Contribution à l'étude des techniques de communication graphique avec un ordinateur - Elements de base des logiciels graphiques interactifs

Michel Lucas

THESE

présentée à

UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

pour obtenir le grade de Docteur ès-Sciences "Mathématiques"

par

Michel LUCAS

CONTRIBUTION A L'ETUDE DES TECHNIQUES DE COMMUNICATION GRAPHIQUE AVEC UN ORDINATEUR - ELEMENTS DE BASE DES LOGICIELS GRAPHIQUES INTERACTIFS

thèse soutenue le 16 Décembre 1977 devant la Commission d'Examen :

Président	:	L. BOLLIET
Examinateurs	:	P.L. HENNEQUIN
		Ph. JORRAND
		J. KUNTZMANN
		O. LECARME
Invité	:	R.A. GUEDJ
Rapporteur	:	G. VEILLON

UNIVERSITE SCIENTIFIQUE
ET MEDICALE DE GRENOBLE

Monsieur Gabriel CAU : Président
Monsieur Pierre JULLIEN : Vice Président

MEMBRES DU CORPS ENSEIGNANT DE L'U.S.M.G.

PROFESSEURS TITULAIRES

MM.	AMBLARD Pierre	Clinique de dermatologie
	ARNAUD Paul	Chimie
	ARVIEU Robert	I.S.N.
	AUBERT Guy	Physique
	AYANT Yves	Physique approfondie
Mme.	BARBIER Marie-Jeanne	Electrochimie
MM.	BARBIER Jean-Claude	Physique expérimentale
	BARBIER Reynold	Géologie appliquée
	BARJON Robert	Physique nucléaire
	BARNOU Fernand	Biosynthèse de la cellulose
	BARRA Jean-René	Statistiques
	BARRIE Joseph	Clinique chirurgicale
	BEAUDOING André	Clinique de pédiatrie et puériculture
	BELORIZKY Elie	Physique
	BERNARD Alain	Mathématiques pures
Mme.	BERTRANDIAS Françoise	Mathématiques pures
MM.	BERTRANDIAS Jean-Paul	Mathématiques pures
	BEZEZ Henri	Pathologie chirurgicale
	BLAMBERT Maurice	Mathématiques pures
	BOLLIET Louis	Informatique (IUT B)
	BONNET Jean-Louis	Clinique ophtalmologique
	BONNET-EYMARD Joseph	Clinique gastro-entérologique
Mme.	BONNIER Marie-Jeanne	Chimie générale
MM.	BOUCHERLE André	Chimie et toxicologie
	BOUCHEZ Robert	Physique nucléaire
	BOUSSARD Jean-Claude	Mathématiques appliquées
	BOUTET DE MONVEL Louis	Mathématiques pures
	BRAVARD Yves	Géographie
	CABANEL Guy	Clinique rhumatologique et hydrologique
	CALAS François	Anatomie
	CARLIER Georges	Biologie végétale
	CARRAZ Gilbert	Biologie animale et pharmocodynamie
	CAU Gabriel	Médecine légale et toxicologie
	CAUQUIS Georges	Chimie organique
	CHABAUTY Claude	Mathématiques pures
	CHARACHON Robert	Clinique oto-rhino-laryngologique
	CHATEAU Robert	Clinique de neurologie
	CHIBON Pierre	Biologie animale
	COEUR André	Pharmacie chimique et chimie analytique
	CONTAMIN Robert	Clinique gynécologique
	COUDERC Pierre	Anatomie pathologique

Mme.	DEBELMAS Anne-Marie	Matière médicale
MM.	DEBELMAS Jacques	Géologie générale
	DEGRANGE Charles	Zoologie
	DELORMAS Pierre	Pneumophtisiologie
	DEPORTES Charles	Chimie minérale
	DESRE Pierre	Métallurgie
	DESSAUX Georges	Physiologie animale
	DODU Jacques	Mécanique appliquée (IUT I)
	DOLIQUE Jean-Michel	Physique des plasmas
	DREYFUS Bernard	Thermodynamique
	DUCROS Pierre	Cristallographie
	GAGNAIRE Didier	Chimie physique
	GALVANI Octave	Mathématiques pures
	GASTINEL Noël	Analyse numérique
	GAVEND Michel	Pharmacologie
	GEINDRE Michel	Electroradiologie
	GERBER Robert	Mathématiques pures
	GERMAIN Jean-Pierre	Mécanique
	GIRAUD Pierre	Géologie
	JANIN Bernard	Géographie
	KAHANE André	Physique générale
	KOSZUL Jean-Louis	Mathématiques pures
	KLEIN Joseph	Mathématiques pures
	KRAVTCHENKO Julien	Mécanique
	KUNTZMANN Jean	Mathématiques appliquées
	LACAZE Albert	Thermodynamique
	LACHARME Jean	Biologie végétale
Mme.	LAJZEROWICZ Janine	Physique
MM.	LAJZEROWICZ Joseph	Physique
	LATREILLE René	Chirurgie générale
	LATURAZE Jean	Biochimie pharmaceutique
	LAURENT Pierre-Jean	Mathématiques Appliquées
	LEDRU Jean	Clinique médicale B
	LE ROY Philippe	Mécanique (IUT I)
	LLIBOUTRY Louis	Géophysique
	LOISEAUX Pierre	Sciences nucléaires
	LONGEQUEUE Jean-Pierre	Physique nucléaire
	LOUP Jean	Géographie
Melle	LUTZ Elisabeth	Mathématiques pures
MM.	MALINAS Yves	Clinique obstétricale
	MARTIN-NOEL Pierre	Clinique cardiologique
	MAZARE Yves	Clinique médicale A
	MICHEL Robert	Minéralogie et pétrographie
	MICOUD Max	Clinique maladies infectieuses
	MOURIQUAND Claude	Histologie
	MOUSSA André	Chimie nucléaire
	NOZIERES Philippe	Spectrométrie physique
	OZENDA Paul	Botanique
	PAYAN Jean-Jacques	Mathématiques pures
	PEBAY-PEYROULA Jean-Claude	Physique
	PERRET Jean	Semeiologie médicale (Neurologie)
	RASSAT André	Chimie systématique
	RENARD Michel	Thermodynamique
	REVOL Michel	Urologie
	RINALDI Renaud	Physique
	DE ROUGEMONT Jacques	Neuro-chirurgie
	SEIGNEURIN Raymond	Microbiologie et Hygiène
	SENGEL Philippe	Zoologie
	SIBILLE Robert	Construction mécanique (IUT I)

MM.	SOUTIF Michel	Physique générale
	TANCHE Maurice	Physiologie
	TRAYNARD Philippe	Chimie générale
	VAILLANT François	Zoologie
	VALENTIN Jacques	Physique nucléaire
	VAUQUOIS Bernard	Calcul électronique
Mme.	VERAIN Alice	Pharmacie galénique
MM.	VERAIN André	Physique
	VEYRET Paul	Géographie
	VIGNAIS Pierre	Biochimie médicale

PROFESSEURS ASSOCIES

MM.	CRABBE Pierre	CERMO
	DEMBICKI Eugéniuz	Mécanique
	JOHNSON Thomas	Mathématiques appliquées
	PENNEY Thomas	Physique

PROFESSEURS SANS CHAIRE

Melle	AGNIUS-DELORD Claudine	Physique pharmaceutique
	ALARY Josette	Chimie analytique
MM.	AMBROISE-THOMAS Pierre	Parasitologie
	ARMAND Gilbert	Géographie
	BENZAKEN Claude	Mathématiques appliquées
	BIAREZ Jean-Pierre	Mécanique
	BILLET Jean	Géographie
	BOUCHET Yves	Anatomie
	BRUGEL Lucien	Energétique (IUT I)
	BUISSON René	Physique (IUT I)
	BUTEL Jean	Orthopédie
	COHEN ADDAD Pierre	Spectrométrie physique
	COLOMB Maurice	Biochimie
	CONTE René	Physique (IUT I)
	DELOBEL Claude	M.I.A.G.
	DEPASSEL Roger	Mécanique des fluides
	FONTAINE Jean-Marc	Mathématiques pures
	GAUTRON René	Chimie
	GIDON Paul	Géologie et minéralogie
	GLENAT René	Chimie organique
	GROULADE Joseph	Biologie médicale
	HACQUES Gérard	Calcul numérique
	HOLLARD Daniel	Hématologie
	HUGONOT Robert	Hygiène et médecine préventive
	IDELMAN Simon	Physiologie animale
	JOLY Jean-René	Mathématiques pures
	JULLIEN Pierre	Mathématiques appliquées
Mme.	KAHANE Josette	Physique
MM.	KRAKOWIACK Sacha	Mathématiques appliquées
	KUHN Gérard	Physique (IUT I)
	LUU DUC Cuong	Chimie organique
	MAYNARD Roger	Physique du solide
Mme.	MINIER Colette	Physique (IUT I)
MM.	PELMONT Jean	Biochimie
	PERRIAUX Jean-Jacques	Géologie et minéralogie
	PFISTER Jean-Claude	Physique du solide
Melle	PIERY Yvette	Physiologie animale

MM.	RAYNAUD Hervé	M.I.A.G.
	REBECQ Jacques	Biologie (CUS)
	REYMOND Jean-Charles	Chirurgie générale
	RICHARD Lucien	Biologie végétale
Mme.	RINAUDO Marguerite	Chimie macromoléculaire
MM.	ROBERT André	Chimie papetière
	SARRAZIN Roger	Anatomie et chirurgie
	SARROT-REYNAULD Jean	Géologie
	SIROT Louis	Chirurgie générale
Mme.	SOUTIF Jeanne	Physique générale
MM.	STIEGLITZ Paul	Anesthésiologie
	VIALON Pierre	Géologie
	VAN CUTSEM Bernard	Mathématiques appliquées

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

MM.	ARMAND Yves	Chimie (IUT I)
	BACHELOT Yvan	Endocrinologie
	BARGE Michel	Neuro-chirurgie
	BEGUIN Claude	Chimie organique
Mme	BERIEL Hélène	Pharmacodynamie
MM.	BOST Michel	Pédiatrie
	BOUCHARLAT Jacques	Psychiatrie adultes
Mme.	BOUCHE Liane	Mathématiques (CUS)
MM.	BRODEAU François	Mathématiques (IUT B) (Personne étrangère habilitée à être directeur de thèse)
	CHAMBAZ Edmond	Biochimie médicale
	CHAMPETIER Jean	Anatomie et organogénèse
	CHARDON Michel	Géographie
	CHERADAME Hervé	Chimie papetière
	CHIAVERINA Jean	Biologie appliquée (EFP)
	CONTAMIN Charles	Chirurgie thoracique et cardio-vasculaire
	CORDONNIER Daniel	Néphrologie
	COULOMB Max	Radiologie
	CROUZET Guy	Radiologie
	CYROT Michel	Physique du solide.
	DENIS Bernard	Cardiologie
	DOUCE Roland	Physiologie végétale
	DUSSAUD René	Mathématiques (CUS)
Mme.	ETERRADOSSI Jacqueline	Physiologie
MM.	FAURE Jacques	Médecine légale
	FAURE Gilbert	Urologie
	GAUTIER Robert	Chirurgie générale
	GIDON Maurice	Géologie
	GROS Yves	Physique (IUT I)
	GUIGNIER Michel	Thérapeutique
	GUITTON Jacques	Chimie
	HICTER Pierre	Chimie
	JALBERT Pierre	Histologie
	JULIEN-LAVILLAVROY Claude	O.R.L.
	KOLODIE Lucien	Hématologie
	LE NOC Pierre	Bactériologie-virologie
	MACHE Régis	Physiologie végétale
	MAGNIN Robert	Hygiène et médecine préventive
	MALLION Jean-Michel	Médecine du travail
	MARECHAL Jean	Mécanique (IUT I)
	MARTIN-BOUYER Michel	Chimie (CUS)
	MICHOULIER Jean	Physique (IUT I)

MM.	NEGRE Robert	Mécanique (IUT I)
	NEMOZ Alain	Thermodynamique
	NOUGARET Marcel	Automatique (IUT I)
	PARAMELLE Bernard	Pneumologie
	PECCOUD François	Analyse (IUT B) (Personnalité étrangère habilité à être directeur de thèse)
	PEFFEN René	Métallurgie (IUT I)
	PERRIER Guy	Géophysique-Glaciologie
	PHELIP Xavier	Rhumatologie
	RACHAIL Michel	Médecine interne
	RACINET Claude	Gynécologie et obstétrique
	RAMBAUD André	Hygiène et hydrologie (Pharmacie)
	RAMBAUD Pierre	Pédiatrie
	RAPHAEL Bernard	Stomatologie
Mme.	RENAUDET Jacqueline	Bactériologie (Pharmacie)
MM.	ROBERT Jean-Bernard	Chimie physique
	Romier Guy	Mathématiques (IUT B) (Personnalité étrangère habilité à être directeur de thèse)
	SCHAERER René	Cancérologie
	SHOM Jean-Claude	Chimie générale
	STOEBNER Pierre	Anatomie pathologie
	VROUSOS Constantin	Radiologie

MAITRES DE CONFERENCES ASSOCIES

MM.	DEVINE Roderick	Spectro physique
	HODGES Christopher	Transition de phases

Fait à SAINT MARTIN D'HERES, NOVEMBRE 1976.

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Monsieur Philippe TRAYNARD : Président

Monsieur Pierre-Jean LAURENT : Vice Président

PROFESSEURS TITULAIRES

MM.	BENOIT Jean	Radioélectricité
	BESSON Jean	Electrochimie
	BLOCH Daniel	Physique du solide
	BONNETAIN Lucien	Chimie minérale
	BONNIER Etienne	Electrochimie et électrométallurgie
	BOUDOURIS Georges	Radioélectricité
	BRISSONNEAU Pierre	Physique du solide
	BUYLE-BODIN Maurice	Electronique
	COUMES André	Radioélectricité
	DURAND Francis	Métallurgie
	FELICI Noël	Electrostatique
	FOULARD Claude	Automatique
	LESPINARD Georges	Mécanique
	MOREAU René	Mécanique
	PARIAUD Jean-Charles	Chimie-Physique
	PAUTHENET René	Physique du solide
	PERRET René	Servomécanismes
	POLOUJADOFF Michel	Electrotechnique
	SILBER Robert	Mécanique des fluides

PROFESSEUR ASSOCIE

M.	ROUXEL Roland	Automatique

PROFESSEURS SANS CHAIRE

MM.	BLIMAN Samuel	Electronique
	BOUVARD Maurice	Génie mécanique
	COHEN Joseph	Electrotechnique
	LACOUME Jean-Louis	Géophysique
	LANCIA Roland	Electronique
	ROBERT François	Analyse Numérique
	VEILLON Gérard	Informatique fondamentale et appliquée
	ZADWORNY François	Electronique

MAITRES DE CONFERENCES

MM.	ANCEAU François	Mathématiques appliquées
	CHARTIER Germain	Electronique
	GUYOT Pierre	Chimie minérale
	IVANES Marcel	Electrotechnique
	JOUBERT Jean-Claude	Physique du solide
	MORET Roger	Electrotechnique nucléaire
	PIERRARD Jean-Marie	Mécanique
	SABONNADIERE Jean-Claude	Informatique fondamentale et appliquée
Mme.	SAUCIER Gabrièle	Informatique fondamentale et appliquée

MAITRE DE CONFERENCES ASSOCIE

M.	LANDAU Ioan	Automatique

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

MM.	FRUCHART Robert	Directeur de Recherche
	ANSARA Ibrahim	Maître de Recherche
	CARRE René	Maître de Recherche
	DRIOLE Jean	Maître de Recherche
	MATHIEU Jean-Claude	Maître de Recherche
	MUNIER Jacques	Maître de Recherche

REMERCIEMENTS

Monsieur Louis BOLLIET, professeur à l'IUT d'informatique de Grenoble, a bien voulu accepter de présider ce Jury. Il me donne ainsi l'occasion de dire que je n'oublie pas que c'est grâce à lui que j'ai été introduit dans le monde universitaire. C'est aussi sur son impulsion que les premières études dans le domaine des techniques graphiques interactives ont été entreprises à Grenoble. Mes motifs de le remercier sont donc multiples, et je le fais de grand coeur.

Monsieur Gérard VEILLON, professeur à l'Institut National Polytechnique de Grenoble (ENSIMAG), a constamment favorisé les travaux de l'équipe de recherche sur les techniques graphiques interactives. Ses encouragements ont ainsi permis à quatre personnes de travailler dans un domaine qui n'était pas directement lié à ses centres d'intérêt. Les quatre thèses qui ont sanctionné ces travaux lui doivent beaucoup. Je tiens à le remercier pour le temps qu'il nous a consacré et à lui demander pardon pour les bousculades que nous lui avons infligées à l'approche des soutenances.

Monsieur Paul-Louis HENNEQUIN, professeur à l'Université de Clermont-Ferrand, a accepté la tâche de juger cette thèse. Les critiques amicales et les suggestions qu'il m'a prodiguées ont beaucoup contribué à l'amélioration du texte initial. Je le remercie de m'avoir ainsi permis de franchir en toute confiance les étapes conduisant à la soutenance.

Monsieur Jean KUNTZMANN, professeur à l'Université Scientifique et Médicale de Grenoble, me fait un grand honneur en siègeant dans ce Jury. Ses remarques judicieuses sur le texte du manuscrit n'ont fait qu'augmenter mon regret que notre collaboration ait été aussi tardive. Je tiens à le remercier pour le soutien actif et amical qu'il m'a apporté, aussi bien dans mes activités de recherche que d'enseignement.

Monsieur Olivier LECARME, professeur à l'Université de Nice, a lu et critiqué le manuscrit avec compétence et efficacité. Je m'en veux de lui avoir gâché quelques fins de semaine, mais je le remercie vivement pour la faiblesse de son temps de réponse.

Monsieur Philippe JORRAND, Maître de Recherches au CNRS, a accepté de lire et commenter le manuscrit, prétextant un besoin de s'instrui dans un domaine étranger à ses préoccupations. Je le remercie du soin qu'il a apporté à cette tâche. Le fait qu'il ait survécu à cette lecture indigeste m'a été un grand encouragement.

Monsieur Richard GUEDJ, Directeur du Laboratoire de Communication Homme-Machine de la THOMSON-CSF, m'a fait l'amitié de critiquer mon travail tout au long d'une intense et fructueuse collaboration. Ses conseils et réflexions m'ont été précieux pour déterminer les orientations de ma recherc Je l'en remercie et espère que notre travail commun continuera encore longte

En plus des membres du Jury, de nombreuses personnes ont lu et critiqué la première version de cette thèse. Je tiens à remercier spécialeme B. DAVID, T. JOHNSON, P. JULLIEN, A. LEMAIRE, B. MOREL et E. SALTEL pour leur contribution efficace.

Bien que la réalisation d'une thèse soit considérée en général comme une performance individuelle, j'ai conscience d'avoir profité (au bon sens du terme) d'un grand nombre de personnes. La tradition voulant que le nombre de pages consacré aux remerciements soit notablement plus restreint que le nombre de pages consacré à la thèse, je me vois contraint de résumer en quelques lignes un projet de *Mémoires*, en plusieurs volumes, mémoires consacrées à remercier en détail tous ceux avec qui j'ai travaillé.

Le premier tome sera réservé aux membres de l'équipe de recherche sur les techniques graphiques interactives. L'esprit d'aventure, de dévoueme et de coopération qui les anime a permis que le mot équipe prenne tout son sens, et ne soit pas un simple qualificatif administratif.

Le deuxième tome passera en revue les différents groupes de trava auxquels j'ai participé. La richesse des discussions et des activités commun m'a permis de mieux saisir les multiples facettes du monde informatique. Je remercierai spécialement les membres du groupe "graphique" de l'AFCET, pour notre longue collaboration. Je pense tout spécialemnnt aux vieux fidèles, J.P. CRESTIN, A. DUCROT, B. GAGEY, M. LEMOINE, P. LERAY, D. LE ROCH, F. PERRIQUET et G. SETIEN.

Un chapitre spécial sera dédié à P. MORVAN, honnorable co-auteur d'un livre orange dont le succès ne cesse de s'affirmer.

Le tome suivant me permettra de remercier tous mes collègues enseignants et chercheurs de l'IMAG. Beaucoup m'ont témoigné une confiance qui me touche beaucoup. Je remercierai spécialement le professeur N. GASTINEL qui s'est toujours intéressé aux problèmes graphiques. Même si nos discussion ont été parfois difficiles, ses encouragements m'ont été précieux.
Une grande partie de ce volume sera consacrée aux activités d'enseignement. J'aurai ainsi l'occasion de remercier J. MOSSIERE, P.Cl. SCHOLL, E. TOURNIER et J. VOIRON avec qui j'ai vécu des expériences d'enseignement tout à fait passionnantes. Je mentionnerai au passage tous les étudiants qui ont subi plus ou moins volontairement ces expériences et ont bien voulu nous faire part de leurs réactions. En particulier, l'équipe a beaucoup reçu des étudian de Troisième Année de l'ENSIMAG.

Un tome au moins sera consacré au personnel du Laboratoire et du CICG, dans sa diversité et sa pluralité. Les différents services (perforation exploitation, reprographie, secrétariats, administration, bibliothèque) ont été à ma disposition vingt quatre heures sur vingt quatre, m'obligeant ainsi à faire de nombreuses heures supplémentaires. Je tiens à remercier toutes es personnes pour leur aide constante et efficace tout au long des dix années passées ensemble, la réalisation de ce document n'étant qu'un exemple parmi c nombreux autres.

Le (les ?) dernier tome sera consacré à ma famille, plus spécialement à ma femme qui, forte de l'expérience acquise lors de la mise au monde de nos deux enfants, m'a fait profité de son calme et de sa patience tout au long de la conception, de la gestation et de l'accouchement de cette thèse. Ces efforts partagés ayant abouti, nous avons pris une grande décisio

nous n'en aurons pas une autre.

TABLE DES MATIERES

2. ELEMENTS POUR LA CONCEPTION DE LOGICIELS GRAPHIQUES INTERACTIFS DE BASE

3. ELEMENTS POUR UN LOGICIEL DE VISUALISATION DE POLYEDRES

4. QUELQUES EXEMPLES DE PROGRAMMES D'APPLICATION

.INTRODUCTION

INTRODUCTION

Le présent document constitue une synthèse des travaux que nous avons effectués au cours des cinq dernières années, dans le domaine des techniques graphiques interactives. La diversité des expériences que nous avons réalisées ou auxquelles nous avons participé nous a conduits à essayer de mettre au clair les notions de base de la conception des logiciels graphiques intéractifs. En ce sens, les lignes qui suivent sont une contribution supplémentaire à nos travaux, et non un simple bilan.

Nous rappellerons d'abord les divers cadres dans lesquels s'est exercée notre activité :

- l'équipe des techniques graphiques interactives du laboratoire IMAG. Cette équipe s'est préoccupée aussi bien de travaux de recherche que de travaux de développement, en particulier du fait de son insertion dans le Centre de Calcul Interuniversitaire de Grenoble.

- Le groupe de travail "graphique" de l'AFCET, qui nous a permis, à travers de fructueuses rencontres, de mieux saisir la diversité des problèmes liés aux techniques graphiques, et, surtout, de nous confronter à d'autres expériences et philosophies.

- Divers groupes d'étude :
  . groupe de travail sur la transmission de graphiques par le réseau CYCLADES,
  . groupe de travail sur la normalisation des logiciels graphiques (AFCET-AFNOR et IFIP),
  . groupe de travail sur l'informatique en tant qu'aide à la conception et à la réalisation de films.

- Participation à plusieurs projets dans le cadre de l'association MICADO (Mission pour la Conception Assistée et le Dessin par Ordinateur), nous donnant l'occasion d'être confrontés aux besoins industriels.

Cette thèse vise donc à rassembler cette expérience de manière cohérente.

Le Chapitre 1 contient la présentation des principaux éléments de la communication graphique avec un ordinateur. Après avoir défini la notion de dessin et celle d'image, nous montrons que le support de visualisation peut recevoir des représentations appartenant à l'un ou l'autre type, moyennant l'utilisation de quelques algorithmes que nous discutons et présentons. En ce qui concerne le dialogue, nous proposons quatre fonctions de haut niveau et montrons qu'elles peuvent être simulées quel que soit le dispositif de communication utilisé. Nous terminons en étudiant la structure des logiciels graphiques interactifs, et proposons de distinguer trois niveaux : la description, la préparation à la visualisation et la génération de la liste de visualisation. Une définition du rôle d'un logiciel graphique interactif de base en découle. Ce découpage en trois parties est alors repris dans les trois chapitres suivants.

Le Chapitre 2 nous permet de comparer deux approches différentes concernant la conception des logiciels graphiques de base. Nous passons en revue quelques logiciels couramment distribués, afin de montrer la diversité qui règne à tous les niveaux de définition. Nous étudions ensuite les travaux de normalisation entrepris à l'heure actuelle à travers l'analyse d'une proposition de norme faite par un groupe de travail américain (GSPC). Nous présentons enfin nos propres travaux qui s'appuient sur des concepts initiaux entièrement différents conduisant à la définition de logiciels d'un type nouveau.

Le Chapitre 3 est consacré à l'étude des éléments de base d'un logiciel de préparation à la visualisation. L'exemple choisi est un logiciel de traitement de polyèdres, logiciel interactif permettant de présenter et d'étudier des scènes tridimensionnelles à l'aide de méthodes diverses. Après avoir passé en revue quelques techniques de projection, transformation et découpage d'une scène, nous présentons rapidement les diverses familles d'algorithmes d'élimination de parties cachées. Nous décrivons l'expérience que nous avons acquise en programmant quatre algorithmes appartenant à des familles différentes en discutant les problèmes rencontrés, tant du point de vue algorithmique que du point de vue des structures de données et optimisations à réaliser. Une étude grossière du coût de ces algorithmes nous permet de proposer une classification en fonction de la complexité de la scène à dessiner.

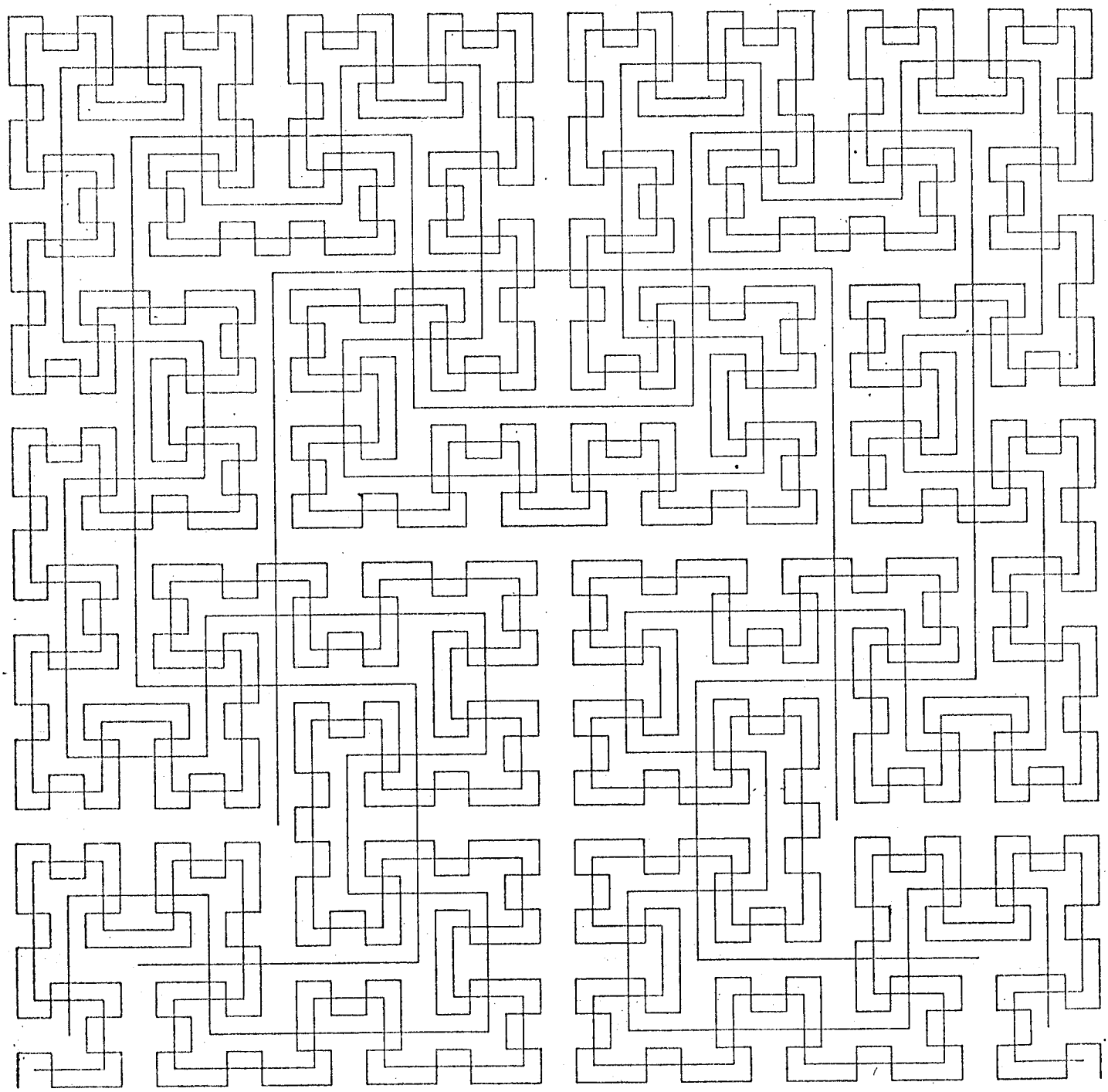
Le Chapitre 4 est consacré à l'étude de quelques applications appartenant à trois grands domaines : la graphique, la conception assistée et la production de films. Nous montrons la grande diversité de structure des logiciels de description et l'importance de la notion d'interprétation graphique de données, qui permet de faire correspondre des dessins différents à des données de base identiques.

En manière de conclusion, nous proposons plusieurs axes de recherche, sur des thèmes appartenant aux différents niveaux de logiciels que nous avons étudiés.

Nous terminons par une bibliographie importante, à travers laquel le nous avons cherché à illustrer non seulement les thèmes abordés dans notre étude, mais aussi à donner des ouvertures sur des thèmes proches qui n'avaient pas leur place ici. Nous pensons avoir ainsi regroupé une sélection des articles de base traitant des techniques graphiques interactives, offrant ainsi un guide permettant de se retrouver à travers les nombreuses publications consacrées à ce domaine.

CHAPITRE 1

ELEMENTS DE BASE DES LOGICIELS GRAPHIQUES INTERACTIFS



courbes de HILBERT

1. ELEMENTS DE BASE DES LOGICIELS GRAPHIQUES INTERACTIFS

1.1 LA COMMUNICATION GRAPHIQUE

Nous présentons dans ce paragraphe les éléments de base de la communication à l'aide de dessins, en nous appuyant sur les travaux de BERTIN [Ber 72]. Nous étudions ensuite les problèmes posés lorsque l'un des interlocuteurs est un ordinateur.

1.11 Les éléments de base de la communication graphique

L'homme s'est servi de tous temps de représentations graphiques pour transmettre des connaissances ou pour noter des situations dont il voulait garder le souvenir. L'utilisation d'images se justifie par le fait que le contenu représenté peut être facilement compréhensible par de larges classes de personnes possédant une culture analogue. C'est ainsi qu'un ensemble de signaux convenablement choisis est compréhensible par de grandes quantités d'individus, alors que la description en langue vernaculaire de la signification de ces signaux n'est accessible qu'à un ensemble plus restreint de personnes. Un exemple peut être trouvé dans l'ensemble de signaux qui ont été développés à l'occasion des jeux olympiques de Munich, destinés à des personnes de pays extrêmement différents mais ayant en commun une bonne connaissance du sport.

Le principal intérêt de l'image est la présentation synthétique qu'elle permet de faire, présentation utilisant l'outil d'analyse puissant qu'est le sens de la vision. En effet, l'homme peut saisir "en un coup d'oeil" les dispositions relatives et les principales liaisons entre les composants d'une image. Plusieurs informations peuvent être regroupées en un seul signal, ce qui donne moins d'éléments à analyser ou à mémoriser. Lorsque le nombre de composants de l'image n'est pas trop élevé (de l'ordre de 7 [Mil 56]), l'ensemble du message transmis par l'image peut être compris quasiment instantanément et conservé. Ce phénomène est à la base des techniques publicitaires qui visent à accrocher le regard pour transmettre de façon si possible durable une information très simple.

