n fois le test C_P^{si} qui consiste à reporter dans l'équation du plan P les coordonnées du sommet si.

Ce test est donc proportionnel au nombre de sommets qui définissent la face f et permet de dire si f peut cacher F ou ne cache pas F.

Détermination des positions relatives des projections de deux faces

Soient F1 et F2 les projections de deux faces, il s'agit de donner une des réponses suivantes :

- une face contient l'autre
- les deux faces sont disjointes ;
- les deux faces se coupent.

Pour effectuer cette comparaison, nous avons utilisé des tests de complexité croissante :

. Tests minimaux : on compare les coordonnées extrêmes suivant les deux axes (figure 2.77.).

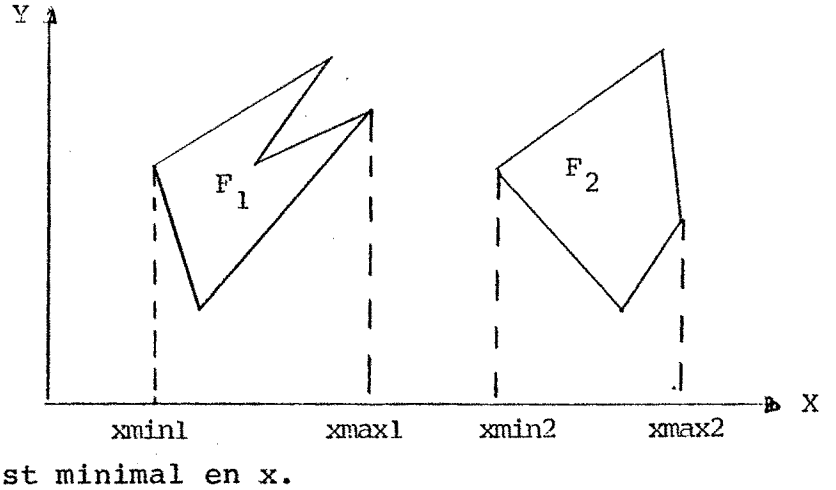


Fig. 2.77. Test minimal en x.

Les deux faces sont disjointes si :

(x max 1 < x min 2) ou (x max 2 < x min 1)

Si aucune des deux conditions n'est vérifiée, on effectue le test équivalent suivant l'axe des ordonnées (figure 2.78.).

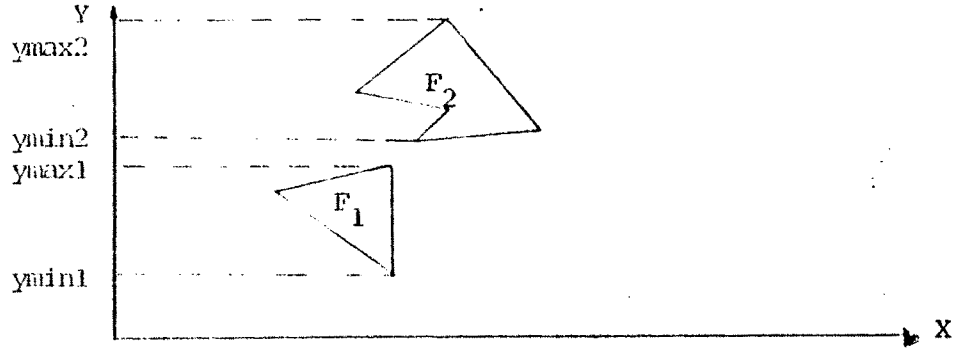


Fig. 2.78. Test minimal en y.

. Comparaison de deux contours : si les tests précédents n'ont pas déterminé que les deux projections sont disjointes, nous utilisons alors une opération C_c^C de comparaison de deux contours qui procède comme suit:

. on recherche si les deux contours se coupent ;

. si ils ne se coupent pas, on détermine lequel contient l'autre ;

. si aucune inclusion n'est trouvée, les deux contours sont disjoints.

Cette opération C_c^C a été décrite auparavant (cf.3.2.3.3). et il est apparu que des méthodes de comparaison de deux contours convexes étaient très performantes. Il est donc important d'utiliser de telles informations sur les contours afin d'améliorer les performances de cet algorithme.

Découpage d'une face par une droite.

Cette opération C_c^D a été décrite en 3.2.3.3. ; nous rappellerons seulement son but :

Le résultat du découpage sera un ensemble de faces ré-

sultant de l'intersection de la droite et du contour (figure 2.79.)

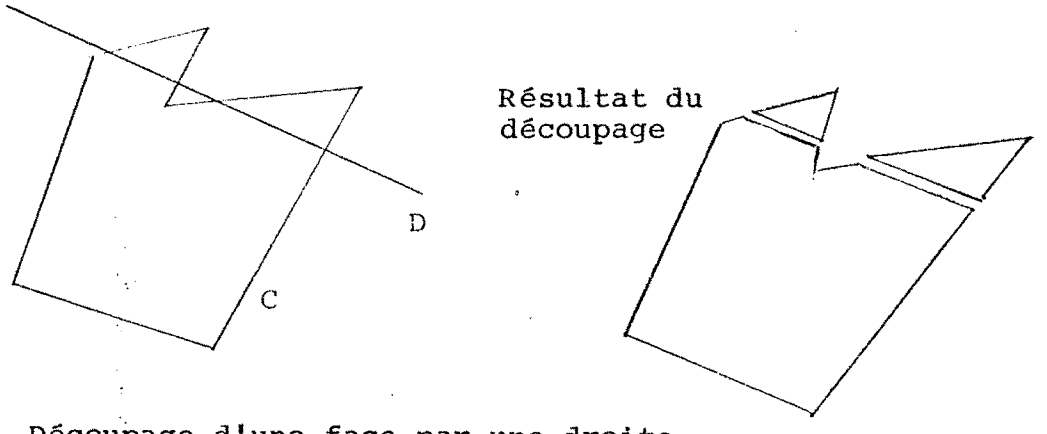


Fig. 2.79. Découpage d'une face par une droite.

2.5.2.4. Fonction stratégie

Présentation

Avant de décrire plus précisémment le principe d'étude de visibilité, nous devons présenter le modèle de visibilité qui est fourni. C'est une liste de faces ordonnées, de telle sorte qu'une face ne peut cacher les éléments suivants dans la liste.

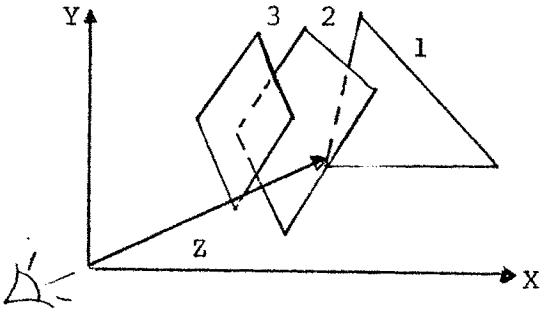


Fig. 2.80. Exemple de modèle de visibilité.

Or, l'ensemble des faces constituant la scène a été trié auparavant en fonction de l'éloignement vis-à-vis de l'observateur. Voyons quelles peuvent être les différences entre cette liste et le modèle de visibilité, c'est-à-dire les insuffisances du tri :

Cas 1 : P a un point plus éloigné de l'observateur que Q et P cache Q (fig. 2.81.a)

Cas 2 : P et Q ont la même clé pour le tri ; elles se trouvent donc dans un ordre quelconque dans la liste (figure 2.81.b).

Cas 3 : Les deux faces se pénètrent. On ne peut pas ordonner ces éléments. Il faudra en découper une fonction de l'autre (fig. 2.81.c).

Cas 4 : Les deux faces se recouvrent mutuellement. Ceci implique qu'au moins une des deux faces n'est pas convexe, car toutes les faces traitées sont planes (fig. 2.81.d). Il est impossible d'ordonner ces deux éléments, il faudra de même que dans le cas précédent, découper une face en fonction de l'autre.

Cas 5 : Recouvrement cyclique entre au moins trois faces (fig. 2.81.e) ; il est impossible d'ordonner ces éléments. En effet, (notons < la relation "est cachée par") on obtient :

$Q < P$ et $R < Q$

et $P < R$

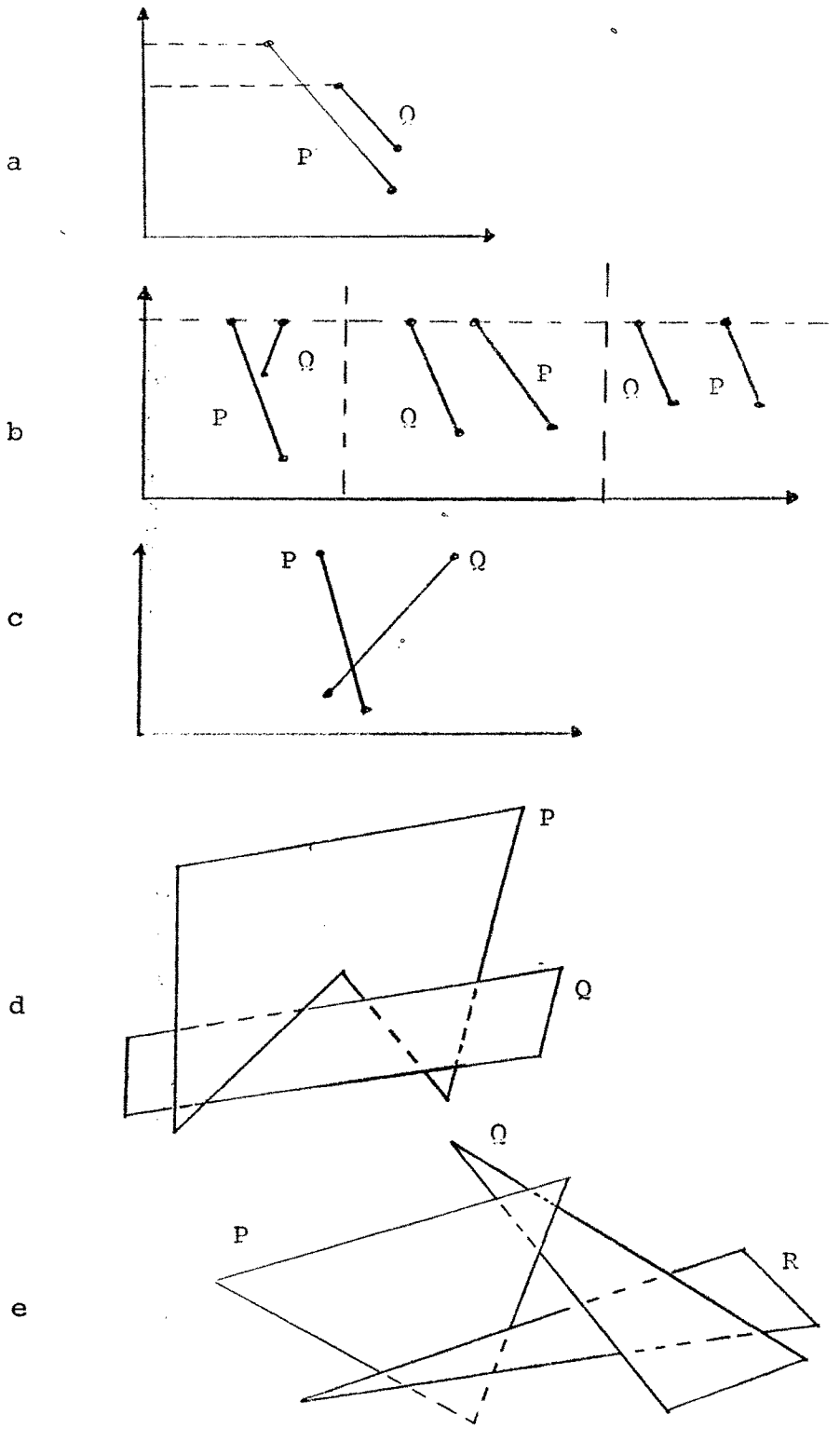


Fig. 2.81. Différents cas d'insuffisance du tri initial.

Parmi tous ces cas qui imposent donc un traitement de la liste ordonnée, on peut distinguer deux catégories :

- Les cas a, b et e qui proviennent de l'insuffisance même du tri ;

- les cas c et d qui proviennent de certaines propriétés topologiques de la scène.

. Cas c : la scène contient des faces qui se pénètrent.

. Cas d : La scène comporte des faces nonconvexes

Si certaines informations d'ordre topologique sont attachées à la scène (pas de faces qui se pénètrent, toutes les faces sont convexes) il ne sera pas nécessaire de rechercher de tels cas et le traitement de la liste se limitera à la détection des autres.

La fonction stratégie va régir l'étude de visibilité des éléments en vérifiant si l'ordre établi par le tri initial est correct et en apportant des modifications à la liste si il ne l'est pas.

Soit P l'élément tête de liste ; l'opération élémentaire est la comparaison de P à un ensemble Q_p défini par (voir figure 2.82.) :

$$Q_p = \{ q \mid z\ max\ q \in]\ z\ min\ p,\ z\ max\ p]\}$$

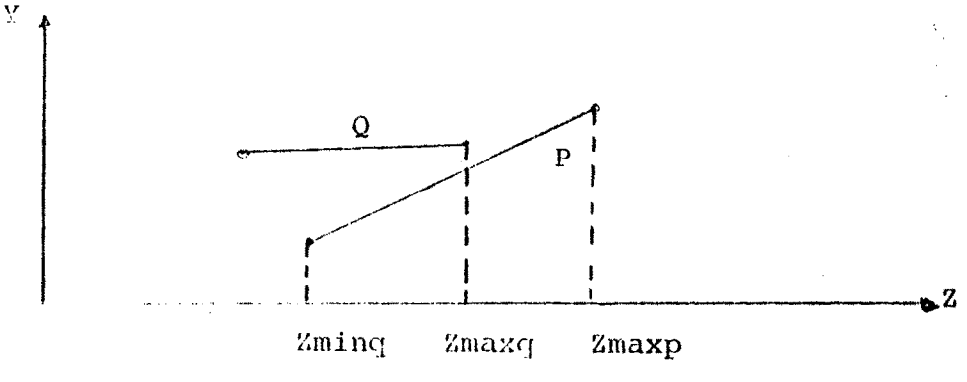


Fig. 2.82. Définition de Q_p.

La fonction stratégie repose sur le fait que l'on dispose de tests simples qui permettent de montrer qu'une face n'en cache pas une autre, mais qui par contre, ne permettent pas d'affirmer qu'une face en

cache une autre. L'opération de base sera une comparaison de deux faces à l'aide de ces tests.

Le fait que l'on ne puisse établir de relation entre deux faces ou plus (cf. cas c , d et e) implique l'emploi d'un marquage des faces afin d'éviter un bouclage dans le traitement.

La fonction stratégie employée peut se décrire de façon simplifiée ∀ p ε MD : comparer (p, q | q ε Qp)

Description de l'algorithme

```
P := Premier ;
tant que P ¬ = nil faire
début
    trouve := faux ;
    tant que ¬ trouve faire
    début
        Q := succ (P) ;
        si Q = nil  alors début Insérer dans modèle (P) ;
                               Oter (P) ; trouve :=vrai ;
                        fin
        sinon
        début
        si z max (Q) < = z min (P) alors
        début
            trouve := vrai ;
            si ¬ marque (Q) alors début insérer dans modèle
                                        (P) ;
                                   ôter (P) ;
                                fin ;
        fin
        sinon
        si  P cache Q trouve := vrai
        fin ;
```

```
SI Q ≠ nil alors
début
   si (Marque (Q)) ou (  Marque (Q) et Q cache P)
   début   découpage (P, Q) ; ôter (élément découpé) :
           Insérer éléments découpés ;
   fin
   sinon
   début
            Insérer en tête (Q)
            Marque (Q) := vrai
   fin
fin
P := Premier
fin algorithme ;
```

Réalisation du test P cache Q

Le but de ce test est de donner une réponse négative à la question "P cache-t-il Q ?". Il est réalisé par une suite de tests de complexité croissante qui sont :

. Test minimaux (cf III 1.2) qui indiquent si les projections des deux faces se recouvrent.

. P appartient-il au demi espace arrière défini par Q (cf III 1.1)

. Q appartient-il au demi espace avant défini par P (cf III.1.).

. Comparaison des deux projections par un test donnant un résultat exact. Pour ceci on utilise une opération C_c^c (pour la description et la réalisation voir 3.2.3.3.).

Marquage des faces

Une face est marquée lorsque, comparée à la face qui est tête de liste, il s'avère qu'on ne puisse pas les ordonner.

Ces différents cas ont été indiqués auparavant.

Le marquage sert de plus à éviter que l'algorithme boucle dans le cas d'un recouvrement cyclique entre plusieurs faces.

Découpage des faces

Soit P la face tête de liste qui est comparée à la face Q. Un découpage d'une des deux faces se produit dans deux cas :

. il se peut que P cache Q et Q est marquée

. Q n'est pas marquée et il se peut que Q cache P

Le problème est de déterminer quelle face doit être découpée.

Dans le deuxième cas, la face P est découpée,

sinon :

si Q n'est pas dans le demi espace arrière de P ,

Q est découpée

si P n'est pas dans le demi espace avant de Q,

P est découpée

sinon :

on crée une droite de découpage qui vérifie les conditions suivantes et qui permettra de découper la face P (choix arbitraire)

Cette droite est choisie parallèle à l'axe

des abscisses si (y max (P) y min (P) > x max (P)

x min (P))

des ordonnées sinon.

2.5.2.5. Ensemble des représentations intermédiaires

La première représentation intermédiaire contient l'ensemble des faces du modèle de description trié par le tri initial Td décrit auparavant.

Les autres représentations intermédiaires obtenues à partir de celle ci et qui conduiront au modèle de visibilité sont issues successivement l'une de l'autre par des modifications qui sont de trois types :

. l'élément tête de liste est supprimé et inséré dans le modèle de visibilité ;

. un élément, lors d'une comparaison, est forcé en tête de liste ;

. une face est découpée. Elle est ôtée de la liste des éléments traités et remplacée par les faces issues du découpage. Ceci necessitera un tri de la liste qui sera réalisé par un tri intermédiaire Ti qui est un tri par insertion.

Dans une représentation intermédiaire se trouvent deux types de faces :

. des faces qui appartiennent au modèle de description ;

. des faces qui proviennent d'un découpage. Il est à noter qu'il est inutile de comparer entre elles des faces provenant du découpage d'une même face, car elles ne peuvent se cacher étant donné que les éléments traités sont plans.

2.5.2.6. Modèle de visibilité

Description

Le modèle de visiblité est constitué d'un ensemble de faces ordonné. L'ordre est en fait celui suivant lequel les faces doivent

être affichées ou traitées par le module de synthèse d'image.

Il est composé de deux types de faces :

. faces qui appartiennent au modèle de description

. faces qui proviennent d'un découpage.