Un autre point d'intérêt de l'utilisation de l'image est la fixation qui se fait dans l'esprit : l'image est un moyen de mémorisation extrêmement efficace. La mémoire visuelle joue, selon les théories communément acceptées, à deux niveaux :

- un niveau de mémorisation temporaire, qui permet de se souvenir de certaines caractéristiques pendant un temps relativement bref (par exemple le temps de comparer mentalement deux images),

- un niveau de mémorisation définitif, qui permet de garder en mémoire des renseignements importants sous forme mnémotechnique (cas typique de la cartographie).

Un dernier rôle de l'image est celui d'inventaire. Ce rôle est particulièrement bien illustré en cartographie, où l'on peut, par exemple, représenter l'ensemble des éléments de production d'un pays sur une carte à l'aide d'un ensemble de signes permettant de distinguer chaque type de production. Une telle carte permet alors, par une étude approfondie, de prendre connaissance rapidement des moyens existants, à partir d'une représentation complexe mais facilement analysable par l'oeil. Cette étude peut également révéler des éléments d'information qui n'étaient pas a priori transcrits : par exemple la concentration d'industries en des régions privilégiées au détriment d'autres régions, l'importance des facteurs géographiques sur le type de cultures etc...

On distingue plusieurs types d'images :

- l'image non figurative (ou abstraite) qui fait appel à l'arrangement de ses différents composants pour provoquer des réactions de nature esthétique par l'intermédiaire du sens de la vision. Ce type d'image est essentiellement employé dans les arts plastiques, mais également dans tout ce qui concerne la décoration au sens large (tapisseries, décors, textiles)

- l'image figurative, qui vise à donner une représentation parfois schématique ou stylisée d'éléments reconnaissables car appartenant au monde réel (ou que l'on peut rapprocher du monde réel). On peut ranger dans cette rubrique la peinture figurative, mais aussi les techniques liées au dessin industriel ou architectural.

- Les graphiques, qui concernent la présentation d'informations à l'aide d'un système de signaux ou symboles connus à l'avance. On cherche généralement à mettre en valeur les corrélations existant entre différentes composantes de l'information à traiter. Les cas les plus classiques concernent les diagrammes (traitement d'une seule composante), les réseaux (deux composantes), les cartes (trois composantes et plus). Notons que l'écriture peut être considérée comme bâtie à partir d'un ensemble de symboles, mais ces symboles sont très élémentaires, ce qui explique la quantité de signes nécessaire pour décrire quelque chose de façon précise.

Le problème de la représentation graphique consiste à choisir une correspondance entre un ensemble d'informations et les éléments d'un langage graphique, langage possédant son vocabulaire et sa grammaire. Composer une image revient donc à définir un langage graphique (fonction de l'information à représenter), à analyser l'information à transcrire pour trouver les correspondances avec les éléments du langage, enfin à organiser les éléments en image. Notons que cet ensemble d'opérations n'est pas toujours réalisable. Par exemple, s'il est assez facile de trouver des transcriptions graphiques pour illustrer des situations se rapportant à la vie de tous les jours, la représentation de certaines abstractions peut se révéler impossible (par exemple : illustrer la notion d'intelligence).

L'image obtenue est ensuite étudiée par la personne à qui elle est destinée. On peut reconnaître deux manières de procéder, suivant que la personne connaît a priori le système de signes utilisé (analyse de l'agencement des signes pour en extraire des informations), ou que le système de signes n'est pas connu à l'avance (reconnaissance des éléments de base de l'image, reconstitution du système de signes utilisé, puis compréhension). Notons que l'on peut également s'intéresser à des séquences d'images, permettant d'illustrer l'évolution d'un phénomène en fonction d'un paramètre qui est souvent le temps. Deux types de variation peuvent être utilisés :

- variation discontinue de l'image, utilisée par exemple dans les montages audio-visuels,
- variation continue de l'image, utilisée dans le cas des films. Nous parlerons alors d'*image animée*.

Nous nous intéresserons aux représentations graphiques utilisant une surface de visualisation à deux dimensions. Que les images soient le fruit du travail humain ou le résultat d'un phénomène naturel (ombres sur un mur, arc-en-ciel, etc.), elles sont constituées de *taches*, éléments de base de la transcription graphique. Les paramètres de définition d'une tache sont les suivants :

- la *forme* , qui suppose que l'on sait reconnaître le contour (la silhouett de la tache. Ce contour est généralement défini par une suite de points.

- l'*emplacement* sur la surface de dessin, souvent repéré par la donnée de l position d'un point particulier appartenant à la tache. Cette variable permet de combiner différentes dispositions, soit pour créer des éléments d'équilibre du dessin, soit au contraire pour attirer l'attention sur un point particulier.

- la *taille* , qui se mesure par la surface occupée sur le plan de dessin. Ce paramètre permet de déduire une tache d'une autre par homothétie.

- l'*orientation*, qui suppose que la tache est repérée par rapport à un syst d'axes qui lui est propre,

- le *grain*, c'est-à-dire le nombre d'éléments séparables pour une unité de surface donnée.

- l'*intensité* (la valeur) de gris, c'est-à-dire le rapport de la surface noire à la surface blanche de la tache (notion pouvant s'étendre à la couleur ou la brillance).

- la *couleur* , qui peut être soit la couleur vraie correspondant à la réalité, soit une fausse couleur, correspondant à un choix arbitraire.

L'image est obtenue en combinant différentes taches, chacune d'elles étant caractérisée par le choix des valeurs des sept paramètres précédents. Nous ferons les deux remarques suivantes :

- deux types de taches sont utilisés habituellement dans le cas des graphiques : les points (correspondant à des taches de forme circulaire) et les lignes (correspondant à des taches beaucoup plus longues que larges).

- tous les paramètres définissant une tache ne sont pas obligatoirement employés : les dessins au trait n'utilisent que des points de diamètre très petit et des lignes, les dessins en noir et blanc ne font pas appel à la couleur.

La création de dessins est une entreprise quelquefois longue et délicate. Comme la nécessité de communiquer à l'aide de graphiques ou d'images figuratives est de plus en plus grande, en raison du flot d'informations à analyser et transmettre, il était naturel que l'on se posât la question de l'utilisation de l'ordinateur pour la réalisation de certains dessins. De plus, le travail en mode dialogué avec un ordinateur ayant montré toute sa richesse, le besoin de communiquer à l'aide de dessins s'est rapidement manifesté. Nous étudierons le domaine de la communication graphique avec un ordinateur, appelé parfois *infographie interactive*.

1.12 L'infographie interactive

L'ensemble des techniques de la communication graphique est trop vaste pour être étudié ici. Nous étudierons les limitations apportées par l'utilisation de l'ordinateur lors de la création de dessins et éventuellement lors de leur analyse. L'ordinateur est utilisé pour son aptitude à manipuler des codes. Les images traitées devront donc être codées sous forme numérique. A partir de ces codes, une représentation graphique sera fournie automatiquement par le dispositif de visualisation. Inversement, une image pourra être transmise au calculateur par le biais de périphériques spécialisés. Ces images devront être codées de manière à obtenir une structure de traitement compréhensible par le calculateur. On distingue plusieurs types de traitement :

- le *codage*, qui consiste à produire à partir des informations à représenter une structure codée permettant d'obtenir une transcription graphique sur le périphérique de visualisation,

- l'*acquisition* d'image, qui permet d'introduire dans la mémoire du calculateur une structure codée par le biais de périphériques spécialisés (caméra, tablette, etc...),

- la *manipulation* d'images, qui permet de passer d'une structure codée à une autre par usage de transformations géométriques, additions ou suppressions de sous-ensembles codés. La nouvelle structure pourra également être dessinée automatiquement,

- l'*analyse* qui permet, à partir d'une structure codée, d'obtenir des renseignements d'ordre quantitatif sur celle-ci (nombre d'éléments, barycentre, distances et longueurs diverses),

- la *reconnaissance de formes* , qui consiste à établir une classification en séparant automatiquement les différents composants de l'image, et en les analysant. L'ordinateur intervient alors pour "comprendre" automatiquement l'image à traiter.

- l'*amélioration* ou la *restauration* des images, qui vise à donner une apparence plus satisfaisante pour l'opérateur. Les principaux traitements concernent l'amélioration des contrastes, l'élimination du flou ou des distorsions géométriques.

Nous pouvons résumer cette discussion par le schéma de la figure 1.1 ci-dessous :

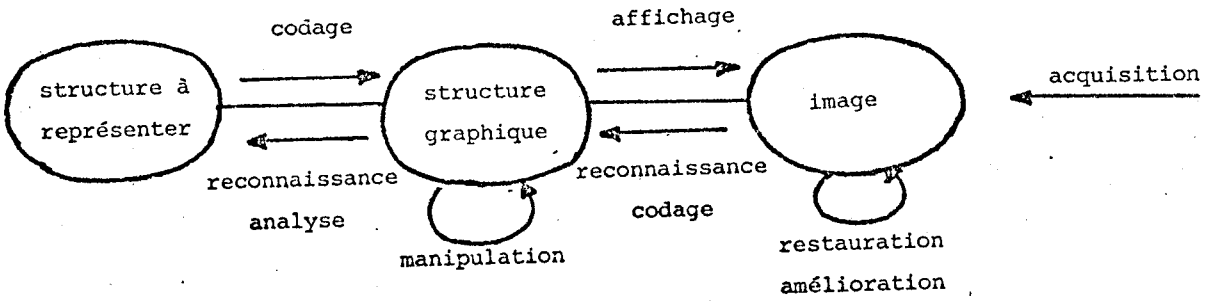


Figure 1.1 différents traitements d'image par ordinateur

Nous nous intéresserons essentiellement aux problèmes de codage et manipulation des images, dans une moindre mesure aux problèmes d'acquisition, cet ensemble de techniques recouvrant le domaine de l'infographie interactive ("computer graphics"). Les autres techniques sont plutôt du ressort de ce que l'on nomme le traitement d'images ("picture processing") et que nous ne traitons pas ici.

Les images engendrées par ordinateur appartiennent aux différentes catégories décrites plus haut (voir figure 1.2) :

- images non figuratives utilisées pour des recherches plastiques. Il est à remarquer que les artistes utilisant les techniques informatiques travaillent surtout dans ce domaine, faisant grand usage de traitements aléatoires, d'analyse combinatoire et de transformations diverses. Le calculateur permet d'explorer plus rapidement ce domaine, et donc de mieux comprendre les lois de composition des images.

- images figuratives, moyen d'expression privilégié des techniques de conception assistée par ordinateur en architecture, génie civil, construction automobile, navale ou aéronautique. L'objet en cours de conception peut ainsi être observé et des vérifications immédiates sur ses caractéristiques entreprises, au vu de l'image présentée.

- graphiques, qui permettent de présenter rapidement les relations liant des informations plus ou moins diverses,

- images animées, soit dans le cadre du contrôle de processus en temps réel soit pour l'aide à la conception de dessins animés.

Cependant, l'infographie interactive se caractérise par le fait que les images présentées sont des images "vivantes", c'est-à-dire susceptibles de modifications. L'image présentée sur l'écran est le reflet de la situation d'un programme à un instant donné, situation pouvant être modifiée au gré de l'opérateur. Si la situation change, il est évident que l'image qui la représente doit changer également, de manière à ce que l'opérateur puisse suivre l'évolution des traitements. Ceci a plusieurs conséquences :

- l'image produite est destinée à être étudiée, en général, pendant un temps relativement bref. Elle ne doit donc pas être très complexe, ou des techniques de réduction de sa complexité (par exemple en la découpant en sous-ensembles) doivent être fournies.

- l'image doit être produite rapidement, afin d'éviter une attente trop longue à l'opérateur. Cependant, cette question du temps de réponse dépend beaucoup de l'application à traiter. S'il s'agit d'applications de contrôle de processus, le temps de réponse doit être réduit au minimum, afin de pouvoir travailler en temps réel. Dans d'autres cas, le temps de réponse pourra varier de la fraction de seconde à plusieurs secondes. Cependant, si les calculs faisant passer d'une situation à une autre sont très longs (par exemple de l'ordre de plusieurs minutes il est clair que le travail en mode dialogué ne pourra être envisagé.

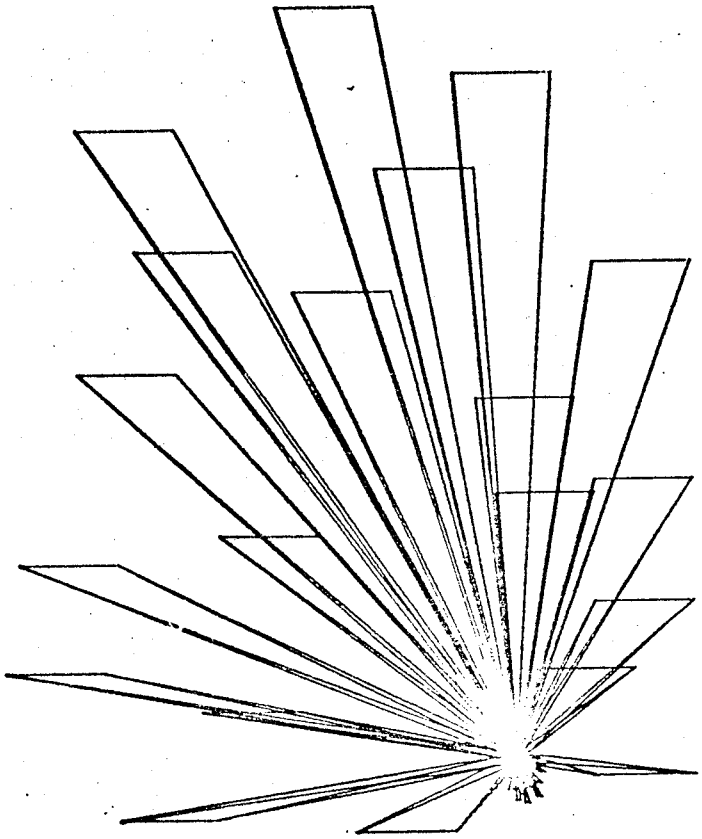
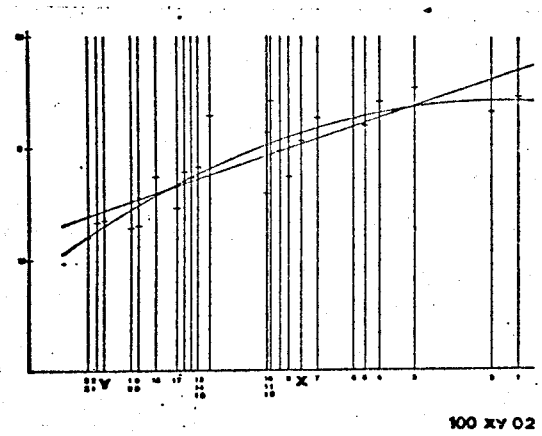
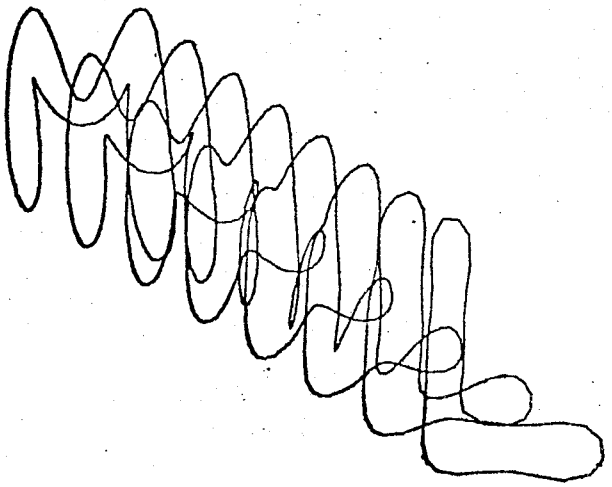


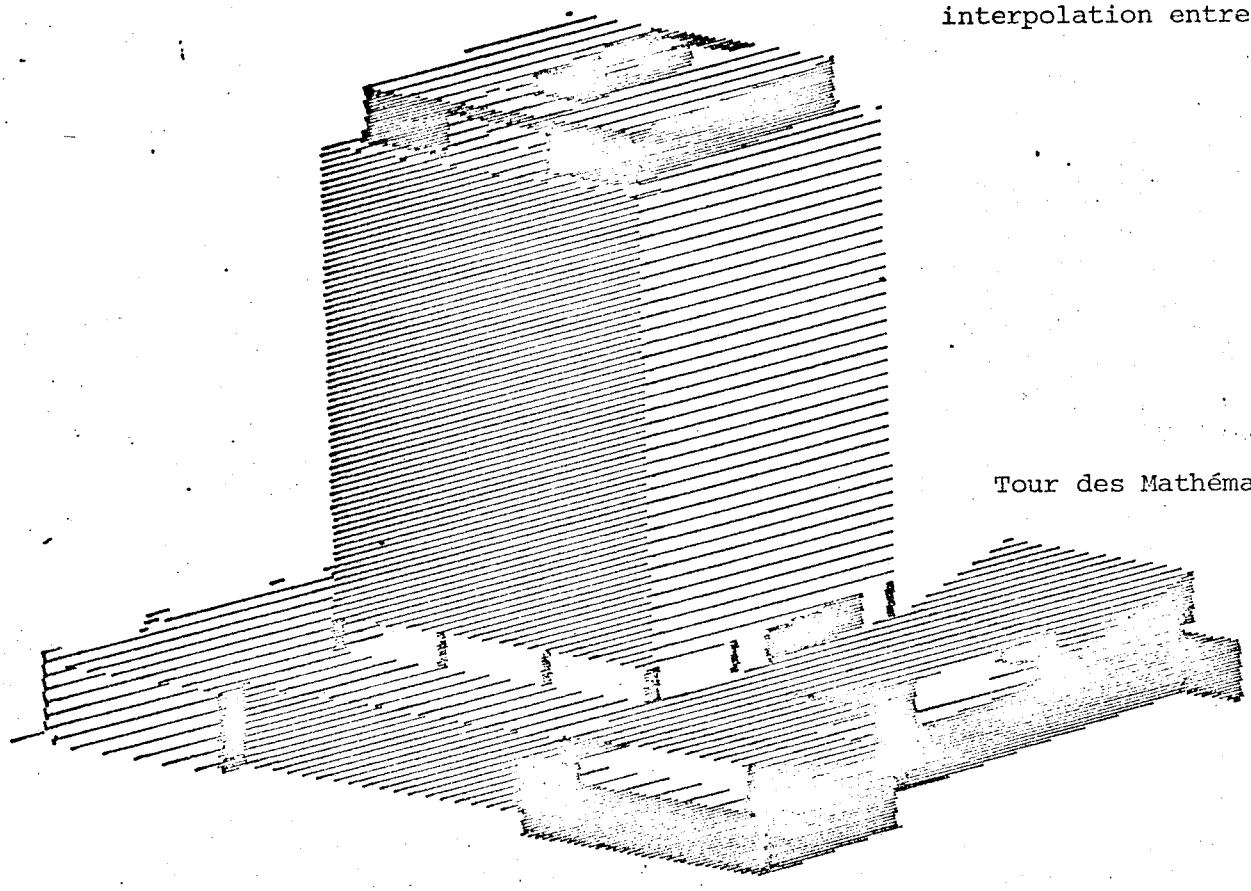
image non figurative



graphique



aide au dessin animé:
interpolation entre dessins- clés



Tour des Mathématiques Appliqué

figure 1.2 Différents types d'images produites par ordinateur

- l'image présentée sur l'écran n'est pas en général un produit fini. Ce n'est qu'une ébauche, souvent grossière, permettant de vérifier un certain nombre de caractéristiques, le dessin final étant obtenu en mode différé, par exemple sur un traceur de courbes.

Il faudra donc disposer de moyens permettant de préciser les modifications de situation souhaitées. L'opérateur doit pouvoir entamer un véritable dialogue avec le calculateur. Cette situation n'est pas particulière à l'infographie interactive. En effet, tout travail en mode dialogué fait appel à cet échange calculateur/opérateur : l'opérateur donne ses ordres grâce à des commandes, assortissant celles-ci de valeurs destinées à fixer tel ou tel paramètre. En général, le dialogue s'établit grâce à un terminal de type machine à écrire. Dans le cas de l'infographie intéractive, les deux fonctions précédentes (sélection d'une commande, introduction d'une valeur) seront reprises au niveau de la console de visualisation, afin de donner une certaine unicité de lieu quant à l'instrument de travail.

De plus, des fonctions purement graphiques seront mises à disposition de l'opérateur, qui lui permettront soit de créer un composant de l'image, soit de modifier l'image elle-même en désignant le ou les composants à traiter. Nous avons proposé [Luc 74 b] les quatre fonctions suivantes en tant qu'éléments de base du dialogue :

- le *menu*, qui permet à l'opérateur de choisir une action à effectuer dans une liste de commandes,

- l'*introduction de valeurs alpha-numériques*, permettant d'affecter une valeur à un (ou plusieurs) paramètre,

- l'*identification* d'un (ou plusieurs) composant d'une image, permettant ainsi d'indiquer l'élément auquel appliquer un certain traitement.

- l'*acquisition d'un élément d'image*, permettant ainsi de construire une représentation destinée au calculateur.

Ces quatre éléments de dialogue peuvent être combinés entre eux pour donner des actions plus complexes. Le problème est de savoir comment s'exprime ce dialogue, quelle sera sa mise en oeuvre au niveau de l'opérateur et au niveau du calculateur.

Le but de l'infographie interactive est de fournir les moyens matériels et logiciels permettant de mener à bien une étude à l'aide d'une communication graphique avec un ordinateur. Nous étudierons rapidement les problèmes liés à la technologie, ce qui nous permettra d'indiquer les limites de la communication graphique par ordinateur. Les problèmes de conception et de réalisation de logiciels graphiques interactifs seront abordés dans les chapitres suivants.

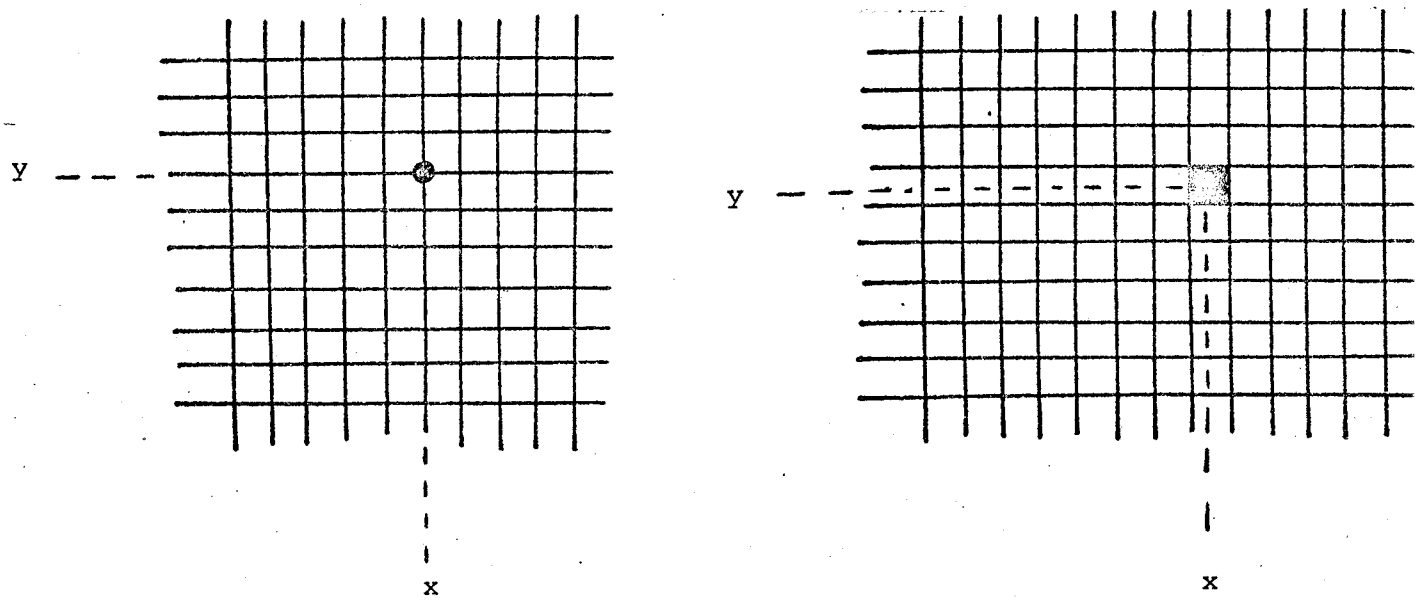
1.2 LA TECHNOLOGIE DE L'INFOGRAPHIE INTERACTIVE

Il ne s'agit pas de faire ici un catalogue des différents matériels et technologies disponibles. Le lecteur pourra se reporter à ([Das 69],[Luc 77a]). Nous nous proposons de mettre en valeur les paramètres communs à ces différentes technologies pour indiquer les moyens graphiques dont nous disposons.

1.21 La technologie de l'affichage

La caractéristique principale est que l'on va construire des images tramées. En effet, les surfaces de visualisation disponibles sont des surfaces constituées d'un quadrillage régulier. Ce quadrillage définit un système de coordonnées permettant de repérer tout point accessible par un couple (x,y). On distingue deux types de surface (cf. figure I.3) :

- les surfaces telles que le couple (x,y) désigne un sommet du quadrillage,
- les surfaces telles que le couple (x,y) désigne un carreau élémentaire.



surface pour le dessin au trait

surface pour le dessin point par poi

Figure I.3 Surfaces de dessin

Le premier type de surface est conçu essentiellement pour la représentation de dessins au trait, dessins obtenus en joignant un certain nombre de positions par des traits au graphisme varié. Le deuxième type de surface est conçu pour obtenir des images point par point, en définissant la couleur ou la luminosité de chacun des carreaux. Nous étudierons séparément les deux types de surface.

I.211 - Les surfaces pour le dessin au trait

Un dessin sera constitué par un ensemble de points et de segments de droite joignant deux positions repérables de la surface de visualisation. La donnée d'un point ou d'un segment de droite se fait soit en coordonnées absolues, c'est-à-dire repérées par rapport au système fixe de l'écran, soit en coordonnées relatives, c'est-à-dire par rapport à un repère mobile d'axes parallèles aux axes absolus (l'origine de ce repère mobile est souvent constituée par la position du dispositif d'écriture).

A chaque élément de dessin sera associé un graphisme, permettant de varier l'apparence du trait ou du point. Les paramètres habituels sont les suivants :

- pour le point : luminosité, taille, couleur
- pour le trait : luminosité, épaisseur, couleur, texture (pointillé, tireté, etc ...)

La figure I.4 donne un exemple d'utilisation de graphismes divers.

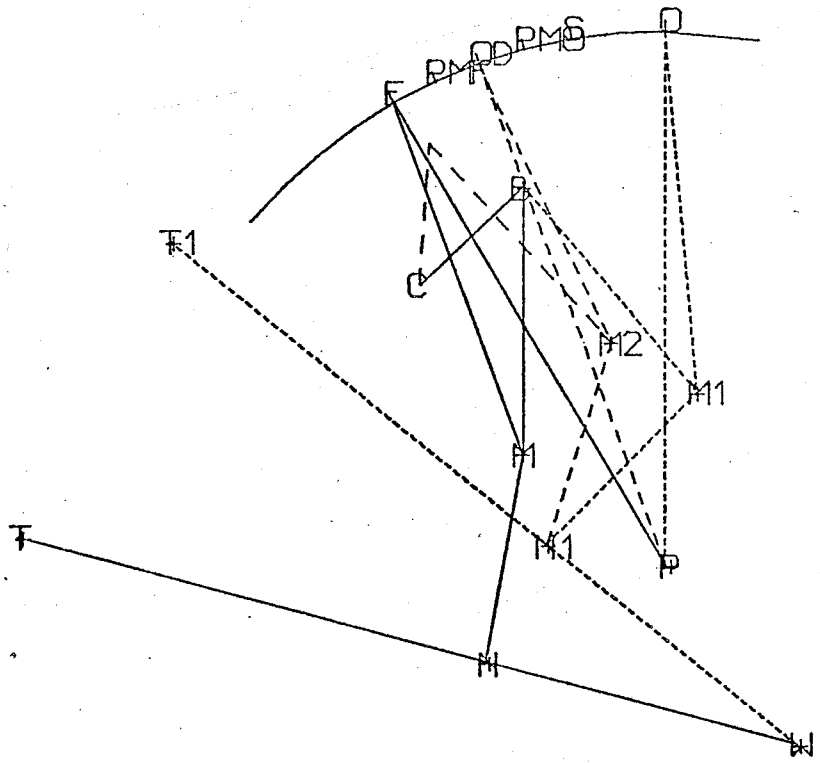


Figure I. 4 Différents types de graphismes dans un dessin au trait

Un dessin sera donc défini par la donnée d'une liste de n-uplets de la forme

{M, G, C}

où M désigne le mode de tracé (point ou segment),
G désigne le type de graphisme utilisé,
C le couple de coordonnées.
Cet ensemble de n-uplets est appelé liste de visualisation.

Pour des raisons d'efficacité, les codes de type M ou de type G s'appliquent non pas à un seul couple de coordonnées, mais à une liste de couples. La liste de visualisation est donc constituée d'une liste de codes opération (M ou G) séparés par leurs données (coordonnées). Le processeur graphique se comporte donc comme une unité centrale ordinaire, chargée de décoder et d'exécuter les instructions d'un programme.

Les opérations de base que l'on peut effectuer sont les suivantes :

- *destruction de la liste de visualisation (effacement total de l'écran),*
- *addition d'une sous-liste à la liste de visualisation.*

La construction d'un dessin se fait en effaçant tout l'écran, puis en constituant peu à peu la liste de visualisation par addition des sous-ensembles représentant chaque composant. Toute modification du dessin (en particulier la suppression d'une partie des composants) peut se faire en effaçant l'écran puis en reconstituant la nouvelle liste de visualisation. Cependant, certains matériels autorisent des opérations d'effacement sélectif à l'aide de l'opération :

- *suppression d'une partie de la liste de visualisation.*

Les opérations de modification sont alors sensiblement moins coûteuses.

Une dernière caractéristique concerne la charge que peut supporter la surface de visualisation. En effet, le nombre d'éléments de la liste de visualisation peut être limité par la vitesse de dessin. L'image créée étant en général fugitive, il importe de la recréer suffisamment souvent pour que l'oeil ne s'aperçoive pas de l'effacement (de 20 à 50 fois par seconde). Si l'image est trop complexe, le dispositif d'écriture peut ne pas arriver à la dessiner suffisamment rapidement, ce qui donne un effet de papillotement ("flickering") très fatiguant pour la vue. Le tableau I.1 donne les principales caractéristiques de quelques technologies couramment utilisées à l'heure actuelle.

	oscilloscope	tube mémoire	panneau plasma	laser
charge	2 à 3000 traits	quelconque	quelconque	quelconque
couleur	1 ou 3 ou 4	1	1	1
luminosités	1 ou de 6 à 8	1 ou 6	1	256
effacement sélectif	oui	non (oui)	oui	non
taille	30 x 30 cm	38 x 38 cm	30 x 30 cm	70 x 100 c
résolution	1024 x 1024	4096 x 4096	512 x 512	70000x100C

tableau 1.1 : Caractéristiques de quelques écrans

Les éléments de base du dessin étant le point et le segment de droite, se pose le problème de représenter d'autres figures. Par composition, il est possible d'obtenir des symboles, dont l'exemple le plus courant est la définition d'un jeu de caractères standard permettant d'afficher quelques éléments de texte. La plupart des consoles disposent d'une instruction laissant le soin à un générateur de caractères de remplir cette tâche. Cependant, le graphisme obtenu n'est pas suffisant pour certaines applications, et il est alors nécessaire de définir de nouveaux alphabets. Chaque caractère sera alors représenté par un ensemble de points ou de traits, les coordonnées étant en général données en relatif, afin de permettre la constitution de chaînes de caractères.

En ce qui concerne les lignes courbes, on peut les dessiner point par point en choisissant les points les plus proches de la courbe réelle. Cependant, si cette méthode est retenue pour des courbes ayant de nombreux points de rebroussement et des courbures très fortes, on préfère généralement approcher une courbe régulière par un ensemble polygonal obtenu en joignant par des segments de droite quelques points de la courbe plus ou moins espacés. Cette solution, meilleure du point de vue du nombre d'éléments nécessaire pour définir la courbe, est en général acceptable du point de vue visuel, l'oeil reconstituant l'allure générale de la courbe. De la même manière, les taches ne sont représentées que par leur contour, l'oeil sachant, dans les cas relativement simples, "remplir" les surfaces suggérées. Nous proposerons plus loin quelques techniques de coloriage de ces zônes, pour le cas où il s'avèrerai indispensable de remplir l'intérieur des contours.

I.212 - Les surfaces pour le dessin point par point.

L'élément de base n'est plus que le point de forme carrée ou rectangulaire ("picture element" ou "pixel"), auquel on associe une couleur ou une intensité. Les dessins seront alors constitués essentiellement de points, ou de taches formées d'ensembles de points. Par opposition aux *dessins* (constitués de traits et de points), nous parlerons d'*images* (c'est de plus le terme technique pour les dessins obtenus sur un écran de télévision, technologie dominante pour les surfaces à pointillage).

Les opérations de base sont

- *l'effacement total de l'écran, par attribution d'une couleur de fond à chaque carreau élémentaire,*

- *l'affectation d'une couleur (ou d'une intensité) à un carreau élémentaire.*

Pour obtenir une image, il sera donc nécessaire d'affecter une couleur ou une intensité à chacun des carreaux élémentaires formant la surface de visualisation. Notons ici qu'à l'inverse de ce qui se passe pour les dessins, l'oeil analyse l'image point par point pour extraire les contours, afin de différencier les éléments la composant. Le coloriage des taches intervient en fait pour permettre de mieux saisir l'espace enclos par la tache (indépendamment d'un rôle éventuellement symbolique).

La technologie des surfaces à pointillage élimine le problème de charge de l'image, celle-ci restant stable. Le vrai problème est celui de la transmission des informations nécessaires pour construire une image : 262 144 octets pour une image formée de 512 x 512 points avec une dynamique de 8 bits pour l'intensité. Ceci explique que de nombreux codes aient été proposés pour réduire cette charge.

I.213 - Inscription d'un dessin sur une surface point par point

Le problème de l'inscription d'un trait sur une surface point par point revient à trouver une approximation telle que l'oeil puisse reconstruire le segment de droite représenté. Plusieurs algorithmes ont été proposés([Bre 65], [Ear 77]), dont la caractéristique principale est d'éviter les opérations de multiplication et de division et de travailler en entier. Nous avons pour notre part proposé un algorithme [Luc 74 b] fondé sur les remarques suivantes :

Soit à tracer le segment de droite (x_a,y_a) (x_b,y_b). Nous prendrons par hypothèse $|(x_a-x_b)| \geq |(y_a-y_b)|$, la discussion étant la même pour les autres cas de figure. La stratégie adoptée consiste à dessiner un maximum de points, ce qui conduit à se déplacer sur l'axe des x (à cause de l'hypothèse de départ). On a alors les relations suivantes :

$$y_i = y_a + (x_i-x_a) * \frac{y_b-y_a}{x_b-x_a}$$

comme

$$x_{i+1} = x_i+1 \qquad \text{(si on suppose } x_a < x_b\text{)}$$

alors

$$y_{i+1} = y_a + (x_i+1-x_a) * \frac{y_b-y_a}{x_b-x_a} = y_i + 1 * \frac{y_b-y_a}{x_b-x_a}$$

Le problème consiste à décider du moment où l'on modifie y, sachant que l'on ne peut ajouter que 0 ou 1. Au départ, le point courant (x,y) est initialisé aux valeurs (x_a,y_a). Au pas suivant, si l'on ne modifie pas y, l'erreur commise par rapport à la position existe est $\frac{y_b- y_a}{x_b-x_a}$.

A chaque pas, l'erreur commise se cumule. Pour la minimiser, on convient que lorsque l'erreur cumulée atteindra (ou dépassera) l'unité, on ajoutera 1 à y. Le processus se répète jusqu'à l'obtention du point final. Nous ferons deux remarques :

- la mise en oeuvre de cet algorithme implique simplement des opérations sur les entiers. En effet, plutôt que de cumuler des réels, ce qui ajouterait des erreurs, il suffit de maintenir la relation entre entiers :

$$n * |(y_b-y_a)| \leq |(x_b-x_a)|$$

ce qui n'utilise que des additions et des soustractions et donne un résultat exact.

- l'algorithme que nous avons publié, issu directement des remarques précédentes, donnait un dessin tel que tous les points calculés se trouvaient soit au dessus, soit au dessous du segment idéal. En effet, les calculs proposés consistent à obtenir les fonctions "plafond" ou "plancher" d'approximation d'un réel par un entier, et non "l'entier le plus proche". Le graphisme obtenu se trouvait ainsi déséquilibré. De plus, le tracé obtenu était différent suivant que l'on partait d'une extrémité ou de l'autre du segment. Suite à une suggestion qui nous a été faite, nous avons pu constater que l'équilibre pouvait être en grande partie rétabli en initialisant la variable de cumul à la moitié de la longueur à parcourir en x (ce qui revient à prendre la partie entière d'un réel auquel on ajoute la valeur 0.5).

Les tracés obtenus sont alors beaucoup plus symétriques.

Nous donnons page suivante l'algorithme complet correspondant à cette version. La notation utilisée est inspirée des langages de la famille ALGOL, mais est suffisamment libre pour éviter les problèmes liés à un langage de programmation spécifique.

```
SEGMENT (entier XA,YA,XB,YB) ;
début {tracé du segment de droite (XA,YA)-(XB,YB) }
| entier DELTAX, DELTAY, XINCR, YINCR, CUMUL, X, Y ;
| X := XA ; Y := YA ; {(X,Y) point courant}
| AFFICHER (X,Y) ;
| XINCR := si XA < XB alors +1 sinon -1 ;
| YINCR := si YA < YB alors +1 sinon -1 ;
| DELTAX := ABS(XA-XB) ; DELTAY := ABS(YA-YB) ;
| si DELTAX > DELTAY {on dessine le maximum de points}
| alors début
|         CUMUL := DELTAX / 2 ;
|         pour I := 1 pas 1 jusqua DELTAX faire
|         début
|         | X := X + XINCR ;
|         | CUMUL := CUMUL+DELTAY ;
|         | si CUMUL ≥ DELTAX
|         | alors début
|         |       | CUMUL := CUMUL - DELTAX ;
|         |       | Y := Y + YINCR
|         |       fin ;
|         | AFFICHER(X,Y)
|         fin
|       fin
| sinon début
|         CUMUL := DELTAY / 2 ;
|         pour I := 1 pas 1 jusqua DELTAY faire
|         début
|         | Y := Y + YINCR ;
|         | CUMUL := CUMUL + DELTAX ;
|         | si CUMUL ≥ DELTAY
|         | alors début
|         |       | CUMUL := CUMUL - DELTAY ;
|         |       | X := X + XINCR
|         |       fin ;
|         | AFFICHER (X,Y)
|         fin
|       fin
fin SEGMENT ;
```

Pour améliorer la présentation du dessin, on peut vouloir remplir l'intérieur d'une tache monochrome définie par son contour. Nous proposons un algorithme se décomposant en deux phases :

- inscription du contour de la tache dans une matrice représentant la surface de visualisation,

- remplissage de l'intérieur de la tache par un balayage ligne par ligne.

La bonne exécution de la deuxième phase implique que l'on sache à tout moment si l'on est situé à l'extérieur ou à l'intérieur de la tache. Le principe retenu est fort simple : on part d'un point supposé à l'extérieur de la tache. On balaye jusqu'à rencontrer un point du contour. Dès cet instant, on remplit les points situés sur cette ligne, jusqu'à rencontrer un nouveau point de contour. Le remplissage est alors interrompu. Le balayage se poursuit par une alternance de remplissages et de recherches de points de contour jusqu'à avoir atteint le bord droit de la matrice (cf Figure I.5).

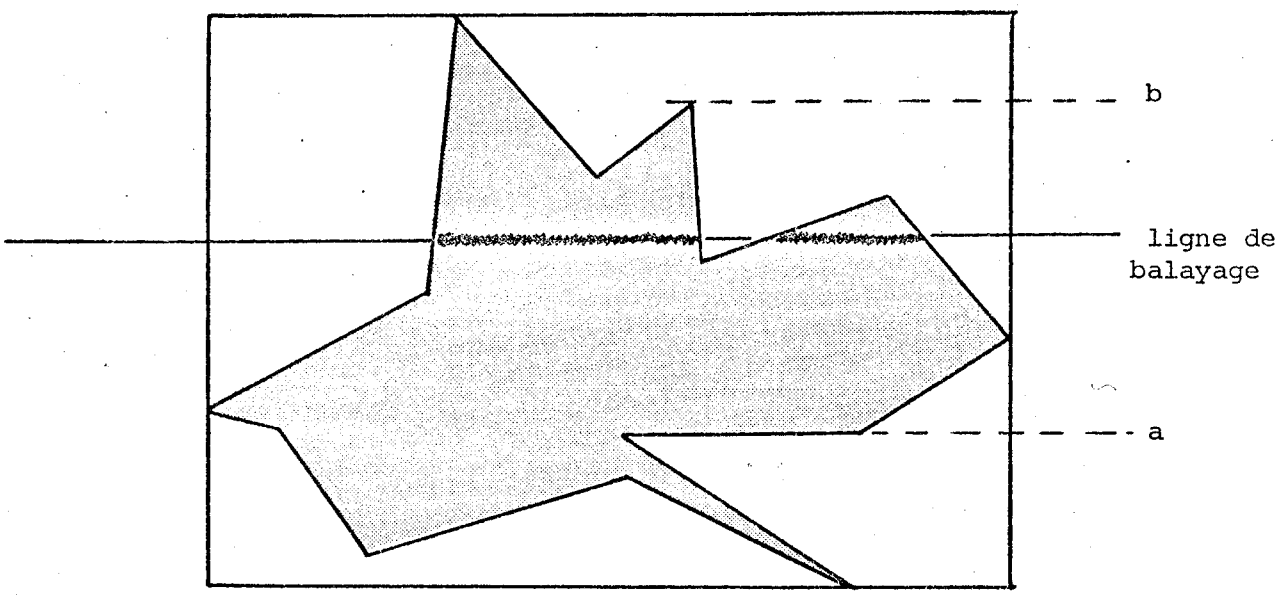


Figure I.5 Principe du remplissage d'une tache

Ce procédé implique que l'on ait toujours un couple de points situés sur le contour délimitant un segment intérieur à la tache. Les difficultés proviennent des cas suivants :

a) - segments horizontaux : on utilise un code disant qu'il faut colorier le segment, mais ne changeant pas la variable booléenne INTERIEUR, (code 2)

b) - sommet commun à deux cotés situés du même côté que l'horizontale passant par ce sommet : code impliquant un coloriage mais pas de modification de INTERIEUR (code 2).

c) - Recouvrements : dus soit au croisement de deux côtés, soit à la précision moindre de la projection. Si ce nombre de recouvrements est pair, modifier la variable INTERIEUR, sinon ne pas la changer.

d) - Segments horizontaux crées par l'approximation point par point : ne noter que les changements d'ordonnée.

L'algorithme qui suit traite ces difficultés.

```
TACHE (entier XMIN, XMAX, YMIN, YMAX ; entier tableau X,Y,Z(*);entier COULE
début {remplissage d'une tache inscrite dans le rectangle XMIN,YMIN,XMAX,YM
      Les coordonnées des sommets de contour sont dans X,Y,Z ; La fonction
      SOMMETSUIVANT fournit l'indice d'un point de contour lorsqu'on suit
      le bord de la tache}
 logique UPL,UPM,POINTE ; entier L,LP,M ;
 logique INTERIEUR ; logique ILYASOMMET ;
 {initialisation à la couleur de fond}
 pour I:=XMIN pas 1 jusqua XMAX faire
 pour J:=YMIN pas 1 jusqua YMAX faire
 IMAGE(I,J) := 0 ;

 {inscription du périmètre}
 LP := PREMIER SOMMET ; L := SOMMETSUIVANT ;
 tantque Y(LP) = Y(L) faire     {sauter les côtés horizontaux}
 début
 | LP := L ;
 | L  := SOMMETSUIVANT
 fin ;

 {parcours du contour de la tache}
 UPL := Y(LP)<Y(L) ; LP := L ;     {sommet de départ}
 ILYASOMMET := vrai ;
 tantque ILYASOMMET faire
 début
 | M := SOMMETSUIVANT ;
 | si Y(L)=Y(M)
 | alors début  {segment horizontal}
 |       | PA := si X(L) < X(M) alors +1 sinon -1 ;
 |       | pour I:= X(L) pas PA jusqua X(M) faire IMAGE(I, Y(L)) := 2
 |       fin
 | sinon début
 |       | UPM := Y(L)<Y(M) ; POINTE := (UPM ≠ UPL) ;
 |       | SEGMENT (X(L),Y(L),X(M),Y(M),POINTE)
 |       fin ;
 | L := M ; ILYASOMMET := (M≠LP)
 fin ;
```

```
|
|  {inscription de la surface}
|  pour J:=YMIN pas 1 jusqua YMAX faire
|  début
|  |  INTERIEUR := faux ;
|  |  pour I:=XMIN pas 1 jusqua XMAX faire
|  |  début
|  |  |  si IMAGE(I,J) = 1   {point de contour}
|  |  |  alors début
|  |  |  |        | INTERIEUR := ¬ INTERIEUR ;
|  |  |  |        | IMAGE(I,J) := COULEUR
|  |  |  |        fin
|  |  |  sinon si INTERIEUR ou (IMAGE(I,J) = 2)
|  |  |  |        alors IMAGE(I,J) := COULEUR
|  |  fin
|  fin
fin TACHE ;


SEGMENT (entier XA, YA, XB, YB ; logique POINTE) ;
début {tracé d'un segment en préparation au remplissage d'une tache}
|  entier DELTAX, DELTAY, XINCR, YINCR, CUMUL, X, Y, YPREC ;
|  X := XA ; Y := YA ; (X,Y) point courant
|  si POINTE {traitement d'un cas b}
|  alors IMAGE(X,Y) := (si IMAGE(X,Y) ≠ 1 alors 2 sinon 1)
|  sinon IMAGE(X,Y) := (si IMAGE(X,Y) = 1 alors 2 sinon 1) ;
|  XINCR := si XA < XB alors +1 sinon -1 ;
|  YINCR := si YA < YB alors +1 sinon -1 ;
|  DELTAX := ABS(XA-XB) ; DELTAY := ABS(YA-YB) ; YPREC := Y ;
|  si DELTAX > DELTAY
|  alors début {traiter les cas d }
|  |        | CUMUL := DELTAX/2 ;
|  |        | pour I:=1 pas 1 jusqua DELTAX-1 faire {pas le dernier point}
|  |        | début
|  |        | |  X:=X + XINCR ; CUMUL := CUMUL+DELTAY ;
|  |        | |  si CUMUL ≥ DELTAX
|  |        | |  alors début
|  |        | |  |        | CUMUL := CUMUL-DELTAX ;
|  |        | |  |        | Y := Y+YINCR
|  |        | |  |        fin ;
|  |        | |  si (YPREC=Y) ou (Y=YB)
|  |        | |  alors début {même Y}
|  |        | |  |        | si IMAGE(X,Y) ≠ 1 alors IMAGE(X,Y) := 2 ;
|  |        | |  |        fin
|  |        | |  sinon début {nouvel Y}
|  |        | |  |        | IMAGE(X,Y) := (si IMAGE(X,Y)=1 alors 2 sinon 1) ;
|  |        | |  |        | YPREC := Y
|  |        | |  |        fin
|  |        | fin
|  |        fin
|  sinon début {pas de problème si on se déplace en y}
|  |        | CUMUL := DELTAY/2 ;
|  |        | pour I:=1 pas 1 jusqua DELTAY faire
|  |        | début
|  |        | |  Y := Y+YINCR ; CUMUL := CUMUL+DELTAX ;
|  |        | |  si CUMUL ≥ DELTAY
|  |        | |  alors début
|  |        | |  |        | CUMUL := CUMUL-DELTAY ;
|  |        | |  |        | X := X+XINCR
|  |        | |  |        fin ;
|  |        | |  IMAGE(X,Y) := (si IMAGE(X,Y) = 1 alors 2 sinon 1)
|  |        | fin
|  |        fin
fin SEGMENT ;
```

I.214 - Inscription d'une image sur une surface de dessin au trait

La première difficulté provient du fait que l'on dispose en général d'un nombre de luminosités restreint (5 ou 6), voire même simplement d'un seul niveau. Dans ce cas, il est parfois utile de pouvoir simuler des variations de luminosité. Plusieurs techniques ont été développées [JJN76] mais pour des raisons de commodité et de simplicité de mise en oeuvre, nous utilisons la technique connue sous le nom de "ordered dither" [Jud 75] . Le principe consiste à remplacer une position de l'écran par plusieurs points contigus, ces points étant allumés ou éteints en fonction de l'intensité à simuler. La matrice de points simulant la luminosité est une matrice n × n notée D_n définie de la façon suivante :

$$D_2 = \begin{vmatrix} 0 & 2 \\ 3 & 1 \end{vmatrix}$$

$$D_n = \left|\begin{array}{cc} 4\ D_{n/2} & 4\ D_{n/2} + 2\ U_{n/2} \\ \hline 4\ D_{n/2} + 3\ U_{n/2} & 4\ D_{n/2} + U_{n/2} \end{array}\right|$$

avec U_n matrice carrée d'ordre n dont tous les éléments sont égaux à 1.

La disposition des valeurs à l'intérieur de la matrice D_n a été choisie de manière à ce que la luminosité simulée sur une large surface possède de bonnes propriétés (pas d'effet d'accumulation, bonne répartition des points allumés etc...).

La matrice permettant de simuler 16 niveaux de luminosité est donc la suivante :

$$D_4 = \begin{vmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{vmatrix}$$

Pour chaque point à afficher, on n'allume que les points de la matrice dont l'intensité est inférieure ou égale à l'intensité à simuler. Nous donnons ci-après l'algorithme de présentation d'une matrice en utilisant cette méthode. La version proposée est prévue pour afficher une matrice de m x n éléments (m ≤ 256 ; n ≤ 256) sur un écran de 1024 × 1024 points. Les paramètres d'appel comportent le minimum et le maximum de la fonction, afin de répartir linéairement les 16 intensités d'une matrice D_4 qui sert ici de base de calcul.

```
LUMINOSITE (réel tableau T(*,*) ; entier M,N ; réel FMIN,FMAX) ;
début {affichage du contenu du tableau T de bornes comprises entre 0 et
       256 à l'aide d'une simulation de 16 niveaux de luminosité ; la
       simulation se fait par utilisation du tableau D supposé rempli
       par ailleurs}.
  réel COEF ; entier XOR, YOR,XL,YL,X,Y ;
  entier F ;
  COEF := 16./(ABS(FMIN-FMAX));
  XOR := (256-N)*2 ;    {centrage sur un écran 1024 × 1024}
  YOR := 1024 - (256-M)*2 ;
  YL := YOR ;
  pour I := 1 pas 1 jusqua M faire
  début
    XL := XOR ;
    pour J:=1 pas 1 jusqua N faire
    début
      F := ARRONDI((T(I,J)-FMIN)*COEF) ;  {couleur associée}
      Y := YL ;
      pour K:=1 pas 1 jusqua 4 faire
      début
        X := XL ;
        pour L:=1 pas 1 jusqua 4 faire
        début
          si F  > D(K,L) alors AFFICHER(X,Y) ;
          X := X+1
        fin ;
        Y := Y-1
      fin ;
      XL := XL+4
    fin ;
    YL := YL-4
  fin
fin LUMINOSITE ;
```

Il est clair que cette méthode conduit à une charge très importante de l'écran. Cette charge peut être inacceptable (papillotement), ou conduire à des transferts de données trop importants. On pourra alors se contenter, au cas où le nombre de luminosités est faible (5 ou 6 maximum), d'utiliser un hachurage des taches à l'aide de segments horizontaux dont l'espacement

sera fonction de la luminosité. Une légère modification de l'algorithme de remplissage de taches présenté en I.213 permet de réaliser ce procédé. Il suffit en effet, au lieu de remplir systématiquement chaque case de la matrice lors de la deuxième phrase de l'algorithme, de conserver trace du point d'entrée dans la surface, jusqu'au moment où on la quitte. On dispose alors des coordonnées des extrémités du segment à tracer. La figure I.6 présente deux aspects de la même scène : une représentation au trait, et un remplissage des taches en utilisant 4 couleurs.

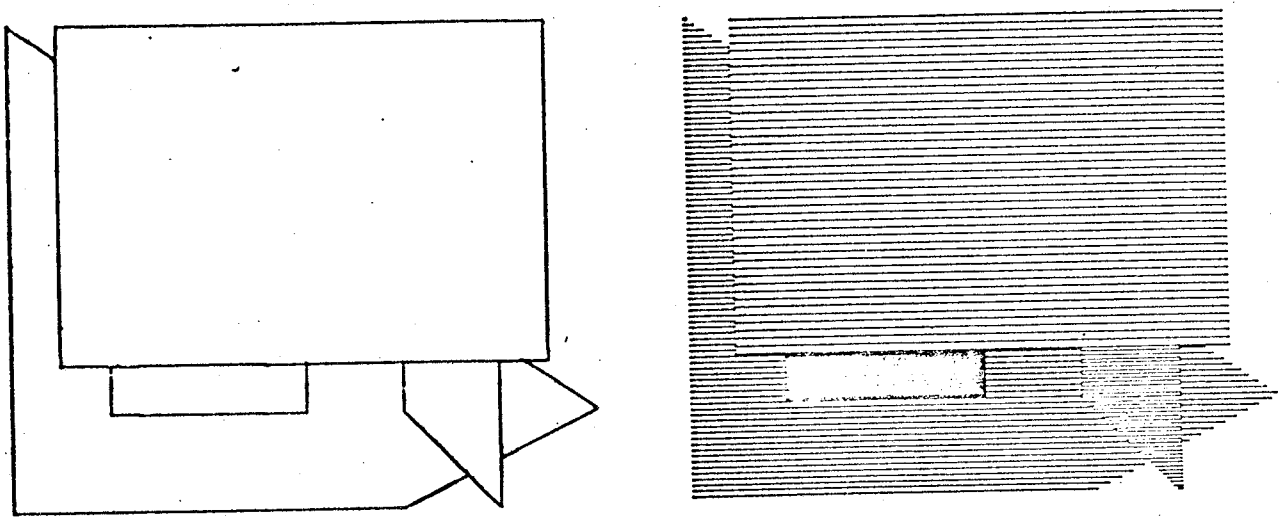


Figure I.6 Exemple de remplissage de taches à l'aide de hachures

Si l'on souhaite maintenant ne représenter que les contours, afin par exemple d'alléger la charge de l'écran, le problème dépend alors de la description de l'image elle-même. Si l'on ne dispose que de la description point par point, il faut rechercher les contours des taches formant l'image. Ce problème est résolu par les techniques de traitement d'images que nous n'abordons pas ici. Si, par contre, on dispose d'une description à base de points et de traits, le problème est alors trivial, puisque l'on est ramené au cas traité en I.213.

I.215 - Les mémoires d'entretien

L'image produite sur un écran graphique est fugitive, par nature, ce qui permet de la modifier. Ceci implique qu'un dispositif quelconque permette d'assurer le balayage constant de l'image, de manière à la laisser affichée le temps voulu. Ce dispositif est appelé *mémoire d'entretien*, et se présente sous des formes extrêmement différentes.

Dans le cas des tubes à rayon cathodique du type oscilloscope, la mémoire d'entretien est une mémoire vive qui contient la liste d'affichage. Le processeur graphique puise dans cette mémoire d'entretien les ordres qui lui sont nécessaires pour assurer la permanence de l'image. Cette technique est à rapprocher de celle utilisée pour les terminaux de télévision, où la mémoire d'entretien est une mémoire à accès direct (mémoire vive, disque) mais où l'information est la définition point par point de l'image. On a donc ici une description de bas niveau. Les autres types de terminaux possèdent des mémoires directement liées à l'écran, cette technologie étant moins coûteuse mais ne permettant plus de retrouver directement des renseignements concernant l'image au niveau de la console de visualisation. Le tableau I.2 résume ce paragraphe :

mémoires en	modèle au trait	modèle point par point
lecture/écriture	oscilloscope	balayage télévision panneau plasma
écriture seule		tube mémoire laser

Tableau I.2 - Classification des mémoires d'entretien

Ce tableau fait ressortir l'intérêt des consoles de type oscilloscope. Le fait que la structure même de l'image au trait soit conservée au niveau de la console va faciliter les opérations de modification de la liste de visualisation et d'identification, par simple modification ou consultation du contenu de la mémoire d'entretien. Ce fait explique que ces terminaux soient rangés dans la catégorie des terminaux hautement interactifs, car les temps de réponse à ces deux types de traitement seront plus courts que pour les autres terminaux. En effet, l'absence de mémoire structurée sera compensée par une simulation au niveau logiciel, simulation coûteuse en place mémoire et temps de calcul.

1.22 La technologie du dialogue

Les consoles de visualisation sont munies de dispositifs de communication visant à permettre la mise en oeuvre de techniques interactives. Nous étudierons ici à grands traits les caractéristiques des principaux matériels employés et leur adéquation aux fonctions de base que nous avons décrites en I.12.

I.221 - Les dispositifs de communication

Il en existe une grande variété et nous nous contenterons de décrire les plus utilisés à l'heure actuelle.

La première catégorie est constituée par l'ensemble des dispositifs à touches. On range sous cette dénomination les appareils dont le fonctionnement (du point de vue du programmeur) se résume aux aspects suivants :

- envoi d'une interruption, ou mise à jour d'un indicateur,
- association d'un code d'identification du dispositif et de la touche enfoncée. Ce code est généralement un entier.

Le dispositif le plus répandu est le clavier alphanumérique, semblable à un clavier de machine à écrire. Il permet d'envoyer des messages au calculateur, sous forme de suites de caractères, suites généralement automatiquement affichées sur l'écran à l'endroit désigné

par un curseur. On trouve également dans cette catégorie les claviers de fonctions, simples boitiers avec une série de touches, dont l'enfoncement provoque une interruption. Parfois, ces touches peuvent être allumées par programme, facilitant la manipulation par l'opérateur.

La deuxième catégorie comprend les dispositifs de désignation liés à l'écran. Le plus répandu et le plus populaire est le photostyle, simple cellule photo-électrique permettant de repérer une tache lumineuse sur l'écran. Cette détection provoque une interruption à laquelle sont associés plusieurs renseignements tels que le couple (x,y) du point "vu" par le photostyle, le code du caractère désigné, l'adresse en mémoire d'entretien de l'instruction graphique en cours de décodage. Le photostyle est très lié à la technologie "oscilloscope". Un autre dispositif moins répandu permet de désigner du doigt certaines portions de l'écran. Ce dispositif est d'une précision très grossière. Enfin, nous rangerons dans cette catégorie les réticules, constitués par deux filaments se croisant sur l'écran et permettant de désigner un point. Les seuls renseignements transmis sont les coordonnées de ce point.

La troisième catégorie de dispositifs comprend les tablettes graphiques. Elles se présentent sous la forme de plaques posées horizontalement. Leur technologie permet de recueillir des signaux représentant la position d'un stylet par rapport aux bords de la tablette. Ces signaux , émis à intervalles réguliers permettent d'assurer le suivi des déplacements du stylet, et donc d'acquérir une suite de coordonnées en continu. Les dessins ainsi obtenus sont de bonne qualité, la précision des tablettes étant tout à fait acceptable. L'usage des tablettes se répand de plus en plus, car elles sont extrêmement intéressantes pour toute les activités de dessin assisté par ordinateur.

Une dernière catégorie d'appareils comprend des dispositifs analogiques permettant de donner non seulement des valeurs différentes aux paramètres qui leur sont attachés, mais également de tenir compte de l'amplitude de la variation. Par exemple, les déplacements d'un symbole asservi à un tel dispositif seront d'autant plus rapides que l'amplitude de variation sera plus forte. Nous rangeons dans cette catégorie les boules roulantes, souris, manches à balais et autres potentiomètres.

En résumé, la variété des dispositifs est grande, les informations fournies sont de bas niveau et ne sont récupérées qu'au prix d'une savante gestion des interruptions. De plus, les consoles de visualisation ne sont pas forcément équipées de l'ensemble des dispositifs de communication, et ne comportent souvent qu'un seul dispositif (par exemple le clavier alphanumérique). Nous étudions au paragraphe suivant la possibilité de simuler les quatre fonctions du dialogue exposées en I.12, quel que soit le dispositif utilisé.

I.222 - Adéquation des dispositifs de communication aux fonctions de dialogue

Nous commencerons cette étude à travers le tableau I.3, qui donne les principales techniques permettant d'utiliser chaque dispositif pour réaliser les fonctions de dialogue. Les cases barrées indiquent des fonctions qu'il n'est pas raisonnable de simuler avec le dispositif considéré.

	identification	menu	valeurs	coordonnées
photostyle	désignation directe sur l'écran du composant à identifier	désignation directe sur l'écran de la commande	entrée carac. par caractère à partir d'une zone alphabet	a) point pa point b) en conti par poursuit
clavier alphanumérique	-donnée du nom de l'élément -guidage d'un symbole	frappe du nom de la commande	frappe de la chaîne de caractères	frappe des suites de coordonnées
clavier de fonctions	guidage d'un symbole par touches directionnelles	appui sur une touche		
tablettes	guidage d'un symbole	a) guidage d'un symbole b) découpage en zônes c) reconnaisseur de symboles	-découpage en zônes -reconnaisseur de caractères	a) point pa point b) en conti
réticule	désignation directe	désignation directe		point par point
manche à balai	guidage d'un symbole	guidage d'un symbole		a) point pa point b) en conti

Tableau I.3 : adéquation des dispositifs de communication aux fonctions de dialogue

Ce tableau montre clairement que tout dispositif n'est pas utilisable pour réaliser l'ensemble des fonctions. Il faut remarquer de plus que des facteurs ergonomiques viennent compléter ce tableau. Par exemple, alors que le photostyle semble d'un maniement plus simple que la boule roulante ou le manche à balai, ces derniers dispositifs lui sont préférés pour deux raisons :

- la position de travail est moins fatiguante, puisque la main repose sur la table de travail,
- l'opérateur peut continuer à observer l'ensemble de l'écran sans qu'une partie soit masquée par la main (fondamental par exemple dans le cas du contrôle aérien).

Le tableau I.4 résume la situation en ce qui concerne le facteur ergonomique, en fonction de différents procédés évoqués ci-dessus.

	identification	menu	valeurs	coordonnées
photostyle	excellent mais portion écran cachée	excellent mais portion écran cachée		a) bon, mais peu précis b) très imprécis
clavier alphanumérique	peu utilisé	peut être fastidieux	moyen habituel donc excellent	fastidieux
clavier de fonctions		excellent		
tablettes	excellent	a) excellent b) bon mais un peu rigide c) excellent le meilleur	a) plus que fastidieux b) limité à l'heure actuelle	a) excellent b) excellent
réticule	excellent	excellent		fastidieux
manche à balai	excellent	excellent		a) excellent b) peu précis

Tableau I.4 : Utilisation des dispositifs de communication : facteur ergonomique

Avant de tirer des conclusions définitives, il faut étudier un dernier facteur, qui est le coût du logiciel associé à ces fonctions. En effet, réaliser une identification sur une console de type

oscilloscope revient à décoder une interruption et à parcourir une table de corrélation, alors que sur les autres types de terminaux, l'identification suppose que l'on ait en mémoire du calculateur une pseudo-liste de visualisation qui permet, par un calcul complet de toute l'image, de trouver l'objet désigné. Il est clair que plus l'image (ou le dessin) est chargée, plus l'opération est coûteuse. Nous étudions dans le tableau I.5 le facteur coût :

	identification	menu	valeurs	coordonnées
photostyle	faible	faible		a) faible b) coûteux temps
clavier alphanumérique	faible	faible	faible (codage valeur)	faible (codage rée
clavier de fonctions		faible		
tablette	très coûteux	a) faible b) faible c) très coûteux	a) faible b) très coûteux	a) faible b) coûteux temps
réticule	très coûteux	faible		faible
manche à balai	très coûteux	faible		a) faible b) coûteux temps

Tableau I.5 : Utilisation des dispositifs de communication : coût logiciel

L'examen de ces trois tableaux montre qu'il n'y a pas de dispositif qui à la fois soit simple à mettre en oeuvre, donne des résultats satisfaisants pour les quatre types de fonctions et soit peu coûteux du point de vue du logiciel. Cependant, des indications peuvent être tirées quand au compromis ou à l'équilibre suffisant pour réaliser une configuration fortement interactive :

- on peut se contenter d'une tablette, les quatre fonctions pouvant être simulées de façon raisonnable, bien que grâce à des techniques logicielles coûteuses.