En résumé, le modèle de visibilité n'est qu'une réorganisation du modèle de description, les seules modifications étant dues au découpage.

Critique

Le principal inconvénient du modèle de visibilité est le nombre de faces qui devra être traité afin d'obtenir l'image. Le traitement (remplissage des faces) sera proportionnel à la surface totale des faces projetées.

Ce modèle de visibilité présente par contre plusieurs avantages qui sont :

. possibilité de rendre les effets de transparence : ceux-ci seront rendus au moment de l'affichage par combinaison de l'aspect d'une face transparente et de l'aspect des faces situées en arrière.

. du fait qu'aucune information d'aspect n'est utilisée pendant l'étude de visibilité il est possible de visualiser une même scène en faisant varier l'aspect de certains de ses éléments sans qu'il soit nécessaire d'effectuer à nouveau une étude de visibilité.

. les calculs sont effectués dans l'espace objet et la grande précision avec laquelle le modèle de visibilité est obtenu permet des visualisations de la scène à des échelles variées.

2.5.2.7. Critiques

Evaluation

Nous mettons en valeurs certaines caractéristiques de la scène qui influent sur le comportement de l'algorithme et donnons le comportement de l'algorithme pour certains cas extrêmes.

Si l'on fait un récapitulatif des opérations géométriques et tris utilisés,

. Deux tris sont utilisés

Td : tri initial des faces du modèle de description. Le tri réalisé est un tri par segmentation.

Soit nbfaces le nombre de faces initiales ; la complexité minimale du tri sera nbfaces, log (nbfaces) et la complexité maximale $(nbfaces)^2$ qui est d'une probabilité assez faible mais qui intervient lorsque les données sont initialement presque triées.

Ti : tri intermédiaire employé lorsque une face découpée est remplacée par les faces résultant du découpage. Le tri parinsertion est de complexité égale n si n est le nombre d'éléments de la représentation intermédiaire traitée.

. Trois opérations géométriques sont utilisées

* C_p^p qui est utilisée lors de la comparaison de deux faces F et G pour déterminer dans lequel des deux demi espaces définis par G F se situe
  Si F comporte f sommets, le traitement est proportionnel à f;
* C_C^D utilisée par le découpage d'une face C par une droite D
  Si c est le nombre de sommets de C, le traitement est proportionnel à c.
  Un point important dans cette opération est le nombre de faces produites.
  si C est convexe, deux faces seront produites ;
  sinon, le nombre de faces produites sera compris entre
  2 et $\frac{c}{2} + 1$
* C_c^c qui est la comparaison de deux contours
  est proportionnel à la somme des nombres de sommets si les contours sont convexes ou au produit sinon.

Quelques cas particuliers se présentent :

. Supposons que pour tout couple de faces (F1, F2), nous ayons la propriété suivante (figure 2.83.) : (Z max 1 < z min 2) ou

(Z max 2 < z min 1)

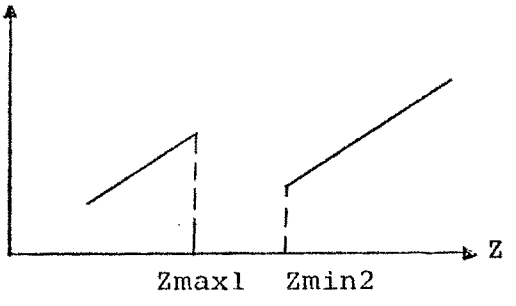


Fig. 2.83. Un cas particulier.

et que de plus, il n'y ait pas de recouvrement cyclique. L'étude de visibilité va consister en la comparaison de chaque face P aux faces de l'ensemble Qp, qui est vide, d'où un comportement en O (nbfaces) dans ce cas particulier. On remarque de plus que n'interviennent ici que les positions relatives en profondeur et non les positions latérales des faces.

. Supposons qu'il n'y ait aucun recouvrement cyclique mais que chaque face soit en recouvrement mutuel avec une face P. Supposons de plus que l'ordre fournit ne soit pas exact pour chaque coupe de faces, c'est-à-dire, qu'on ait la situation suivante (figure 2.84.).

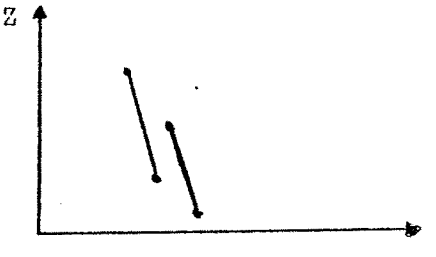
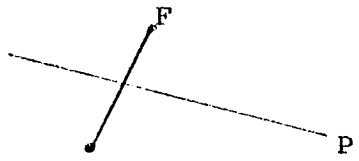


Fig. 2.84.

Soit r le nombre moyen de sommets par face dans la scène. Une première étape va consister en un découpage de chaque face par P. Chaque découpage produit au maximum $(\frac{r}{2} + 1)$ (figure 2.85.).



F fournit $\frac{r}{2}+1$ faces au maximum

Fig. 2.85. Découpage d'une face.

Soit k x $(\frac{II}{2} + 1)$ le traitement sera proportionnel

si α = k x nbfaces à α^2 soit

(nbfaces) x (nbfaces produites)^2

. Influence du recouvrement cyclique

Soient (P_1 - Pn) n faces telles qu'il y ait un recouvrement cyclique. Celui-ci sera détecté lorsque toutes les faces auront été mises à tour de rôle en tête de liste. Ceci sera fait pour n études, chacune comportant (n - 1) comparaisons de faces. Le comportement dû à la présence d'un recouvrement cyclique sera donc proportionnel à (nbfaces)^2

Cette étude a mis en valeur différentes caractéristiques de la scène qui influent sur le comportement de l'algorithme et qui sont :

Le nombre de sommets par face :

Facteur qui intervient dans les trois opérations géométriques.

La convexité ou non convexité des faces qui intervient dans les deux opérations géométriques C_c^c et C_c^D. De plus, cette caractéristique de la scène permet de dire s'il y aura ou n'y aura pas de recouvrement mutuel entre deux faces, car pour que ceci se produise, il faut qu'une des

deux faces ne soit pas convexe. Ce facteur intervient aussi dans le nombre de faces issues du découpage d'une face par une droite.

La répartition latérale des faces : si les faces sont "bien espacées" latéralement, le fait qu'elles ne peuvent se cacher pourra être déterminé en employant des tests minimaux de coût peu élevé.

La répartition en profondeur des faces : En effet, un élément P est comparé à tous les éléments de Qp, plus cet ensemble comportera d'éléments, plus le nombre de comparaisons sera élevé.

On peut dire de façon simplifiée que le cardinal de cet ensemble sera inversement proportionnel à la distance entre les faces et à l'inclinaison moyenne des faces.

Le recouvrement cyclique entre plusieurs faces dont la présence augmente considérablement le nombre de comparaisons.

. Prétraitements appliqués au modèle de description

Nous distinguons deux types de prétraitements qui peuvent être appliqués au modèle de description avant d'effectuer l'étude de visibilité.

Prétraitement avec suppression d'éléments

On peut considérer deux traitements qui sont :

- la suppression des faces arrières d'un objet dont aucune face avant n'est transparente ;

- la suppression des faces parallèles à l'axe de vue et qui ne peuvent donc pas être visibles.

L'intérêt d'un tel prétraitement est bien sûr de diminuer le nombre de faces à traiter mais on peut relever par contre deux inconvénients :

a) le prétraitement fait intervenir l'aspect des faces et on ne pourra donc pas, par la suite, modifier cet aspect sans réeffectuer une étude de visibilité.

b) la suppression des faces ne permet pas d'obtenir un modèle de visibilité qui n'est qu'une simple réorganisation du modèle de description et qui permettrait dans un contexte d'emploi dynamique de le réutiliser afin d'obtenir des vues successives.

. Réorganisation du modèle de description

On découpe la scène en zones et en boîtes

Une zone est un ensemble de faces dont l'état, visible ou invisible, peut être indépendant. Deux zones n'ont aucune influence mutuelle au point de vue de la visibilité (figure 2.86.).

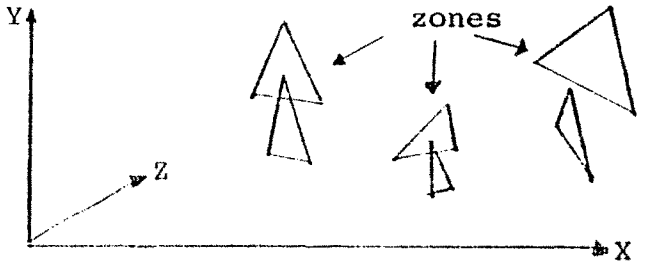


Fig. 2.86. Découpe d'une scène en zones

Une zone peut être décomposée en un ensemble de boîtes Deux boîtes sont séparées par un plan parallèle au plan xoy (figure 2.87.)

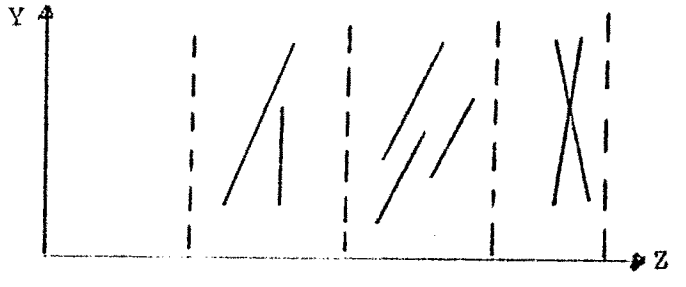


Fig. 2.87. Décomposition d'une zone en boîtes.

L'intéret des zones est de réduire le nombre de faces qui doivent être comparées, intérêt d'autant plus grand vu le comportement de l'algorithme d'étude de visibilité.

L'ensemble des boîtes formant une zone est ordonné par éloignement décroissant vis à vis de l'observateur, chaque boîte fournit un modèle de visibilité qui sera inséré dans le modèle de visibilité lié à la zone suivant l'ordre de cette boîte dans la liste.

L'intérêt de cette décomposition est multiple :

. les éléments à trier par Td et Ti sont en nombre moindre

. soit P une face à traiter les éléments formant à Qp appartiennent à la même boîte que P.

En fait ce prétraitement peut être poussé plus loin. On peut en effet décomposer à leur tour chaque boîte en zone et continuer ce prétraitement jusqu'à ce que les boîtes ne puissent être décomposées. On remarque que si à l'issue de ce prétraitement les boîtes obtenues à la fin de cette décomposition ne contiennent chacune qu'une face, l'étude de visibilité est en fait réalisée (figure 2.88.).

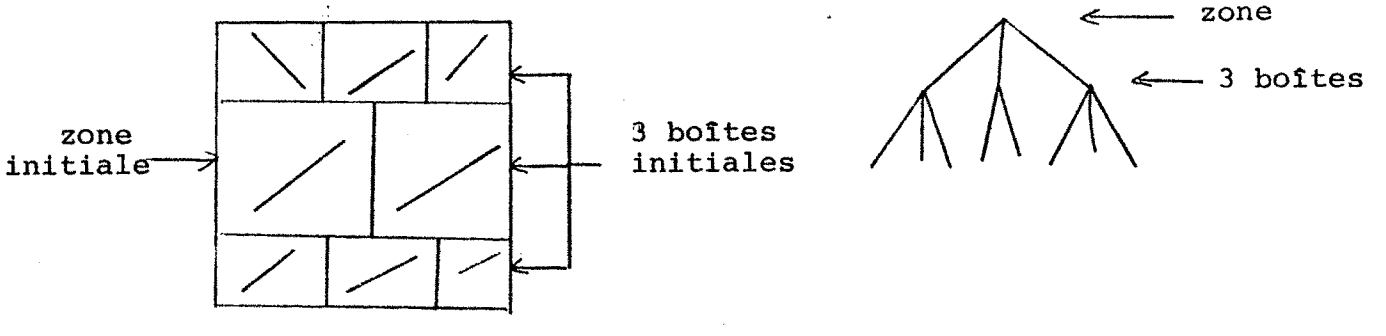


Fig. 2.88. Décomposition et structuration d'une scène.

Si cette décomposition est représentée par une structure arborescente, il suffira d'en effectuer un parcours pour obtenir le modèle de visibilité.

On peut aussi appliquer un prétraitement aux objets convexes. Soit un objet convexe qui n'est pénétré par aucun élément de la scène. Cet objet peut être décomposé en deux ensembles : l'ensemble des faces arrières, l'ensemble des faces avants.

Le deuxième ensemble peut être considéré comme une seule face dont le contour extérieur est la silhouette de cet objet. Seule la face ainsi obtenue sera traitée ensuite dans l'étude de visibilité. Au lieu de l'insérer dans le modèle de visibilité on insérera d'abord les faces arrières de l'objet, puis les faces avants.

Cette méthode qui ne modifie pas la scène permet ainsi de réduire le nombre de faces à traiter.

2.5.3. Conclusion

Cet algorithme présente un ensemble d'avantages qui sont en fait tous liés à la nature du modèle de visibilité qui est fourni :

. pour la synthèse d'images il permet de rendre les effets de transparence et de modifier l'aspect d'éléments du modèle de description sans qu'une étude de visibilité soit à réeffectuer.

. le fait que le modèle de visibilité soit une réorganisation du modèle de description permet de le réutiliser lors d'une étude de visibilité postérieure. Si l'application désire un ensemble de vues successives d'une scène où l'observateur change de position, il sera intéressant de réutiliser le modèle de visibilité précédent, comme modèle de description. On peut penser que le nombre de changements d'une vue à une autre sera minime et de plus, certains cas de figure comme le recouvrement mutuel auront déjà été résolus.

Cette étude a mis par contre en évidence le comportement de cet algorithme qui dépend du nombre total d'éléments à traiter et ceci parfois de façon exponentielle.

2.6. Conclusion

L'étude de la réalisation de ces quatres algorithmes a mis en valeur différents critères de choix qui sont résumés dans la figure 2.88. Nous avons évoqué pour chaque algorithme quelles étaient les possibilités de décomposition en traitement en parallèle.

Si on peut dire qu'il n'y a pas de meilleur algorithme, on peut toutefois écarter l'algorithme de Warnock qui pêche tant par la pauvreté du dessin obtenu que par son comportement. Nous avons mis aussi en valeur dans ce chapitre les caractéristiques de la scène qui influent sur le comportement des algorithmes. Nous indiquons ci-dessous pour chacun quelles sont ces caractéristiques et pour quelles fonctions elle interviennent :

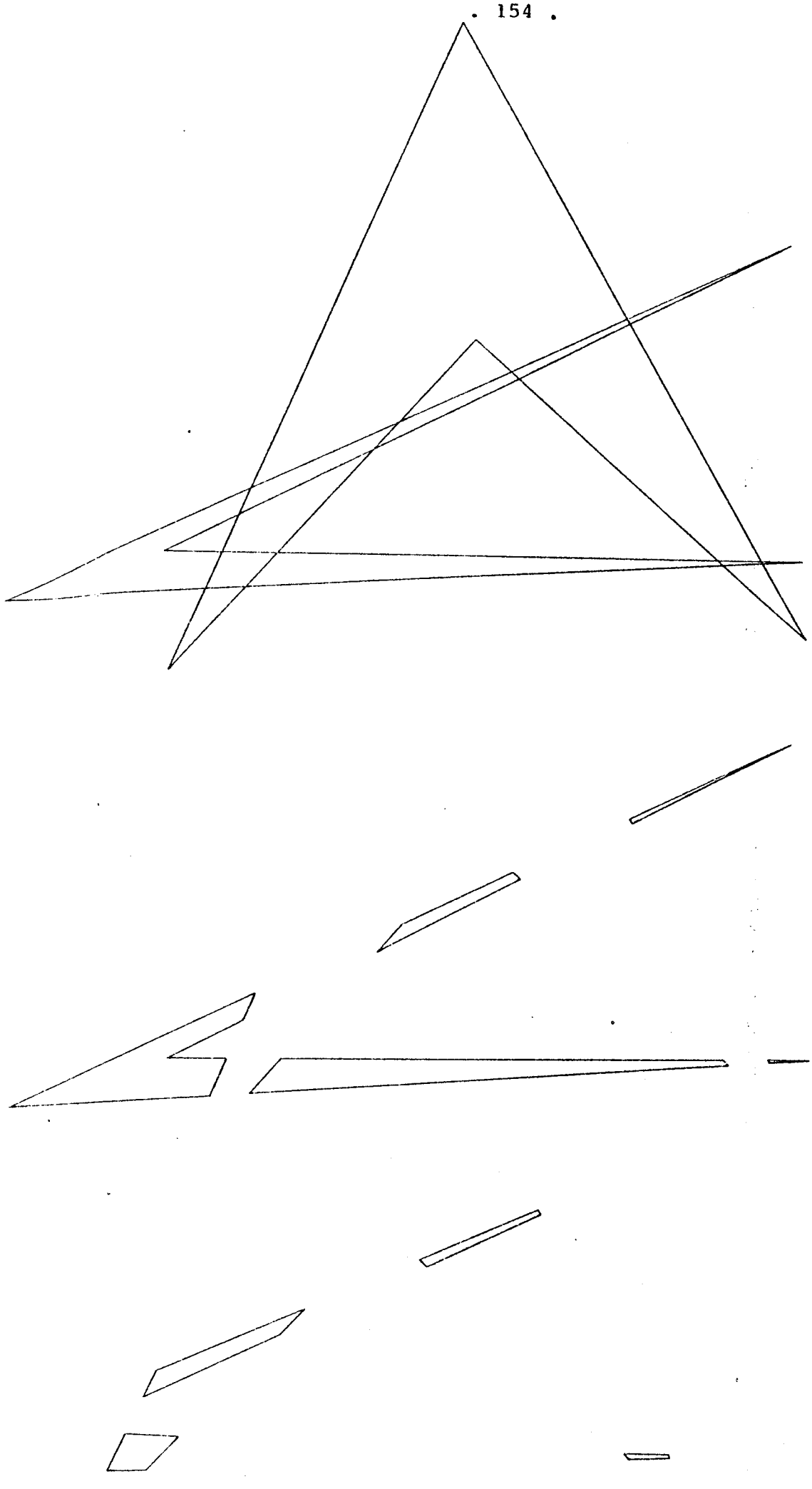
- algorithme de Watkins
  - Convexité ou non convexité des faces : nombre de plans à étudier ; nombre de segments dans un plan.
  - Nombre de segments (nbseg) et nombre de segments visibles (nvis) :
    - étude de visibilité (méthode Pile O(nbseg), méthode boîte O(nbseg x log(nvis))
    - tri intermédiaire
  - pénétration de faces : comparaison de segments.
- algorithme de Warnock
  - nombre de faces : tri initial
  - surface de la scène projetée : choix de la fenêtre initiale et nombre maximum de découpages ;
  - répartition en z et forme, surface, orientation des éléments visibles : analyse d'une fenêtre ;
  - pénétration des faces : test C_P^P
  - nombre d'arêtes par contour : tests C_C^C, C_C^P

- algorithme de Atherton et Weiler
  - . nombre de faces : tri initial
  - . nombre de sommets par contour : tests C_P^P, C_F^F
  - . complexité du recouvrement entre deux faces : test C_F^F
  - . convexité ou non convexité des contours : test C_F^F
- algorithme de Newell, Newell et Sancha
  - . nombre de faces : tri initial
  - . nombre de sommets : tests C_c^D, C_p^p
  - . convexité ou non convexité : recouvrement mutuel test C_c^D
  - . séparabilité des faces : test C_c^c .

Une caractéristique commune à tous ces algorithmes est le nombre de zones qui permet un traitement en parallèle.

	Scènes	Image	Dessin	Affichage arêtes d'intersection	Transparence	Remplissage	Mémoire	Programmation	Comportement
WARNOCK	faces polygonales sans trou Pénétration autorisée	Moyen	Mauvais	Oui	Non	Moyen - le nombre de contours à remplir peut être important	3 * nbsom + 37 * nbfaces	Simple	Mauvais
WATKINS	faces polygonales Trous Pénétration pour la methode boîte	Bon	Moyen	Non	Possible	bon - Adapté au balayage télévision	3 * nbsom + 33 * nbfaces	Complexe	bon - La méthode Pile est dans l'ensemble meilleure que la méthode boîte
ATHERTON	faces polynales Trous Pénétration	Bon	Bon	Non	Possible	bon - Nombre minimal de contours à remplir	5 * nbsom + 40 * nbfaces	Moyen	Moyen
NEWELL	faces polynales Trous Pénétration	BOn	Mauvais	Non	Oui	Mauvais - Grand nombre de contours à remplir	3 * nbsom + 40 * nbfaces	Simple	Moyen

Fig. 2.88.



CHAPITRE III

LES TRAITEMENTS ELEMENTAIRES

3.1. Introduction

Dans la dénomination "traitements élémentaires" on regroupe en fait deux types de traitements qui sont :

. les opérations géométriques qui sont les fonctions de transition dans la formalisation des algorithmes de visibilité ;

. les opérations de tri qui sont les composants Td et Tid'un algorithme

Nous étudierons ici diverses solutions qui peuvent être apportées à chacun de ces problèmes en mettant en valeur différents points :

. le problème se décompose-t-il en sous-problèmes élémentaires ?

. quel est le comportement de l'algorithme réalisant cette opération

. peut-on attendre des améliorations en utilisant des informations précalculées sur les éléments qui sont traités ?

. La méthode peut-elle être cablée ou être décomposée en traitements en parallèle ?

Nous ferons d'abord un rappel des opérations géométriques utilisées dans divers algorithmes afin de souligner deux points :

- différents algorithmes de visibilité utilisent les mêmes opérations géométriques ;

- toute opération géométrique se trouve au coeur d'un algorithme de visibilité du fait de son emploi répété lors de la comparaison d'éléments qui appartiennent soit au modèle de description soit à une représentation intermédiaire.

Algorithme de Catmull (cf. [Cat 74]) : technique de Z. buffer.

Cet algorithme, travaillant dans l'espace écran, détermine en chaque point élémentaire de la surface de visualisation, quel est l'élément visible il faut donc, pour chaque droite issue d'un point de l'écran et perpendiculaire à celui-ci, déterminer quels polygones elle coupe et quel est le point d'intersection le plus proche de l'observateur. Ce problème peut se formuler

de la façon suivante :

. pour chaque point écran, déterminer dans quels contours polygonaux il est contenu, ce qui est un problème de comparaison point-polygone dans un plan.

. déterminer la profondeur des points d'intersection à l'aide des équations des plans de support des polygones.

Algorithme de Watkins (cf. [RWE 69])

L'algorithme de Watkins se ramène à l'étude de visibilité dans un plan en déterminant pour celui-ci, quels sont les polygones qui le coupent. Il faut déterminer l'intersection d'un polygone et d'un plan (horizontal).

On verra dans la suite de cette étude, l'analogie entre le calcul des segments ou portions de segments visibles dans un plan, et la détermination des intersection parmi un ensemble de segments coplanaires.

Algorithme de Newell et Sancha (cf. [NNS 73])

Les calculs géométriques utilisés dans cet algorithme sont :

- de deux polygones, déterminer lequel est le plus proche de l'observateur ;

- intersection d'un polygone par un plan, le résultat étant un ensemble de polygones coplanaires.

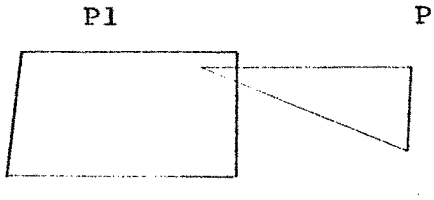
Algorithme de Atherton et Weiler (cf. [AtW 78])

Cet algorithme utilise deux opérations géométriques :

- de deux polygones, déterminer lequel est le plus proche de l'observateur ;

- découpage d'un polygone par un autre, ceci dans un plan.

Exemple : soit P1 et P2



P1 est coupé par P2. Le résultat sera le découpage de P1 en deux polygones.

Algorithme de Warnock (cf. [War 69])

L'algorithme de Warnock utilise deux opérations géométriques :

- un algorithme de découpage qui compare un polygone avec un carré ou un rectangle, et qui détermine si le polygone est extérieur à la fenêtre, ou l'englobe, ou la coupe.

- une comparaison de profondeur entre polygones qui sont en influence par rapport à une même fenêtre.

Algorithme de Appel (cf. [App 67])

Cet algorithme travaille dans l'espace objet et fournit un dessin au trait. Il utilise le calcul de l'intersection d'une droite et d'un polygone dans un espace à trois dimensions. Ceci se ramène à deux problèmes élémentaires :

- intersection d'une droite et d'un plan;

- un point est-il contenu dans un contour polygonal ?

On peut formaliser la notion d'opération géométrique de la façon suivante : une opération géométrique est en fait la comparaison de deux éléments i et j qui peuvent être, dans le contexte qui nous intéresse, un point, un segment, un contour, un plan...

Une opération géométrique peut combiner d'autres opérations élémentaires afin de réaliser cette comparaison (le découpage de deux contours peut utiliser une opération de calcul d'intersection entre segments).

Enfin, le résultat d'une comparaison peut être une valeur logique (un point est, ou n'est pas, intérieur à un contour) ou une liste d'entités géométriques (liste de points d'intersection).

Ceci nous conduit à la formalisation suivante :

Une opération géométrique sera décrite par :

$C_i^j = (i, j, O, S)$

. i, j sont les éléments comparés,

. O sont les opérations utilisées,

. S est l'ensemble de sortie.

Cette formalisation met en valeur :

. l'ensemble d'opérations utilisées qui peut être plus ou moins adapté au contexte d'utilisation ;

. les éléments i, j et S permettent de choisir pour une même opération C_i^j celle qui fournit l'élément de sortie S désiré.

. l'ensemble O met en valeur les divers liens qui existent entre les différentes opérations C_i^j.

Parmi toutes les opérations récapitulées dans le tableau suivant, un nombre réduit, qui entraient en jeu dans les algorithmes étudiés, ont été réalisées, les références indiquées aideront à une réalisation éventuelle des opérations non étudiées.

	Point	Segment	Droite	Contour	Plan	Polyèdre
Point	C_P^P : 2 points sont-ils confondus	C_S^P : position d'un point par rapport à un segment	C_D^P : position d'un point par rapport à une droite	C_C^P : position d'un point par rapport à un contour [Dat 77][Sha 77]	C_P^P : position d'un point par rapport à un plan	C_{Po}^P : position d'un point par rapport à un polyèdre
Segment		C_S^S : intersection de 2 segments [Lev]	C_D^S : intersection d'une droite et d'un segment [SUH 74]	C_C^S : intersection d'un segment par rapport à un contour [Nes73]	C_P^S : position d'un segment par rapport à un plan	C_{Po}^S : position d'un segment par rapport à un polyèdre
Droite			C_D^D : intersection de 2 droites	C_C^D : intersection d'un contour et d'une droite [SUH74]	C_P^D : intersection d'une droite et d'un plan	C_{Po}^D : intersection d'une droite et d'un polyèdre
Contour				C_C^C : intersection de 2 contours [Sha77][Atw 78]	C_P^C : intersection d'un contour et d'1 plan - [Bro76]	C_{Po}^C : intersection d'un contour et d'un polyèdre
Plan					C_P^P : intersection de 2 plans	C_{Po}^P : intersection d'un plan et d'un polyèdre [tC75]
Polyèdre						C_{Po}^{Po} : intersection de 2 polyèdres [Com68] [Mar72] [Tup78]

3.2. Etude de quelques opérations géométriques

De toutes les comparaisons citées nous avons choisi d'en présenter un nombre limité du fait de leur fréquence d'emploi dans divers algorithmes d'étude de visibilité et de leur importance au coeur de ces algorithmes.

L'étude portera sur :

. la comparaison de segments ,

. la comparaison point-contour,

. le découpage de contour par une droite ou un contour.

Pour chaque problème, diverses solutions sont présentées qui diffèrent par la nature des éléments traités, le comportement et les opérations élémentaires utilisées.

Cette étude permet de faire une revue, certes non exhaustive, des solutions apportées à ces problèmes et de guider dans le choix de l'une d'entre elles, en fonction du contexte d'utilisation.

3.2.1. Comparaison de segments :

3.2.1.1. Intersection de deux segments de droite : C_s^s

Cette opération est à la base d'un nombre important de comparaisons qui ont été citées auparavant et diverses solutions ont été apportées à ce problème parmi lesquelles quatre seront évoquées ici :

. la méthode analytique,

. la méthode vectorielle,

. la méthode paramétrique,

. la méthode min-max.

Méthode analytique

La méthode repose sur l'équivalence suivante :

AB et CD se coupent

<===>

Les points A et B sont de part de d'autre de la droite

support du segment CD

et

les points C et D sont de part et d'autre de la droite support du segment AB.

(Voir figure 3.1.).

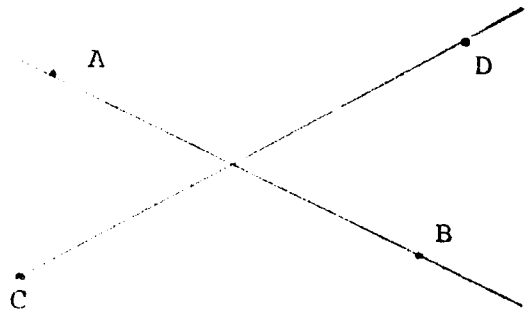


Fig. 3.1. Méthode analytique.

Méthode vectorielle

Cette méthode utilise les notions de produit vectoriel et de produit scalaire.

Les vecteurs $\vec{AC}$, $\vec{AD}$, $\vec{BC}$, $\vec{BD}$ sont situés dans un plan x O y. Les produits vectoriels $\vec{AC} \wedge \vec{AD}$ et $\vec{BC} \wedge \vec{BD}$ sont des vecteurs perpendiculaires à ce plan. S'il sont de sens opposé, leur produit sera négatif, et il sera positif dans le cas contraire.

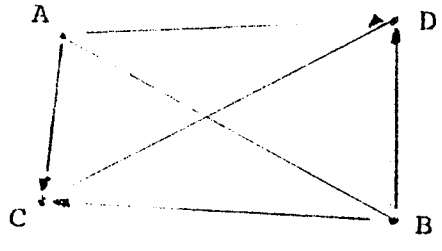


Fig. 3.2. Méthode vectorielle.

$$\text{AB et CD se coupent} \Longleftrightarrow \begin{matrix} (\vec{AC} \quad \vec{AD}) \cdot (\vec{BC} \quad \vec{BD}) < 0 \\ (\vec{CA} \quad \vec{CB}) \cdot (\vec{DA} \quad \vec{DB}) < 0 \end{matrix}$$

Méthode paramétrique

La méthode repose sur une approche paramétrique pour caractériser les droites support des segments.

La droite support du segment AB est définie par les équations paramétriques :

$$x = x_A + (x_B - x_A)\ t_1$$

$$y = y_A + (y_B - y_A)\ t_1$$

De même pour la droite support du segment CD :

$$x = x_C + (x_D - x_C)\ t_2$$

$$y = y_C + (y_D - y_C)\ t_2$$

Un point (x, y) appartenant au segment AB est tel que : $0 < t_1 < 1$;

et un point (x,y) appartenant au segment CD est tel que : $0 < t_2 < 1$.

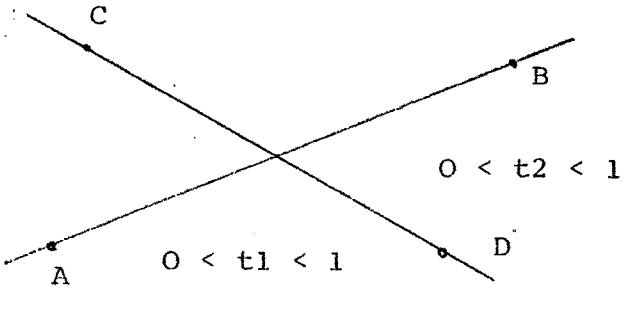


Fig. 3.3. Méthode paramétrique.

Le point d'intersection des deux droites a pour coordonnées t_1 sur AB, t_2 sur CD. Il suffit de vérifier :

$$\text{AB et CD se coupent} \Longleftrightarrow \begin{matrix} t_2 \in]0,1[\\ t1 \in]0,1[\end{matrix}$$

Méthode MIN-MAX

Cette méthode est proposée par W.K. GILOI (cf.[Gil 78], p. 157). Elle consiste à calculer les coordonnées (x_I, y_I) du point d'intersection des segments AB et CD ; puis à vérifier que ce point est à l'intérieur de la partie commune aux rectangles bâtis sur les segments AB et CD : (voir figure 3.4.)

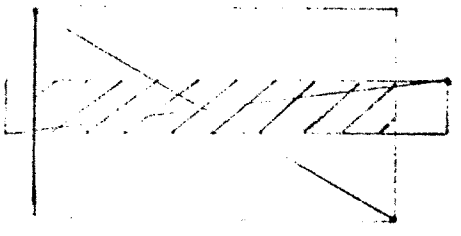


Fig. 3.4. Méthode Min-Max.

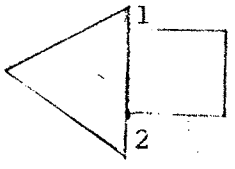
Conclusion

Le tableau ci-dessous, tiré de [Luc80], montre qu'il existe des différences appréciables d'une méthode à l'autre.

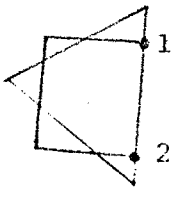
	+	x	?	:=
Méthode analytique	14	14	3	0
Méthode vectorielle	12	10	3	0
Méthode paramétrique	6	6	5	7
Méthode MIN-MAX	9	12	17	1

Il faut ajouter à ces évaluation le traitement de cas particuliers : segments parallèles, colinéaires... En particulier, la définition exacte des cas d'intersection dépend du contexte d'utilisation. Par exemple, pour la comparaison de deux contours, méthode qui compare les

segments, le traitement des cas particuliers nécessite l'examen des arêtes adjacentes dans le contour (voir figure 3.5.).



1.2 ne sont pas retenus comme des intersections



1.2 sont retenus comme des intersections.

Fig. 3.5. Cas particuliers.

Si, de plus, on tient compte des rapports existant entre les temps d'exécution des opérations élémentaires (+, x, test, affectation), les écarts entre les différentes méthodes varient du simple au double.Le choix d'un algorithme d'intersection devra donc se faire après une étude approfondie des données à traiter, afin de déterminer le meilleur programme.

3.2.1.2. Recherche d'intersection dans un ensemble de segments

Nous présentons une solution à ce problème, ayant beaucoup d'analogie avec les méthodes d'analyse de visibilité pour un ensemble de segments coplanaires.

Le problème est de déterminer toutes les intersections parmi un ensemble de N segments. La méthode triviale consiste à comparer tous les segments, ce qui conduit à un algorithme en $O(N^2)$ inacceptable lorsque un grand nombre de segments est à traiter. (exemple : masque de circuits intégrés). Nous présentons ici une solution due à Bentley et Ottman [BeO 78] obtenue par modification d'un algorithme présenté par Shamos et Hoey ([ShH 76])

Principe

L'algorithme procède par balayage de gauche à droite, à l'aide d'une droite verticale, de la portion de plan où les segments sont

présents.

Soit N le nombre de segments:

. les 2 N abscisses sont triées en ordre croissant, et sont représentées dans une structure de données Q, qui est un tas (minimier)

. Pour chaque position de la droite verticale, il est possible d'ordonner les segments présents suivant une relation d'ordre totale représentée par une s.d.d R qui est un arbre équilibré. Les changements d'ordre entre segments lors du balayage permettront de déterminer les intersections. (figure 3.6.).

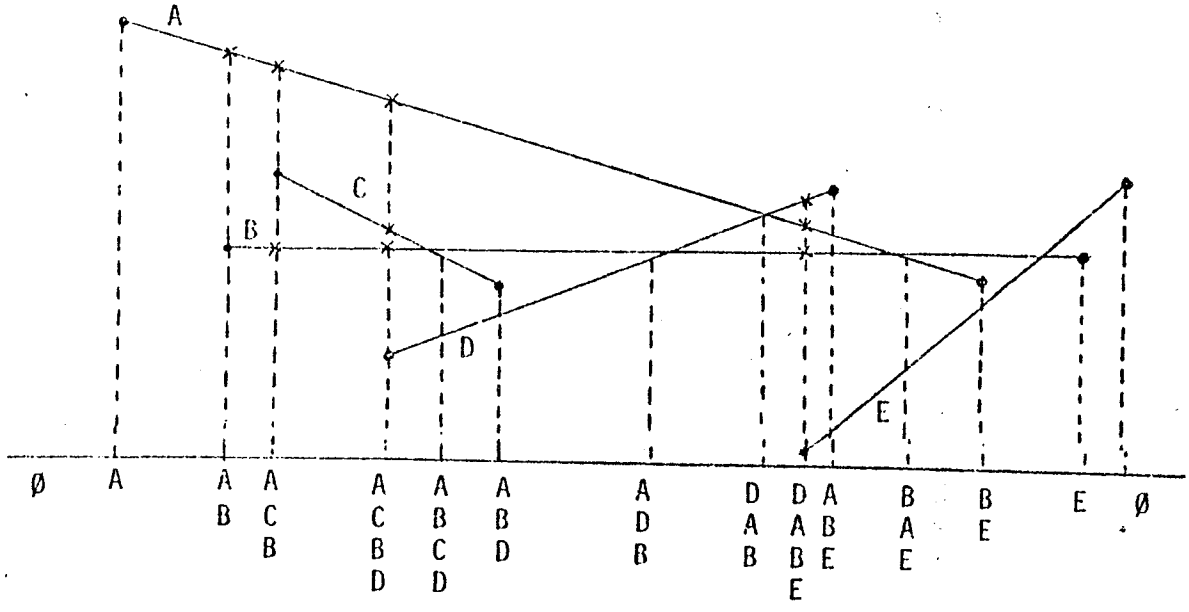


Fig. 3.6. Exemple de traitement (tiré de [Luc80])

Les points d'intersections seront insérés à leur place dans l'ensemble Q des abscisses.