- une bonne association est celle du clavier alphanumérique et d'une tablette. Les quatre fonctions peuvent se réaliser de façon très agréable pour l'opérateur.

- pour les applications à base de désignations ou de menus exigeant un temps de réponse très rapide (l'image étant peu chargée), le manche à balai convient très bien, même tout seul.

Une dernière remarque avant de conclure : il est important de noter que certains dispositifs peuvent permettre de simuler une fonction de dialogue, même dans des conditions peu satisfaisantes. En effet, il peut arriver que l'on ait intérêt, par exemple, à entrer une suite de coordonnées à l'aide d'un clavier alphanumérique, soit pour avoir une grande précision (donnée à l'avance), soit parce que le dispositif habituel est en panne. Dans ce dernier cas, l'opérateur préfèrera sacrifier momentanément son confort pour ne pas arrêter l'exploitation en cours.

La possibilité d'utiliser indifféremment les différents dispositifs de communication sera donc une qualité appréciée. De plus, l'utilisation de fonctions de dialogue permet d'accroître la portabilité du programme d'application, puisqu'elles pourront être simulées quelle que soit la configuration du poste de travail utilisé, même si les différentes installations ne disposent pas des mêmes dispositifs d'entrée ou de sortie. On dispose ainsi d'un moyen de résoudre une partie du problème de l'indépendance des programmes par rapport au matériel.

1.23 Conclusion

Nous avons donc constaté qu'aussi bien en ce qui concerne l'affichage que le dialogue il existe des technologies très différentes, a priori mieux adaptées pour telle ou telle fonction bien précise.

C'est ainsi qu'en ce qui concerne l'affichage, nous avons été conduits à distinguer entre dessins et images. Nous avons cependant montré (en donnant les algorithmes nécessaires), que l'on peut oublier la technologie de la surface de visualisation si l'on doit construire des représentations graphiques présentant une qualité moyenne, ou ne faisant pas appel aux meilleures possibilités d'un matériel donné.

De même, nous avons montré que la grande diversité des dispositifs de communication pouvait être oubliée en raisonnant en termes de fonctions de dialogue et en acceptant éventuellement d'utiliser un dispositif qui n'est pas le mieux qualifié pour aider à l'accomplissement de cette fonction.

La conclusion que nous voulons tirer est donc que l'on peut envisager la définition et la réalisation de logiciels graphiques généraux, relativement indépendants du matériel utilisé, permettant la mise en oeuvre d'applications qui ne soient pas ultra spécialisées.

1.3 LA STRUCTURE DES LOGICIELS GRAPHIQUES INTERACTIFS

1.31 Schéma général

Un logiciel graphique interactif est constitué par un ensemble de programmes destinés à être insérés dans un programme d'application qui sera exploité en mode dialogué à travers une communication graphique avec le calculateur. Pour réaliser son programme, le programmeur d'application doit décider quelles opérations d'entrées/sorties, parmi toutes celles qui seront nécessaires à la bonne marche des calculs, seront effectuées à l'aide d'une console de visualisation (cf. figure I.7).

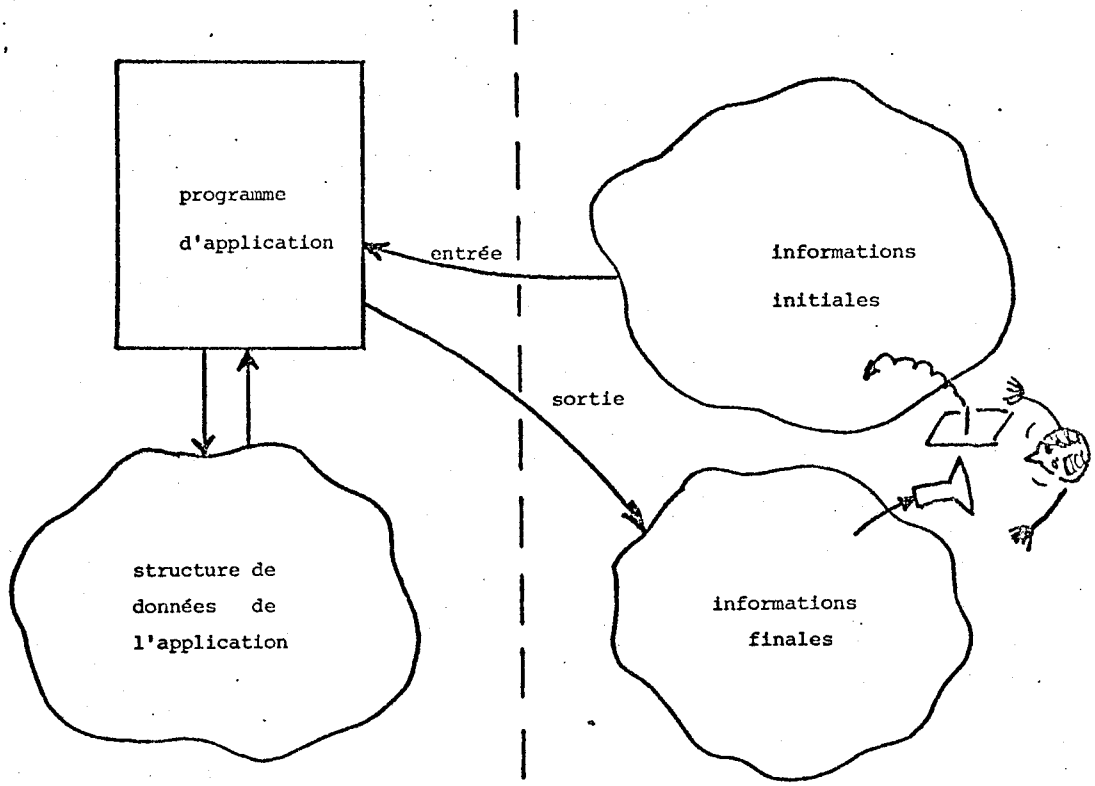


Figure I.7 Schéma général d'un programme d'application en mode dialogué

Nous avons vu en I.2 que la description finale d'un dessin est une liste de visualisation composée d'éléments de très bas niveau (listes de points et de segments de droite). Il est évident que la plupart des programmes d'application n'ont pas vocation de traiter de tels éléments. Les données qu'ils manipulent sont adaptées au type de calcul qu'ils ont à effectuer, mais n'ont a priori pas de raison spéciale de se trouver sous la forme de listes de points et de segments. Il faudra donc qu'à un moment donné l'opération de transcription des données en éléments

graphiques se réalise. Le schéma le plus général de la chaîne de programmes nécessaires comporte alors les éléments suivants : (cf. figure I.8) :

- un logiciel de *description* qui permet, à partir des informations fournies par le programme d'application, de coder les éléments qui permettront d'obtenir une représentation graphique. Le code utilisé est en général à base d'éléments géométriques, mais peut être entièrement indépendant du dessin proprement dit. Nous appellerons *scène* le résultat d'un tel codage.

- un logiciel de *préparation à la visualisation* qui, à partir de la scène codée, compose une scène bidimensionnelle sur une surface de visualisation fictive. Le code utilisé est un code à base de points et de segments de droite, qui peut également être totalement indépendant du code utilisé pour produire le dessin.

- un logiciel *élémentaire*, qui permet d'obtenir le dessin (l'image) final à partir des indications fournies par le logiciel de préparation à la visualisation. Le logiciel élémentaire a donc pour rôle d'établir la liste de visualisation réelle et d'assurer la gestion de l'écran.

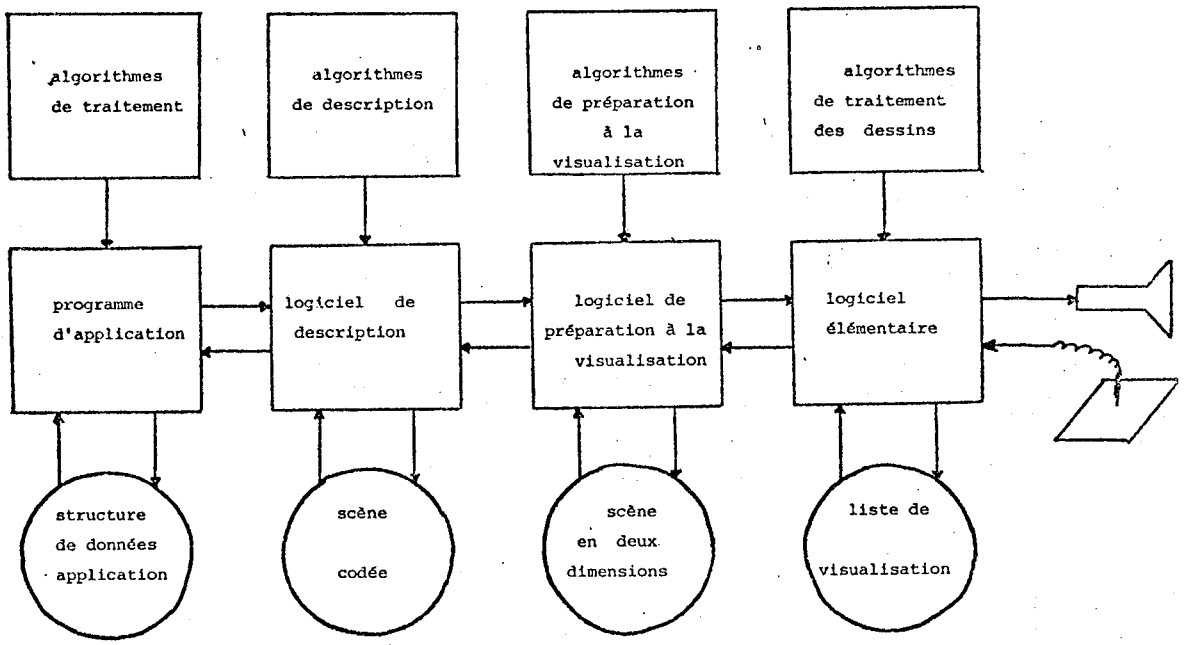


Figure I.8 Schéma général d'un logiciel graphique interactif

Chacun de ces logiciels peut être à son tour divisé en trois parties :

- une bibliothèque d'algorithmes,
- une structure de données,
- un moniteur d'enchaînement (ou interpréteur).

Les bibliothèques d'algorithmes constituent en fait l'éventail des possibilités offertes. Par exemple, un logiciel de description permettra d'utiliser un ou plusieurs codes différents, un logiciel de préparation à la visualisation saura traduire un ou plusieurs types de codes, un logiciel élémentaire permettra d'utiliser un ou plusieurs types de matériels.

Les structures de données peuvent être spécifiques de chaque logiciel, et les techniques de représentation utilisées être extrèmement différentes (listes, anneaux, relations etc...). La présence de ces différentes structures est un des problèmes de fond des techniques graphiques interactives, problème non résolu à l'heure actuelle. La question principale est de savoir comment, à partir d'une information affichée, on peut retrouver le niveau représenté. On voit en effet que lorsque l'on désigne (par exemple) un élément de dessin (d'image), on peut vouloir s'adresser à l'une des quatre structures de données (quitte à ce que l'opération entreprise ait éventuellement un effet sur les autres). Il faudra donc disposer d'un mécanisme permettant d'identifier un élément sur l'écran et de lui associer son homologue dans la structure de données concernée. Cependant, ce problème est considérablement compliqué par le fait que deux types de relations sont présentes dans les structures utilisées :

- les relations logiques, issues de l'application (donc du plus haut niveau), et qui doivent bien entendu être répercutées jusqu'au niveau le plus bas,

- les relations de description, qui visent simplement à faciliter la description des scènes ou des dessins (images) par l'utilisation de composants modèles que l'on reproduit à une transformation géométrique près. C'est ainsi qu'une entité manipulée au niveau du logiciel

d'application peut être représentée par une scène décrite à l'aide d'un catalogue d'objets, chacun de ces objets étant à son tour représenté graphiquement par la composition de sous-dessins élémentaires.

Le problème sera donc de pouvoir donner un moyen d'identification (*nom*) permettant de savoir à quelle structure s'adresse l'opération entreprise, afin d'éviter, dans la mesure du possible, de procéder à une analyse fastidieuse en partant systématiquement de l'application. Un autre problème sera celui de la cohérence des renseignements fournis : si un relevé de points au niveau élémentaire se traduit par une liste de coordonnées, cette même liste peut avoir une signification logique qui est la seule présentant un intérêt au niveau le plus haut. Les différentes couches de logiciel auront donc pour rôle de filtrer les informations transmises et de les restituer dans la forme utilisable soit au niveau inférieur (affichage), soit au niveau supérieur (dialogue).

La partie moniteur (interpréteur) de chacun de ces logiciels est en fait la partie visible aux yeux du programmeur (d'où l'appellation abusive de cette seule partie par le nom générique, tel que nous l'avons fait dans la figure I.8). Chaque moniteur reçoit une suite de commandes visant à utiliser les algorithmes de sa bibliothèque. Le programmeur dispose du répertoire des commandes utilisables, ainsi que des paramètres qui leur sont attachés. Nous nommerons par la suite *primitive* la donnée d'une commande et des paramètres qui lui sont liés. Les données nécessaires à la bonne exécution d'une commande sont soit transmises par le logiciel appelant, soit dans la structure de données afférente au logiciel appelé.

La séparation entre les différents niveaux de logiciels n'est pas toujours aussi nette, soit que certains niveaux manquent, soit que les primitives soient mélangées. Les situations les plus fréquemment rencontrées sont les suivantes :

- absence du niveau de description, spécialement pour les applications purement graphiques, qui ont pour objet de manipuler des dessins dans le plan,

- mise en commun d'algorithmes situés à des niveaux différents. L'exemple le plus typique est celui des transformations géométriques, qui peuvent servir aussi bien pour la description de la scène à représenter que pour la mise en page finale sur l'écran. La tentation est alors grande, pour des raisons d'efficacité, de confondre le niveau de description et le niveau de dessin,

- mise en commun de structures de données, soit pour éviter la duplication d'informations, soit pour tout regrouper dans une "structure de données graphique".

Il nous paraît cependant très important de mettre en évidence le rôle joué par chacun de ces logiciels. En effet, le développement de programmes d'application utilisant les techniques graphiques interactives implique à un moment donné de choisir entre :

- développer un logiciel entièrement autonome, comportant les trois niveaux mentionnés, avec toutes les simplifications possibles,

- développer des couches de logiciel permettant de ne pas réécrire à chaque application les programmes élémentaires, mais comportant dès lors des interfaces obligatoires conduisant à d'éventuelles duplications.

Nous étudions dans les paragraphes suivants les caractéristiques principales des trois niveaux de logiciel que nous avons mentionné.

1.32 Les logiciels de description

Le rôle des logiciels de description est d'extraire de la structure de données de l'application les éléments nécessaires à la codification d'une scène dont on obtiendra plus tard une représentation graphique. Le choix du code dépend en général de l'application, soit parce qu'il est très simple d'extraire les éléments à coder des informations

telles qu'elles sont traitées, soit parce que le code choisi se prête particulièrement bien à certains traitements ultérieurs, soit encore pour de simples raisons d'encombrement de la mémoire : il est moins coûteux de conserver l'image d'un cercle sous la forme (coordonnées du centre, rayon) plutôt que sous la forme de la centaine de points éventuellement nécessaires à l'obtention du dessin sur l'écran. De nombreux codes ont été utilisés, les plus couramment utilisés étant les codes suivants :

- codes utilisant des directions élémentaires et réduisant la description d'une scène à des chaînes de directions élémentaires sur des distances plus ou moins longues,

- codes syntaxiques, faisant intervenir un vocabulaire de base (tracés élémentaires) et des règles de composition (grammaires).

- codes géométriques, utilisant des figures de géométrie élémentaires et les combinant par le biais de transformations géométriques. Les objets les plus utilisés sont les polyèdres (assemblages de facettes ou solides les surfaces gauches approchées par des morceaux de quadriques, les surfaces gauches approchées par des fonctions polynomiales.

Le lecteur intéressé trouvera en annexe 1 une étude détaillée de quelques codes. Nous montrerons au chapitre 4 l'extrême diversité des logiciels de description. Ceci nous conduit à prévoir que l'on ne pourra pas définir un logiciel de description universel, adaptable à toute application. Par contre, un des axes de recherche que l'on pourrait envisager est l'étude de classes d'applications utilisant des logiciels de description ayant des bases communes. (Par exemple, logiciels adaptés à la cartographie, logiciels adaptés à la mise en forme de tableaux (courbes, histogrammes etc...), logiciels de construction de polyèdres).

1.33 Les logiciels de préparation à la visualisation

Le rôle des logiciels de préparation à la visualisation est de créer une ébauche du dessin (ou de l'image) à partir de la scène codée Le produit du logiciel de visualisation est une scène bidimensionnelle composée de points et de segments de droite (éventuellement de textes).

On distingue entre logiciels de visualisation de scènes à deux dimensions et logiciels de visualisation de scènes à trois dimensions.

Les logiciels de visualisation de scènes bidimensionnelles partent d'une scène plane pour créer une autre scène sur une surface de visualisation fictive, caractérisée par un système de coordonnées choisi par le programmeur en fonction de l'application traitée. Les primitives de base de ces logiciels se répartissent en deux groupes :

- les primitives de *décodage*, qui permettent de passer des primitives de description à des ensembles de points, segments et caractères. De nombreuses techniques ont en particulier été développées pour la génération de coniques et courbes diverses.

- les primitives de *mise en page*, qui permettent de disposer les ensembles créés précédemment sur la surface de visualisation fictive. Ces primitives comportent essentiellement la définition de sous-espaces de visualisation (fenêtres) et les transformations géométriques habituelles (homothétie, rotation, translation).

Les logiciels de visualisation de scènes tridimensionnelles ont pour rôle essentiel de transformer ces scènes en scènes bidimensionnelles donnant une représentation plus ou moins réaliste visant à restituer une notion de profondeur. Nous étudierons quelques techniques de restitution de polyèdres au chapitre 3. Nous remarquerons qu'en dernier ressort ce sont les primitives de visualisation à deux dimensions qui seront utilisées.

Il est bien évident que la partie décodage des logiciels de visualisation est très dépendante du code utilisé par le logiciel de description. La diversité des codes employés interdit de songer à une normalisation de ces modèles. Par contre, l'interface avec les logiciels élémentaires peut être défini sans ambiguïté puisque les éléments produits sont tous identiques : points, segments et caractères.

1.34 Les logiciels élémentaires

Le rôle de ces logiciels est a priori relativement simple : inscrire la scène bidimensionnelle fournie par le logiciel de visualisation sur la surface de l'écran. Il s'agit donc d'un simple transcodage, puisque la description initiale est déjà faite en termes de primitives élémentaires correspondant aux éléments de base des terminaux graphiques. La principale qualité d'un logiciel élémentaire sera de fournir une certaine indépendance par rapport au matériel utilisé. Nous avons vu en I.2 la grande diversité technologique aussi bien du point de vue de l'affichage que du point de vue du dialogue.

Différentes techniques ont été mises en oeuvre pour assurer cette indépendance. Nous ne passerons pas ici en revue les différents problèmes rencontrés, renvoyant le lecteur aux différentes publications existant sur le sujet (voir [LeL-]).

Nous rappellerons simplement deux points essentiels :

- pour assurer correctement les fonctions interactives, il est nécessaire de disposer à un moment donné d'un modèle sous forme d'une liste de visualisation de dessin (et non d'image). De plus, il est nécessaire de disposer de tables de corrélation liant les éléments du dessin aux noms fournis par le programmeur pour identifier certaines parties de la représentation graphique. Le logiciel élémentaire simulera donc éventuellement la liste d'affichage et devra gérer l'ensemble des noms associés aux éléments du dessin.

- deux grands types de techniques sont utilisées pour résoudre le problème de l'indépendance vis-vis du matériel (voir [Led 77]) :

  . L'utilisation d'une console virtuelle, qui revient à simuler en amont de chaque console une console fictive, des interpréteurs permettant de passer aux consoles réelles (cf. figure I.9),

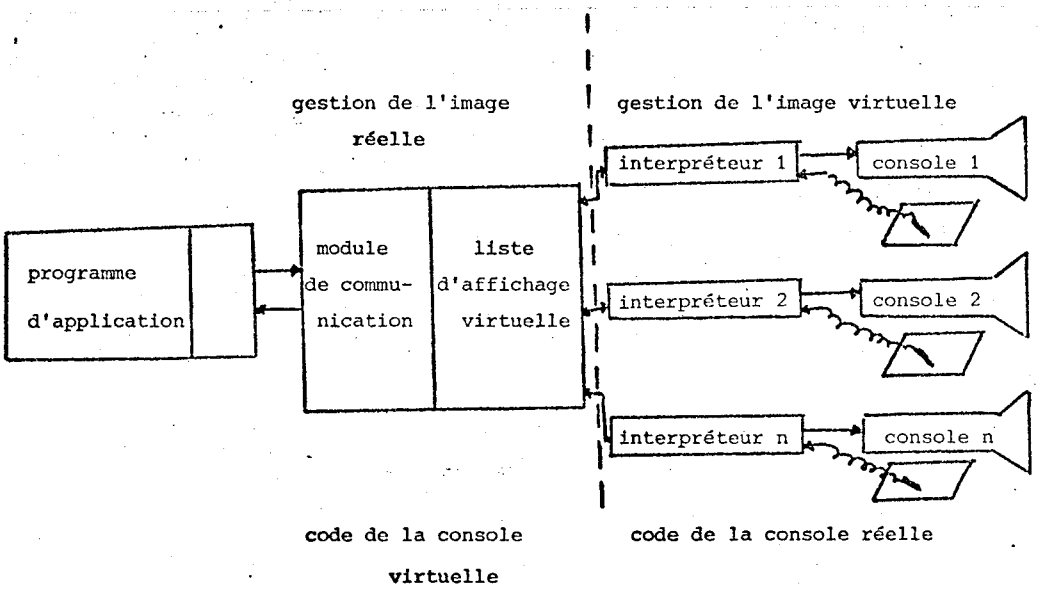


Figure I.9 Schéma d'un logiciel fondé sur la notion de console virtuelle

. L'utilisation de logiciels adaptatifs, qui visent à donner plus d'autonomie aux différents interpréteurs en leur laissant une charge de travail plus grande (cf. figure I.10). Ce deuxième type de logiciel, bien que plus difficile à mettre en oeuvre permet une meilleure utilisation de chaque type de matériel.

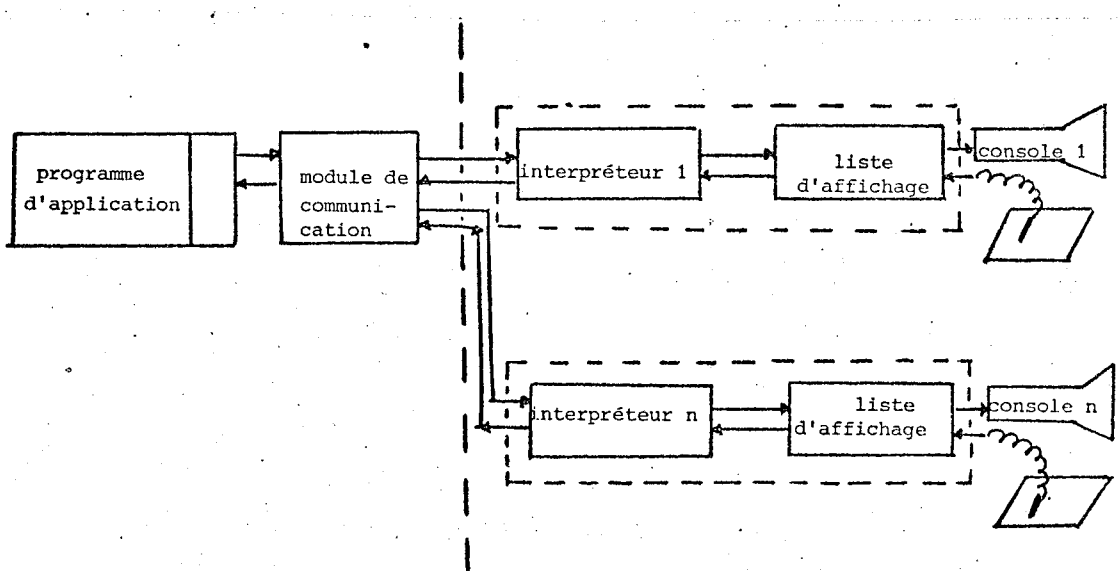


Figure I.10 · Schéma d'un logiciel fondé sur la notion d'adaptation

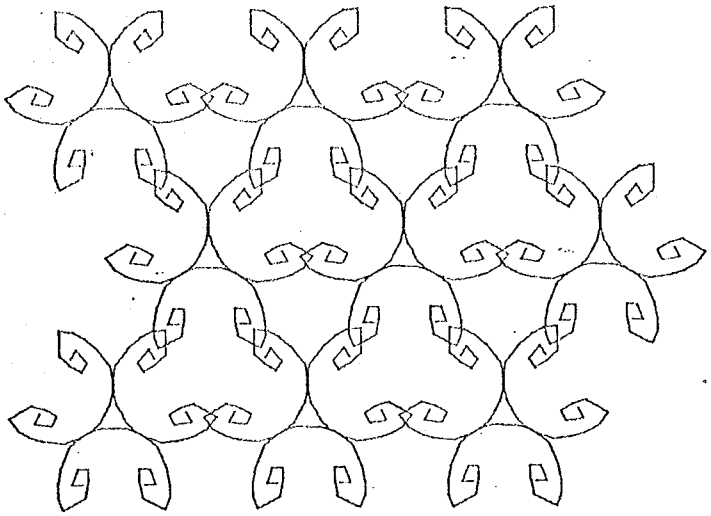
1.4 CONCLUSION

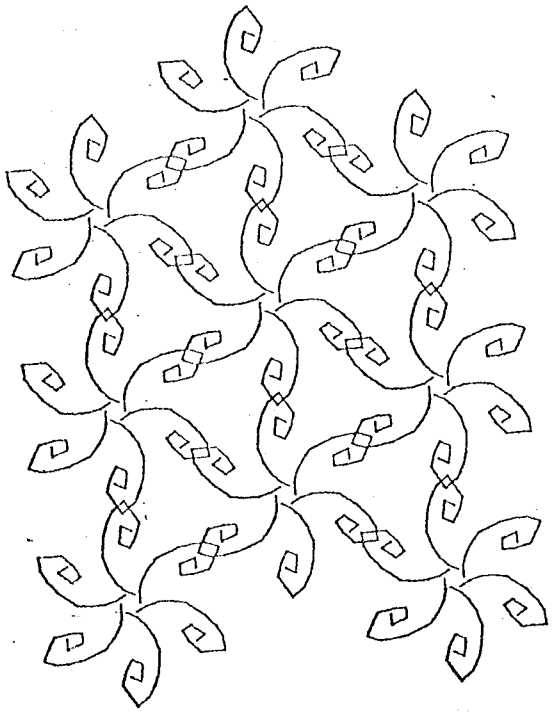
Nous souhaitons souligner pour conclure l'importance des techniques logicielles associées à l'infographie interactive. Les différents étages que nous avons signalé forment autant de couches de programme qu'il faut traverser pour, partant de l'information brute, aboutir à la représentation graphique. Le problème du temps de réponse étant psychologiquement très important, il n'est pas étonnant que toutes les combinaisons aient été essayées pour réduire les temps de calcul. Ceci s'est fait en particulier au détriment de la séparation des niveaux de logiciel, et il n'est pas rare de rencontrer des logiciels qui mélangent les différents niveaux. En particulier, beaucoup de logiciels confondent la description et le tracé, ce qui conduit à des difficultés importantes. Nous en verrons un exemple au chapitre 2.

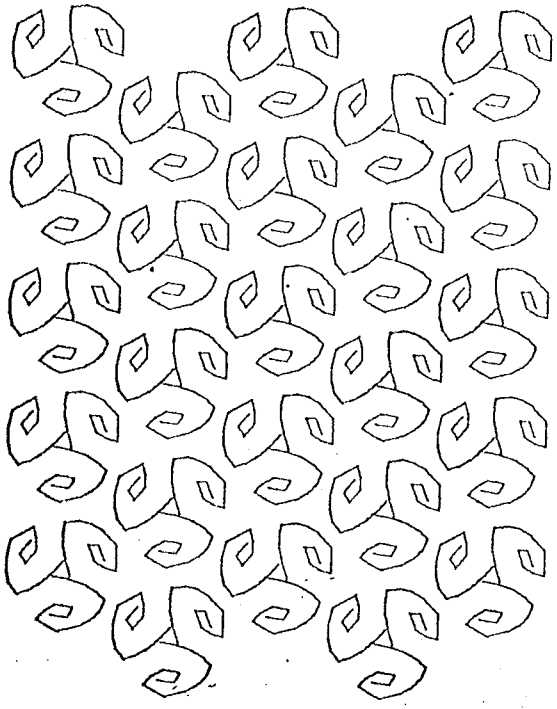
Cependant, certaines de ces confusions sont inévitables, et même bénéfiques. En particulier, la séparation nette entre logiciel élémentaire et logiciel de visualisation pourrait être extrèmement pénalisante. L'exemple de la mise en page du dessin définitif est relativement parlant : plutôt que de laisser faire cette mise en page au plus haut niveau, on peut la faire au niveau de l'écran lui-même. Etant donné que l'écran a son propre système de coordonnées, qui devrait être inconnu du niveau supérieur (condition minimale d'indépendance par rapport au matériel !), les primitives de mise en page seront alors ramenées au niveau du logiciel élémentaire (on pourrait dire que le logiciel de préparation à la visualisation enregistre ces commandes, mais ne les exécute pas, laissant le soin de le faire au niveau inférieur). On voit donc que le logiciel qui permettra d'obtenir réellement le dessin sur l'écran sera une combinaison du logiciel élémentaire et des primitives de mise en page du logiciel de préparation à la visualisation. Nous nommerons *logiciel graphique interactif de base* ("core") un tel logiciel et étudierons les problèmes de conception au chapitre 2. Un exemple de logiciel interactif de visualisation de polyèdres sera étudié au chapitre 3. Divers logiciels de description seront étudiés au chapitre 4 à travers la présentation de quelques applications.

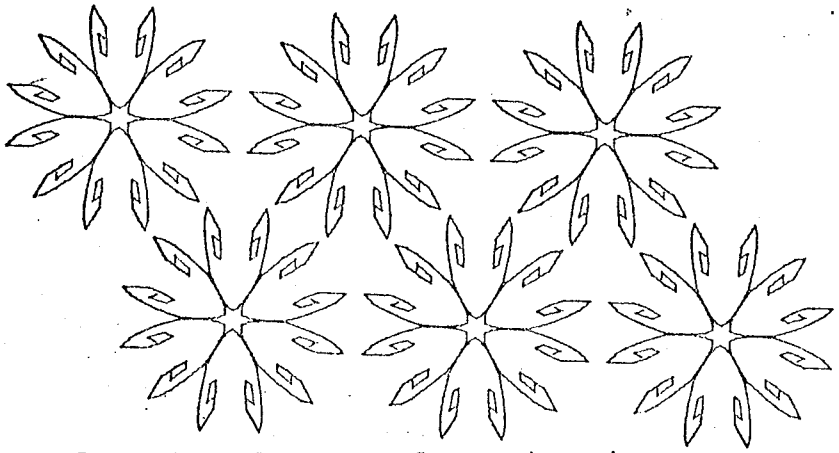
CHAPITRE 2

ELEMENTS POUR LA CONCEPTION DE LOGICIELS GRAPHIQUES INTERACTIFS DE BASE









transformations du groupe des tapisseries

2. ELEMENTS POUR LA CONCEPTION DE LOGICIELS GRAPHIQUES INTERACTIFS DE BASE

2.1 ETUDE DE QUELQUES LOGICIELS GRAPHIQUES INTERACTIFS

Nous nous intéressons dans ce chapitre aux problèmes posés par la conception de logiciels graphiques interactifs de base. Il s'agit essentiellement de discuter le point de vue du programmeur d'application qui utilisera les primitives fournies par le réalisateur du logiciel. Nous discuterons l'aspect sémantique des primitives, mais pas l'aspect syntaxique (abordé dans [MoL 76]). Nous étudierons les caractéristiques de la plupart des logiciels graphiques actuels à travers la comparaison de quatre d'entre eux :

- GSP (Graphic Subroutine Package), ensemble de sous-programmes conçu pour la programmation des consoles de visualisation IBM 2250 [IBM -]. Les langages de programmation les plus couramment utilisés sont FORTRAN, ALGOLW, COBOL, PL/1, LISP et Assembleur 360.