Quand l'abscisse traitée sera celle d'une intersection il faudra echanger les positions des segments concernés dans R et vérifier si après cet échange, les segments nouvellement adjacents se coupent.

Algorithme

L'algorithme est le suivant (d'après [Luc 80])

```
DEBUT
  p = lère abscisse à traiter
  TANTQUE Q non terminé FAIRE
    SI p est une abscisse-début
      ALORS insérer dans R le segment, à sa place dans l'ordre vertical
            SI ce segment coupe celui au-dessus ou au-dessous
              ALORS insérer l'abscisse de l'intersection
                    à sa place dans Q
            FINSI
      SINON SI p est une abscisse-fin
              ALORS SI les segments au dessus et au-dessous
                        se coupent
                        ALORS insérer l'abscisse de l'intersection
                              à sa place dans Q.
                    FINSI
                    enlever le segment de R
              SINON (p est l'abscisse de l'intersection de 2 segments S,T)
                     reporter cette intersection
                     échanger les positions des segments S et T
                      SI le segment T coupe celui au-dessus
                        ALORS insérer l'abscisse de l'intersection
                              à sa place dans Q
                      FINSI
                      SI le segment S coupe celui au-dessous
                        ALORS insérer l'abscisse de l'intersection
                              à sa place dans Q
                      FINSI
            FINSI
    FINSI
    p = abscisse suivante de Q
  FINTANTQUE
```

Conclusion

. Le comportement de cet algorithme est O (nlogn + k logN) qui n'est pas le comportement minimal O (nlogN + k) indiqué par Shamos et Hoey, bien que celui-ci soit atteint lorsque les segments à traiter sont horizontaux ou verticaux. On remarque que lorsque k est proche de N^2, cette méthode est moins bonne que la méthode triviale.

. On remarque l'importance du type de structure de données utilisées, tas et arbre équilibré, qui pourraient être employées pour les méthodes d'analyse de visibilité.

. Cet algorithme pourrait être utilisé afin d'étudier la visibilité d'un ensemble de segments coplanaires et de plus permettrait de traiter la transparence puisqu'on possède à chaque pas de l'algorithme un ordre pour l'ensemble des segments concernés. De plus, il peut être étendu au traitement d'arcs de cercles dans le cas où ceux-ci possèdent une concavité unique.

3.2.2. Comparaison point - Contour C_c^P

3.2.2.1. Méthode par calcul d'intersections

Principe

Le polygone traité peut être convexe ou non.

Cette méthode consiste à considérer une demi-droite issue de x et à dénombrer le nombre d'intersections avec le polygone (figure 3.7.).

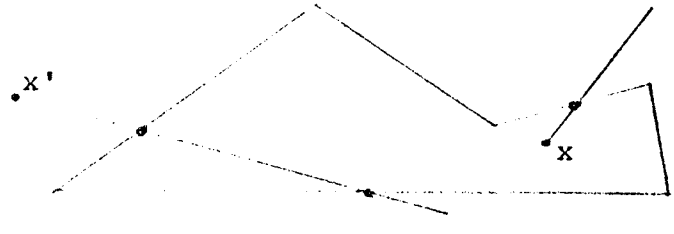


Fig. 3.7. Méthode par dénombrement d'intersections.

Le nombre d'intersections est pair (x') : le point est extérieur au contour.
Le nombre d'intersections est impair (x) : le point est intérieur.
Un point situé sur le contour sera décelé au moment du calcul de l'intersection de la demi-droite avec le segment auquel il appartient.

Cette opération peut donc s'écrire :

$C_c^p = (P, C_q, (C_d^s, C_d^p), S)$

Cq indique un contour quelconque (C_c indiquera par la suite un contour convexe). L'ensemble S est une variable logique, mais si l'indication des cas particuliers est nécessitée par le contexte d'utilisation S pourra prendre quatre valeurs :

. le point est intérieur,

. le point est extérieur,

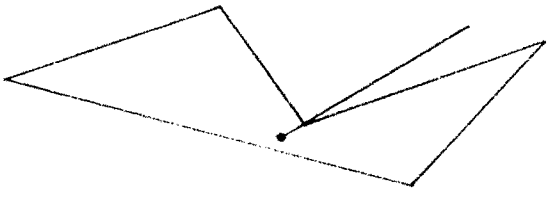
. le point est sur un segment de contour,

. le point est en un sommet du contour.

Cas particuliers

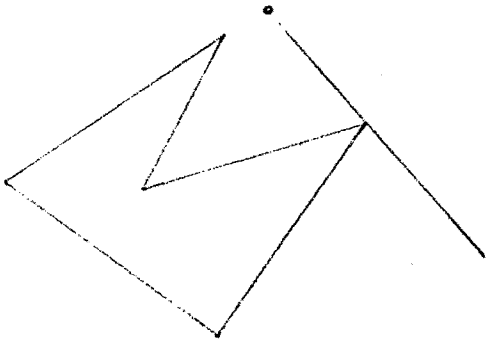
Cette méthode nécessite la détection et le traitement de cas particuliers, qui impliquent un surplus de calcul.

Cas 1 : La demi-droite passe par un sommet du polygone.



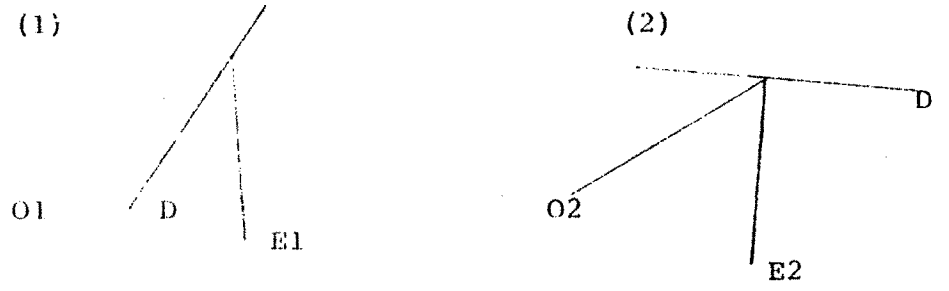
On ne devra considérer qu'un point d'intersection.

Cas 2 :



Dans ce cas, on devra considérer qu'il existe deux points d'intersections.

Ces deux cas peuvent être distingués de la façon suivante :



Dans le cas (1), les points O_1 et E_1 sont situés de part et d'autre de la droite D ; dans le cas (2), les points O_2 et E_2 sont situés d'un même côté par rapport à la droite D.

- la demi-droite supporte une arête du contour.

On ne doit obtenir que deux points d'intersection, ceci implique de parcourir séquentiellement les sommets du contour et mémoriser si, à l'extrémité du segment précédent, on a décelé un point d'intersection. Ici intervient la précision des calculs, qui permettra de décider de l'égalité d'un point des données, et d'un point obtenu par calcul, donc entaché d'erreur.

Algorithme

CØ : méthode par dénombrement d'intersections ;

CØ : nbinter = nombre d'intersections

XP, YP point à tester ; n = nombre d'arêtes du contour

((XØ, YØ), (XE, YE)) arête.

```
nbinter ; = o ;
XØ : = X (premier sommet) ;
YØ : = Y (premier sommet) ;
XE : = X (deux sommet) ;
YE : = Y (deux sommet) ;
Pour I : = 1 jusqu'à n faire
début
   CØ : on compare l'arête (XØ, YØ) (XE, YE) à la demi droite
      X = XP et Y > YP .
   nbinter : nbinter + intersection (XØ, YØ, XE, YE, cas particulier,
             contour)
   si cas particulier alors
   début
      I : = 1 ;
      nbinter : = o ;
      modifier demi droite
    fin
    sinon si contour alors
    début
      CØ : le point est sur le contour ;
      I : = n + 1 ;
    fin
        XØ : = XE ; YØ : = YE ; XE : = X (sommet suivant) ;
        YE : = Y (sommet suivant).
    fin
    si contour alors
    début
        si nbinter reste 2 = o alors
        CØ nombre pair d'intersections ;
        dehors : = vrai ;
```

```
        sinon dehors : = faux ;
    fin ;
fin algorithme.
```

Critique

Les avantages de cette méthode sont :

- il n'y a pas de restriction sur le contour traité,

- la demi-droite issue du point à comparer au contour, peut être choisie parallèle à un des axes du repère, ce qui entraîne une simplification des calculs ;

Les inconvénients de cette méthode sont :

- les cas particuliers peuvent soit être traités comme tels, soit évités en effectuant une rotation d'un angle ε de la demi-droite autour du point,

- un comportement en O(n) n étant le nombre de sommets du contour. En effet, cette méthode applique n fois un calcul qui est celui de l'intersection des deux segments (arête, demi-droite). La comparaison n'est pas nécessairement exhaustive dans un cas seulement, celui où le contour est convexe et lorsqu'on a décelé deux intersections qui ne sont pas un sommet du contour.

- dans le cas d'une implémentation, la précision des calculs devra être au moins égale à celle des données.

- les performances de cette méthode sont directement liées à la réalisation de l'opération de calcul de l'intersection de deux segments.

- du fait même de sa généralité, cette méthode ne tire aucun profit de la convexité ou non-convexité du contour traité.

3.2.2.2. Méthode des angles capables

Principe

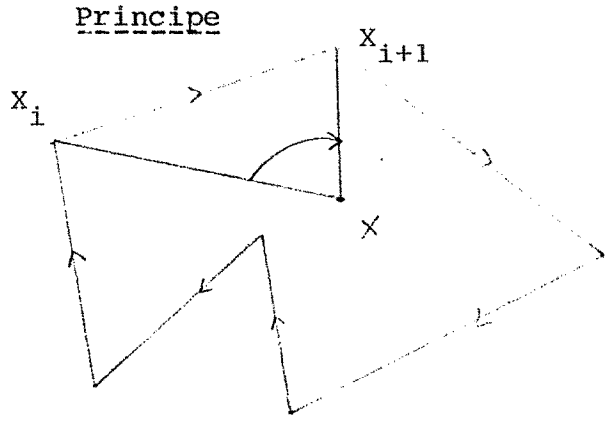


Fig. 3.8. Méthode des angles capables

Le calcul se fait par un parcours séquentiel des sommets du contour. Pour chaque arête, on calcule l'angle $\alpha_i = (\vec{xx}_i, \vec{xx}_{i+1})$

Si la somme des α_i est nulle, le point x est à l'extérieur du contour P. Sinon, la somme des angles est égale à $2k\pi$ et le point est à l'intérieur du contour

Cette méthode de comparaison se décompose suivant la formalisation de la façon suivante :

$= (P, Cq(n), (nK_A, C_S^P), S)$

l'opération K_A étant celle d'un calcul d'angle.

Remarques

Cas des points situés sur le contour P (figure 3.9.

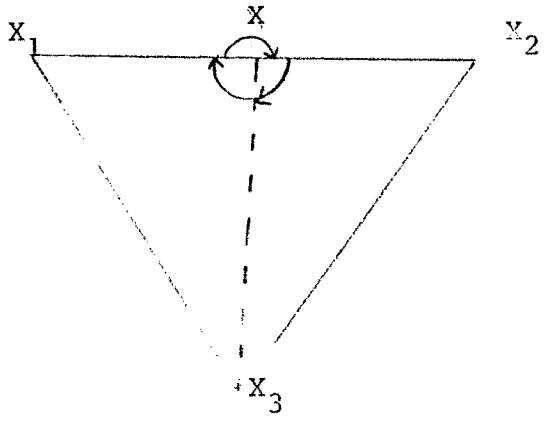


Fig. 3.9. Cas particulier.

La somme des angles α_1, α_2, α_3 sera égale à 2π le point x sera donc considéré comme étant intérieur au contour et aucune information supplémentaire ne sera fournie pour indiquer la position particulière qu'il possède par rapport au contour, information parfois nécessaire pour la suite du traitement dans un algorithme d'élimination de parties cachées.

. Déterminer la position d'un point par rapport à un contour va donc nécessiter un test exhaustif de toutes les arêtes (calcul de α_i, $i \in [1,n]$). On peut toutefois éviter ce test exhaustif dans certains cas. Supposons que le contour C soit convexe :

la fonction $\alpha = f(i) = \sum_{j=1}^{i} \alpha_j$ est une fonction monotone lorsque le point P est à l'intérieur du contour, mais ne l'est pas lorsque le point est extérieur.

Il est donc possible d'affirmer qu'un point est extérieur lorsqu'on décèle un changement de monotonie et d'éviter ainsi le calcul exhaustif de tous les angles.

Algorithme

```
CØ : méthode des angles capables ;
XØ : = X (premier sommet) ; vx1 : = XP - XØ ;
YØ : = Y (premier sommet) ; vy1 : = YP - YØ ;
XE : = X (deux sommets)   ; vx2 : = XP - XE ;
YE : = Y (deux sommets)   ; vy2 : = YP - YE ;
angle : = o ;
Pour I : = 1 jusqu'à n faire
début
    v sin : = vx1 * vy2 - vy1 * vx2 ;
    v cos : = vx1 * vx2 + vy1 * vy2 ;
    angle : = angle + atan (v sin, v cos) ;
```

CØ arc tangente (v/v)

vx1 : = vx2 ; vy1 : = vy2 ;

vx2 : = XP - X (sommet suivant) ;

vy2 : = YP - Y (sommet suivant) ;

fin ;

dehors : = abs (angle) > 1 ;

fin algorithme.

Critique

Cet algorithme possède essentiellement deux avantages :

- le contour P traité est quelconque ;

- la précision des calcul peut être médiocre car la valeur calculée peut prendre deux valeurs qui sont 0 ou 2 kπ à ε près.

Les inconvénients sont :

- le comportement est en O(n), car on étudie toutes les arêtes du contour,

- on utilise des fonctions trigonométriques,

- aucun renseignement n'est fourni sur les points situés sur le contour P.

3.2.2.3. Méthode Dahan Le Tuan (cf. [DaT 77])

Principe

Cet algorithme ne traite que des contours convexes et suppose une orientation a priori de ceux-ci (exemple : lors du parcours du contour, l'intérieur se situe à droite).

L'hypothèse de convexité permet de déterminer si le point est extérieur au contour de la façon suivante (figure 3.10).

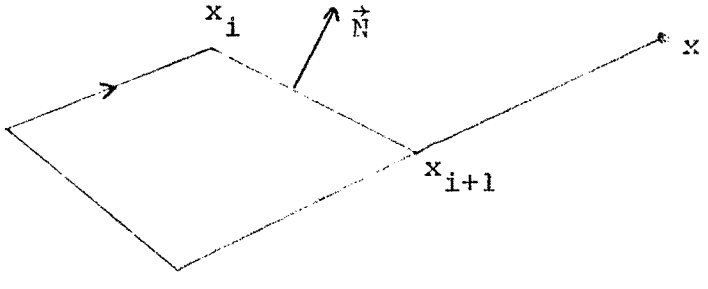


Fig. 3.10. Méthode Dahan Le Tuan.

La condition nécessaire et suffisante pour que x soit extérieur au contour P, est $I(x_i, x_{i+1}) / xx_{i+1}.n<=0$.

CNS déduite du fait qu'un contour convexe se situe entièrement dans un des demi-plan défini par l'une quelconque de ses arêtes

L'orientation qui permet de déterminer rapidement une normale extérieure, peut être rendue inutile en remplaçant le test ci-dessus par celui-ci : soit Q_i l'équation de l'arête $x_i\ x_{i+1}$

Le test devient :

$$(x_i\ \ x_{i+1})/Q_i(x).Q_i(x_{i+2}) <=0$$

Si le produit est nul, le point x est sur la droite de support de $x_i\ x_{i+1}$.

Il faut déterminer si x est sur l'arête ou à l'extérieur

$\overline{xx_i}.\overline{xx_{i+1}} <= 0$ x est sur le contour .

$\overline{xx_i}.\overline{xx_{i+1}} > 0$ x est extérieur

Si cette méthode permet d'éviter une comparaison exhaustive entre point et contour, dans le cas où x est extérieur, elle nécessite par contre l'étude de tous les côtés du contour dans le cas où x est intérieur.

Afin de palier dans une certaine mesure cet inconvénient, les auteurs de cette méthode proposent de découper le contour en triangles.

Pour que x soit intérieur à P, il faut et il suffit qu'il soit intérieur à un triangle.

Cette méthode revient donc à comparer un point à un triangle. Pour ceci, on applique la méthode décrite plus haut. Déterminer qu'un point est intérieur à un triangle, nécessitera 3 applications du test.

Critique

Un polygone à n côtés se décomposera en n-2 triangles. Le nombre total d'arêtes sera : 2n-3.

. un point extérieur nécessitera :

au minimum une évaluation $Q_i(x).Q_i(x')$

au maximum 2n-3 évaluation de ce type.

. un point intérieur au contour P nécessitera :

au minimum 3 évaluations de $Q_i \times Q_i$

au maximum $2n-3$ évaluations.

Le principal avantage de cette méthode est d'éviter un test systématique de toutes les arêtes du contour. Mais il est par contre difficile d'évaluer quel est le comportement de cet algorithme, bien qu'on puisse donner des bornes minimum et maximum du nombre de tests nécessaires. D'autre part, on remarque l'importance de la borne max, qui pénalisera le traitement d'un contour au nombre d'arêtes élevé du fait de la triangulation du contour, ce qui ôte une grande part de l'intérêt de cette méthode, si l'on considère de plus le coût du découpage en triangles, qui est en O(n).

Algorithme

```
CØ méthode de Dahan le Tuan ;
I : = 1 ; fini : = faux ;
tantque  (I< = N) et (∧ fini) faire
début
    produit scalaire : = P.x⃗ i. xi⃗.xi+1 ;
si produit scalaire <o alors
début
     dehors : = vrai
     fini : = vrai ;
fin
```

```
sinon
si produit scalaire = o alors
début
     si P X i. P X i+1 < = o alors frontière : = vrai
                                sinon dehors : = vrai ;
     fini : = vrai
fin
i : = i + 1 ;
fin ;
si ∧ fini dehors : = faux ;
fin Algorithme.
```

3.2.2.4. Méthode de Shamos (cf. [Sha 75])

Cette méthode résoud le problème pour un polygone convexe avec un comportement en O(logn), un prétraitement en O(n) ayant été effectué.