- GRAF, extension graphique de FORTRAN destinée à remplacer GSP [HCY 67],

- un ensemble défini par NEWMAN et SPROULL [NeS 74] et qui sert de modèle à de nombreuses réalisations actuelles,

- une extension graphique de BASIC proposée par LUEHRMAN et al. [Lue 74].

Ces quatre logiciels ont été choisis comme représentatifs de la plupart des logiciels distribués actuellement, les variantes portant sur le nombre de primitives, leur dénomination, le mode d'insertion dans le langage de programmation.

Nous étudierons successivement les primitives se rapportant à l'affichage, à la structuration du dessin et au dialogue.

2.11 Les primitives d'affichage

Nous rangeons sous cette appellation l'ensemble des primitives permettant d'obtenir un dessin sur l'écran. On distingue :

- les primitives de *tracé* , qui permettent de définir la suite de traits, points ou caractères formant le dessin,

- les primitives de *mise en page* , permettant de définir les espaces de coordonnées utilisés,

- les primitives de *gestion de l'écran*, permettant de modifier le dessin présent sur l'écran par addition ou suppression de parties du dessin.

La caractéristique principale des primitives de tracé est l'utilisation d'un *point courant* représentant la position actuelle du dispositif de dessin. Cette notion a une origine historique, liée au fait que les premiers logiciels graphiques ont été développés pour des traceurs de courbes. La programmation de ces dispositifs fait intervenir le déplacement d'une plume de manière à obtenir le dessin voulu. Le programmeur écrit un algorithme de dessin, qui consiste à définir le chemin que doit suivre la plume (le faisceau). En particulier, il peut chercher à établir un algorithme de dessin assurant un temps de tracé minimum (ceci revient à chercher le plus court chemin pour parcourir toutes les arêtes du graphe du dessin). Ce fait conduit à deux conséquences importantes :

- on trouve des primitives permettant de guider le faisceau soit en position d'écriture, soit en position de déplacement sans écriture.

- on dispose en général de la possibilité de définir des sous-dessins, c'est-à-dire des portions définies en coordonnées relatives, la mise en place se faisant par rapport à la position du faisceau à un moment donné.

Les primitives de tracé permettent également d'obtenir des graphismes différents, du type point par point, ligne brisée ou suite de segments disjoints. La donnée de ces différents graphismes se traduit souvent par l'apparition de primitives spécialisées dans tel ou tel type de tracé.

Le principal problème lié à cette approche consiste à définir quelle est la position du point courant après une opération. Si nous prenons l'exemple du tracé de segments de droite, la suite d'instructions :

MOVE x_o, y_o
DRAW x_1, y_1

peut conduire aux résultats suivants :

- le faisceau trace le segment d'origine (x_o, y_o) et d'extrémité (x_1, y_1) et reste à la position (x_1, y_1)

ou

- le faisceau trace le segment d'origine (x_o, y_o) et d'extrémité (x_1, y_1) puis revient en position (x_o, y_o).

La première solution est extrèmement intéressante pour dessiner une suite de segments jointifs (ligne brisée), la seconde pour dessiner des segments en étoile (cas fréquent de certains graphiques), Le point remarquable est que le choix de l'une des solutions rend la programmation de l'autre extrèmement lourde, puisque l'on sera contraint après chaque tracé d'utiliser une primitive *MOVE* pour ramener le faisceau en bonne position de départ.

Une autre source de problèmes vient de la possibilité de découper la scène graphique à l'aide de fenêtres. Il peut arriver alors que des portions de dessin soient situées en dehors de la fenêtre et ne soient pas représentées sur l'écran. Il est alors clair que la position courante du faisceau ne correspond à rien, sinon à la position d'un point virtuel, se déplaçant dans l'espace utilisateur. Ce point courant est donc un outil de description de scène, et il y a confusion entre les fonctions du logiciel de description et celles du logiciel graphique de base telles que nous les avons définies en 1.4.

En ce qui concerne la gestion de l'écran, on trouve des primitives permettant d'effacer tout ou partie du dessin. Les notions intéressantes sont les suivantes :

- différenciation dans certains logiciels entre effacement de l'écran et destruction de la liste de visualisation (pour permettre par exemple des effets de clignotement en alternant effacements et affichages sans avoir à reconstruire la liste à chaque pas),

- la possibilité de choisir entre un affichage immédiat (le dessin correspondant à la primitive appelée est visualisé aussitôt) et un affichage différé (le dessin correspondant à la (aux) primitive appelée n'est affiché que sur commande, l'évocation de la primitive de tracé ne donnant lieu qu'à constitution d'une pseudo-liste de visualisation).

Le tableau 2.1 donne un exemple de primitives d'affichage utilisée dans trois logiciels graphiques.

La remarque essentielle portera sur deux caractéristiques du logiciel proposé par NEWMAN et SPROULL :

- généralisation des primitives *MOVE* et *DRAW* au "tracé" en trois dimensions,
- inclusion de l'ensemble des transformations géométriques élémentaires, aussi bien à deux qu'à trois dimensions.

La confusion entre logiciel de description et logiciel de base est ici de plus en plus évidente.

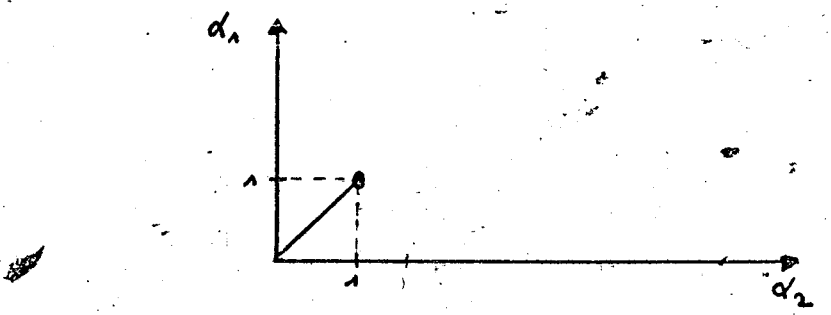
	GSP	GRAF	NEWMAN et SPROULI
tracé position du faisceau	*STPOS(nfig,x,y,correl)* *MVPOS(nfig,dx,dy,correl)*	*PLACE(x,y)*	*MOVETO(x,y,[z])* *MOVE(dx,dy,[dz])*
points	*PPNT(nfig,xtab,ytab,correl,null, null,nb)*	*POINT(x,y)*	
segments	*PSGMT(nfig,xtab,ytab,correl,null, null, nb)* *PLINE(nfig,xtab,ytab,correl,null, null,nb)*	*LINE(x,y)*	*LINETO(x,y,[z])* *LINE(dx,dy,[dz])*
textes	*PTEXT(nfig,texte,nb,correl,null, null,null, x, y)*	*CHAR(texte,nb,mode)*	*DISPLAY(texte)*
mise en page	*SDATL(nfig,x,y,x',y')* *SGDSL(nfig,x,y,x',y',x_1,y_1,x'_1,y'_1)*		*WINDOW(x,y,x',y'* *VIEWPORT(x,y,x',*
gestion écran	*RESET(,fig)* *EXEC(nfig)*	*A=A+B+...* *ERASE(A)* *RESET(A)* *BLANK* *PLOT(A)*	*DELETE(nfig)* *APPEND(nfig)* *CLEAR* *UPDATE*
structuration	*INGDS(unité,nfig)*	*DISPLAY nfig1,nfig2*	*OPEN(nfig)* *CLOSE(nfig)*
transformations			*SETMATRIX(A)* *SAVEMATRIX* *RESTOREMATRIX* *TRANSLATE(x,y,[z* *SCALE(sx,sy,[sz].* *ROTATE(rx,ry,[rz*
divers	*SDATM(nfig,modex,modey)* *SGRAM(nfig,mode)* *SCCIS(nfig,mode)*		

Tableau 2.1 : Primitives d'affichage

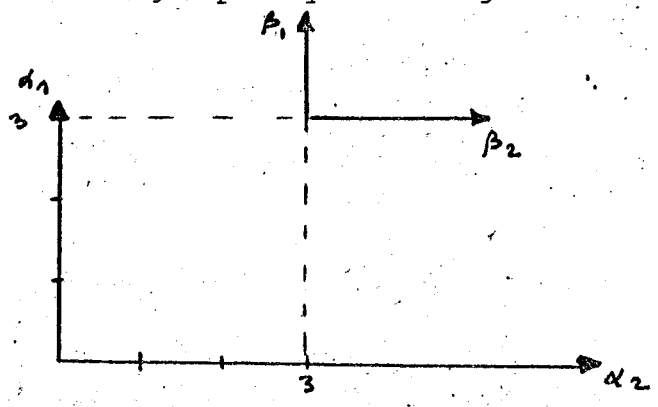
Pour montrer le danger de cette pratique, nous choisirons l'exemple donné dans [Gue 76].
Soit la suite d'instructions suivante :

INIT	début d'un dessin
MOVE(0,0)	mise en place du point courant
DRAW(1,1)	tracé d'un segment de droite
TRANS(3,3)	translation du système d'axes
DRAW(2,2)	tracé d'un segment de droite

qui implique un changement de repère au milieu d'instructions de tracé.
Après les trois première instructions, le point courant se trouve placé en (1,1) par rapport à un système de coordonnées α :



L'instruction *TRANS(3,3)* provoque un changement de repère, que nous nommerons β



La suite du dessin est conditionnée par l'interprétation choisie pour le nouvelle position du point courant. Cinq variantes sont fréquemment rencontrées :

1 - position indéfinie (implique obligatoirement un *MOVE* dans le nouveau système)

2 - automatiquement (0,0) origine du nouveau système,

3 - $(-\infty, -\infty)$ pour montrer qu'il y a une erreur (à rapprocher de la première solution),

4 - (1,1) le point courant bouge en suivant le nouveau système d'axes (coordonnées inchangées),

5 - (-2,-2) le point courant ne bouge pas, mais ses coordonnées sont exprimées dans le nouveau système.

On peut trouver des exemples de ces cinq réalisations différentes dans les logiciels actuels, la solution 4 étant classique dans tous les logiciels pour traceurs de courbe, la solution 5 étant plus utilisée pour les logiciels graphiques.

Nous étudions à part l'extension à BASIC [Lue 74] car elle repose sur deux principes la différenciant nettement des logiciels habituels :

- les primitives de description du mouvement du faisceau ne sont plus *MOVE* et *DRAW* (c'est-à-dire des déplacements plume haute ou plume basse), mais des opérations composées que nous étudierons ci-dessous,

- il est fait une utilisation intensive des règles habituelles de syntaxe de BASIC, en complète analogie avec les opérations d'entrée/sortie ordinaires.

Le premier point est fondé sur la remarque que les primitives *MOVE* et *DRAW* sont en fait difficiles à utiliser, en particulier lorsque l'on a à tracer une suite de segments à l'aide d'une boucle *pour*.

LUEHRMAN remarque que les primitives *MOVE* et *DRAW* correspondent en fait aux opérations suivantes :

MOVE x,y
. mettre la plume en position haute
. amener le faisceau en x,y

DRAW x,y
. mettre la plume en position basse
. amener le faisceau en x,y (un segment est donc tracé)

il propose de les remplacer par les primitives

LIFT
. mettre la plume en position haute

LINE x,y
. amener le faisceau en x,y
. mettre la plume en position basse (un segment sera tracé si la plume était en position basse).

avec la notation syntaxique suivante :

PLOT x,y ; équivalent de *LINE x,y*
PLOT équivalent de *LIFT*
PLOT x,y équivalent de { *PLOT x,y ;* / *PLOT*

On voit que *PLOT X,Y ;* permet de dessiner des lignes brisées, alors que *PLOT X,Y* permettra de dessiner des suites de segments disjoints. Cette notation est analogue à l'utilisation du ; dans les instructions d'écriture, où il permet d'écrire sur une même ligne une suite de valeurs.

L'exemple choisi par LUERHMAN est celui du tracé du graphe de sin(x). Il propose de comparer les trois écritures suivantes :

utilisation de *MOVE* et *DRAW*

```
10 LET X=0
20 MOVE X, SIN(X)
30 FOR X=.1 TO 6.28 STEP .1
40 DRAW X, SIN(X)
50 NEXT X
```

Ecriture sur console

```
30 FOR X=0 TO 6.28 STEP .1
40 PRINT X, SIN(X)
50 NEXT X
```

proposition

```
30 FOR X=0 TO 6.28 STEP .1
40 PLOT X, SIN(X)
50 NEXT X
```

L'extension proposée vise à rendre un programme graphique identique à un programme ordinaire. Les avantages attendus en sont une plus grande facilité d'apprentissage et un moindre risque de commettre des erreurs lors de l'écriture des programmes.

Dans le reste de son article, LUEHRMAN propose essentiellement des primitives de mise en page (*WINDOW, LIMITS*), d'affichage d'une suite de segments (*MAT PLOT*), d'affichage de texte (*PLOT label SIZE s ANGLE a FONT f*) et l'extension au tracé en trois dimensions par utilisation d'une troisième coordonnée.

Nous nous contenterons de constater pour l'instant la diversité qui règne au niveau du nombre de primitives, de leur syntaxe, de leur sémantique, alors qu'elles s'appuient apparemment sur le même concept de départ.

2.12 Les primitives de structuration

Nous avons vu en 1.21 que la liste de visualisation était composée d'un ensemble d'instructions de tracé et de données permettant de réaliser le dessin. En général, cette liste est munie d'une certaine structure, permettant de différencier des sous-ensembles qui seront traités globalement et individuellement. Les raisons de cette structuration sont multiples :

- nécessité de respecter au niveau du dessin un découpage correspondant aux éléments de la structure de données d'application qui sont représentés,

- facilité de description d'un dessin en utilisant des symboles (ou encore sous-dessins) que l'on peut répéter, à une transformation géométrique près,

- efficacité des opérations de modification d'un dessin, permettant de ne toucher qu'à une partie de la liste de visualisation au lieu de la recomposer entièrement.

Ce problème de la structuration du dessin est encore mal connu actuellement. Le débat porte sur les points suivants :

- quelle structure adopter ? la plus communément utilisée est la structure arborescente, mais elle ne convient pas à toutes les applications. De plus, la profondeur des arbres utilisés varie d'une application à l'autre, et conduit à une gestion du dessin très délicate.

- faut-il séparer la structuration correspondant à l'application de la structuration correspondant à la construction du dessin ? Cette séparation permet alors d'autoriser un emboîtement quelconque au niveau de la désignation des éléments du dessin, mais un nombre relativement limité de niveaux quant à la modification. L'expérience prouve qu'en général les niveaux de modification correspondent logiquement aux niveaux de dénomination ("naming").

Les logiciels généraux distribués actuellement proposent habituellement un niveau de dénomination (le dessin est décomposé en figures), parfois deux niveaux (chaque figure peut être à son tour décomposée en sous-figures) [Fol 76].

En ce qui concerne les primitives de structuration, on trouve deux grandes philosophies :

- le regroupement des primitives d'une figure se fait à l'aide d'un nom de figure, généralement figuré par un entier, qui relie les différents appels (voir GSP),

- le regroupement des primitives se fait par le biais d'une déclaration de début de figure (*OPEN*) et d'une fin de figure (*CLOSE)*. Toutes les primitives appelées entre ces deux fonctions appartiennent à la même figure (voir NEWMAN et SPROULL).

L'intérêt de la deuxième solution semble résider dans les deux points suivants :

- diminution du nombre de paramètres, puisque le nom de la figure n'est passé qu'une seule fois et non à chaque appel de primitive,

- présentation plus "structurée" du programme graphique.

Par contre, elle implique des contraintes très fortes, en particulier l'interdiction de construire simultanément deux figures (pas de nouveau *OPEN* tant que la figure en cours de construction n'est pas terminée). Cette contrainte ne semble pas du tout nécessaire, ce qui explique que la première solution soit également très fréquente.

2.13 Les primitives de dialogue

Si au niveau des primitives de tracé, il existe un semblant d'accord sur le concept de point courant au niveau des primitives de dialogue on rencontre les situations suivantes :

- pas de primitives de dialogue réellement adaptées au traitement graphiqu Par exemple, la seule primitive offerte dans l'extension à BASIC concerne la lecture d'une suite de coordonnées au clavier alphanumérique par l'instruction *PLOT INPUT*.

- primitives de dialogue fondées sur un traitement d'interruptions de bas niveau, le programmeur ayant à décoder les renseignements obtenus pour en déduire la fonction réalisée. Le tableau 2.2 présente les primitives de GSP.

CRATL (nomunité, niveau)	déclaration d'un niveau d'interruptio
ENATN(niveau, liste d'interruptions)	autorisation de certaines interruptio
DSATN(niveau, liste d'interruption)	interdiction de certaines interruptio
RQATN(niveau, réponse, synchrone, tableau, liste)	attente d'interruption
MLITS(niveau, mode, liste de touches)	allumage programmé des touches de fonction
SLPAT(nfig, mode)	autorisation de désignation d'une fig
LOCPN(nfig, x, y)	collecte d'un couple de coordonnées
ICURS(nfig, null, clé, position)	mise en place du curseur alphanumériq
RCURS(nfig)	suppression du curseur
GSPRD(nfig, adresse, nombre d'octets)	lecture d'une zône de la mémoire d'entretien

Tableau 2.2 - primitives de dialogue de GSP.

- primitives de dialogue de haut niveau, fondées sur la définition de dispositifs logiques. Le tableau 2.3 présente les primitives proposées par NEWMAN et SPROULL.

DRAG(nfig)	mise en place d'une figure
FIX(nfig)	
V←READPOSITION	collecte d'un couple de coordonnées
V ← INK	collecte d'une suite de couples de coordonnées
V ← HITDETECT(x, y, nfig)	identification
V ← RECOGNIZE	reconnaissance d'un symbole manuscrit

Tableau 2.3 : Primitives de dialogue proposées par Newman et Spro

En ce qui concerne GSP, il est clair que les primitives proposées permettent de programmer les fonctions de dialogue que nous avons proposées en 1.12. Cependant, l'expérience montre que cette programmation peut être délicate. Par exemple, la programmation d'une entrée de valeur alphanumérique implique la gestion de la mémoire d'entretien octet par octet. Les programmeurs d'application n'ont en général pas l'expérience nécessaire pour bien utiliser ces primitives. En particulier, l'écart entre le niveau des langages utilisés (FORTRAN par exemple) est trop grand.

Ces considérations ont conduit NEWMAN et SPROULL ainsi que d'autres auteurs ([FoW 76], [Luc 74b], [NeS 74]) à proposer des primitives de haut niveau, s'intégrant parfaitement aux langages de programmation utilisés et s'éloignant des dispositifs de communication réels. Cependant, si nous étudions la proposition NEWMAN et SPROULL, nous pouvons constater qu'elle n'est pas cohérente :

- absence de fonctions de menu et d'introduction de valeurs alpha-numériques,

- promotion au rang de fonctions de techniques interactives très particulières, telles que la reconnaissance en temps réel de symboles manuscrits.

En fait, chaque logiciel graphique propose son ensemble de primitives de dialogues, sans que réellement les auteurs soient d'accord sur un ensemble de base.

2.14 Nécessité d'une normalisation

Bien qu'a priori la définition des primitives d'un logiciel graphique de base semble simple, l'étude que nous venons de faire montre que les logiciels distribués offrent une grande variété dans le choix des primitives, tant du point de vue syntaxique que sémantique. L'échange de programmes d'application graphique devient alors rapidement irréalisable, les logiciels utilisés étant par trop différents et incompatibles. Le travail de transcription d'un langage de commande à un autre devient vite une opération fastidieuse et source de nombreuses erreurs.

Cependant, la croissance rapide du nombre d'installations mises en service a rendu ce problème crucial. Les utilisateurs souhaitent qu'un accord se fasse sur les primitives, pour qu'ils puissent être assurés d'un service minimum identique d'un logiciel à un autre. En particulier, l'utilisation grandissante des réseaux d'ordinateur a conduit à rechercher la définition de protocoles d'accord pour l'utilisation de terminaux graphiques à travers un réseau. Cette recherche a fatalement abouti à la constatation qu'il s'agissait en fait de

définir des normes acceptables par la majorité des gens. [Spt 75].

Depuis mai 1976, un certain nombre de travaux ont été entrepris, qui ont abouti à des résultats importants, dans diverses directions. Nous étudierons au paragraphe 2.2 une proposition de norme faite par un groupe du comité d'étude pour la standardisation de l'ACM (Graphic Standards Planning Committee). Nous présenterons en 2.3 nos propres travaux.

2.2 LA PROPOSITION DE NORME DU GRAPHIC STANDARDS PLANNING COMMITTEE

2.21 Historique

Le mouvement qui a conduit à de nombreux travaux sur la normalisation des logiciels graphiques a pris naissance sur une initiative du groupe de travail IFIP WG 5.2 (Computer Aided Design). En août 1974, une commission fut créée, en vue d'étudier le problème de la standardisation. Plusieurs réunions eurent lieu, qui conduisirent aux trois constatations suivantes :

- aucun logiciel graphique distribué n'était susceptible de devenir une norme,

- les concepts de base des techniques graphiques interactives n'étant pas suffisamment perçus, il était trop tôt pour proposer une normalisation,

- il valait mieux se préoccuper de méthodologie, afin de préparer des bases saines à une éventuelle normalisation.

Un séminaire de quatre jours, sur le thème "Méthodologie des techniques graphiques", eut lieu à SEILLAC (France), au mois de mai 1976. Les personnes assistant à ce séminaire étaient d'origines très diverses, tant du point de vue des pays représentés Europe et Amérique du Nord), que du point de vue des activités (universitaires, industriels, utilisateurs, concepteurs). Cette diversité d'expériences garantissait une bonne représentation des problèmes rencontrés habituellement.

Les participants formèrent plusieurs sous- groupes, chargés chacun d'étudier un sujet bien déterminé : spécification formelle, transformations, primitives graphiques, conception d'un logiciel graphique, primitives de dialogue, structure des systèmes et programmes graphiques. Chaque sous-groupe devait rédiger un certain nombre de propositions, en étayant celles-ci des raisons d'accord ou de désaccord. En particulier, les points d'achoppement furent soigneusement répertoriés, en vue de rechercher les raisons profondes de désaccord. L'ensemble des travaux du séminaire doit être publié prochainement. Nous ne donnerons

ici que les quatre résolutions qui furent adoptées [Gue 76].

Résolution 1
Il est nécessaire de faire clairement la distinction entre le logiciel graphique de base ("core") et le logiciel de description ("modelling system").

Résolution 2
Il est très souhaitable d'établir une taxonomie de la structure des programmes graphiques.

Résolution 3
Il est nécessaire de donner une définition formelle des principaux concepts qui sont à la base des techniques graphiques.

Résolution 4
Il est recommandé que les utilisateurs, concepteurs et réalisateurs de logiciels graphiques étudient en commun les logiciels distribués actuellement, afin de voir quelles modifications pourraient être apportées à ces produits, de manière à les rapprocher des recommandations émises par le séminaire.

A la suite de ce séminaire, de nombreux travaux ont eu lieu, tendant à amplifier les résultats obtenus. Nous passons en revue quelques lignes générales avant d'étudier plus particulièrement les travaux américains.

Les pays scandinaves ont choisi de s'attaquer au problème de la modification de logiciels existants en vue d'assurer une meilleure compatibilité entre eux. Disposant déjà de GINO-F [Woo 73] et de GPGS [VCV 77] , ils cherchent à construire un logiciel graphique permettant de bénéficier des services de ces deux logiciels. Le nouveau produit a pour nom IGP [BØ 77] . Sa définition est très avancée, ce qui fait qu'il sera probablement expérimenté très rapidement.

En allemagne, un groupe de travail (FNI Working Group 5.9) cherch à définir un logiciel graphique de base très complet. La grande originalité de ces travaux est de donner une spécification formelle des modules mis en oeuvre ([Eck 77], [EPS 77]). La technique de spécification utilisée est celle qui a été proposée par D. PARNAS [Par 72] . La définition complète du logiciel est également très avancée en ce qui concerne la visualisation.

Les grandes associations de normalisation se sont mises de la partie, désireuses d'encourager les efforts dans le domaine de la standardisation des logiciels graphiques. Citons l'ISO qui, par le biais de l'association britannique de normalisation, a réuni une conférence à Londres en février 1977. Un groupe de travail a été crée sous son égide, qui doit en particulier chercher à unifier les efforts faits au niveau européen [ISO 77].

En France, un groupe commun AFNOR-AFCET a tenu plusieurs réunion Son objectif est de proposer un autre type de norme, correspondant à une approche différente de celle utilisée habituellement pour la définition des logiciels graphiques. En particulier, la notion de point courant serai absente de cette proposition.

Aux Etats-Unis, enfin, le groupe de normalisation affilié à l'ACM (GSPC) a créé en juillet 1976 quatre sous-groupes chargés d'étudier différents aspects de la normalisation des logiciels graphiques. Ces quatr sous-groupes sont :

- le sous-groupe "core", qui a proposé une norme pour un logiciel graphique interactif de base [GSP 77b],

- le groupe "state-of-the-Art", qui s'est chargé d'étudier les points communs et les divergences de quelques logiciels graphiques couramment utilisés [GSP 77a] ,

- le sous-groupe "interface", chargé de définir quelques règles permettant d'assurer un lien correct entre le logiciel graphique de base et le logiciel de description, [GSP 77c] ,

- le sous-groupe "Méthodologie", chargé d'étudier différentes structures de programmes d'application.

Nous étudions ci-dessous la proposition de norme. Nous donnerons les grandes lignes de celle-ci, les détails exacts pouvant être trouvés dans le rapport. L'annexe 2 contient la liste complète des primitives.

2.22 Niveaux de logiciels

La proposition de norme que nous allons présenter est destinée à permettre la définition d'un logiciel graphique interactif de base pour traceurs de courbes et surfaces de visualisation au trait. Les langages de programmation auxquels il servira d'extension sont du type FORTRAN ou PL/1. Le choix des primitives doit permettre d'atteindre un certain degré d'indépendance par rapport au matériel et par rapport aux systèmes d'exploitation. Comme tous les matériels n'offrent pas les mêmes qualités, quatre niveaux de mise en oeuvre sont proposés :

- niveau 1 ("basic") correspondant aux applications non interactives, ne produisant que des dessins (utilisation de traceurs de courbes, de microfilms, etc...).

- niveau 2 ("buffered") correspond encore à des applications non interactives. La différence essentielle est que certaines figures peuvent être déclarées de type permanent, permettant de les réutiliser d'un dessin à l'autre.

- niveau 3 ("interactive") premier niveau permettant d'utiliser les primitives de dialogue.

- niveau 4 ("complete") L'ensemble complet des primitives proposées, y compris les transformations d'images, est disponible.

Les trois premiers niveaux sont compatibles entre eux lorsqu'on monte dans la hiérarchie. Le niveau 4, plus délicat à mettre en oeuvre, a été divisé en trois niveaux compatibles :

. niveau 4a translation en deux dimensions
. niveau 4b translation, rotation, échelle en deux dimensions
. niveau 4c translation, rotation, échelle en trois dimensions

Le tableau 2.4 récapitule les différentes possibilités en fonction des niveaux.

fonctions	niveau 1	niveau 2	niveau 3	niveau 4
primitives de tracé et attributs	oui	oui	oui	oui
préparation à la visualisation	oui	oui	oui	oui
contrôle	oui	oui	oui	oui
figures temporaires	oui	oui	oui	oui
figures permanentes	non	oui	oui	oui
attributs dynamique				
. mise en valeur, affichage immédiat	non	oui	oui	oui
. désignation	non	non	oui	oui
primitives de dialogue	non	non	oui	oui
transformations de dessins	non	non	non	oui

Tableau 2.4 : Possibilités offertes par chaque niveau de logiciel

2.23 Les primitives d'affichage

Les primitives d'affichage sont classées en deux catégories :

- les primitives de tracé ("output primitives"),
- les primitives de définition du graphisme ("attributes").

Les primitives de tracé sont utilisées par le programmeur d'application pour décrire une scène dans son propre système de coordonnées. Le point fondamental est que toutes ces primitives travaillent dans l'espace utilisateur et que leur action définit la position d'un point courant ("current position" ou 'CP") dans cet espace.

Trois types de données peuvent être traitées :

- des segments de droite,
- des chaînes de caractères,
- des marqueurs.

La position initiale de chaque tracé est spécifiée par la position du point courant. Celle-ci résulte soit de la position finale du tracé précédent, soit d'une mise en place à l'aide d'une primitive de type *MOVE*.

En ce qui concerne les segments de droite, nous noterons les points suivants :

- il existe un jeu de primitives permettant de dessiner aussi bien dans le plan que dans l'espace,

- les coordonnées peuvent être données en absolu ou en relatif (par rapport à la position actuelle du point courant),

- il est fait distinction entre le tracé d'un segment de droite (*LINE*) et le tracé d'une ligne brisée (*POLYLINE*).

- toutes les primitives de tracé de segment de droite modifient la positi du point courant.

En ce qui concerne le tracé de chaînes de caractères, nous fero les remarques suivantes :

- tout tracé de chaîne commence à la position actuelle du point courant.
- quatre qualités de texte sont prévues ("low-quality", "medium-quality", "high-quality", "special-output").
- le tracé de chaque caractère modifie l'emplacement du point courant. Cette modification dépend des caractéristiques associées à la chaîne à tracer (hauteur, largeur, taille, orientation, qualité). Par exemple, la rotation d'une chaîne de caractères ne donne pas les mêmes résultats suivant la qualité employée (!), et donc la position finale du point courant diffère d'un cas sur l'autre.

La notion de marqueur ("marker") est introduite pour permettre le repérage sur l'écran de positions à l'aide de symboles spéciaux. Le symbole sélectionné est affiché centré sur une position passée en paramètre. Quelle que soit la forme du marqueur, la position finale correspond à la position de centrage.

Les primitives de définition du graphisme permettent d'associer à chaque primitive de tracé un mode de dessin. Les graphismes attribués sont statiques et ne peuvent être modifiés une fois la scène décrite. Les principaux paramètres sont :

- la couleur, la luminosité (applicables à toute primitive),
- l'épaisseur, la texture (pour les segments de droite),
- la police, la taille, l'espacement, la qualité (pour les caractères).

Un nom ("pick-id") peut être associé à toute primitive, correspondant à une valeur de corrélation.

Des primitives d'interrogation sont définies, qui permettent de connaître la valeur courante de chacun des attributs. Des valeurs standard sont proposées.

2.24 Les primitives de structuration

Elles sont explicites, différenciant la modification de la dénomination. Un seul niveau de modification est prévu, la figure ("segment"), représentant un ensemble de primitives de tracé. La dénomination ("naming") peut s'adresser à une ou plusieurs primitives d'une même figure, par le biais des valeurs de corrélation.

Une figure est définie par une suite d'appels de primitives de tracé, suite délimitée par l'appel d'une primitive *CREATE* qui permet d'attribuer un nom de figure, et l'appel d'une primitive *CLOSE*. La création d'une figure répond aux règles suivantes :

- il n'y a pas emboîtement des définitions,
- il n'existe pas deux figures portant le même nom,
- tous les paramètres de visualisation sont figés et ne peuvent être modifiés au cours de la création de la figure,
- il existe deux types de figures :
    . les figures permanentes ("retained") : la liste des primitives les composant est conservée (pseudo-liste de visualisation),
    . Les figures temporaires ("non retained segments"), qui sont affichées immédiatement et ne sont pas conservées.

Des primitives sont fournies qui permettent d'effacer une figure de type permanent et de détruire la liste de visualisation associée

Les principaux attributs liés à une figure sont :

- le type de la figure (temporaire ou permanent),
- certaines particularités telles que affichage immédiat ou différé, mise en valeur (par clignotement, surbrillance ou autre moyen), désignation autorisée ou non,
- des attributs de mise en page (translation, rotation, changement d'échel le), que ce soit dans le plan ou dans l'espace (?).

Tous ces attributs sont dynamiques, sauf le type de la figure défini une fois pour toutes. Des valeurs par défaut sont définies. Des primitives d'interrogation sont proposées, permettant de connaître les valeurs courantes des attributs.

Nous noterons surtout ici l'apparition de primitives de mise en page aussi bien dans le plan que dans l'espace, ce qui montre bien que nous sommes en présence d'un logiciel de description et non d'un logiciel de base.