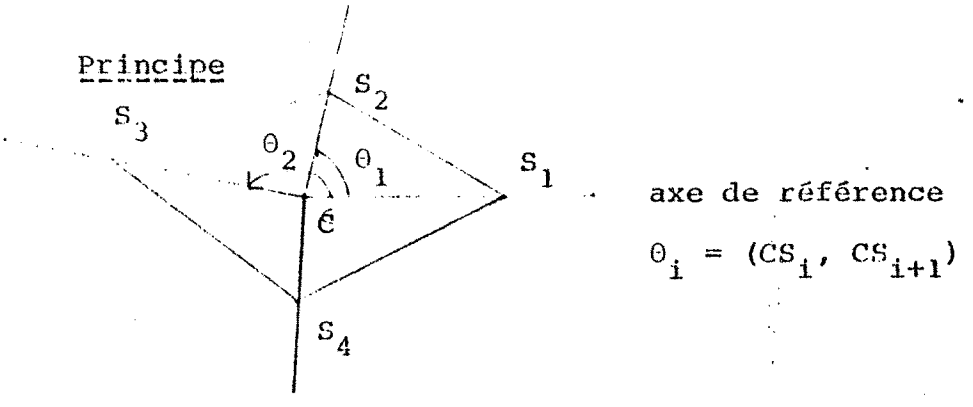


Fig. 3.11. Méthode de Shamos.

Il faut premièrement déterminer un point c intérieur au contour, ce qui est facilement réalisable puisque le contour est supposé convexe.

Les demi-droites joignant c aux sommets du contour partitionnent le plan en n secteurs. c est considéré comme l'origine d'un système de coordonnées polaires. A chaque arête du contour est donc associé un angle (cf.fig.3.11.).Le prétraitement consiste à initialiser une structure de données faisant la correspondance arête-angle. Ceci est fait en O(n) puisque le contour est supposé orienté.

Soit x le point à comparer au contour. On détermine son angle polaire, puis une recherche dichotomique permet de déterminer le secteur auquel il appartient en un temps en O(logn).

Le secteur contenant le point x étant déterminé, il suffit de tester de quel côté de l'arête se situe le point (figure 3.12.).

Fig. 3.12. Secteur défini par une arête.

Le contour étant orienté, il suffit de tester le signe de Q_i.

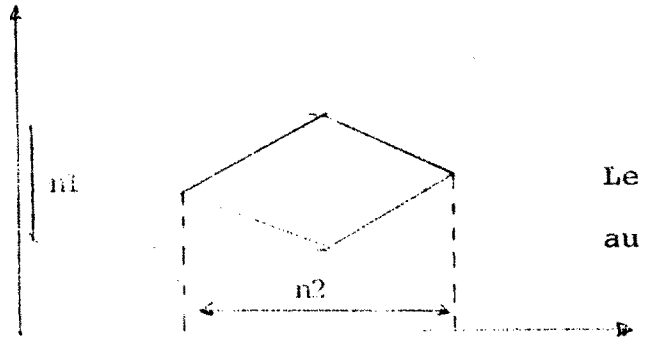
Algorithme : CØ méthode de Shamos :

Prétraitement ;

```
CØ Soit (xr, yr) le point origine intérieur au contour
   Tangle (1 : n) le tableau des angles polaires ordonnées par valeur
   croissante ;
   vx1 : = XR - X(1) ;
   vy1 : = YR - Y(1) ;
   vx2 : = XR - XP ;
   vy2 : = YR - YP ;
   angle : = arc tangente (vx1 * vy2 - vy1 * vx2, vx1 * vx2 + vy1 * vy2) ;
   Recherche dichotomique (angle, tangle, n, indice) ;
   CØ : le point se situe dans le secteur défini par ( (xR, yR),(x (indice));
   si (coea(indice) * xp + coe b (indice) yp > coe c (indice) ) alors
       dehors : = vrai ;
   sinon dehors : = faux ;
```

Critique

Le principal désavantage de cette méthode est le calcul des angles polaires, qui implique l'utilisation de fonctions trigonométriques inverses coûteuses en temps de calcul. Cette solution semble par contre tirer le plus grand profit des hypothèses sur le contour : convexité et orientation, et donne une solution au comportement en O(logn). Cette méthode sera d'autant plus performante que le nombre de points à tester par rapport au contour sera important, le prétraitement n'étant à effectuer qu'une seule fois. Dans l'algorithme de Catmull, cette méthode aura une grande efficacité si la scène est composée de contours convexes, puisqu'un grand nombre de points sont à tester par rapport à un contour (figure 3.13.)



Le nombre de points à tester sera au minimum égal à n1.n2

Fig. 3.13. Application de cette méthode.

3.2.2.5. Méthode par codage ([NeS 73])

Principe

Considérons un petit domaine autour du point à tester. Celui-ci définit huit domaines codés de 0 à 7 (figure 3.14.)

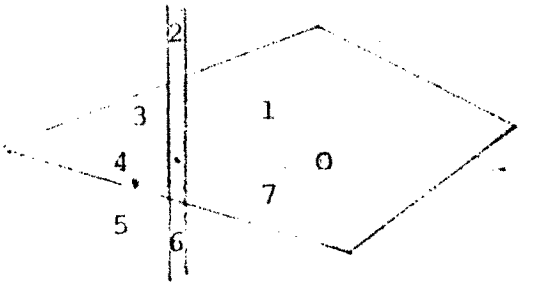


Fig. 3.14. Méthode par codage

A chaque segment est associé un code fonction de la position de ses extrémités par rapport aux domaines :

Soit le segment s = AB Code (S) = Code (B) -·Code (A)

Ce code donne en fait le nombre de frontières traversées par le segment.

Le point à tester sera intérieur au contour si la somme des codes des arêtes est multiple de 8 et non nul, extérieur sinon.

Algorithme

```
CØ n = nombre de sommets ;
  Ø : = premier_sommet ; CØ : = code (Ø) ; Code : = 0 ;
  tant que il y a sommet faire
  début
      E : = sommet_suivant ; CE : = code (E) ;
      CS : = CE - CØ ;
      si abs (cs) > 3 alors
         si CS < o alors CS : = CS + 8
                    sinon CS : = CS - 8
      finsi
      code : = code + CS ;
      CØ : = CE ; sommet_suivant
   fin ;
      si code ¬ = 0 alors point_extérieur
                    sinon point_intérieur.
```

Critique

Cette méthode de programmation simple a pour principal inconvénient de nécessiter le traitement de toutes les arêtes, ce qui conduit à un comportement en O(n), n étant le nombre de sommets. Un intérêt est cependant la simplicité du calcul du code d'un sommet si l'on choisit les frontières des domaines parallèles aux axes du repère.

3.2.2.6. Conclusion

Du tableau récapitulatif (page 183) on peut faire certaines remarques :

- la résolution du problème géométrique "comparaison point-contour" peut tirer grand profit de l'utilisation d'informations intrinsèques sur les figures à traiter (convexité ou non convexité du contour, orientation).

Des trois solutions opérant sur des contours convexes celle de Shamos semble tirer le meilleur profit de cette information.

Des deux solutions qui traitent des contours convexes celle des angles capables semble la plus intéressante. On peut lui reprocher d'utiliser des fonctions trigonométriques mais l'autre méthode utilise l'intersection de segments, opération qui n'est pas d'un coût à négliger. De plus cette méthode doit traiter des cas particuliers, qui pénalisent encore le comportement de cet algorithme.

3.2.3. Algorithmes de découpage

Nous allons étudier plusieurs algorithmes de découpage qui diffèrent par les éléments traités, les éléments et renseignements produits. Les deux premiers algorithmes comparent un contour polygonal et un segment de droite, les 3 dernières méthodes comparent deux contours polygonaux.

3.2.3.1. Algorithme de Sutherland (cf. [NeS 73])

Principe

L'action de base de cet algorithme est de comparer un segment et un rectangle à bords parallèles aux axes afin de déterminer si :

. le segment coupe le rectangle ;

. le segment est à l'intérieur du rectangle ;

. le segment est à l'extérieur.

Méthode	Nature du contour	Cas particuliers	Points sur le contour	Actions élementaires	Comportement	Remarques
Dénombrement d'intersection	quelconque	demi-droite passant par un ou plusieurs sommets	déterminés par C_D^S	C_D^S comparaison droite-segment	O(n)	$(P, {}_SCq(p), (nC_D^S, C_D^P), S)$
Angles capables	quelconque		déterminés par C_S^P	arc tangente (K_A)	O(n)	$(P, Cq(p), (nK_A, C_S^P)\ S)$
Dahan le Tuan	Convexe orienté		déterminés par C_D^P	comparaison droite-point	nombre d'opérations k $k \in [i,n]$	$(P, C_c(n), kC_D^P,$ $k = n$ si point $k \in [i,n]$ si le point est extérieur
Shamos	convexe orienté		déterminés par C_D^P	- recherche dichotomique - arc tangente (n) - comparaison droite point C_D^P	O(log n) après préparation	$(P, C_c(n), kC_D^P, S)$ préparation du contour en O(n) méthode intéressante lorsque le contour est utisé plusieurs fois
Codage	contour quelconque			- tests de coordonnées	O(n)	$(P, C_g(n), O(n), S)$. méthode exhaustive

Le plan est partitionné en 9 régions à l'aide du rectangle (figure 3.15.)

1001	1000	1010
0001	0000	0010
0101	0100	0110

Fig. 3.15. Codage du plan.

Chaque région est identifiée par un code de longueur 4. Un bit étant associé à une droite. L'intérieur a le code nul.

. Cet algorithme utilise deux opérations élémentaires qui sont :

- le codage d'un point par rapport au contour qui dans ce cas particulier d'un rectangle aux côtés parallèles aux axes se fait par test des coordonnées du point avec les abscisses et ordonnées des sommets du rectangle. Ceci nécessite 4 tests élémentaires au maximum (comparaison de réels) mais dans le cas d'un rectangle quelconque nécessiterait l'utilisation des équations des droites de support des côtés du rectangle.

- le calcul du point d'intersection du segment et d'une droite de support d'un côté du rectangle.

Le processus s'arrête lorsque les 2 extrémités ont un code nul ou lorsque le "et" des deux codes n'est pas nul ce qui signifie que le segment est d'un même côté par rapport à une droite de support, et donc à l'extérieur. '

Critique

Cet algorithme présente deux principaux avantages :

- il ne fait au plus que quatre comparaisons segment demi-plan.

Il présente une extension possible :

- le contour peut être généralisé à un contour convexe quelconque. Pour ceci, il faudra utiliser un code de longueur n si n est le nombre d'arêtes du contour.

Les inconvénients de cet algorithme viennent de deux points principaux :

- il effectue un calcul d'intersection coûteux où les cas particuliers (segments verticaux, horizontaux) doivent être distingués. Ce coût augmentera en fonction de la nature du contour (côtés non parallèles aux axes).

- les performances de cet algorithme sont liées à la nature du contour :

  . n le nombre d'arêtes ;

  . arêtes parallèles aux axes ou non, fait qui influe à la fois sur le codage et sur le calcul d'intersection.

Remarque

Nous présentons ici une variante de l'algorithme précédemment décrit, qui évite le calcul d'intersection entre une arête et une droite parallèle aux axes (cf. [NeS 73]) et qui divise le segment étudié en son milieu lorsqu'on ne peut pas décider de sa position par rapport au rectangle.

Principe

Chaque extrémité du segment initial est traitée de la manière suivante qui est :

Pour une extrémité on recherche le point le plus éloigné d'elle qui est encore dans la surface du rectangle. Ceci est déterminé par divisions successives du segment, en son milieu.

```
point procédure RECH (G, D)
d
  si (D dans la surface) alors
  début
      tant que  (G = D) faire
      d  M : = (G + D)/2 ;
         si  (M dans la surface) alors D : = M
         Sinon
      fin ;
      si M dans surface retourner (M) sinon nil
  fin
  sinon Retourner point (D)
f
```

La détermination du segment compris dans la surface se fait par les deux appels :

```
P1 : = Rech (G,D) ;
P2 : = Rech (D,G) ;
```

Critique

Cet algorithme utilise deux opérations élémentaires qui sont :

- déterminer si un point est intérieur à un contour (ici un rectangle) (voir les méthodes décrites auparavant).
- calculer le milieu d'un segment.

si cet algorithme est utilisé pour traiter des figures dans l'espace écran où les coordonnées des points ont des valeurs entières et soit 2^k la taille maximum d'un segment, un appel à la procédure rech nécessitera k passages dans la boucle principale, dans le pire des cas.

Faison une évaluation du nombre d'opérations nécessaires

	Point contour	milieu du segt
les 2 extr sont inté	2	0
1 extr est inter	< k + 1	< k
0 extr int	< 2 * k + 2	<2 * k

k tel que 2^k est la résolution écran.
k dont un ordre de grandeur est 10.

Un intérêt majeur de cette méthode est qu'elle évite le calcul d'intersection entre segment en se ramenant à un calcul de milieu de segment qui peut être réalisé par décalage sur les valeurs des extrémités Ceci signifie que cette méthode est particulièrement adaptée à une réalisation cablée.

D'autre part, cette comparaison segment-contour peut être faite en menant en parallèle le même traitement pour chaque extrémité initiale :

- comportement logarithmique en fonction de la taille du segment ;
- adaptation à une réalisation cablée ;
- parallélisme possible.

3.2.3.2. Méthode de Sutherland et Hodgman (cf. [SuH 74])

Principe

Cet algorithme traite un polygone convexe ou non convexe vis-à-vis d'une droite, un observateur était supposé situé dans un les deux demi-plans définis pour cette droite (voir figure 3.16.).

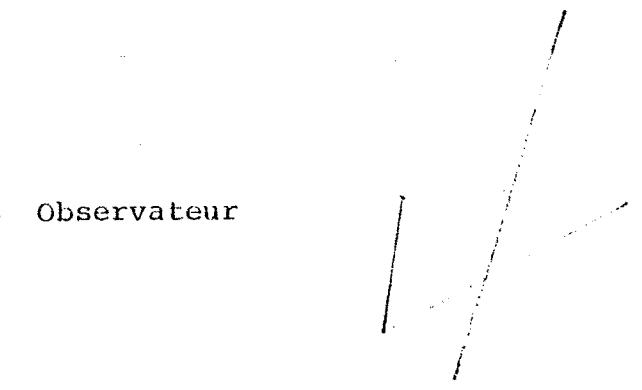


Fig. 3.16. Découpage d'un contour.

Cet algorithme fournit le contour polygonal ou les contours polygonaux compris dans ce demi-plan.

Le polygone est défini par une suite de sommets P1,...Pn

Chaque sommet est considéré comme le sommet terminal d'une arête et zéro, un, ou deux sommets sont produits suivant les 4 configurations possibles entre l'arête et la droite :

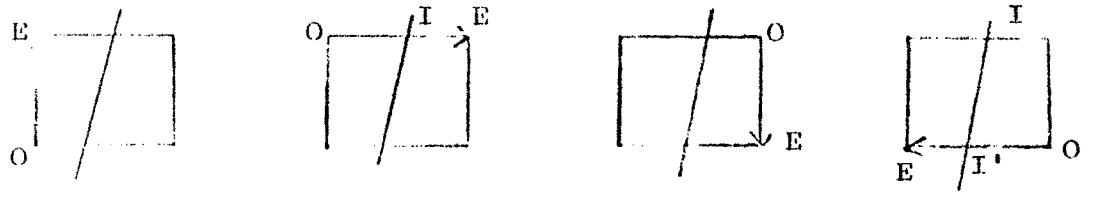


Fig. 3.17. Positions d'un segment par rapport à la droite de découpage.

en (1) le segment OE est produit

(2) le segment OI est produit

(3) aucun élément produit

(4) les segments I I' et I'E sont produits.

Critique

Cet algorithme a été le premier qui utilisait le fait que l'élément traité est un polygone et qui ne considérait pas que le coupage d'un polygone par une droite était une simple extension du coupage de deux segments.

Le comportement de cet algorithme est en O(n) si n est le nombre de sommets du polygone. Une attention particulière doit être portée

aux polygones non convexes qui peuvent produire des polygones dégénérés (figure 3.18.).

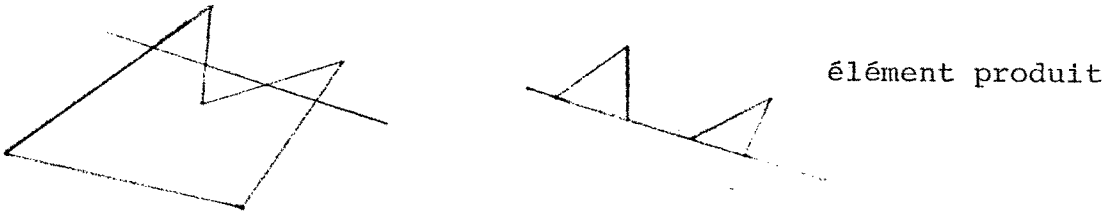


Fig. 3.18. Polygone dégénéré.

Cet algorithme peut être utilisé afin de réaliser le coupage de deux polygones en appliquant successivement le traitement aux différentes arêtes du polygone de coupage.

3.2.3.3. Intersection de deux polygones convexes (cf. [Sha 75])

Les deux polygones traités sont des polygones convexes les contours sont orientés dans un sens arbitraire (sens direct par exemple)

Principe

Soient A et B les deux polygones à traiter

- le polygone B subit un prétraitement identique à celui indiqué en 2.1.4. (figure 3.19.).

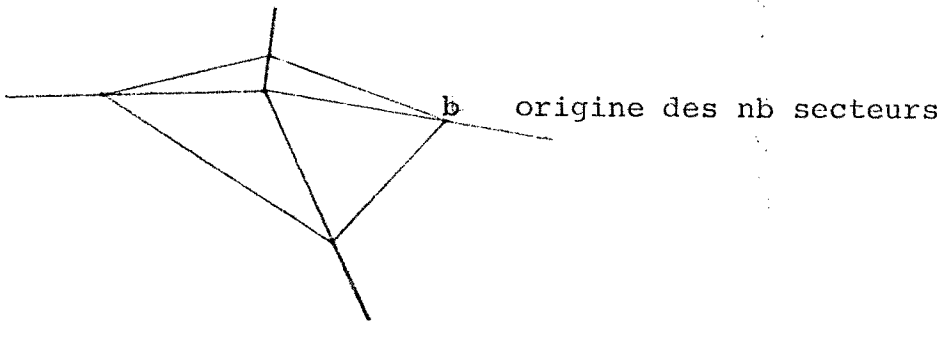


fig. 3.19. Méthode de Shamos.