2.25 Les primitives de préparation à la visualisation

Les primitives regroupées sous cette rubrique permettent de définir complètement les transformations à appliquer aux figures pour obtenir une représentation graphique sur l'écran. Les auteurs du rapport utilisent l'analogie avec un appareil photographique. L'image obtenue est déterminée par la position, l'orientation de l'appareil. Elle est présentée sur une ou plusieurs surfaces de visualisation.

Les transformations en vue de la visualisation servent à deux fins :

- spécifier la portion de scène qui présente un intérêt,
- spécifier la transformation géométrique qui fait passer des coordonnées utilisateur à un système de coordonnées normalisé.

La remarque fondamentale est que les scènes considérées appartiennent toujours à l'espace à trois dimensions, le dessin de scènes planes étant considéré comme un cas particulier. Ceci explique que le processus de visualisation obéisse au scénario suivant :

- la scène, décrite en coordonnées utilisateur grâce aux primitives de tracé, est supposée enclose dans un volume, qui est une pyramide si l'on fait une projection perspective, un parallélépipède rectangle si l'on fait une projection cylindrique.

- les frontières du domaine de vision sont déterminées par une fenêtre, domaine rectangulaire situé dans le plan de vue et soit le centre de projection (dans le cas des projections perspectives), soit la direction de projection (pour les projections cylindriques).

- la scène est découpée par rapport au volume défini précédemment. L'ensemble des constituants internes à ce volume est projeté dans la fenêtre. La scène à deux dimensions ainsi obtenue est projetée dans la cloture ("viewport"), espace de l'écran réservé à cet effet.

Nous ferons les remarques suivantes :

- pour faciliter la tâche des programmeurs ne travaillant que dans le plar un jeu de primitives pour le traitement en deux dimensions est fourni. Le principe disant qu'on ne travaille que dans l'espace est donc contou

- la notion de fenêtre a été étendue par rapport à l'acception habituelle. En effet, les bords de la fenêtre peuvent être inclinés par rapport aux axes Ox et Oy, la direction étant donnée par les composantes d'un vecteur de visée ("view-up vector").

- des primitives sont fournies qui permettent de travailler à une échelle réelle (pour faciliter des sorties correctes de plans ou de cartes, par exemple).

2.26 Les primitives de dialogue

Deux principes ont guidé la définition des primitives de dialog

- définition d'un ensemble de dispositifs logiques, divisé en deux sous-ensembles suivant le mode de collecte des renseignements,

- gestion des dispositifs logiques en termes de traitement d'interruption par établissement de files d'attente, autorisation ou interdiction de certains événements.

Bien que les dispositifs logiques choisis aient une correspondance direct avec des dispositifs de communication spécifique, il n'est pas interdit de pallier l'absence d'un dispositif par des techniques logicielles.

Il existe cinq types de dispositifs logiques :

- les dispositifs d'identification *(PICK)*
- les dispositifs de lecture de chaînes de caractères *(KEYBOARD)*
- les dispositifs de sélection d'action *(BUTTON)*
- les dispositifs de collecte de coordonnées *(LOCATOR)*
- les dispositifs de lecture de valeurs numériques *(VALUATOR)*

Chaque dispositif logique est identifié par un nom formé de deux parties
- son type ("device class")
- son numéro dans sa classe ("device number").

Deux modes de fonctionnement sont prévus :

- certains dispositfs ne sont lus qu'une seule fois, avant retour au programme d'application (*PICK*, *KEYBOARD*, *BUTTON*). Ils ont un fonctionnement sur interruption.

- les autres dispositifs sont régulièrement relevés à des intervalles de temps donnés (*LOCATOR*, *VALUATOR*).

La gestion des dispositifs de communication est assurée grâce à une trentaine (!) de primitives, que l'on peut ranger dans les catégories suivantes :

- autorisation ou interdiction d'utiliser un dispositif logique,
- traitement des dispositifs à relevé périodique (lecture d'une valeur),
- traitement des dispositifs à interruption (attente d'interruption, gestion de la file d'interruptions),
- lecture des réponses aux différentes demandes,
- réponse aux actions de l'opérateur (curseurs, valeurs numériques, allumage des touches des claviers de fonction etc...)

Deux remarques peuvent être faites à ce niveau :

- la notion de dispositif logique n'est qu'un masque par rapport aux dispositifs réels. Ceci est clair lorsque l'on considère le choix des traitements : par exemple, la collecte de coordonnées (*LOCATOR*) est manifestement prévue par une tablette ou un manche à balai. Nous sommes donc fort loin de fonctions de dialogue évoluées.

- les primitives proposées sont celles que l'on utilise lorsque l'on programme le traitement des interruptions au niveau le plus bas (assembl

2.27 Les primitives de contrôle

Il s'agit d'un ensemble de primitives permettant de contrôler le fonctionnement du logiciel et son adaptation au matériel utilisé. Les principales fonctions assurées sont :

- initialisation et terminaison d'une session,
- choix d'une console (pour des installations ou des applications multi-consoles),
- modification des valeurs par défaut,
- vérification des paramètres de la version de logiciel utilisée, permettant au programmeur de prévoir plusieurs cas de traitement au niveau du programme d'application
- aide à la gestion des erreurs éventuelles,
- gestion de l'écran, en particulier dans le cas des tubes mémoire,
- contrôle du découpage par fenêtres.

Cet ensemble de primitives n'offre aucune cohérence interne, et rassemble en fait quelques particularités de certains logiciels.

2.28 Liens spéciaux avec le programme d'application

Il s'agit de deux possibilités

- transmission de matrices de transformations issues d'un logiciel de description,
- utilisation de possibilités non standards (par exemples liées à un matériel spécialisé).

2.29 Critique

La proposition de norme que nous venons de décrire re
un travail considérable (évalué par ses auteurs à 18 hommes/an)
de primitives retenu a été choisi -après de nombreuses discussi
raisons des différents choix sont explicitées dans le rapport,
ainsi de juger des critères fondamentaux qui ont été retenus l
l'élaboration de celui-ci. Cependant, le travail de normalisat
pas terminé et nous relèverons quelques points soulevés dans l
lui-même et qui restent à traiter :

- primitives de tracé : faut-il réellement une position courant
  est la position courante lors de l'utilisation de *CREATE SE*
  CLOSE ? quelles autres primitives doivent être ajoutées ? Q
  les traitements appropriés aux textes ? Comment l'opérateur
  être prévenu d'une action ?

- structuration du dessin : quelles hiérarchies et structures
  être offertes ? Peut-on réutiliser une figure pour lui ajoute
  mitives alors que *CLOSE* a été invoquée ? Faut-il vraiment de
  temporaires ?

- attributs : les attributs de tracé doivent-ils être totalemen
  figures ? Quelles sont les valeurs permises, par défaut ? Co
  certains attributs n'existant pas du point de vue technologi

- préparation à la visualisation : ces transformations font-el
  du logiciel de base ? Faut-il séparer les opérations en deux
  des opérations en trois dimensions ?

Enfin deux questions relatives aux langages de programmation :

- doit-il exister un logiciel de base pour chaque langage de pr
- faut-il limiter l'utilisation de certains traits d'un langag
  l'emploi du logiciel graphique de base ?

Les questions posées prouvent à l'évidence que d'autres voies de recherche peuvent être proposées. Pour notre part, tout en reconnaissant l'intérêt de cette proposition pour les logiciels actuels, plusieurs points nous semblent très sujets à caution :

- le choix d'une description dynamique du dessin à l'aide d'un point courant
  Ce choix introduit automatiquement une confusion entre logiciel de description et logiciel graphique de base en faisant un amalgame entre le mode de description et le mode de tracé. C'est ainsi que cette norme ne peut être utilisée pour autre chose que des dessins au trait, la notion de point courant n'ayant pas de sens pour définir une tache. Nous pensons qu'il s'agit là d'une restriction trop forte, qui n'a pas de vraie raison d'être. Les principaux arguments avancés en faveur de ce choix concernent le "naturel d'une telle description" et surtout une plus grande efficacité : moins de paramètres à passer en appel et possibilité de choisir un parcours minimal pour réaliser le dessin. En ce qui concerne le premier argument, il n'est très fort que dans la mesure où l'enchaînement des différentes parties du dessin peut se faire en utilisant des positions communes telles que la position finale d'un tracé corresponde justement à la position initiale du tracé suivant. Cette situation est en fait exceptionnelle, ce qui conduit à utiliser constamment la primitive *MOVE*. Quant à la recherche d'un parcours minimum, s'il est vrai que certains programmeurs s'en préoccupent, c'est beaucoup plus à cause de la lenteur des traceurs de courbes que pour obtenir un tracé rapide sur une console de visualisation.
  En tout état de cause, le logiciel graphique élémentaire peut être amené à modifier le parcours, à l'insu du programmeur. C'est par exemple le cas pour un affichage sur une surface de visualisation point par point, où le dessin est découpé en bandes horizontales. Le choix fait par le programmeur n'a alors plus aucun intérêt, sinon un intérêt purement intellectuel (par exemple dessiner une figure sans jamais lever la plume et sans repasser deux fois au même endroit).

- les primitives de tracé choisies ne sont pas très cohérentes. Différencier le tracé d'un segment de droite de celui de plusieurs segments de droite n'est pas réellement très intéressant sinon pour économiser un paramètre : le nombre de segments à tracer. La politique adoptée pour le tracé des caractères montre également que la notion de point courant est mal adaptée à autre chose que du dessin au trait. En effet, une bonne qualité de dessin de caractères implique automatiquement le remplissage de formes, donc l'utilisation de taches qui sont justement proscrites de la norme.

- en ce qui concerne la structuration, le fait de ne pas disposer du même nombre de niveaux en ce qui concerne la modification et en ce qui concerne la dénomination ne nous paraît pas très heureux. Ce choix met en évidence la nécessité de représenter des structures différentes (voir 1.31), mais la solution retenue pourraît être mieux équilibrée (un niveau ou deux niveaux pour la modification et la dénomination).

- la mise au rang des attributs de figure des fonctions de mise en page n'est pas très claire. Il vaudrait mieux, pour des questions de cohérence de niveau, utiliser des primitives de même niveau que les primitives de définition de fenêtres et de clotures, ce qui mettrait en valeur la mise en page des figures par rapport au tracé proprement dit.

- le mode de spécification du dialogue est de trop bas niveau. Il s'agit en fait de la programmation de périphériques au niveau assembleur. Pour des raisons très contestables d'efficacité, les dispositifs logiques proposés sont en fait très dépendants de dispositifs réels, et ne sont pas réellement définis en termes de fonctions de dialogue. Le niveau de programmation du dialogue ne correspond pas au niveau des langages de programmation auxquels est destinée la norme.

En résumé, si la norme proposée est une certaine remise en ordre des notions utilisées à l'heure actuelle, d'autres axiomes peuvent être pris pour base de conception de logiciels de base. Nous présentons dans le paragraphe suivant un logiciel graphique interactif de base qui tente de répondre aux critiques que nous venons de formuler.

2.3 LE LOGICIEL GRAPHIQUE INTERACTIF DE BASE GRIGRI

2.31 Historique

Le logiciel graphique de base interactif GRIGRI représente la dernière étape d'une lignée de trois logiciels développés pour l'utilisation des matériels du Centre Interuniversitaire de Calcul de Grenoble. Les principales caractéristiques de cet environnement sont les suivantes :

- existence d'un matériel puissant (IBM 360/67, IRIS 80) avec des périphériques divers (IBM 2250 modèle I, TEKTRONIX 4010 et 4014),

- exploitation sous un système de partage de temps (CP/CMS), offrant de nombreux services (gestion de fichiers, langages divers),

- communauté d'utilisateurs très divers, composée aussi bien d'informaticiens de métier que de personnes n'ayant que les connaissances nécessaires à l'écriture de programmes simples (programmation FORTRAN, ALGOL W ou PL/1).

A l'origine, la programmation se faisait en utilisant GSP (voir 2.1). Cependant, les difficultés d'utilisation et de mise au point de tels programmes avaient conduit dans un premier temps à expérimenter un système de programmation graphique autonome, permettant de gérer l'ensemble des activités d'écriture, mise au point et exploitation d'un programme à partir de la console (EULALIE/EUPHEMIE [Cle 69] , [Lec 70]). Malgré les qualités de ce système du point de vue de la gestion des programmes, son utilisation fut faible. La raison majeure tient au fait que le logiciel graphique n'était autre que GSP avec un habillage syntaxique plus moderne, mais encor difficile d'emploi.

Par réaction à cette situation, nous avons développé un premier logiciel (AW 2250 [LaM 74] , qui, bien que réalisé à l'aide de GSP, avait été défini de manière à réduire radicalement le nombre de primitives offertes (10 au lieu d'une cinquantaine, cf. Annexe 2). Si l'utilisation de l'écran était moins efficace (en nombre d'octets !), elle était au moins à la portée d'un plus grand nombre de gens. La principale leçon que nous avons tirée de cette expérience était qu'un nombre de primitives

relativement réduit était un avantage décisif. En effet, la mise en service de ce logiciel permit rapidement à un grand nombre d'utilisateurs de réaliser des programmes graphiques. Les principales faiblesses de ce logiciel se situaient au niveau de sa trop grande dépendance par rapport au terminal IBM 2250. En particulier, le dialogue était entièrement spécifié par référence au système de gestion d'interruptions de GSP, ce qui conduisait à une programmation de trop bas niveau.

La première version du logiciel GRIGRI [LLL 75] visait à donner des primitives de dialogue de haut niveau. La notion de dispositif logique telle que nous l'avons décrite en 1.12 était développée, permettant de faire disparaître toute mention des dispositifs employés. Cependant, si le côté dialogue révèla rapidement son intérêt, par contre l'affichage montra de graves insuffisances. Les points forts de cette première version furent les suivants :

- indépendance par rapport au matériel (utilisation indifférente des terminaux du CICG),
- grande facilité de programmation du dialogue et d'utilisation des dispositifs de communication,
- grande simplicité de la programmation, qui a permis de ne plus avoir à aider les utilisateurs lors de la programmation de leurs applications.

Les points faibles se trouvaient au niveau de l'affichage :

- absence de primitives de traitement de textes,
- impossibilité de définir différents graphismes (fournis par le matériel !
- niveau des primitives d'affichage inférieur au niveau des primitives de dialogue, interdisant d'en employer toute la puissance. Par exemple, une primitive de type *VECTEUR* ne permettait d'afficher qu'un ensemble de traits de même type (une ligne brisée par exemple), alors que la primitive *POSITION* permettait d'en introduire plusieurs.

Le nombre de primitives est encore faible (13), mais la qualité des service rendus (en particulier au niveau du dialogue) est sans commune mesure avec AW2250 (cf. annexe 2).

La version actuelle de GRIGRI [LLM 77] est fondée sur les points suivants :

- description statique du dessin, par utilisation de tableaux de données de types différents (coordonnées, textes) et spécification de l'interprétation graphique de ces données (graphismes).

- Symétrie des primitives d'affichage et de dialogue,

- indépendance du logiciel par rapport à son environnement d'exploitation.

Les primitives de cette version sont répertoriées dans l'Annexe 2. Leur nombre est de 18, mais la puissance du logiciel s'est accrue de nouveau de manière très importante.

Nous étudierons dans ce paragraphe les particularités de cette dernière version en nous plaçant au point de vue du programmeur. Le lecteur intéressé par la réalisation pratique de ce logiciel trouvera la description complète des techniques employées dans [Led 77].

2.32 Les primitives d'affichage

Elles se décomposent en trois catégories :

- déclaration de données,
- définition de l'interprétation graphique des données,
- gestion de l'écran.

2.321 Les déclarations de données

Les primitives de déclaration de données permettent de prévenir le logiciel élémentaire du type de données qu'il aura à traiter. La version actuelle permet de traiter des listes de traits et des listes de caractères définis de la manière suivante :

POINTS (nfig, tablong, tabx, taby, tablegende)
TEXTES (nfig, tablong, tabtextes, tabx, taby).

La primitive *POINTS* permet de spécifier l'arrangement d'un ensemble de coordonnées, structuré en sections. Une section est représentée par un sous-tableau de coordonnées. L'ensemble des sections est spécifié grâce à *tablong* qui contient la longueur de chacune de sections considérées (cf. figure 2.1). Les sections sont numérotées séquentiellement dans leur ordre de définition dans *tablong*.

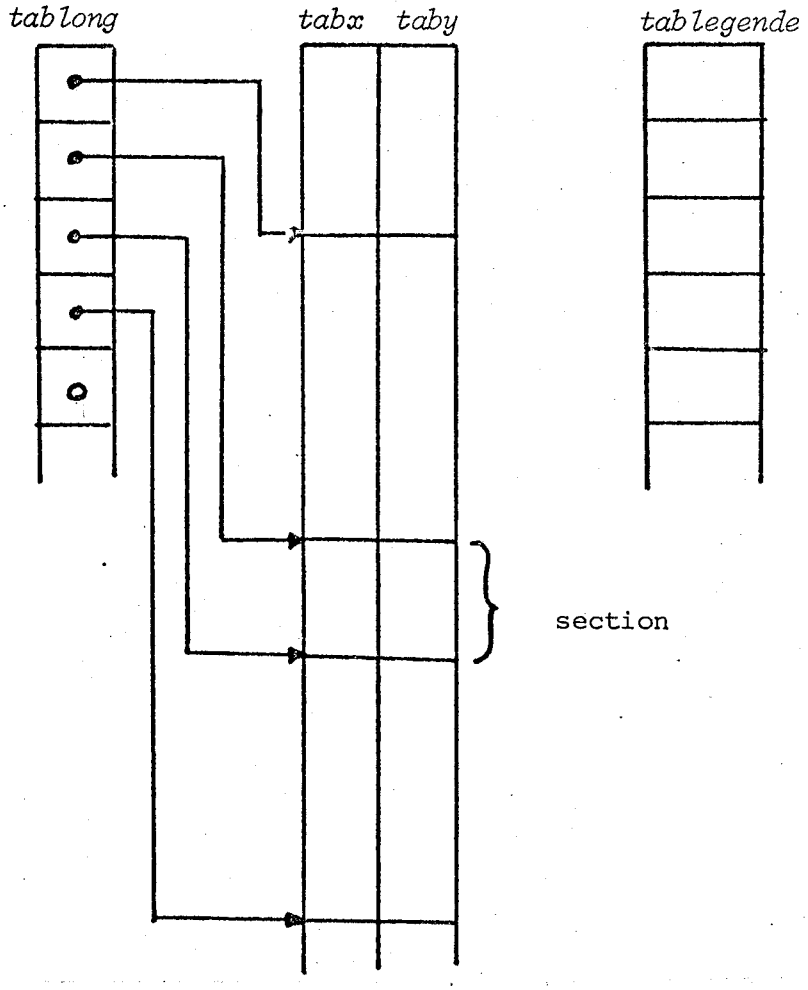


Figure 2.1 Déclaration de données de type POINTS

Le tableau des légendes permet d'associer à chaque section un marqueur, pouvant faciliter l'identification d'un élément sur l'écran. Le marqueur est centré sur les coordonnées du premier couple de la section concernée.

Les sections permettent de définir, pour un ensemble de points donné, un découpage en sous-ensembles auxquels on pourra associer individuellement un graphisme. Les différents tableaux utilisés sont liés à une figure par *nfig* ce qui constitue le premier niveau de dénomination de la structure de l'image.

La primitive *TEXTES* est analogue à la primitive *POINTS*. Les textes à afficher sont rangés dans le tableau *tabtextes*. Celui-ci est découpé en sections grâce à *tablong* , permettant ainsi d'obtenir des affichages avec des caractéristiques différentes d'une partie du texte à l'autre (polices, épaisseur, majuscules etc...).

Ces deux primitives définissent en fait l'interface avec le programme appelant GRIGRI, mais ne provoquent aucun affichage. Elles peuvent être appelées à tout moment, permettant ainsi d'associer au moment de l'affichage proprement dit les tableaux correspondant à la figure à traiter.

2.322 - L'interprétation graphique des données

L'interprétation graphique des données est précisée grâce à la primitive

MODE(nfig, ncorrel, nsection, descripteur)

Cette primitive permet d'associer un graphisme à une (ou plusieurs) section de la figure *nfig*.

Les principaux graphismes autorisés sont les suivants :

- points couleur, texture, épaisseur, intensité, mode de tracé, marquage ...
- textes couleur, police, taille, orientation ...

Le graphisme choisi est spécifié à l'aide d'une combinaison de codes contenue dans le descripteur. Ce graphisme restera valable aussi longtemps que la primitive *MODE* n'aura pas été réutilisée avec le même numéro de section.

La primitive *MODE* est également utilisée pour associer une valeur de corrélation à la section concernée. Nous considérons en effet que cette valeur est une caractéristique comme une autre, ce qui explique qu'elle soit spécifiée à ce niveau.
La programmation est facilitée par l'existence des deux conventions suivantes :

- il existe des graphismes standard, correspondant aux cas les plus usuels, qui sont attribués automatiquement aux sections. Un mode "neutre" permet de changer de corrélation sans changer de graphisme,

- il est possible d'affecter le même mode à plusieurs sections en utilisant une notation qui est détaillée en 2.33.

2.323 - La gestion de l'écran

La gestion de l'écran se fait à l'aide des deux primitives suivantes :

AFFICHER (nfig, ncorrel, nfen, nclot)
EFFACER (nfig, ncorrel)

L'appel à la fonction *AFFICHER* provoque la construction d'un élément du dessin, par interprétation des valeurs contenues dans l'interface associé à la figure *nfig* à l'aide des modes précisés section par section.

2.33 La structuration du dessin

GRIGRI offre deux niveaux de dénomination et de modification :

- le niveau de la figure, qui se fait implicitement en regroupant les données à l'aide du nom *nfig* passé en paramètre de la déclaration de données.

- le niveau des sections, qui se fait explicitement par utilisation de la valeur de corrélation lors de l'attribution des caractéristiques de chaque section.

Les noms de figure et de corrélation sont des entiers choisis par le programmeur d'application.

L'affichage et l'effacement peuvent se faire globalement au niveau de la figure, ou section par section (ou ensemble de sections de même corrélation). Pour faciliter la description de ces sous-ensembles, une notation abrégée est utilisée, permettant de désigner un élément ou plusieurs éléments. Le tableau 2.5 récapitule les cas possibles et la convention associée.

	nfig	*ncorrel*	*nsection*
= 0	toutes figures	toutes sections	toutes sections
> 0	la figure *nfig*	les sections corrélées *ncorrel*	la section *nsection*
< 0	toutes les figures sauf *nfig*	toutes les sections sauf celles corrélées *ncorrel*	toutes les sections sau *nsection*

Tableau 2.5 - Conventions de dénomination

2.34 Les primitives de préparation à la visualisation

Elles sont au nombre de deux

FENETRE(numfen, gx, gy, dx, dy)
CLOTURE(numclot, gx, gy, dx, dy)

Ces primitives permettent de définir différents systèmes de coordonnées utilisateur et plusieurs espaces de dessin sur écran. On notera les points suivants :

- la référence à une fenêtre et une clôture sera obligatoire lors de la création du dessin,
- la donnée de la clôture se fait dans un système de coordonnées indépendant de ceux utilisés par les écrans réels ;
- il n'y a plus association directe avec une figure. La définition des espaces de travail est indépendante de la structure logique du dessin.

2.35 Les primitives de dialogue

L'expérience acquise lors de la mise en service de la première version de GRIGRI nous avait montré que le concept de dispositif logique fondé sur les quatre actions de base d'écritures en 1.12 était très apprécié des utilisateurs. Cependant, nous avions constaté également que ces primitives élémentaires étaient toujours utilisées en combinaison. En particulier, la notion de répétition d'une action élémentaire, telle qu'elle existait dans la primitive *POSITION* (collecter n couples) devait être généralisée aux autres primitives de base. La version actuelle offre les mêmes primitives de base, avec un paramètre de répétition supplémentaire.

2.351 - Les messages à l'opérateur

Les primitives évoquées ci-dessous permettent de composer des messages destinés à prévenir l'opérateur du déroulement de son programme. En particulier, des zones sont réservées, qui permettent d'afficher les valeurs de certains paramètres du calcul. Les primitives sont les suivantes :

MESSAGE (nmess, descripteur, nclot, x,y)
EFFMESS (nmess)

Le descripteur contient le message et la définition des zones de présentation de valeurs. Le message est affiché dans la clôture *nclot*, à la position spécifiée par les coordonnées x,y (dans le système de la clôture). La primitive *EFFMESS* permet d'effacer un ou plusieurs messages de l'écran, en utilisant la convention de dénomination décrite précédemment.

Les primitives

EDENTIER (nmess, nzone, entier)
EDREEL (nmess, nzone, réel)
EDCHAINE (nmess, nzone, chaîne)

permettent de compléter les zones réservées d'un message grâce au contenu des variables spécifiées.

Il est à remarquer que les messages à l'opérateur ne font pas partie du dessin proprement dit. Ils ne peuvent donc remplacer la primitive *TEXTES*.

2.352 - Introduction de données alphanumériques

Elle est réalisée en association avec un message à l'opérateur, à l'aide des primitives suivantes :

RVAL (nb, nmess, tréel)
NVAL (nb, nmess, tentier)
CVAL (nb, nmess, tchaîne)

Les *nb* valeurs consécutives sont introduites dans le tableau spécifié.

2.353 - Collecte de coordonnées

Elle se fait par le biais de la primitive

POSITION (nb, nfig, nfen, nclot)

La donnée de *nfen* et de *nclot* permet de spécifier la transformation permettant de passer de l'espace écran (où sont recueillies les coordonnées) à l'espace utilisateur. Les *nb* coordonnées recueillies sont rangées à la suite de la dernière section existant dans le tableau spécifié par la donnée de *nfig* . Lors du relevé des *nb* coordonnées, l'opérateur peut spécifier un découpage en sections. Le tableau de sections associé à la figure *nfig* est automatiquement mis à jour.

Il est clair que l'option consistant à remplir systématiquement les tableaux à partir de la dernière section était la plus simple que nous pouvions prendre. On pourrait imaginer des opérations plus complexes, permettant de remplacer une section par les points recueillis. Cependant, la lourdeur du logiciel associé à ce type de travail (opérations de tassement ou de décalage des tableaux) ne nous a pas paru être d'un intérêt suffisant pour la mettre en oeuvre. Cependant, le lecteur notera que l'on peut créer des données graphiques par relevé, en remplissant peu à peu un

catalogue dont les sections représenteraient les différents éléments.

2.354 - Identification

La primitive

IDENTIFIER (nb, nfig, tfig, tcorrel)

permet d'obtenir dans les tableaux *tfig* et *tcorrel* la liste des noms (numéro de figure et numéro de corrélation) associés aux éléments du dessin désignés par l'opérateur. Une convention de numérotation sur *nfig* permet d'autoriser la désignation sur les éléments d'une seule figure, de toutes les figures présentes sur l'écran ou de toutes les figures sauf une. Cette faculté permet ainsi de définir des portions de dessins sur lesquelles des opérations pourront se faire, et d'autres portions au contraire protégées. Ceci est par exemple très important lorsqu'on se sert d'un catalogue de symboles qu'il faut pouvoir tour à tour désigner pour sélectionner un élément et protéger lors d'opérations d'effacement sélectif.

2.355 - Menu

La sélection de plusieurs actions se fait grâce à :

MENU (nb, liste, tretour)

Les actions choisies sont représentées par leur numéro, rangées dans l'ordre chronologique de désignation dans le tableau *tretour*. Le programmeur pourra donc exploiter ce tableau soit en effectuant les opérations demandées l'une après l'autre, soit au contraire en les regroupant. Par exemple, l'ensemble des opérations de transformation d'une figure peut se réaliser en une seule opération en calculant la matrice produit des matrices de transformations successives données par un menu multiple.

2.356 - Mise en oeuvre par l'opérateur

Lorsqu'une fonction de dialogue est exécutée, une indication du type d'action attendu est affichée sur l'écran. L'opérateur choisit alors le type de dispositif qu'il estime le plus approprié à l'action qu'il va réaliser. Par exemple, lors d'une collecte de coordonnées, il choisira

indifféremment le photostyle, la tablette, le clavier alphanumérique ou tout autre moyen mis à sa disposition au niveau de la console réelle. En plus de cette liberté du choix du moyen physique, nous voulons mentionner aussi que la réalisation proposée permet une gestion identique des actions élémentaires. Lors de l'exécution d'une demande de fonction de dialogue, l'opérateur dispose des fonctions suivantes :

- contrôle du nombre d'actions élémentaires effectuées, au cas où un maximun a été spécifié. Si celui-ci a été atteint, il y a retour au programme appelant.
- contrôle du nombre d'actions effectué par utilisation d'un signal de fin de dialogue, permettant à l'opérateur d'interrompre la suite d'action élémentaires.
- annulation de la dernière action élémentaire réalisée.
- annulation de l'ensemble des actions élémentaires réalisées depuis le début d'exécution de la fonction.

La similitude des actions de dialogue est ainsi mise en valeur au niveau opératoire.

2.36 Extensions du logiciel GRIGRI

Les extensions proposées ci-dessous concernent aussi bien l'affichage que le dialogue. Elles font suite à des demandes d'utilisateurs, ou au désir d'offrir des outils plus perfectionnés.

2.361 - Extension des primitives d'affichage

Au niveau de la mise en page, l'obligation de déclarer les limites de la fenêtre utilisée conduit les programmeurs à calculer systématiquement les coordonnées maximales et minimales de la scène à présenter. On peut imaginer d'autoriser une mise en page automatique, par recherche de la meilleure échelle permettant de cadrer une scène dans une clôture donnée. Nous proposons donc la convention suivante, portant sur la définition d'une fenêtre par les coordonnées (gx,gy) et (dx,dy) :

gx < dx	et	gy < dy	fenêtre entièrement définie par le programme
gx < dx	et	gy ≥ dy	fenêtre définie seulement en largeur : calculer la hauteur abs(ymax-ymin)
gx ≥ dx	et	gy dy	fenêtre définie seulement en hauteur ; calculer la largeur abs(xmax-xmin)
gx ≥ dx	et	gy ≥ dy	fenêtre non définie ou fenêtre non déclarée : calculer hauteur et largeur

dans le cas où l'on calcule une hauteur et/ou une largeur, les valeurs obtenues sont placées dans les variables correspondantes.

Un autre point concerne le découpage de la scène. La version actuelle effectue un découpage automatique en comparant les éléments de la scène par rapport à la fenêtre définie. Or cette pratique est inefficace pour deux raisons :

- beaucoup d'utilisateurs travaillent dans un domaine de coordonnées spécifique, identique à la fenêtre, et aucun élément du dessin n'est à priori, hors de la fenêtre,
- l'extension que nous venons de proposer risque d'augmenter le nombre de dessins cadrés exactement dans une fenêtre.

Nous avons alors proposé de mettre au niveau du mode un code permettant de spécifier si l'opération de découpage doit être entreprise, l'option standard étant qu'il n'y a pas de découpage à réaliser.

En ce qui concerne l'affichage proprement dit, nous proposons d'ajouter la primitive

TACHES (nfig, tablong, tabx, taby, faces, tablegende)

permettant de définir des ensembles de taches. Une tache sera décrite par la liste de ses points de contour, liste terminée par 0. Une section sera composée par un ensemble de taches, chaque élément du tableau *tablong* étant constitué du nombre de taches comprises dans la section correspondante (cf. figure 2.2).