- si le point b est intérieur à A :

. on détermine dans quel secteur se situe chaque sommet du contour A. Chaque recherche s'effectue en log n_b opérations d'où un coût en O (n_a log n_b) pour cette partie de l'algorithme. Du fait de

l'orientation des contours, chaque secteur ne peut être examiné qu'une fois plus le nombre de sommets qu'il contient d'où un comportement en $O(n_a + n_b)$

. on effectue un parcours du contour en examinant chaque arête $a_i a_{i+1}$

Différents cas se présentent :

. les deux points sont dans le même secteur :

- ils sont intérieurs ou extérieurs il n'y a pas d'intersection ;

- il sont de part et d'autre du contour, il y a une intersection

. les deux points sont extérieurs au polygone mais appartiennent à des secteurs différents. Il y a o ou ∿ intersections. Il faut tester les arêtes des secteurs concernés et l'arête $a_i\ a_{i+1}$ (figure 3.20a)

- un point est intérieur, l'autre extérieur mais dans des secteurs différents. Il y a une intersection de $a_i\ a_{i+1}$ et de l'arête du secteur où le point est extérieur (figure 3.20.b).

- le polygone d'intersection si il existe est obtenu en parcourant alternativement A et B. Le changement se faisant lors du passage sur un point d'intersection.

Fig. 3.20. Recherche des points d'intersection.

- si le point b est extérieur à A

en b le contour A sous-tend en angle x, on décompose A en deux suites d'arêtes qui seront traitées par la méthode citée ci-dessus.

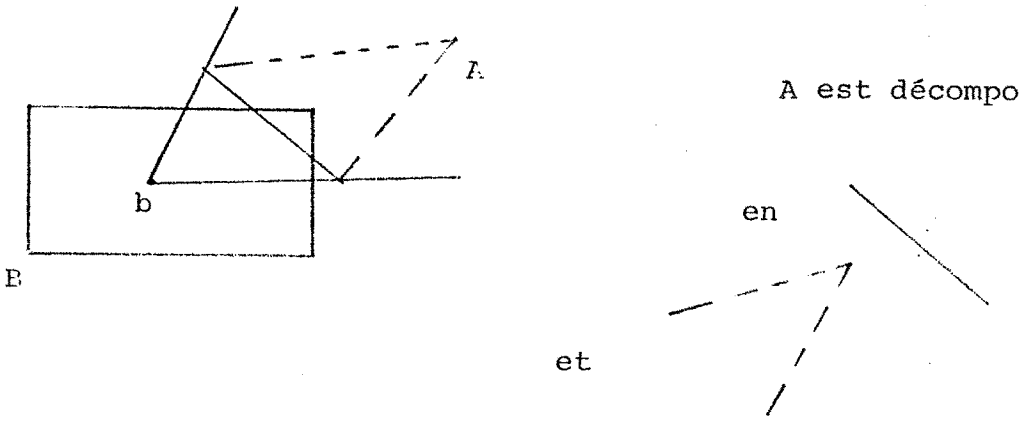


Fig. 3.21. b extérieur à A.

Critique

Le principal avantage de cet algorithme est son comportement linéaire en fonction du nombre d'arêtes des contours alors que la solution triviale qui consiste à comparer tous les couples d'arêtes est en $O(n_a + n_b)$.

L'opération élémentaire utilisée est le calcul d'intersection de segments pour lequel quelques cas particuliers doivent être traités lorsqu'un sommet d'un contour se trouve sur une arête ou en un sommet de l'autre contour (figure 3.22.).

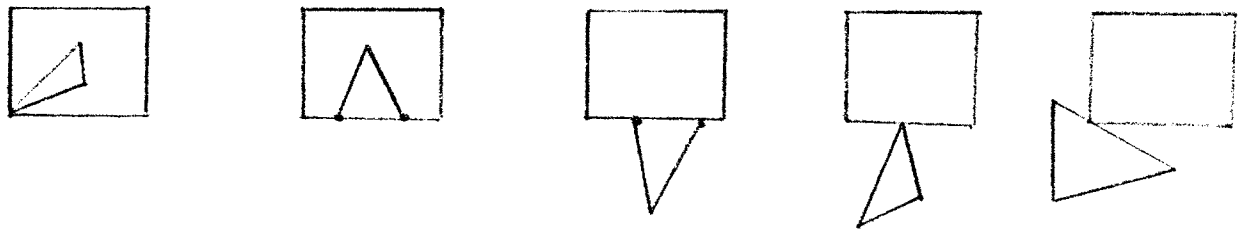


Fig. 3.22. Cas particuliers.

Les cas d'intersections seront décelés du fait qu'un polygone a un sommet intérieur à l'autre contour. Ici interviendra la précision des calculs qui permettra de dire si un point est sur une arête ou si deux points coincident.

3.2.3.4. Découpage de faces : l'algorithme de Atherton et Weiler

Présentation du problème

Cette opération géométrique se décrit suivant la formalisation présentée auparavant par C_p^F = (F1, F2, Op, S) où :

- F1 et F2 sont deux faces planes décrites par un contour extérieur quelconque orienté (matière à droite lors du parcours) et des contours intérieurs orientés en sens inverse qui définissent des trous.

- Op est l'ensemble des opérations géométriques de comparaisons qui sont utilisées.

- S est la liste des éléments produits :

Soit F1 la face de coupage et F2 la face découpée : S sera composé de deux listes : la liste des faces provenant de F2 et intérieures à F1 et la liste des faces provenant de F2 extérieures à F1.

Les faces produites ont les mêmes caractéristiques et la même représentation que les faces traitées.

Description de l'algorithme

Chaque contour est représenté par deux listes circulaires qui correspondent aux deux sens de parcours possible (figure 3.23.).

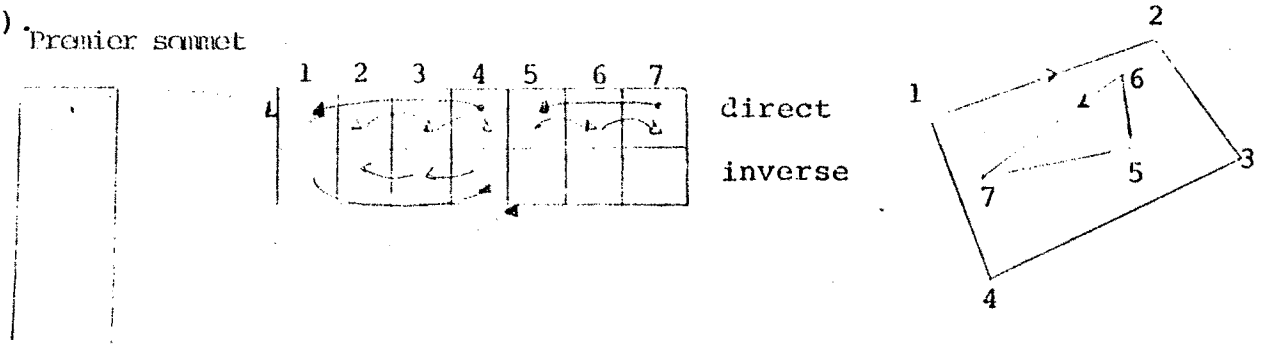


Fig. 3.23. Représentation des contours.

L'algorithme se décompose essentiellement en deux étapes, la recherche d'intersection entre contours et la détermination des éléments de sortie.

Les contours sont quelconques (convexes ou non convexes).

La méthode de recherche d'intersection qui est utilisée sera une comparaison arête par arête des deux contours. Cette solution est optimale dans le cas de deux contours non convexes car le nombre de points d'intersection peut être égal au produit du nombre d'arêtes (figure 3.24.).

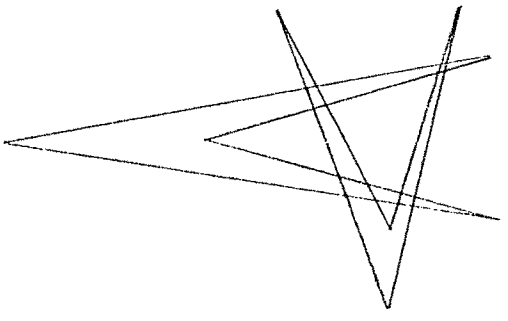


Fig. 3.24. Nombre maximum d'intersections

L'opération élémentaire utilisée sera donc la comparaison arête-arête (C_s^s) qui indiquera s'il y a intersection et si oui les coordonnées de ce point.

Les points d'intersection sont classés en deux types par rapport à l'arête appartenant à la face de découpage.

1er type : on rentre dans la face de découpage (figure 3.25.a)

2e type : on sort de la face de découpage (figure 3.25.b).

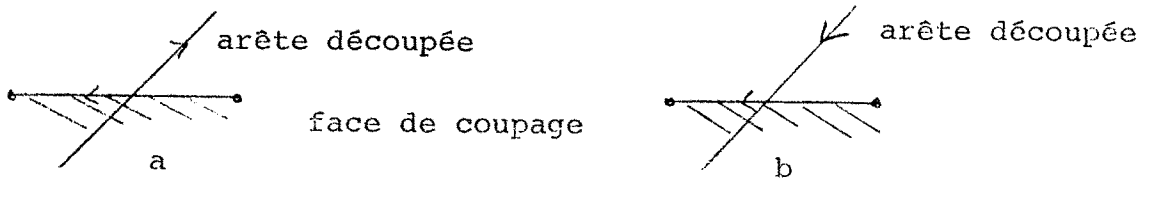


Fig. 3.25. Type des points d'intersection.

On remarque que sur chaque contour les types des points d'intersection sont alternés. Il s'agit, lors du traitement de deux contours de déterminer le type du premier point d'intersection décelé. Ceci est réalisé en testant le signe d'un produit vectoriel.

Si deux contours se coupent, les points d'intersection sont insérés dans chacun d'eux et des pointeurs permettent à partir de chaque point d'intersection, d'accéder aux points voisins sur chaque contour. Si une arête comporte plusieurs points d'intersection il est alors nécessaire de les ordonner afin de respecter l'orientation initiale du contour.

Si aucun point d'intersection n'a été décelé, entre deux contours extérieurs, il est alors nécessaire de comparer les contours extérieurs de chaque face afin de déterminer si :

. les deux faces sont disjointes ;

. une face contient l'autre.

Pour cela, on utilise une opération de comparaison point-contour (C_p^C) qui est appliquée 1 ou 2 fois suivant le résultat positif ou négatif du premier test.

A. La face de découpage contient la face découpée

. les faces ne contiennent aucun trou

la face découpée est intérieure à la face de découpage

. la face de découpage contient un trou englobant la face découpée : celle-ci est extérieure à la face de découpage (fig. 3.26 a)

. sinon, on détermine la face intérieure et les faces extérieures portions de la face découpée (figure 3.26.b).

Faces intérieures :

Soit C_{ext}^C le contour extérieur de la face découpée, et C_{tr_i} les trous de la face de découpage, le contour extérieur de cette face sera $(C_{ext} - \overset{U}{i} (C_{ext} \text{ A } C_{tri})$.

On verra plus loin comment obtenir ce type de contour Les trous de cette face seront l'union des trous de la face découpée et de la face de découpage qui sont en conflit. (figure 3.26.c).

Faces extérieures à la face de découpage

Elles proviennent de l'intersection d'un trou de la face de découpage avec le contour extérieur ou un trou de la face découpée. (figure 3.26.d).

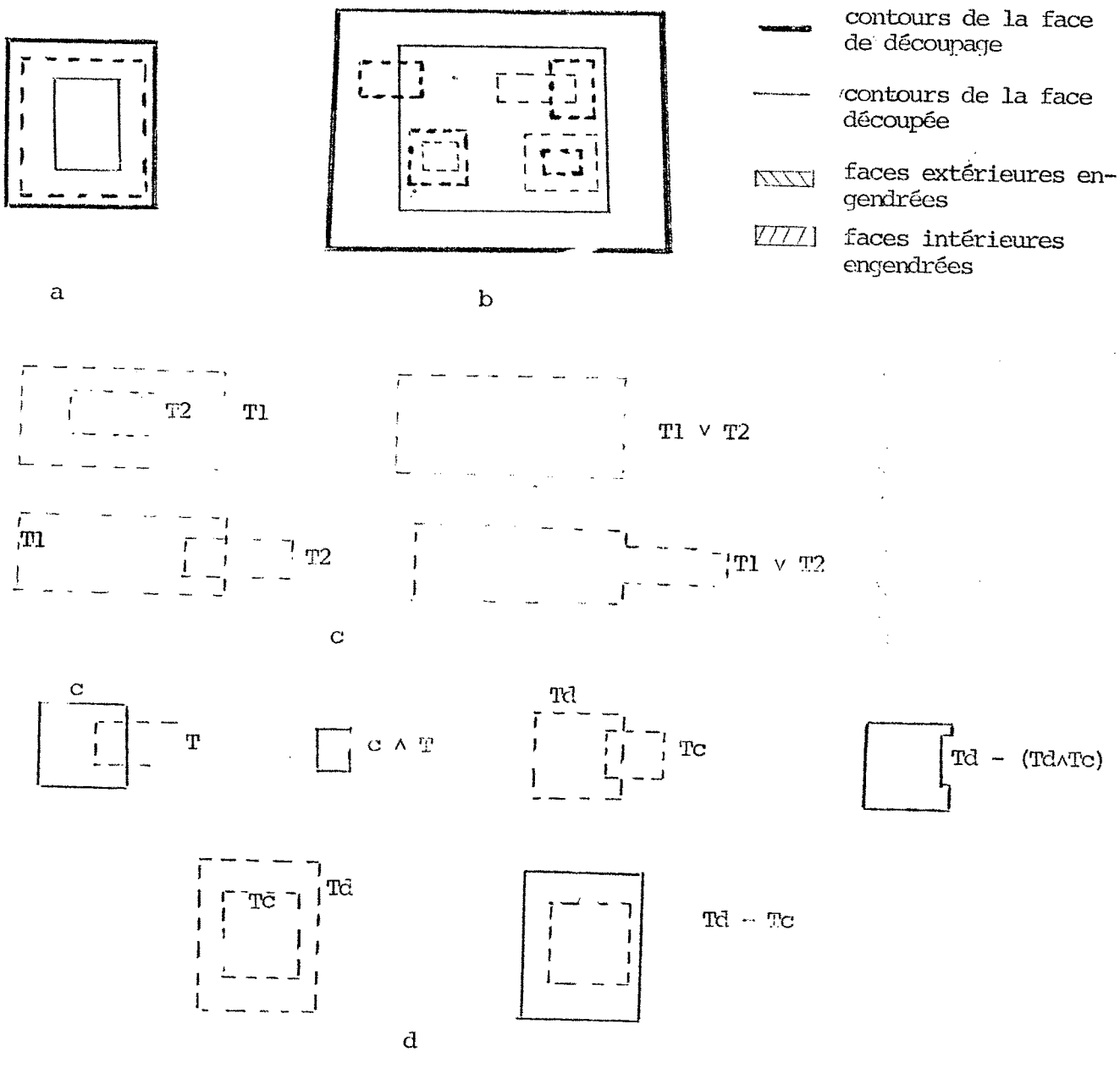


Fig. 3.26. Détermination des éléments de sortie.

B. La face de découpage est contenue par la face découpée

On produit (figure 3.27)

- une ou plusieurs faces intérieures :

  son contour extérieur est celui de la face découpée.

  les contours trous produits sont :

  . les trous de la face découpée si un de ces trous contient la face de découpage.

Sinon, l'union du contour extérieur de la face de découpage et des trous qui l'intersectent.

Les faces extérieures s'obtiennent de façon identique à celle du paragraphe précédent.

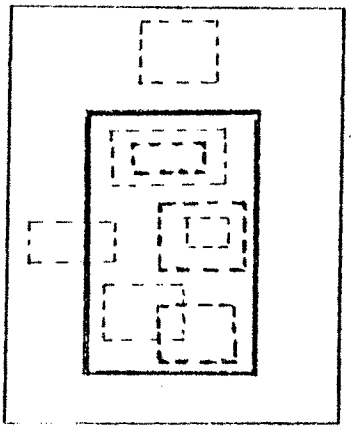


Fig. 3.27. Autre configuration.

Si les contours extérieurs se coupent, le traitement est différent (figure 3.28.).

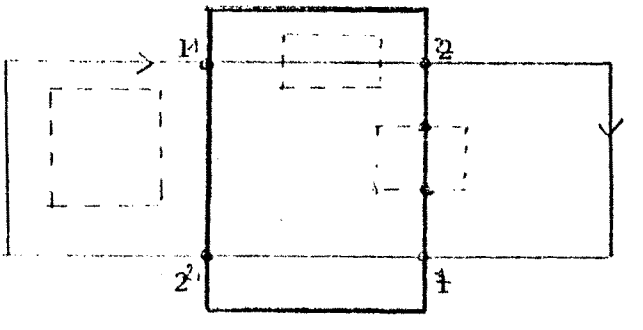


Fig. 3.28. Intersection des contours extérieurs.

Faces provenant de la face découpée et extérieures à la face de découpage.

. contours extérieurs : on obtient tous les contours extérieurs en faisant un parcours à partir de tout point d'intersection de type 2 qui appartient aux contours extérieurs des deux faces en jeu

Règles de parcours :

- parcourir le contour découpé à partir d'un point d'intersection de type 2 dans le sens direct.

- si l'on rencontre un point d'intersection parcourir l'autre contour (le contour de découpage est parcouru en sens inverse)

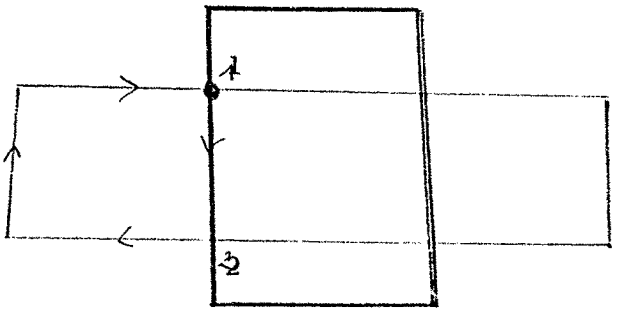


Fig. 3.29. Contours extérieurs.

. on détermine les trous de la face découpée qui sont contenus dans les faces produites.

Faces intérieures

. contour extérieur :

Partir d'un point de type 1 en suivant le contour découpé en sens direct.

En un point d'intersection parcourir l'autre contour en sens direct.

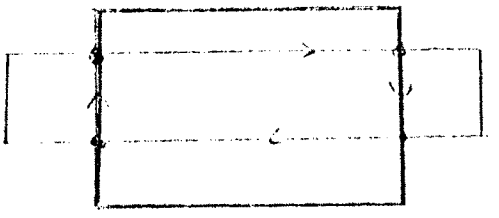


Fig. 3.30. Face intérieure.

. trous : ils sont fournis par l'union des trous de deux faces contenus dans ce contour extérieur.

Evaluation

. Soit F1 décrite par n1 contours et F2 par n2, cet algorithme nécessite de connaitre les positions relatives de chaque contour. On applique donc une comparaison C_c^c au nombre de n1 * n2 fois

n1.n2 $T(C_c^c)$, et lorsqu'aucune intersection n'est trouvée, on emploie les tests C_c^p au plus 2 fois par couple étudié. Les contours traités sont quelconques on effectue un test exhaustif de toutes les arêtes.