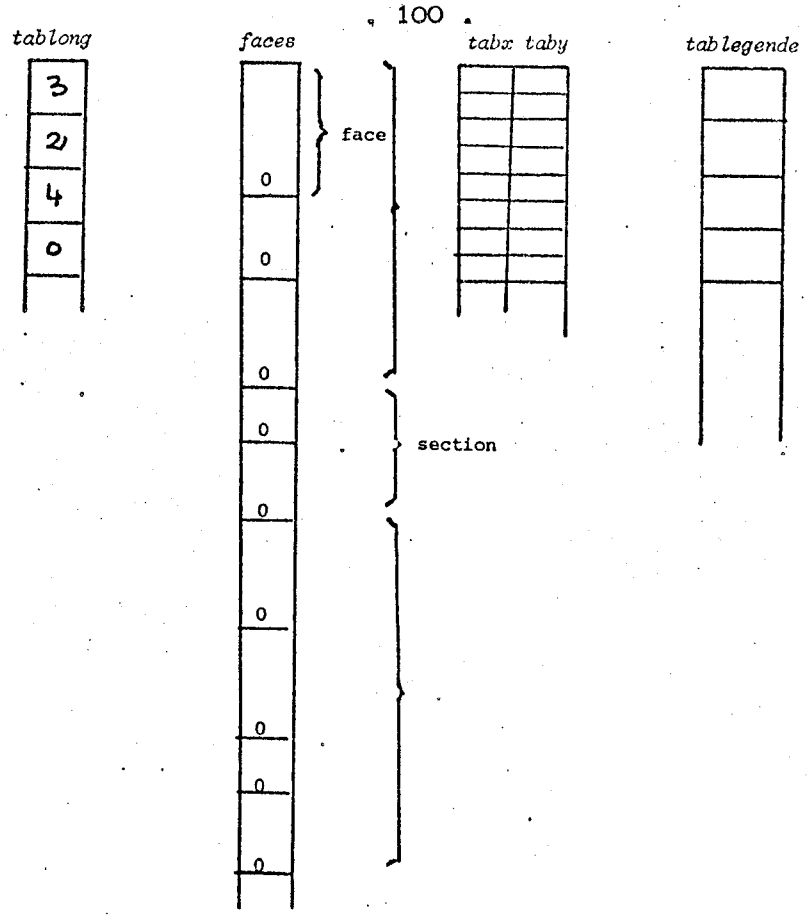


Figure 2.2 Définition d'un ensemble de taches

Les caractéristiques associées grâce à la primitive *MODE* concerneraient essentiellement la couleur, l'intensité lumineuse, le grain. L'introduction de cette primitive permettrait ainsi de disposer d'un logiciel élémentaire permettant de traiter tous les éléments de base d'un dessin, ainsi qu'ils ont été définis en I.1.

2.362 - Extension des primitives de dialogue

Nos propositions concernent trois niveaux :

- respect de la règle de symétrie, entre primitives d'affichage et de dialogue,
- addition de nouvelles primitives élémentaires,
- généralisation des possibilités de combinaison d'actions élémentaires.

En ce qui concerne les problèmes de symétrie, on peut constater que la fonction *POSITION* est le pendant de l'affichage des traits de la primitive *POINTS*. Rien n'existe au niveau de l'introduction de chaînes de caractères. N'ayant pas eu à traiter à ce jour d'applications de manipulation interactive de chaînes de caractères, nous ne formulerons pas de proposition à ce sujet. Par contre, en ce qui concerne la primitive *TACHES*, nous avons eu l'occasion de suggérer le mécanisme suivant, fondé sur

l'utilisation d'une tablette et d'un manche à balai (ou potentiomètre quelconque). La définition d'une tache se fait en combinant :

- le "squelette" de la tache (ligne de points équidistants des bords de la tache) donné en utilisant la tablette,
- une largeur de trait, donnée à l'aide du manche à balai.

L'opérateur peut ainsi, en agissant simultanément sur le manche à balai et le stylo de la tablette définir des taches constituées en fait de traits d'épaisseur variable (cf. figure 2.3).

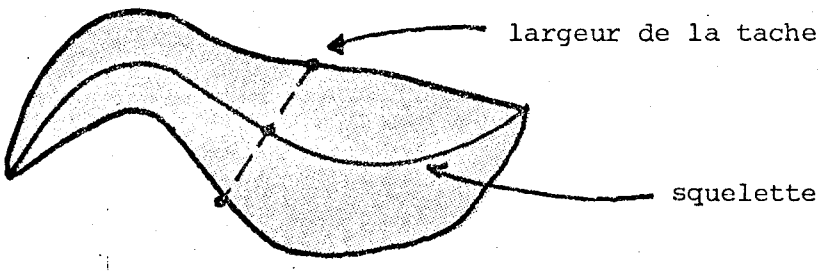


Figure 2.3 Dessin d'une tache

Il est intéressant de remarquer que l'on peut mettre en jeu un mécanisme analogue pour effacer certaines portions de l'écran. On dispose alors d'une véritable "gomme" qui efface une zone à contour variable.

Il est bien évident que d'autres mécanismes peuvent être proposés. Ce mécanisme s'est avéré très intéressant pour une application utilisant un terminal télévision et destinée à permettre la construction de synoptiques.

L'addition de nouvelles primitives de dialogue revient à estime qu'un mode d'utilisation des techniques interactives, bien que réalisable en combinant deux actions élémentaires, ne se fait pas dans de bonnes conditions. Un exemple typique concerne l'action de placement d'un élément de dessin sur l'écran. Cette action se décompose en général en deux temps

- désignation d'un élément du dessin,

- désignation d'un point de l'écran permettant de localiser l'emplacement définitif (ou temporaire) d'affichage de l'élément désigné. .

Les . primitives fournies permettent de simuler cette fonction. Cependant, une technique fort intéressante qui consiste à déplacer l'élément désigné à l'aide d'un dispositif de désignation (photostyle ou tablette ou autre) jusqu'à ce que la position idéale soit atteinte ne peut être réalisée sans un coût logiciel exorbitant. Nous proposons alors d'utiliser la primitive suivante

PLACER (nb, nfig, tabfig, tabcorrel, tabx, taby)

qui permet à l'opérateur de répéter nb placements, la trace des travaux effectués étant conservée dans les tableaux *tabfig, tabcorrel* (éléments désignés) et *tabx, taby* (emplacements définitifs retenus). Ainsi, les techniques mentionnées ci-dessus pourront être réalisées dans de bonnes conditions au niveau du logiciel de base.

L'introduction de cette nouvelle primitive montre que les combinaisons d'actions que nous avons proposées en introduisant un facteur de répétition ne suffisent pas. En effet, nous avons ainsi proposé une combinaison "verticale", c'est-à-dire ne mettant en jeu que le même type d'actions élémentaires.Il arrive cependant que pour certains types d'interaction, il soit nécessaire de programmer la situation suivante (par exemple) :

- donner la possibilité d'attendre soit un menu, soit un ensemble d'identifications, pour pouvoir désigner des éléments puis l'action à faire, puis une autre série d'éléments, une nouvelle action à réaliser etc.. On voit donc qu'il s'agit maintenant de réaliser des enchaînements d'actions de natures différentes. Nous parlerons alors de "combinaison" horizontale.

Pour réaliser une combinaison horizontale, nous proposons d'utiliser deux types de primitives :

- des déclarations d'actions,

- des activations d'actions de dialogue

Les déclarations d'action permettront de spécifier des couches horizontales d'actions en définissant en fait les interfaces dans lesquels seront rangés les résultats. On remarquera au passage l'analogie avec les déclarations de données. Ces déclarations peuvent par exemple être réalisée par les primitives de dialogue définies précédemment en 2.35 auxquelles on ajoute un paramètre d'identification. Ces primitives ne sont donc pas opératoires.

La mise en fonction se fait par l'utilisation de la primitive

DIALOGUE (p,m,v,i)

où les variables p,m,v,i permettent de spécifier quel type d'action sera utilisé, avec la convention suivante :

$p,m,v,i = 0$	pas d'action de ce type
$p,m,v,i > 0$	une action de ce type, repérée par le numéro p,m,v, ou i (p pour position, m pour menu, v pour valeur et i pour identification).

Du point de vue de l'opérateur, il pourra donc enchaîner plusieurs actions de type différent sans être obligé de parcourir des listes de menus longues et fastidieuses. Un dernier problème se pose, qui est de savoir si l'utilisation de différents dispositifs pour chacune des fonctions pourra toujours se faire sans ambiguité maintenant que plusieurs fonctions peuvent cohabiter. La question est de savoir dans quelle mesure le logiciel peut déduire, lors de la première intervention, quelle est l'action envisagée en fonction du dispositif utilisé. En fait, une rapide étude de tous les cas possibles montre que les cas d'ambiguïté sont rares, correspondent à des cas de figure peu fréquents et peuvent être levés dans ces cas uniquement par utilisation d'un signal supplémentaire.

Nous donnons dans le tableau 2.6, pour un appareil disposant d'un photostyle, d'une tablette, d'un clavier alphanumérique et d'un clavier de fonctions, un exemple de cas possibles. Nous avons noté dans

chaque case la fonction réalisée par le dispositif de communication porté en abscisse (la lettre s indique une fonction système).

combinaison	photostyle	tablette	clavier alphanumérique	clavier fonc
P M V I	p m i	p	p v	m s
P M V	p m	p	p v	m s
P M I	p m i	p	p	m s
P V I	p i	p	p v	s
M V I	m i		v	m
P M	p m	p	m	m s
P V	p	p	p v	s
P I	p i	p	p	s
M V	m		v	m
M I	m i			m
V I	i		v	
P	p	p	p	s
M	m			m
V			v	
I	i			

Tableau 2.6 : composition horizontale de fonctions de dialogue

Nous ferons les remarques suivantes :

- il n'y a jamais de problème avec la tablette,
- le choix du photostyle pour une collecte de coordonnées est fait à l'aide d'une touche de fonction (touche système). Comme la différenciation entre un élément de menu et un élément de figure lors d'une identification est triviale, il n'y a aucun cas d'ambiguïté.
- les seuls problèmes interviennent lorsque deux fonctions doivent être réalisées soit par le clavier de fonctions, soit par le clavier alphanumérique. On constate alors que le nombre de cas ambigus est de 8, sur

un total de 45 possibilités offertes par le logiciel.

En fait les dernières ambiguïtés sont levées en utilisant une touche supplémentaire. L'opérateur est prévenu de l'ambiguïté par un signal quelconque et précise ses intentions à l'aide de cette touche. Dans toutes les autres situations, la mise en oeuvre n'est pas modifiée.

2.37 Critique

Dès sa mise en service le logiciel GRIGRI a rencontré un succès non négligeable auprès des utilisateurs. Les principales raisons de cette réussite tiennent aux qualités suivantes :

- cohérence de l'ensemble défini. En particulier, le fait que ce logiciel se cantonne à un rôle de logiciel élémentaire (aux quelques primitives de mise en page près) permet de couper court à toute tentative d'extension par des primitives trop dépendantes d'une application donnée,

- simplicité de la programmation par rapport à l'étendue des services rendus. Ceci est particulièrement sensible au niveau du dialogue, mais le fait de pouvoir construire toute une figure dans le même tableau en jouant sur les sections pour exprimer des graphismes différents est un autre exemple d'intérêt de ce logiciel. En effet, cette particularité permet de préserver, même au niveau de la programmation, la structure logique d'un dessin, en ne donnant qu'un seul ordre d'affichage pour l'ensemble de la figure, et non pour chaque section (cas des logiciels habituels).

- indépendance par rapport à l'environnement d'exploitation, assurant une utilisation sur des matériels différents, mais également à partir de FORTRAN, PL/1, ALGOL W et autres langages de même nature.

La principale critique formulée concerne le fait qu'il s'agisse d'un logiciel de base et qu'il ne soit pas accompagné de primitives de description géométrique et d'outils de visualisation, tant pour des scènes à deux dimensions que pour des scènes à trois dimensions. Nous décrivons dans le chapitre 3 un exemple de logiciel de visualisation de

scènes en trois dimensions qui est en cours d'exploitation. Le chapitre 4 donnera des exemples de logiciels de description qui utilisent, en fin de chaîne, GRIGRI.

La principale conclusion que nous voulons tirer est qu'il est donc possible de définir des logiciels graphiques de base sans utiliser le concept de point courant. GRIGRI assure une bonne partie des fonctions énumérées dans la proposition de norme américaine et est donc de ce point de vue entièrement compatible avec l'esprit de cette proposition.

2.4 CONCLUSION

Nous nous placerons sur le terrain de la normalisation des logiciels graphiques interactifs. La norme proposée par le groupe américain sera adoptée après quelques modifications quels que soient les travaux futurs qui seront faits. D'ores et déjà, la publication d'algorithmes graphiques est proposée dans une rubrique spéciale du Computer Graphics SIGGRAPH-ACM, à condition que les primitives utilisées soient celles spécifiées dans le rapport([Cla 77]). Cependant, nous pensons qu'il ne s'agira que d'une première norme, et que d'autres seront proposées.

En particulier, les idées qui ont été à la base de la conception de GRIGRI méritent d'être approfondies. L'écriture de nombreux programmes d'application devrait permettre en particulier de voir quelle influence le niveau des primitives proposées pourra avoir sur la structure des programmes graphiques. Nous pensons que les points clés de notre proposition sont les suivantes :

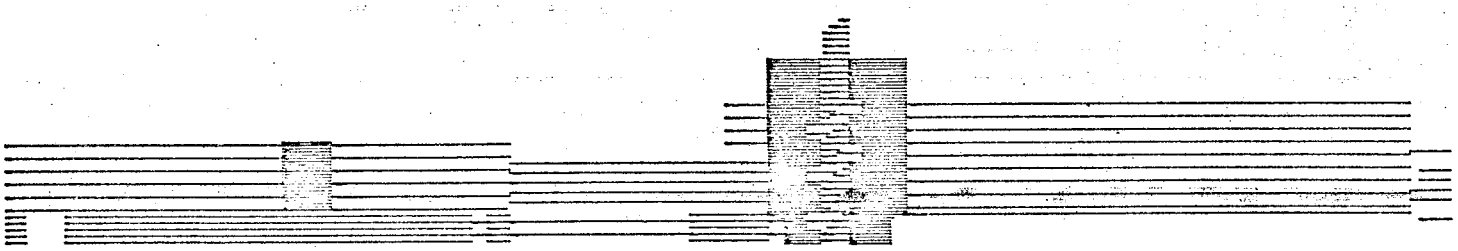
- unification des concepts de données à représenter et des graphismes à utiliser,
- claire séparation des types de données à traiter, tout en conservant une structure de définition commune,
- séparation de la description de dessin du mode de tracé, laissent libre champ aux techniques les plus performantes quel que soit le support du dessin. En particulier, on peut imaginer que l'optimisation du tracé soit un trait d'un logiciel élémentaire (par exemple pour une sortie sur traceur de courbes), cette tâche n'étant plus confiée au programmeur.

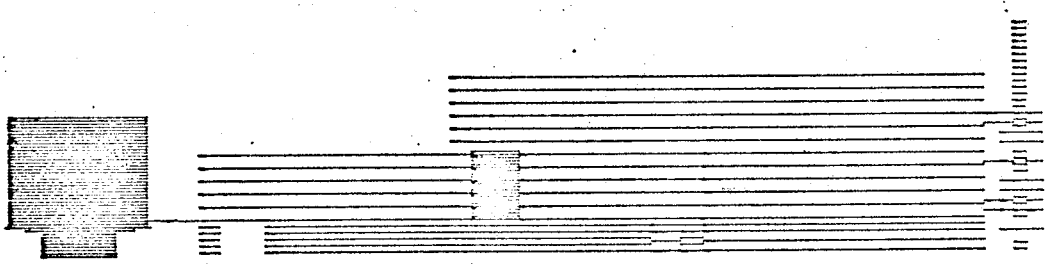
- symétrie des primitives d'affichage et de dialogue, assurant un même niveau de programmation et une mise en oeuvre totalement libérée des techniques au niveau langage machine. Nous notons en particulier que les pritimives de dialogue proposées par le groupe GSPC pourraient (devraient) être employées pour réaliser ces logiciels élémentaires. La transportabilité des programmes serait également assurée.

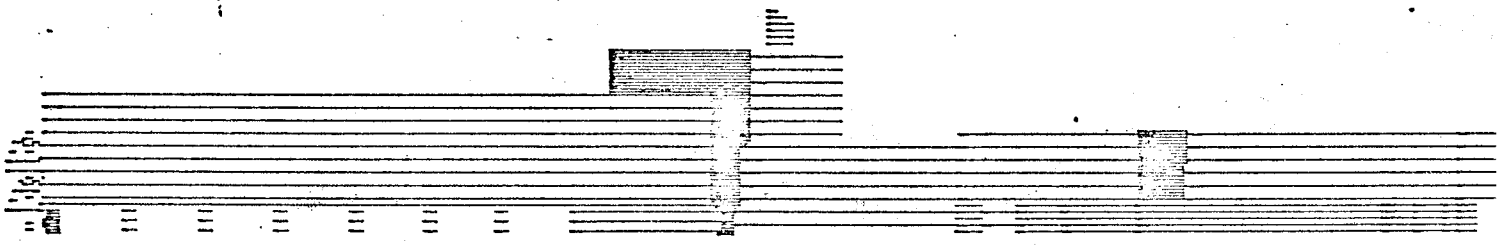
Le fait que GRIGRI soit utilisé prouve la réalité des idées que nous proposons. Une brêche a donc été ouverte dans le monopole *MOVE-DRAW*, et nous souhaitons qu'elle soit explorée par de nombreuses équipes de recherche.

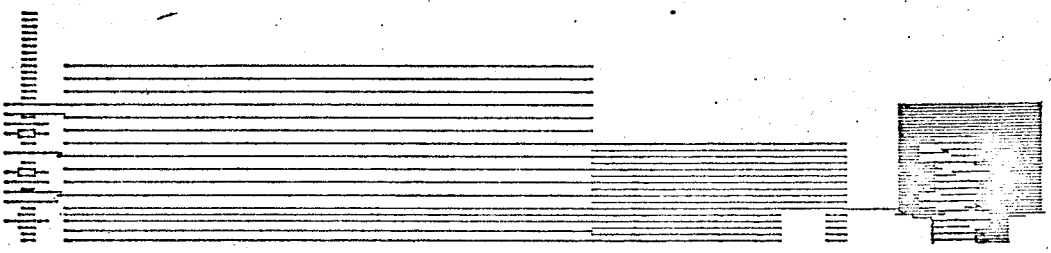
CHAPITRE 3

ELEMENTS POUR UN LOGICIEL DE VISUALISATION DE POLYEDRES









IMAG bâtiments a, b et c (CICG)

3. ELEMENTS POUR UN LOGICIEL DE VISUALISATION DE POLYEDRES

3.1 LE LOGICIEL VISU3D

Nous présentons dans ce chapitre l'expérience que nous avons pu acquérir lors du développement d'un logiciel interactif de visualisation de scènes composée d'ensembles de polyèdres (ce logiciel a pour nom VISU3D). Les polyèdres à visualiser sont décrits à l'aide d'un logiciel de description du type de FORTRAN-3D ou EUCLID (voir ANNEXE 1). Nous avons choisi de ne traiter que des polyèdres (et non des surfaces analytiques) parce que les méthodes développée jusqu'à maintenant ne permettent de présenter en un temps raisonnable que des objets de cette famille.

3.11 Description de la scène

Sous le nom générique de "polyèdres", les programmes de visualisation traiten en fait trois types d'objets bien différents :

a) des assemblages de "tiges" rectilignes (appelées *arêtes*) qui forment des graphes de R^3.

b) des assemblages de plaques polygonales planes (appelées *faces*),

c) des volumes polyèdraux.

On peut remarquer que :

- tout volume de type c) permet de définir un ensemble de plaques de type b).
- tout assemblage de type b) permet de constituer un ensemble de tiges de typ a).

La réciproque n'est pas vraie : un ensemble de tiges ne définit pas forcément des faces planes, un ensemble de faces planes ne définit pas forcément un volume.

Une scène peut être composée d'objets appartenant aux trois types définis ci-dessus. Nous nous sommes intéressés à des scènes composées uniquement d'assemblages de plaques polygonales planes, supposées dotées d'une seul couleur. Les données d'entrée du logiciel de visualisation sont constituées des informations suivantes :

- le nombre de sommets de la scène (*NBSOM*),
- la liste des coordonnées des sommets, exprimées dans le système utilisateur (tableaux *X*, *Y*, *Z*),
- le nombre de faces (*NBFACES*),
- la liste des faces (tableau *FACES*), chaque face étant représentée par sa cou leur suivie de la liste des sommets de contour, liste terminée par la valeu L'ordre des sommets dans la liste définit le sens de parcours du contour de face.
- un tableau de sections (tableau *TABLONG*), en complète analogie avec la noti de section définie en 2.321. Une section représente un ensemble de faces re présentées consécutivement dans le tableau *FACES*. Le ième élément de *TABLON* contient le nombre de faces de la ième section.

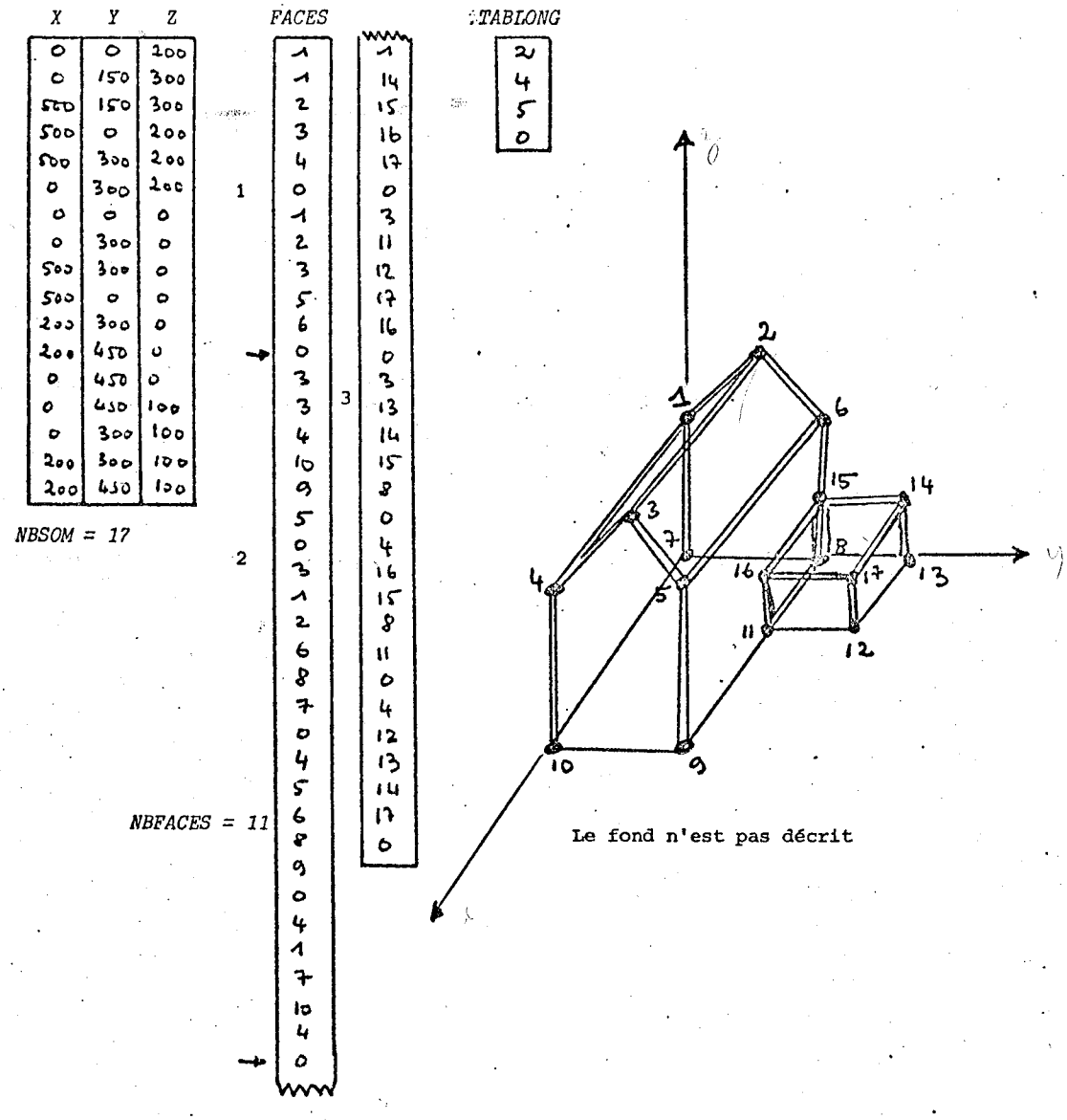


figure 3.1 Structure de données pour la description d'une scène

Le tableau 3.1 donne quelques chiffres concernant les principales caractérist
ques de quelques scènes traitées à l'aide de VISU3D dans le cadre d'applicati
réelles.
Les principales remarques que nous ferons sont les suivantes :

- le nombre total d'arêtes (*NBARETES*) représente la somme des nombres d'arête
  utilisées pour décrire chaque face. Ce nombre est au moins égal à *3 x NBFACE*
  (on traite au moins des triangles). L'expérience montre que la majorité des
  descriptions utilise des quadrilatères. Pour les évaluations futures, nous
  utiliserons *NBARETES = 4xNBFACES*.
- le nombre d'arêtes utilisées (*NBARUTIL*) représente les arêtes réellement
  existantes (on ne compte pas deux fois une arête appartenant à deux faces).
  En général, il y a toujours une proportion importante d'arêtes communes à
  deux ou plusieurs faces. Nous utiliserons par la suite l'approximation
  NBARUTIL = 3xNFACES.

		NBSOM	NBFACES	NBARETES	NBARUTIL	nombre d'arêtes par faces		NBARMOY
						nombre	fréquence	
IMAG bâtiments a, b, c		280	106	488	419	4 6 8 16	88 8 9 1	4.60
IMAG bâtiment d		288	93	413	371	4 5 6 8 12	79 1 8 4 1	4.44
IMAG Tour des Mathématiques Appliquées		198	81	372	329	4 5 6 8 10 12	66 6 2 4 1 2	4.60
Objets divers	1	48	60	264	131	4 6	48 12	4.4
	2	48	48	204	118	4 6	42 6	4.25
	3	20	14	64	49	3 4 10	4 8 2	4.57
	4	14	24	72	44	3	24	3.0
	5	40	22	120	100	4 20	20 2	5.45
Isoèdres	1	23	42	126	63	3	42	3.0
	2	43	82	246	123	3	62	3.0
	3	19	34	102	51	3	34	3.0
	4	39	74	222	111	3	74	3.0
"église"		45	38	148	78	3 4	14 24	3.89
Essais	1	88	66	264	220	4	66	4.0
	2	152	114	456	380	4	114	4.0
	3	56	38	168	140	4 6 8	32 4 2	4.42

tableau 3.1 Caractéristiques de quelques scènes traitées par VISU3D

3.12 Les commandes de VISU3D

L'objectif de ce logiciel est de permettre de voir, analyser et comprendre une scène. Les commandes proposées à l'opérateur se divisent en plusieurs catégor:

- commandes de *composition* de la scène à observer. Ces fonctions permettent s(d'extraire un sous-ensemble à examiner par combinaison de sections, soit d'éliminer une partie de la scène en pratiquant une coupe.

- commandes de *mise en place* dans l'espace, afin de pouvoir varier l'angle d'observation. L'opérateur peut demander des rotations autour des trois axe: de coordonnées, spécifier un point d'observation ou donner les valeurs des trois angles d'Euler.

- Commandes de *présentation* de la scène, permettant à l'observateur de choisi: un mode d'observation à travers une projection orthogonale ou une projectio: perspective, avec une éventuelle élimination des parties cachées.

- commandes *d'analyse* de la scène, permettant d'obtenir des renseignements concernant les dimensions de la scène, sa composition (sommets, faces).

- commandes de *mise au point* de la scène, permettant de modifier des valeurs erronnées (coordonnées d'un sommet, couleur d'une face, ordre des faces etc

Il faut remarquer que nous n'avons pas donné la possibilité de modi fier la structure même de la scène. Les seules opérations autorisées sont les transformations géométriques, qui ne sont pas répercutées sur la structure de données elle-même. On ne peut donc ajouter ou supprimer un élément de descrip tion. Nous avons volontairement cantonné VISU3D à un rôle d'observation et de contrôle, sans lui donner la possibilité de construction (et donc de descrip-tion).

Nous étudions dans les paragraphes suivants les principaux problèmes de réali sation de ce logiciel.

3.2 PRESENTATION DE LA STRUCTURE DE LA SCENE

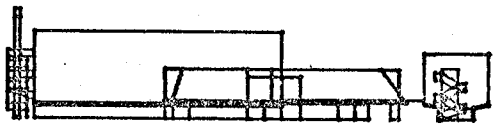
Nous nous intéressons ici à la présentation de l'ensemble d'arêtes issu des faces de la scène. Cette présentation est faite automatiquement en début d'étude, car elle permet de localiser très rapidement les erreurs de description (numérotation des sommets incorrecte, coordonnées erronnées etc...). Nou passons en revue les techniques algorithmiques liées aux commandes de compos tion, de mise en place et de présentation.

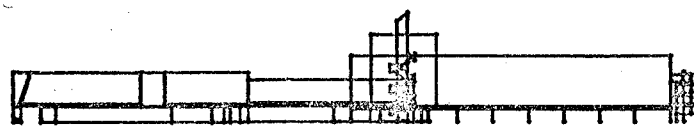
3.21 Composition de la scène à étudier

Nous présentons ici les techniques liées à l'obtention d'un sous-ensemble ou d'une partie de la scène.

En ce qui concerne la définition d'un sous-ensemble de la scène, e se fait en donnant le numéro des sections que l'on veut étudier. L'opérateur ainsi la possibilité d'examiner certaines parties de la scène pour mieux exe cer son contrôle. Il n'y a pas de difficultés particulières à mentionner, si l'utilisation d'une structure de données propre au logiciel, contenant des r seignements analogues à ceux décrits en 3.11, permettant de ne pas modifier structure initiale.

En ce qui concerne la définition d'un plan de coupe, nous utilisons une prés tation "trois vues" (voir plus loin en 3.22). L'opérateur définit la trace d plan de coupe sur l'un des plans de projection en donnant deux points de cel ci (voir figure 3.2).





COTE

FACE

AMPLIX= 78.900

AMPLIY= 110.550

AMPLIZ= 18.000

IMAG bâtiments A,B,C

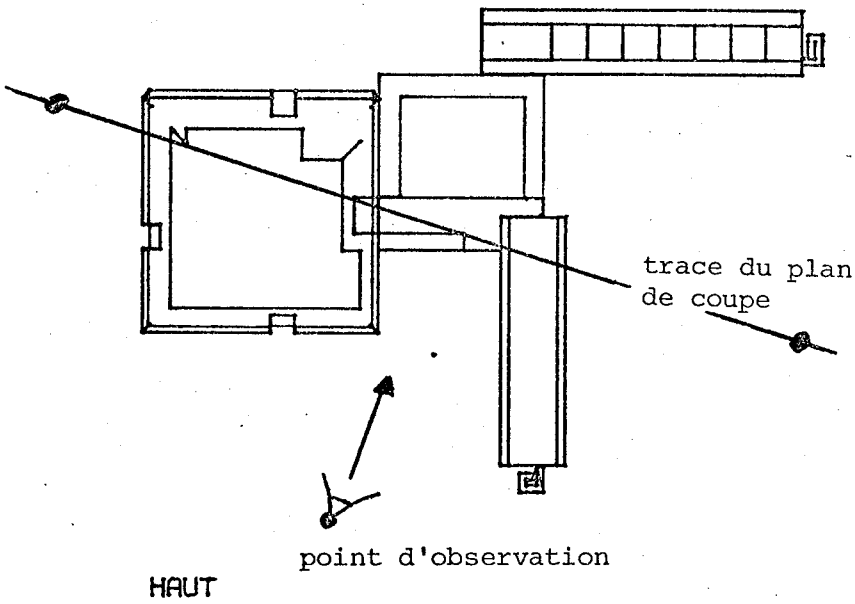


figure 3.2 Définition d'un plan de coupe

Le plan de coupe est défini par le plan perpendiculaire au plan de projectic de la vue choisie passant par les deux points désignés. L'opérateur définit la direction de vue en donnant un troisième point sur l'écran.

Les faces définissant la scène se classent alors en trois catégories :

- faces contenues entièrement dans le demi-espace comprenant le point de vue elles doivent être rejetées,
- faces contenues entièrement dans le demi-espace ne comprenant pas le point vue : elles sont à présenter telles quelles,
- faces possédant une intersection avec le plan de coupe. Elles doivent alor être étudiées, de manière à déterminer quelle portion est comprise dans le demi-espace à projeter.

Le problème consiste donc à étudier, pour une face, une suite de sommets $s_1,\ldots,s_p$ pour en déduire une nouvelle suite (éventuellement vide) $s'_1, \ldots,s'_q$ de sommets tous situés dans le demi-espace à conserver. Pour ce faire, nous utilisons l'algorithme de SUTHERLAND et HODGMAN [SuH 74] , qui consiste à par courir le polygone arête par arête en examinant la position de l'extrèmité fi nale de chaque arête par rapport au plan de découpage. Quatre cas se présente (voir figure 3.3) :

a) - les deux extrèmités de l'arête sont dans le demi-espace à conserver : sortir le sommet final (noté P).

b) - aucune extrêmité de l'arête n'est située dans le demi-espace à conserver ne rien faire.

c) - l'arête quitte le demi-espace à conserver : sortir le point d'intersecti de l'arête et du plan de coupe (noté I).

d) - l'arête entre dans le demi-espace à conserver : sortir le point d'intersection I et l'extrêmité finale P.