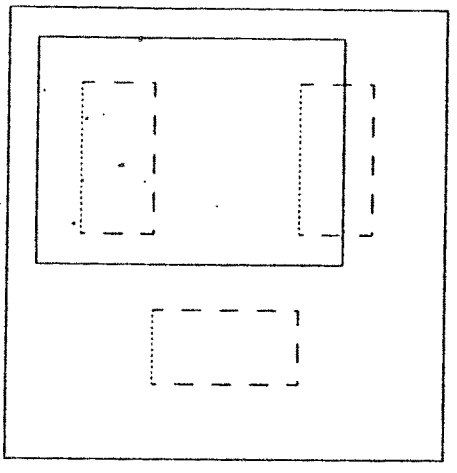
Soit C_1, a_1 arêtes, C_2, a_2 arêtes, le temps $T(C_{C1}^{C2})$ sera proportionnel à $a_1 \times a_2$.

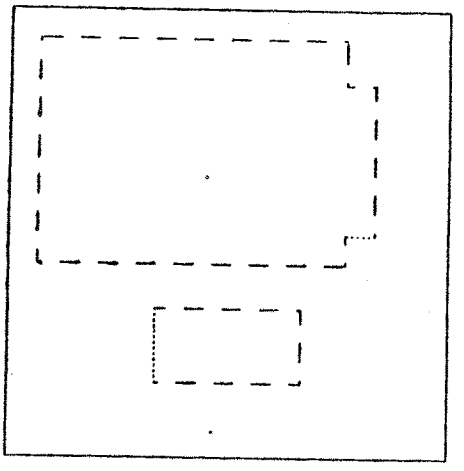
L'autre opération utilisée est celle de détermination de contour. Le nombre des contours produits est égal au plus au nombre d'intersections et pour un contour l'opération est proportionnelle à sa longueur.

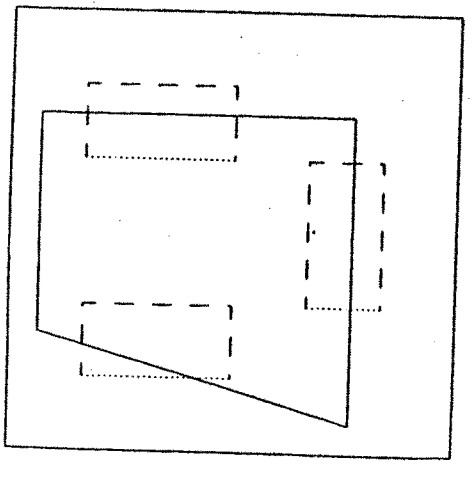
Cet algorithme emploie essentiellement deux opérations C_c^c et C_c^p. L'étude des différentes solutions à ces opérations a montré que des méthodes performantes peuvent être utilisées lorsque les contours traités sont convexes.

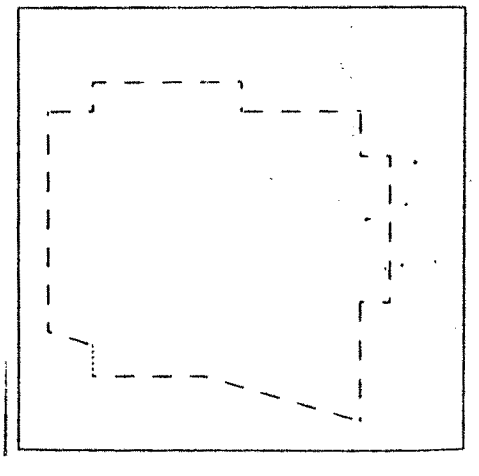
L'opération C_c^c a un comportement proportionnel à la somme du nombre d'arêtes et non à leur produit (cf. 3.23.).

L'opération C_c^p a un comportement proportionnel au logarithme du nombre d'arêtes et non à ce nombre (cf. 3.22.).









S'il est intéressant d'utiliser de telles informations sur les contours traités il le serait de les conserver pour les éléments qui sont produits, car dans certains contextes d'utilisation les éléments produits auparavant pourraient être à leur tour traités par cette opération de découpage. Pour cela, on fera quelques remarques simples qui ne recouvrent pas tous les cas qui peuvent être rencontrés :

. les deux contours sont convexes :

- l'intersection est convexe ;

- les contours extérieurs si ils ne contiennent que deux points d'intersection appartenant à la même arête du contour de découpage sont convexes. (figure 3.31.).

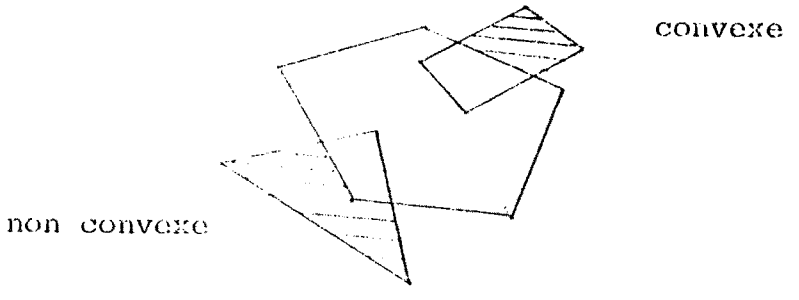
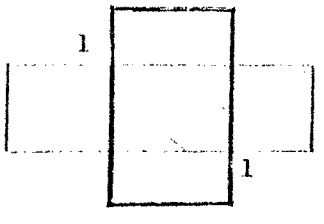


Fig. 3.31. Information attachée aux éléments de sortie.

. Quels que soient les contours traités, compter le nombre de sommets, s'il est égal à 3, le contour produit est bien sûr convexe.

Si chaque point d'intersection de type 1 (2) est pris comme point de départ pour l'obtention d'un contour intérieur (extérieur) certains contours seront produits plusieurs fois. Il est nécessaire de ne pas utiliser les points d'intersection de type 1 (2) rencontrés dans un contour intérieur (extérieur) produit comme nouveau point de départ (figure 3.32.).



le contour intérieur serait produit deux fois

Fig. 3.32. Marquage des intersections.

Certains cas particuliers demandent à être traités avec attention :

un sommet d'un contour appartient à une arête de l'autre contour ou deux arêtes ont une portion commune.

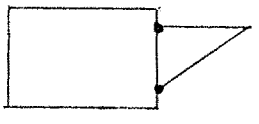
Les problème rencontrés sont de deux types :

. quels points d'intersection faut-il retenir ?

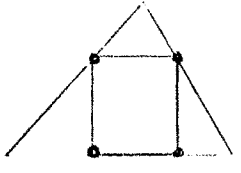
. comment déterminer le type des intersection singulières restantes ?

. validité d'un point d'intersection.

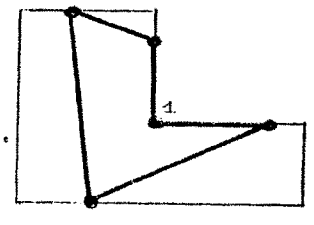
Un point d'intersection singulier sera retenu si au moins une des arêtes qui le définit possède des points intérieurs à un des deux contours en jeu (figure 3.33.).



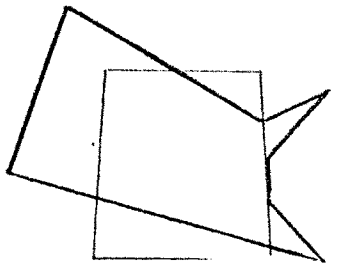
Les deux points sont rejetés



Les 4 points sont retenus



Seul le point 1 est rejeté



Auncun point d'intersection singulier n'est rejeté

Fig. 3.33. Validité d'un point d'intersection.

Ce traitement permet d'éviter la production de contours dégénérés tels que par exemple dans le cas ci-dessus.

- Le type est déterminé si aucun point d'intersection non singulier n'a été obtenu auparavant en prenant en compte les arêtes qui ont permi de ne pas rejeter ce point.

Conclusion

Le principal inconvénient de cette méthode vient de la généralité des faces traitées qui va pénaliser les performances de cet algorithme. On peut toutefois prendre en compte des informations sur les contours (convexité) afin d'utiliser des méthodes adaptées. Un autre inconvénient est la lourdeur du traitement des cas particuliers.Par contre l'avantage essentiel est l'identitée entre les formats des éléments d'entrée et de sortie dont on verra un exemple d'application dans la réalisation de l'algorithme de visibilité proposé par Atherton et Weiler.

3.2.3.5. Conclusion

Cette étude a permis d'analyser des solutions différentes par la nature des contours traités.

La méthode de Shamos tire profit du fait de traiter seulement des contours convexes.

On a mis en valeur la difficulté du traitement des cas particuliers qui, si du point de vue de l'analyse sont facilement résolus, obligent par contre à un surcroit de programmation important.

Méthode	Elements en entrée	Elément(s) en sortie	Operations élémentaires utilisées	Comportement	Remarques
Comparaison segment contour par calcul de milieu de segment	le contour est un rectangle	le segment ds la surface délimitée par le contour	- comparaison point-contour - calcul du milieu d'un segment.	O(log(taille segment)	- adapté cablage - parallèlisme possible - généralisation à un contour convexe quelconque.
Comparaison segment contour Sutherland Sproull	le contour est un rectangle à bord parallèles aux axes	la portion de segment qui coupe le contour	- codage d'un point par rapport à un contour - calcul d'intersection segment droite	O(n) n'étant le nombre d'arêtes du contour	- importance du contour sur le comportement de l'algorithme - généralisation à un contour convexe quelconque
Intersection de deux polygones CHAMOS	- polygones convexes - contour orienté	polygone d'intersection	- calcul d'intersection - fonction trigométriques	O(n) n'étant le nombre d'arêtes des polygones	- intérêt du comportement dû aux contraintes sur les contours traités ($C_c(n)$, $C_c(P)$, (K_A, dicho, C_s^s) ($C_q{}^1$..., $C_q{}^q$))
Intersection de deux polygones Atherton & Weiler)	- polygones quelconques (avec trou) (contour orientés)	- polygones intérieurs et extérieurs -contours orientés)	- intersection de segments	$(\sum n_j^k).(\sum n_i^{k'})$ n_j^k j^e arête du contour k	- complexité du traitement des cas particuliers - formats d'entrée et de sortie identiques - (P(n), P(p), (n x p C_s^s, n.p.P_A) (P_1,...,$P_{n.p}$))
Sutherland Hodgman	- Polygone quelconque	Polygone contenu dans le demi plan où est situé l'observateur	- intersection d'un segment & d'une droite -position d'un point par rapport à 1 droite	O(n)	-formats d'entrée et de sortie identiques - (Cq(n), D, Cn x C_d^p, knC_D^s) (Cq1,..., C_q^p

Le traitement des contours non convexes à un comportement fonction du produit du nombre d'arêtes ce qui est le comportement optimal car le nombre de points d'intersections peut, dans ce cas, être égal au produit du nombre d'arêtes.

3.2.4. Conclusion

Cette étude de différentes solutions à quelques problèmes géométriques a mis en valeur plusieurs points, soulignés par la formalisation qui a été présentée en introduction.

- l'importance de différentes opérations élémentaires (C_D^P, C_S^S, C_D^S) tant du fait de leur utilisation dans divers algorithmes que de leur fréquence d'emploi dans un même algorithme. La réalisation de ces opérations devra attirer toute l'attention de celui qui est confronté à ces problèmes géométriques.

- l'apport d'informations intrinsèques sur les contours (convexité, orientation) qui permet d'utiliser des méthodes plus performantes que des méthodes générales.

- pour une même opération géométrique, cette étude a mis en valeur la supériorité de certaines solutions. On pourra d'autre part faire le choix d'une methode en fonction des éléments que l'on a à traiter et des éléments que l'on désire obtenir.

3.3. Les tris

Nous nous interesserons ici seulement aux algorithmes de tris interne bien que cette hypothèse ne puisse recouvrir tous les cas d'application. Cette étude mettra en valeur différents critères de choix d'un algorithme en fonction d'un contexte d'utilisation.

3.3.1. Exemples

La réalisation de divers algorithmes d'étude de visibilité nous a confronté au choix d'algorithmesde tri. Deux algorithme distincts ont été

choisis :

- le tri rapide,
- le tri bulle.

Le premier est utilisé lorsqu'aucun renseignement n'est fourni à priori sur les données à trier : réalisation du tri initial Td dans l'algorithme de Warnock, Atherton et Newell.

Le second est employé lorsqu'au contraire, on peut considérer que les données sont "presque triées" : réalisation du tri Ti dans l'algorithme de Watkins. Les quelques exemples montrent l'utilisation d'un critère particulier, renseignement sur l'état des données, pour le choix d'un algorithme de tri.

3.3.2. Quelques critères de choix

Ce paragraphe n'a pas pour but l'étude des algorithmes de tri, mais met en valeur certains critères qui doivent être pris en compte lors de la réalisation d'un algorithme de tri.

Nous avons retenu quatre algorithmes afin de faire apparaître différents critères :

. l'algorithme de tri rapide :

Son comportement minimal est en O(nlogn), son comportement maximal et moyen est en O(n2). Le comportement maximal est atteint lorsque les données sont triées ou en ordre inverse. Le choix du pivot influe aussi sur le comportement de l'algorithme ([MeB 78])

Il utilise un espace mémoire de travail dont la taille suit un comportement en O(logn).

. l'algorithme de tri bulle [MeB 78]

Son comportement minimal, atteint lorsque les données sont triées est en O(n), son comportement maximal est en O(n2).

. l'algorithme de tri arbre [MeB 78]

C'est un des algorithmes au comportement le plus sûr puisque son comportement moyen et maximal est en O(nlogn).

. l'algorithme de Shell

Son comportement minimal est en O(n) et maximal en O(n3/2) Utilisant l'accès direct, il est déconseillé de l'employer pour trier des listes ou lorsque la gestion mémoire se fait par un système de pagination.

3.3.3. Conclusion

Ces quelques exemples ont mis en valeur les critères suivants :

- comportement moyen, maximal, minimal.
- comportement lorsque les données sont triées
- espace mémoire utilisé,
- type de la structure de données utilisée
- système d'exploitation,

D'autres critères peuvent entrer en jeu qui sont :

- le domaine de variation des clés à trier ;
- possibilité d'avoir des opérateurs cablés ;
- stabilité du tri ;
- implantation en parallèle qui permet non plus de faire une comparaison espace-temps, mais un compromis espace, temps, processeur ([Batcher], [Pre 79])

On citera bien sûr [Knu 73] et l'article [Mar 71] où un tableau de choix d'algorithme est présenté, qui ne se limite pas aux tris internes.

```
Algorithme de tri bulle
/* tri d'une liste linéaire avec double chainage */
Change ← vrai
tantque change faire
    change ← faux ;
    seg ← tête_de_liste ;
    tant que seg ¬= nil faire
       i ← suivant (seg) ;
       si i = nil allera fin ;
       si clé (seg) > clé (i) alors
           change ← vrai ;
            k ← précédent (seg) ;
           si  k ¬= nil alors suivant (le) ← i
           fsi
           précédent (i) ← k
           précédent (seg) ← i ;
           k ← suivant (i) ;
           si  k ¬= nul alors précédent (k)  ← seg
           fsi
           suivant (seg) ← k ;
           suivant (i)   ← seg ;
           si tête_de_liste = seg alors tête_de_liste ← i
           fsi
       sinon
           seg ← suivant (seg)
       fsi
   fin faire
   fin :
fin faire
```

Algorithme de Tri rapide

Tri d'une liste linéaire

```
s ← 1
Pilel (s) ← 1
Piler (s) ← n
nontrie   ← vrai
tantque nontrie faire
    l ← pilel (s)
    r ← piler (s)
    s ← s - 1
    on sépare ← vrai
    tantque on separe faire
       i ← l ; j ← r ; x ← clé ((l + r/2)
       change ← vrai
       tantque change faire
           tantque clé (i) < x   faire x ←  x + 1
           tantque clé (j) > x   faire j ←  j - 1
           si (i < j) alors
              échanger (clé (i), clé (j))
              échanger (lien (i), lien (j))
              i ← i + 1 ; j ←  j - 1.
              Finsi
              change ← (i < = j)
           finfaire
           si (j - l)   (r - i) alors
              si i < r alors
                 s ← s + 1 ;
                 pilel (s) ← i ;
                 piler (s) ← r
```

```
            finsi
            sinon
                si  l < j alors
                    s ← s + 1
                    pilel (s) ← l ;
                    piler (s) ← j ;
                finsi
                l ← i
            finsi
        on sépare ← (l < r)
    fin faire
        nontrie ← (s ¬1 = 0)
fin faire.
```

3.4. Conclusion

Cette étude des fonctions de transition et des tris Td et Ti a mis en valeur différentes caractéristiques de la scène qui influent sur leurs performances et sur le choix d'une méthode adaptée.
Parmi ces caractéristiques, nous citerons :

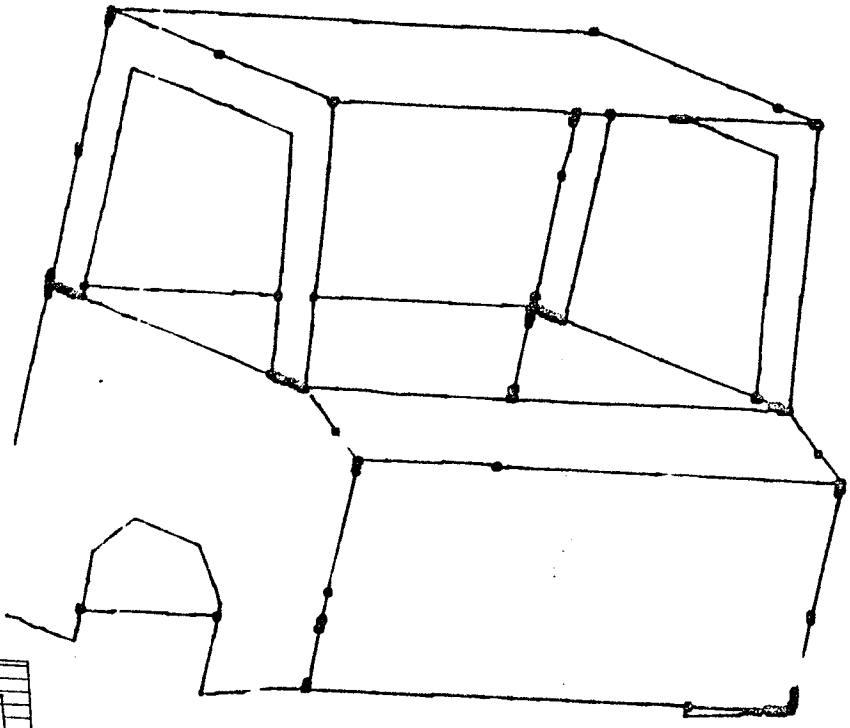
. la nature convexe ou non convexe du contour. On possède des méthodes plus performantes lorsque lors d'une comparaison point-contour ou contour-contour, ceux-ci sont convexes. Dans des algorithmes de visibilité tel celui de Catmull, il sera important de tenir compte de la nature de ceux-ci.

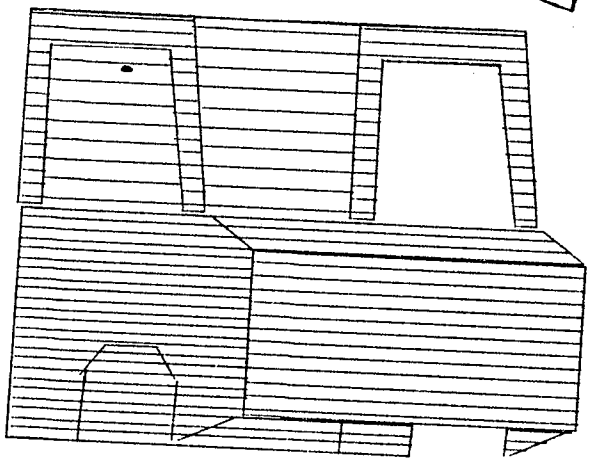
. le nombre de sommets du contour qui en général influe de façon linéaire sur le comportement de ces algorithmes. Il sera peut-être intéressant de trianguler les faces, ce qui de plus permettra de ne traiter que des contours convexes.

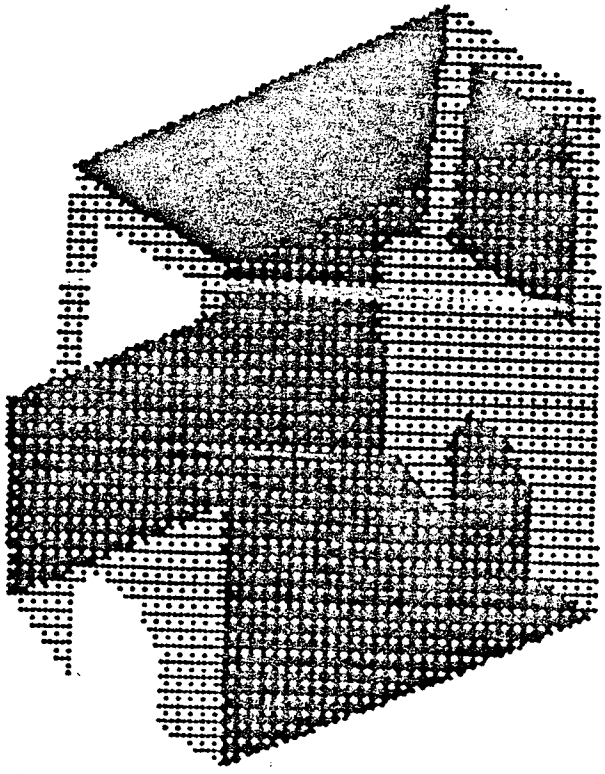
. le traitement de cas particuliers, lorsque deux contours sont adjacents. Ceci montre l'intérêt pour un objet de posséder une matrice d'adjacence pour ses faces.

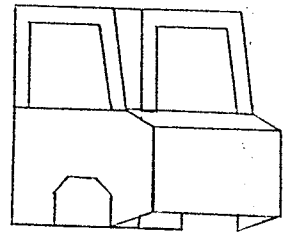
. l'état "presque trié" des données ce qui peut, par exemple, lors d'une élimination de parties cachées dynamique conduire à modifier le choix d'un tri initial Td et de prendre un tri bulle qui sera mieux adapté que le tri rapide.

C O N C L U S I O N









CONCLUSION

Au stade de réalisation actuel nous possédons trois algorithmes d'étude de visibilité opérationnels :

- l'algorithme de Warnock a été abandonné du fait de sa pauvreté du point de vue temps d'exécution et qualité du dessin
- les algorithmes de Watkins, d'Atherton et Weiler, et de Newell, Newell et Sancha nous permettent d'envisager un grand nombre d'applications et ceci pour deux raisons :
- les modèles de description peuvent être composés de faces polygonales quelconques pouvant se pénétrer
- les modèles de visibilité permettent d'obtenir des dessins au trait de grande précision et des images, ainsi que de rendre l'effet de transparence. Les limitations actuelles sur la qualité des réalisations sont en fait dues à la pauvreté des possibilités offertes par le matériel.

D'autre part nous avons mis en évidence, lors de cette étude différents points qui sont :

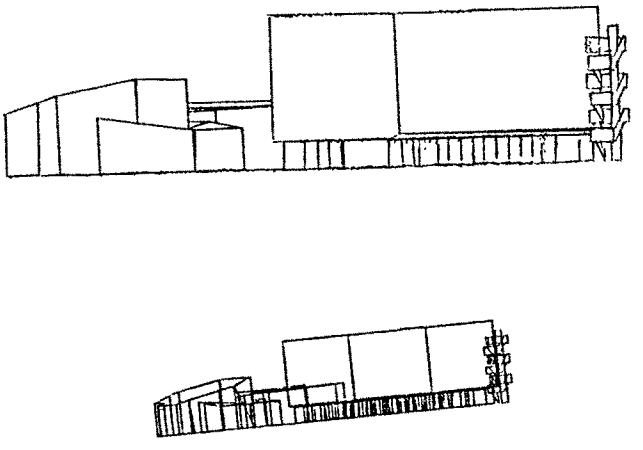
- l'intérêt d'acquérir des informations sur la scène au cours de la phase de modélisation. Ces informations présentent l'avantage d'être calculées une fois pour toutes et permettent d'améliorer le comportement des algorithmes d'étude de visibilité pour le traitement de diverses vues d'une même scène
- nous avons énoncé différents critères de choix et de comparaison entre algorithmes qui permettent de ne considérer que celui qui est adapté au contexte d'utilisation (matériel, modèle de description, modèle de visibilité)
- pour chaque algorithme nous avons indiqué les possibilités de réalisation câblée, de décomposition en tâches parallèles. Pour chacun nous avons donné les possibilités qu'ils ont de traiter des éléments autres que des faces polygonales.
- enfin nous avons donné des solutions à plusieurs opérations géométriques qui pourront être utilisées dans un contexte autre que celui qui nous a préoccupé.

En ce qui concerne les axes de recherche nous en citerons deux qui nous semblent les plus importants :

-si nous avons dégagé un certain nombre de caractéristiques de la scène qui influencent le comportement des algorithmes d'étude de visibilité il serait important de les évaluer, en fonction des domaines d'application. Ceci permettrait de choisir l'algorithme le mieux adapté parmi ceux ayant le même couple modèle de description, modèle de visibilité.

- peu de travaux ont été à l'heure actuelle effectués sur l'étude de visibilité dynamique. Celle-ci doit tenir compte de la cohérence de la scène lors du passage à une vue suivante. Nous avons, par exemple, indiqué que l'algorithme de Newell, Newell et Sancha nous semblait adapté à ce contexte, vu le modèle de visibilité qu'il fournit..

BIBLIOGRAPHIE



Bâtiments de l'Ensimag : Algorithme de Watkins

BIBLIOGRAPHIE

[App 67] APPEL
The Notion of Quantitative Invisibility and the Machine Rendering of Solids. Proceeding ACM Nat. Conf. (1967)

[Arc 72] M. ARCHULETA
Hidden Surface Line Drawing Algorithm. Utah University, June 1972

[ARS 79] A. APPEL, J. ROHLF, A. STEIN
The Haloed Line Effect for Hidden Line Elimination. SIGGRAPH-ACM Computer Graphics 79

[AtW 78] ATHERTON, WEILER
Hidden Surface Removal Using Polygon Area Sorting. SIGGRAPH-ACM ACM Computer Graphics 78

[BeC 78] J. BECK, M. CYRUS
Generalized two-and three-dimensional Clipping.Comput. & Graphics, Vol. 3, 1978

[BeO 79] J. BENTLEY, T. OTTMANN
Algorithms for Reporting and Counting Geometric Intersections. I.E.E.E. Transactions on Computers, Vol. C 28, n°9, septembre 79

[Bli 78] J.F. BLINN
Clipping using Homogeneous Coordinates. SIGGRAPH-ACM Computer Graphics, 1978

[Bok 70] W.J. BOUKNIGHT, KELLEY
An Algorithm for Producing Halftone Computer Graphics Presentations with Shadows and Movable Light Sources. Spring Joint Computer Conference, 1970

[Bou 70] W.J. BOUKNIGHT
A Procedure for Generation of Three-Dimensional Half-tone Computer Graphics Representation. CACM, Vol. 13, n°9, septembre 1970

[Bro 76] D.K. BROTZ
Intersecting Polyhedra with Successive Planes.Computer and Graphics, Vol. 2, n°1, 1976

[Cat 74] E.E. CATMULL
A Subdivision Algorithm for Computer Display of Curved Surfaces. Ph.D. Thesis, University of Utah, décembre 1974

[Cat 78] E. CATMULL
A Hidden-Surface Algorithm with Anti-Aliasing. Computer Graphics 12(3):6, august 1978

[Cla 76] J.H. CLARK
Hierarchical Geometric Models for Visible Surface Algorithms. CACM, Vol. 19, n°10, octobre 1976

[Com 68] P.G. COMBA
A Procedure for Detecting Intersections of Three Dimensional Objects. JACM, Vol. 15, n°3, july 1968

[Dat 77] DAHAN, LE TUAN
Approche théorique d'une technique : perspective et ombres calculées. Thèse de Docteur-Ingénieur, ENST, Paris, 1977

[Eas 75] J.S. EASTMAN
An Efficient Scan Conversion and Hidden Surface Removal Algorithm. Comput. & Graphics, Vol. 1, 1975

[Enc 75] J. ENCARNACAO
Computer-Graphics : Programmierung und Anwendung von graphischen Systemen. R. Oldenburgh Verlag, Munich, 1975

[Fra 78] W.R. FRANKLIN
Combinatorics of Hidden Surface Algorithms. Harvard University Cambridge, Massachusetts, TR-12-78

[FuK 79] FUCHS, KEDEM
Predetermining Visibility Priority in 3-D Scenes. SIGGRAPH-ACM Computer Graphics, 1979

[GaM 69] R. GALIMBERTI, U. MONTANARI
An Algorithm for Hidden Line Elimination. CACM, Vol. 12, n°4, avril 1969

[Gil 78] W.K. GILOÏ
Interactive Computer Graphics : Data Structure, ... Prentice Hall, 1978

[Gri 70] J.G. GRIFFITHS
Drawing Opaque Solids using a Incremental Plotter. Computer Journal, Vol. 16, n°1, july 70

[Gri 73] J.G. GRIFFITHS
Drawing Pictures of Solid Objects using a Graph Plotter. Computer Aided Design

[Gri 75] J.G. GRIFFITHS
A Data Structure for the Elimination of Hidden Surfaces by Patch Subdivision. Computer Journal, Vol. 7, n°3, july 1975

[Gri 78] J.G. GRIFFITHS
Bibliography of Hidden-Line and Hidden-Surface Algorithms. Computer Aided Design, Vol. 10, n°3, may 1978

[Gri 79] J.G. GRIFFITHS
Eliminating Hidden Edges in Line Drawings. Computer Aided Design, Vol. 11, n°2, march 1979

[HaG 77] G. HAMLIN, C.W. GEAR
Raster-scan Hidden Surface Algorithm Techniques. Computer Graphics SIGGRAPH ACM, Vol. 11, n°2, summer 1977

[Jar 75] J.F. JARVIS
Two Simple Windowing Algorithms. Software Practice and Experience, Vol. 5, 115-112 (1975).

[Jon 71] C.B. JONES
A New Approach to the Hidden-Line Problem. Computer J., Vol. 14, n°3, (august 1971)

[KaG 79] M. KAPLAN, D. GREENBERG
Parallel Processing Techniques for Hidden Surface Removal. SIGGRAPH-ACM Computer Graphics, 1979

[KyG 79] D.S. KAY, D. GREENBERG
Transparency for Computer Synthesized Images. SIGGRAPH-ACM. Computer Graphics, 1979

[Klo 75] W.F. KLOS
Einheithliche formale Beschreibung, qualitative und quantitative Untersuchung von Visibilätsverfahren.Diplomaufgabe. Universitäts des Saarlands, 6.75

[Knu 73] D.E. KNUTH
The Art of Computer Programming. Tome 3, Sorting and Searching; Addison-Wesley, 1973

[LCW 80] LANE, CARPENTER, WHITTED, BLINN
Scan Line Methods for Displaying Parametrically Defined Surfaces. CACM, january 1980, Vol. 23, N°1.

[Lev 76] J. LEVIN
A Parametric Algorithm for Drawing Pictures of Solid Ojbects Composed of Quadric Surfaces. CACM, Vol. 19, n°10, octobre 1979

[Lou 70] P.P. LOUTREL
A Solution to the Hidden-Line Problem for Computer-Drawn Polyhedra. IEEE Transactions on Computers, C-19, 3, mars 1970

[Luc 77] M. LUCAS
Contribution à l'étude des techniques de communication graphique avec un ordinateur. Eléments de base des logiciels graphiques interactifs. Thèse d'état, Grenoble, décembre 1977

[Luc 80] M. LUCAS et autres
Eléments pour la réalisation des logiciels graphiques interactifs. à paraitre

[Mah 72] R. MAHL
Visible Surface Algorithms for Quadric Patches. IEEE transactions on computers, Vol. C-21, n°1, january 1972

[Mar 71] W.A. MARTIN
Sorting. Computing Surveys, Vol. 3, n°4, december 1971

[Mar 72] K. MARUYAMA
A Procedure to Determine Intersections Between Polyhedral Objects. International Journal of Computer and Information Sciences, Vol. 1, n°3, 1972

[Mar 79a] F. MARTINEZ
La synthèse d'images réalistes. Rapport bibliographique, I.M.A.G., janvier 1979

[Mar 79b] F. MARTINEZ
An Approach to the Modelling and Display of Landscapes. Proceedings of Eurographics, Bologna, Italy, october 1979

[Mat 72] Y. MATSUSHITA
Hidden Lines Elimination for a Rotating Object. CACM, Vol. 15, april 1972

[MeB 78] B. MEYER, C. BAUDOIN
Méthodes de Programmation. Editions Eyrolles 1978

[MeM 77] MERY, MIEULLET
Etude et réalisation d'un algorithme d'élimination de parties cachées : l'algorithme de NEWELL-SANCHA. Rapport de projet de 3ème année, E.N.S.I.M.A.G., juin 1977

[Mol 76] MORVAN, LUCAS
Images et Ordinateur. Introduction à l'infographie interactive. Larousse, 76

[MuP 78] D.E. MULLER, F.P. PREPARATA
Finding the Intersection of two Convex Polyhedra. Theoritical Computer Science 7 (1978)

[Nes 73] W.M. NEWMAN, R.F. SPROULL
Principles of Interactive Computer Graphics
Mac Graw Hill, First Edition, 1973

[Nes 79] W.M. NEWMAN, R.F. SPROULL
Principles of Interactive Computer Graphics. Mac Graw Hill, New-York, Second Edition

[NNS 72] M.E. NEWELL, R.G. NEWELL, T.L. SANCHA
A New Approach to the Shaded Picture Problem. Proceedings ACM National Conference, 1972

[Pre 78] F.P. PREPARATA
New Parallel Sorting Schemes. IEEE Transactions on Computers, Vol. C-27, N°7, july 1978

[Puk 77] R. PUK
General Clipping on an Oblique Viewing Frustrum. SIGGRAPH-ACM Computer Graphics IS77

[Rob 63] L.G. ROBERTS
Machine Perception of Three-Dimensional Solids. TR 315, MIT Lincoln Laboratory, mai 1963

[RWE 69] ROMNEY, WATKINS, EVANS
Real Time Display of Computer General Halftone Perspective Pictures Information processing 68. North Holland publishing company Amsterdam (1969)

[Sch 78] B. SCHACHTER
Decomposition of Polygons into Convex Sets. IEEE Transactions on Computers, Vol. C-27, n°11, nov. 78

[Sha 75] SHAMOS
Geometric Complexity. Proceedings of the 7th ACM Symposium on the Theory of Computing, may 1975

[ShH 76] SCHAMOS, HOEY
Geometric Intersection Problems. Proc. 17th Annu. Conf. Foundations of Computer Science, Oct. 76

[SpS 73] R. SPROULL, J.E. SUTHERLAND
A Clipping Divider . Proc AFIPS FJCC Vol 33

[SSS 74] I.E. SUTHERLAND, R.F. SPROULL, R.A. SCHUMACKER
A Characterization of Ten Hidden-Surface Algorithms. ACM Computing Surveys, Vol. 6, n°1, mars 1974

[StC 75] M.B. STEPHENSON, H.N. CHRISTIANSEN
A Polyhedron Clipping and Capping Algorithm and a Display System. Computer Graphics SIGGRAPH-ACM, vol. 9, n°3, Fall 1975

[SuH 74] I.E. SUTHERLAND, G.W. HODGMAN
Reentrant Polygon Clipping. CACM, Vol. 17, n°1, janvier 1974

[Tan 78] S.L. TANIMOTO
A Graph-Theoritic Real-Time Visible Surface Editing Technique. SIGGRAPH-ACM Computer Graphics, 1978

[War 69] WARNOCK
A Hidden Surface Algorithm for Computer Generated Halftone Pictures. TR 4.15 Computer Science Department, University of Utah, 1969

[Wat 70] G.S. WATKINS
A Real Time Visible Surface Algorithm. University of Utah, UTEC, CSs 70101, juin 1970

[Wei 66] WEISS
Be Vision. JACM, Vol. 13, n°2, april 1966

[Wil 72] H. WILLIAMSON
Algorithm 420, Hidden Line Plotting Program. CACM, Vol. 15, N°2, février 1972

[Wil 78] P.J. WILLIS
Proximity techniques for Hidden-Surface Removal. Computers and Digital Technique, october 1978, Vol. 1, n°4

[Wri 73] T.J. WRIGHT
A Two-Space Solution for the Hidden Line Problem for Plotting Functions of Two Variables. IEEE Transactions C-23, n°1, janvier 1973.

dernière page de la thèse

AUTORISATION DE SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 Avril 1974,

VU les rapports de présentation de Messieurs :

- R. GUEDJ, Directeur du laboratoire de Communication Homme-Machine de la THOMSON-CSF - ORSAY -
- M. LUCAS, Professeur à l'Université de NANTES

Monsieur Philippe B O U L L E

est autorisé à présenter une thèse en soutenance pour l'obtention du diplôme de DOCTEUR-INGENIEUR, spécialité "Génie Informatique".

Grenoble, le 4 Septembre 1980

Le Président de l'I.N.P.G.

Ph. TRAYNARD
Président
de l'Institut National Polytechnique
P.O. le Vice-Président,