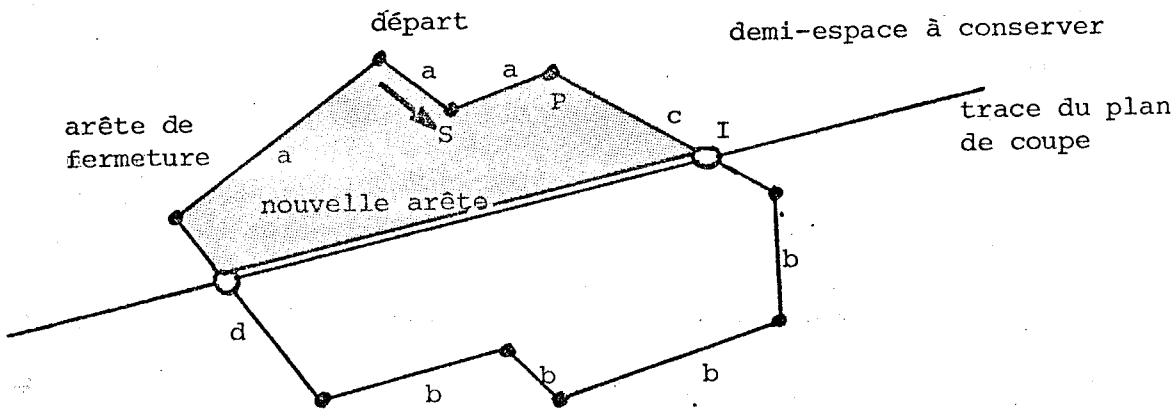


figure 3.3 Découpage d'une face

L'algorithme de traitement d'une face peut s'écrire :

```
DECOUPE_FACE ;
début {découpage d'une face par rapport à un plan de coupe ; les fonctions
       PREMIER_SOMMET et SOMMET_SUIVANT permettent de parcourir la suite d'arêt
       bordant la face. Cette suite est terminée par la valeur 0}
  entier P ; logique PREMIER ;
  réel XP,YP,ZP, XS,YS,ZS, XF,YF,ZF, XI,YI,ZI ;
  P := PREMIER_SOMMET ; PREMIER := vrai ;
  tantque P¬ = 0 faire
  début
    XP := X(P); YP := Y(P); ZP := Z(P) ; {coordonnées du sommet}
    si PREMIER
    alors début
            PREMIER := faux ;
            XF := XP; YF := YP; ZF := ZP {conservation du premier sommet}
          fin
    sinon si ARETE_COUPE {test par rapport au plan de coupe}
          alors début
                  CALCUL_INTERSECTION(XI,YI,ZI) ;
                  SORTIR (XI,YI,ZI)
                fin ;
    XS := XP; YS := YP; ZS := ZP; {extrémité finale devient extrémité initial
    si S_VISIBLE
    alors SORTIR (XS,YS,ZS) ;
    SOMMET_SUIVANT
  fin ;
  {traitement de l'arête de fermeture}
  si SF_COUPE
  alors début
          CALCUL_INTERSECTION (XI,YI,ZI) ;
          SORTIR(XI,YI,ZI)
        fin
fin DECOUPE_FACE ;
```

Trois remarques permettent de faciliter la mise en oeuvre de l'algorithme :

- la présentation de la coupe se fait "vue de face". Il y a alors tout intérêt à effectuer dans un premier temps une rotation de la scène de manière à ce q le plan de coupe soit perpendiculaire à l'axe des x, le point de vue se trou vant du côté des x positifs. Le test permettant de savoir s'il faut conserve ou non un sommet se fera alors simplement en comparant la coordonnée en x à la cote du plan de coupe (au lieu de reporter les trois coordonnées dans l'équation du plan).

- une fois cette rotation effectuée, le calcul de l'intersection d'une arête avec le plan de coupe se calcule très simplement de la façon suivante :

$$p_i = p_1 + (p_2 - p_1) \frac{|x_1 - x|}{|(x_1 - x)| + |(x_2 - x)|} \qquad x = \text{cote de la coupe}$$

- la procédure *SORTIR* a deux comportements différents :

  . lorsqu'il s'agit d'un sommet du polygone en cours de traitement, il su fit de ranger son numéro dans la liste représentant la face en cours d construction.

  . lorsqu'il s'agit d'une intersection, il faut créer un nouveau sommet (donc ranger les coordonnées XI,YI,ZI) et ajouter le numéro de ce nouv sommet dans la liste constituant la nouvelle face.

Nous conclurons en faisant remarquer que les commandes de composit de scène conduisent à créer de nouvelles données à partir des données initia (nouveaux sommets, nouvelles faces). Il y aura donc création d'une scène inte médiaire sur laquelle s'appliqueront les autres commandes. Nous n'avons cepen dant pas autorisé l'utilisation des commandes de composition sur un sous-ensemble de la scène.

3.22 Techniques de projection et transformations géométriques

Nous regroupons ici les techniques liées aux commandes de mise en place dans l'espace et de présentation. En effet, la réalisation de ces commandes fait intervenir des matrices de transformation qu'il est intéressant de combiner produit, afin d'optimiser les calculs pour l'obtention de la figure finale. Nous traiterons des techniques de projection et des techniques de transforma géométrique.

Les techniques de projection sont utilisées pour passer de la scèn en trois dimensions à une scène en deux dimensions qui permet de donner une représentation graphique de la scène originale. La technique la plus simple consiste à faire une projection orthogonale sur les plans xoy, yoz ou xoz.

Nous avons choisi d'utiliser la norme habituelle de présentation en trois vues (face, haut, profil) (voir [Rib 74]) correspondant aux points de vue de la figure 3.4 :

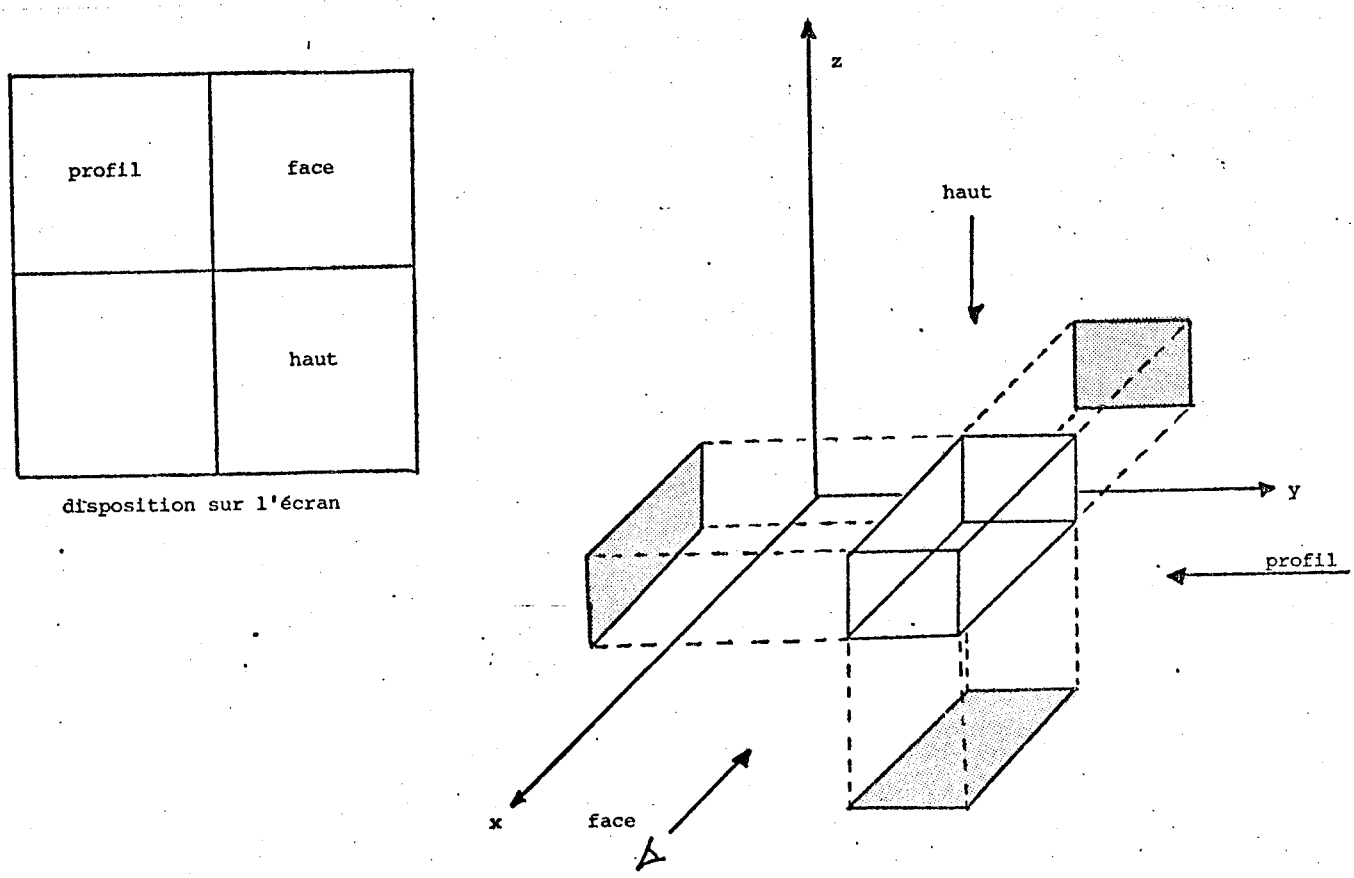


figure 3.4 Observation de la scène pour une présentation trois vues

Un exemple de cette présentation est donné en figure 3.5.

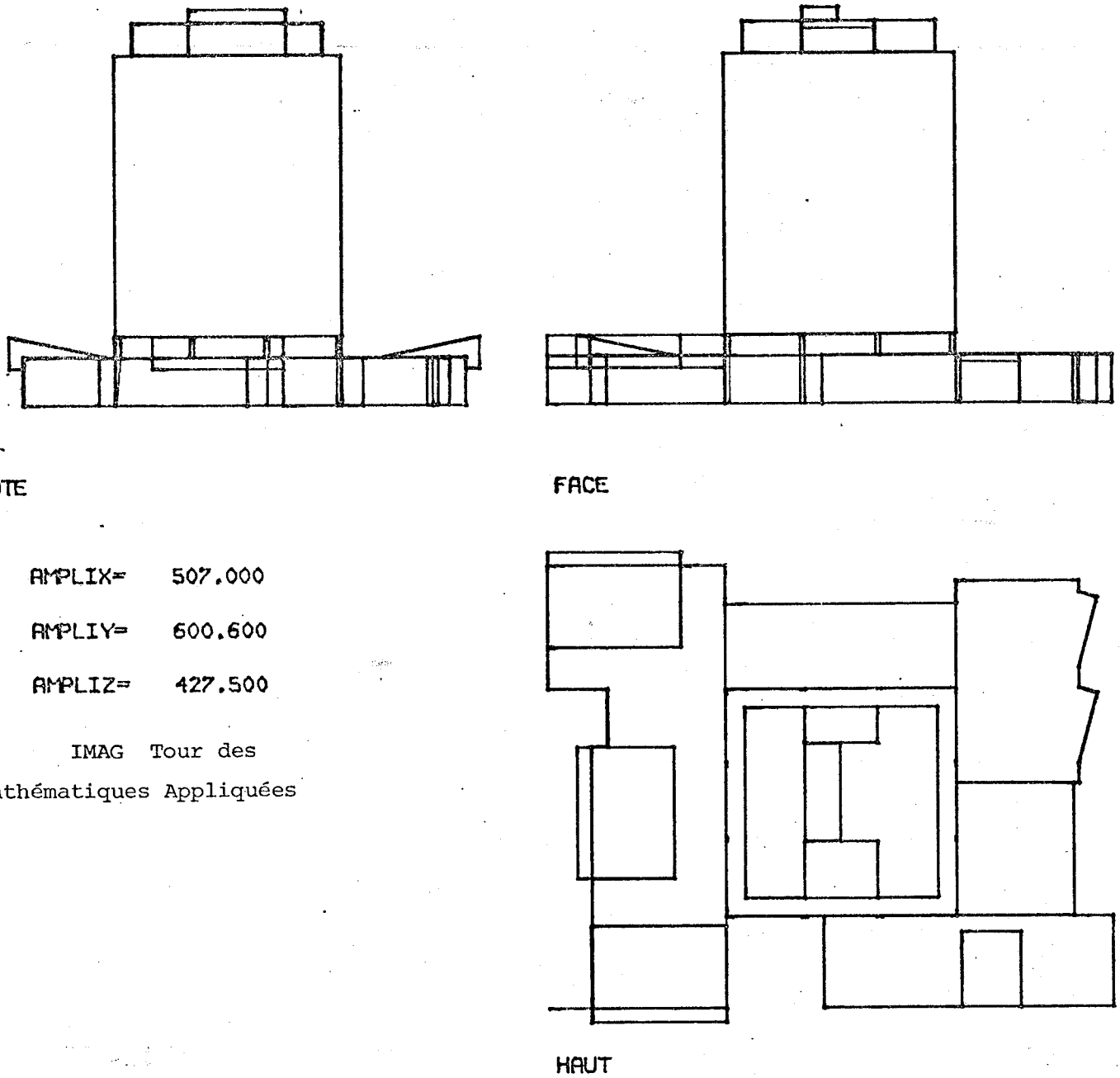


figure 3.5 Présentation "trois vues" d'une scène

On peut constater qu'il est nécessaire de pouvoir changer de point de vue, a: de mieux mettre en valeur l'ensemble des arêtes. Ceci se fait par combinaiso: de rotations autour des axes de coordonnées, en multipliant les coordonnées des sommets par les matrices de rotation suivantes (en coordonnées homogènes

$$R_x = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad R_y = \begin{vmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \quad R_z = \begin{vmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

La figure 3.6 présente une même scène sous différents angles de vue.

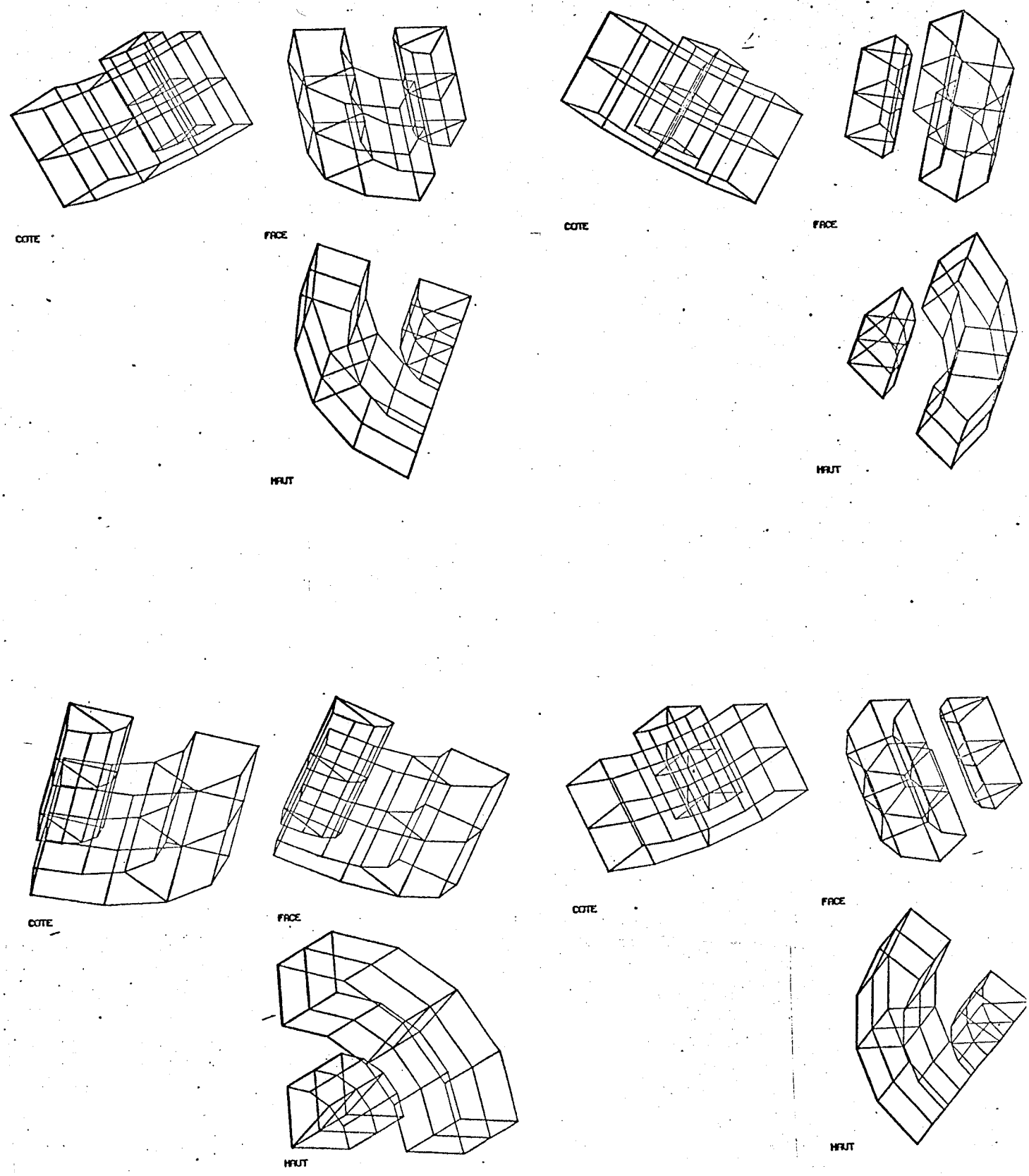


figure 3.6 Présentation d'une scène sous différents angles de vue

Une technique de projection en "vue perspective" est également proposée à l'opérateur, lui permettant d'obtenir une représentation plus proche de ce que l'oeil est habitué à analyser. Dans ce cas, seule la vue de face est présentée. Cette présentation n'est pas proposée automatiquement car, si elle es intéressante pour le cas de l'architecture (par exemple), elle se révèle tout fait superflue dans d'autres cas (étude d'un entrefer d'électro-aimant par la méthode des éléments finis par exemple). L'effet de perspective est obtenu en centrant la scène sur l'origine des axes, et en divisant les coordonnées y et par la cote en x. Nous sommes en effet dans le cas de la figure 3.6.

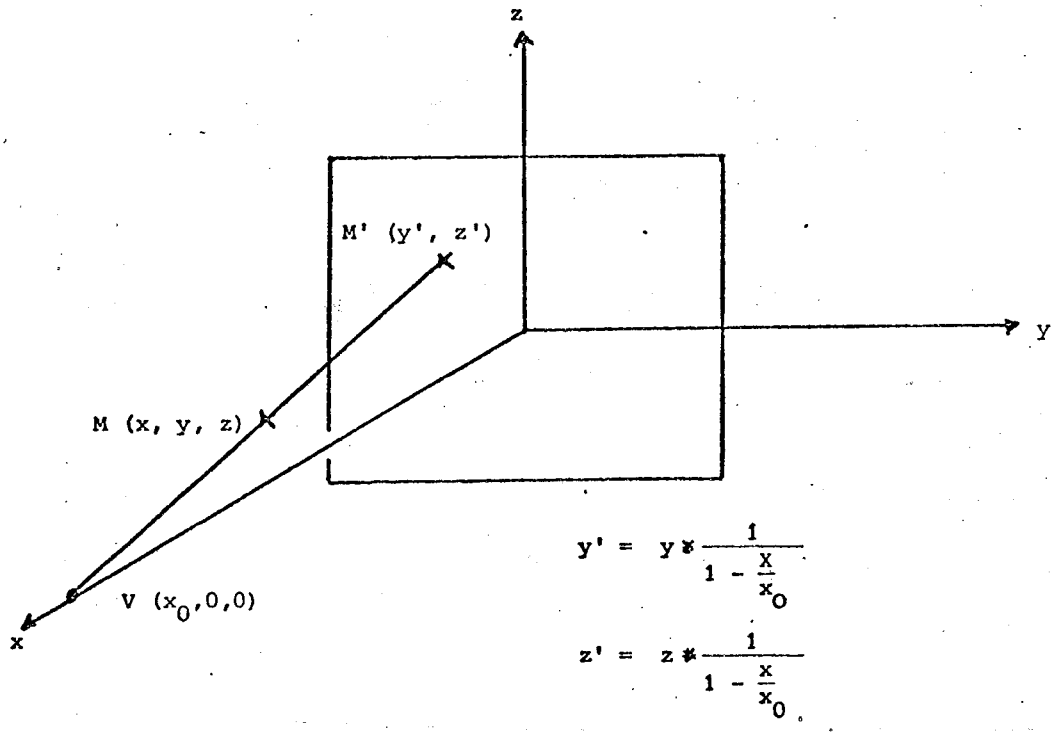


Figure 3.6 Calcul de la perspective

La matrice de transformation utilisée est la suivante :

$$P = \begin{vmatrix} 0 & 0 & 0 & -\frac{1}{x_0} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Le point de vue peut être choisi par l'utilisateur, ou calculé automatiquement de manière à ce que l'angle défini par le point de vue et les milieux des bord gauche et droit de l'écran soit d'environ 32° (valeur utilisée par les architectes voir [Str -].

Il est évident que d'autres projections pourraient être utilisées (cavalière, dimétrique, trimétrique ou autre). Pour les applications que nous avons traité la projection perspective était la plus intéressante du point de vue visuel.

La figure 3.7 donne un exemple de présentation avec vue perspective.

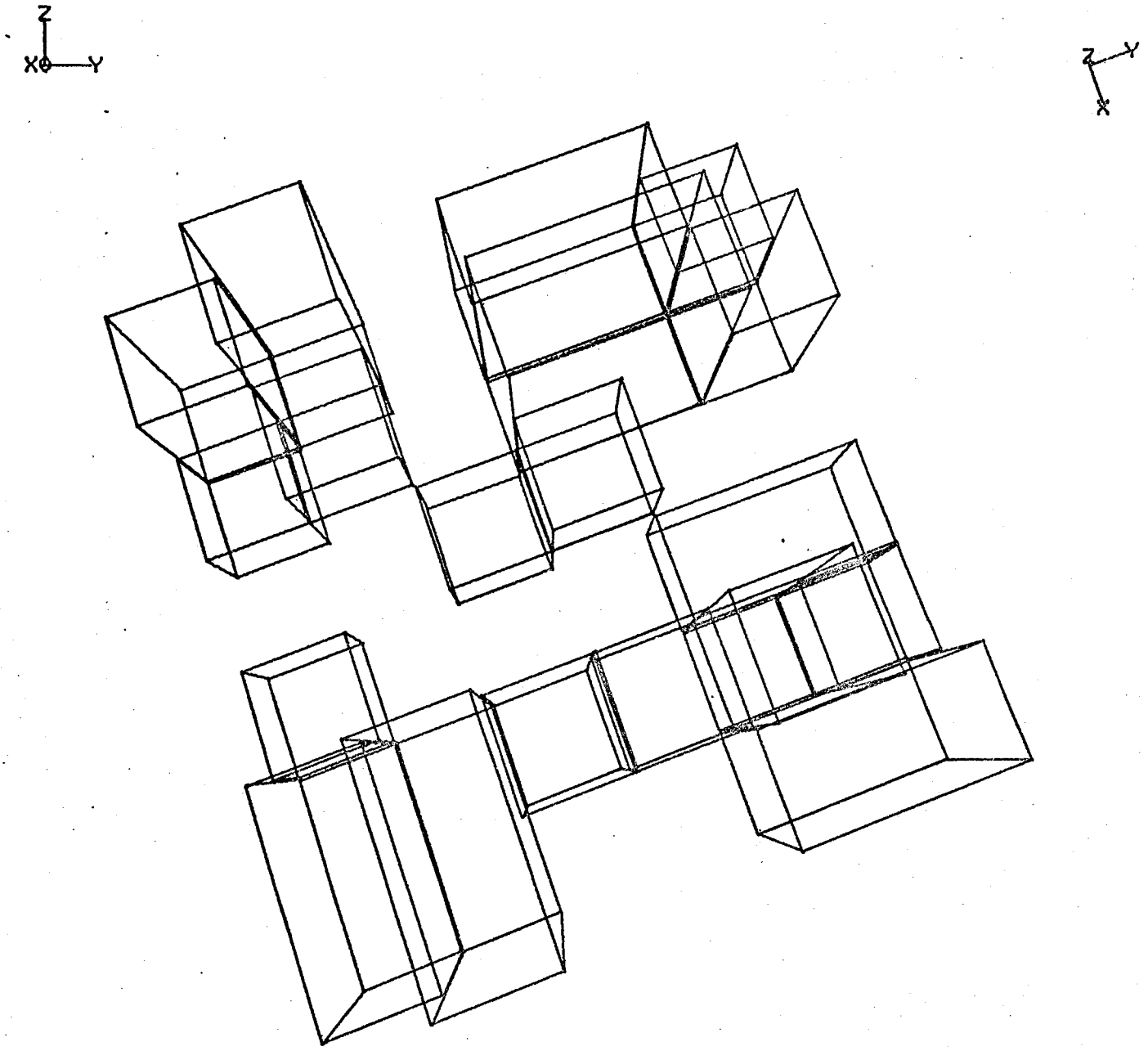


figure 3.7 Présentation perspective (vue en plongée)

Un dernier type de transformation concerne les échelles sur les axes x, y et

En effet, le système de coordonnées employé par l'utilisateur est quelconque niveau du programme d'application, et il est nécessaire d'ajuster la taille du dessin sur l'écran. Nous n'avons pas inclus dans VISU3D de mise à l'échell automatique (par exemple pour couvrir toute la surface de l'écran), cette pra tique étant fort gênante lorsque l'on désire étudier des objets de dimension différentes en gardant la même échelle d'une scène à l'autre. L'opérateur peu demander un changement d'échelle sur chacun des trois axes. Le calcul se fait à l'aide de la matrice de transformation suivante :

$$E = \begin{vmatrix} e_x & 0 & 0 & 0 \\ 0 & e_y & 0 & 0 \\ 0 & 0 & e_z & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

L'ensemble des transformations étant commandé à l'aide d'un menu multiple (cf 2.355), les calculs sont faits à l'aide d'une seule matrice, produit des différentes matrices évoquées ci-dessus.

3.23 L'élimination des parties cachées

Lorsque la scène est trop complexe, la présentation sous forme d'ensembles d'arêtes ne permet plus d'apprécier les dispositions relatives des différent faces qui la composent. Il est alors nécessaire de proposer une représentati qui tienne compte du fait que les faces sont supposées opaques : un observat ne voit plus alors qu'un sous-ensemble de celles-ci, sous-ensemble formé des faces (ou portions de faces) situées le plus près de lui, les autres étant cachées à ses yeux.

De nombreux algorithmes d'élimination de parties cachées ont été publiés (voir Bibliographie). Nous proposons de distinguer deux grandes caté gories :

- Les algorithmes qui présentent une scène sous la forme d'un dessin constitué d'un ensemble de traits correspondant aux arêtes (ou parties d'arêtes) visibles par l'observateur. On parle dans ce cas *d'élimination de lignes cachées* ("hidden lines algorithm").

- les algorithmes qui présentent une scène sous la forme d'une image constituée d'un ensemble de taches correspondant aux faces (ou portions de faces) visibles par l'observateur. On parle alors de *recherche des surfaces visibles* ("visible surfaces algorithm").

Les algorithmes de composition de dessins vont traiter les arêtes bordant les faces de la scène, pour déterminer quelle arêtes (ou parties d'arêtes sont visibles et doivent être dessinées. Deux approches ont été utilisées :

- comparaison des arêtes par rapport à des volumes :
  algorithme de ROBERTS [Rob 63] qui traite des polyèdres convexes,

- comparaison des arêtes aux faces : algorithmes de APPEL [App 67] . LOUTREL [Lou 70] (polyèdres quelconques), de GALIMBERTI et MONTANARI [GaM 69] (solides particuliers).

Cependant, la complexité de ces algorithmes étant fonction du produit du nombre de face par le nombre d'arêtes, une autre famille d'algorithmes a vu le jour à partir de l'idée suivante : pour accélérer le processus de dessin, on va essayer de dessiner non plus arête par arête mais face par face (plus exactement on cherche à dessiner d'un seul coup le contour d'une face). En effet, si l'on parvient à déterminer (par exemple) qu'une face est toute seule dans une portion d'espace donné, il est inutile de comparer ses arêtes aux autres faces de la scène : on peut dessiner directement son contour. Le problème est donc maintenant de comparer, sur une portion d'espace donnée, des faces entre elles. Si la compléxité des tests est au moins quatre fois inférieure à celle des tests utilisés précédemment, ces algorithmes seront a priori plus rapides (puisqu'il y en moyenne au moins quatre arêtes par face). L'algorithme le plus connu est celui de WARNOCK [War 69].

Les algorithmes de composition d'image ont pour but de calculer quelle est la couleur à affecter à chaque point de la surface de visualisati en fonction de la portion de scène vue par l'observateur. L'algorithme le pl simple que l'on puisse imaginer part de la remarque suivante : supposons qu' chaque point de l'écran nous menions une droite perpendiculaire à celui-ci. Chacune de ces droites va percer 0, une, deux,...,n faces de la scène consid rés. Si l'on calcule, pour chacune de ces droites, quel est le point d'intersection le plus proche de l'oeil, il suffit de noter la couleur assoc à la face ainsi retenue. L'ensemble des points collectés définira l'image fi le. Etant donné que pour un écran télévision la définition courante est de *512×512* points, il est clair que la mise en oeuvre d'une telle idée ne peut faire en calculant *250 000* × *NBFACES* intersections.

Deux techniques de réduction de cette complexité sont utilisées :

- les algorithmes qui construisent l'image ligne par ligne, une ligne corres pondant à l'étude d'une section de la scène à représenter par un plan per- pendiculaire à la surface de visualisation. Le problème revient alors à ét dier les positions relatives des segments de droite ainsi déterminés. L'image est obtenue en étudiant un nombre de sections relativement restrei Les algorithmes les plus connus sont ceux de WATKINS [Wat 70] , ROMNEY [Rom 70] ou BOUKNIGHT [Bou 70] .

- les algorithmes qui construisent l'image point par point, en procédant à u classement des faces par rapport à l'observateur et en projettant ces face dans l'ordre ainsi déterminé. Le plus connu est l'algorithme de NEWELL, NEWELL et SANCHA [NNS 72] .

Bien que l'on connaisse une bonne centaine d'articles sur le sujet le point remarquable est qu'à chaque congrès un (ou plusieurs) algorithme es décrit. Chaque publication annonce que l'algorithme le plus simple et le plu efficace a été trouvé. Le problème du choix d'un algorithme pour l'insérer d un logiciel de présentation de polyèdres se posant, nous avons réalisé quatr programmes correspondant à quatre algorithmes représentatifs des familles év quées ci-dessus :

- production de dessins : algorithme de GALIMBERTI-MONTANARI (3.31), algorithme de WARNOCK (3.32),

- production d'images : algorithme de WATKINS (3.41), algorithme de NEWELL, NEWELL et SANCHA (3.42).

Les difficultés de réalisation ainsi que les résultats et conclusions de cette expérience sont décrits en détail dans les paragraphes qui suivent.

3.3 ELIMINATION DE PARTIES CACHEES : COMPOSITION DE DESSINS

Nous nous intéressons dans ce paragraphe aux algorithmes qui permettent de n'afficher que les arêtes ou parties d'arêtes visibles pour un observateur si à l'infini, les facettes étant supposées opaques. Le problème est alors de déterminer les arêtes appartenant aux faces visibles pour l'observateur. Nous étudierons :

- l'algorithme de GALIMBERTI et MONTANARI qui permet d'obtenir une solution exacte du problème posé,

- l'algorithme de WARNOCK qui donne une solution approchée du même problème.

3.31 L'algorithme de GALIMBERTI et MONTANARI

3.311 - Principe

Les polyèdres concernés sont des solides tels que chaque arête appartient à deux faces, et deux seulement. La structure de données présentée en 3.11 est suffisante, mais l'orientation des faces doit être obligatoirement choisie de telle sorte qu'une arête appartenant à deux faces soit parcourue dans un sens pour une face $(s_i s_j)$ et dans l'autre sens pour l'autre face $(s_j s_i)$. De plus, sens de parcours doit permettre de calculer une normale à la face dirigée ver l'extérieur du polyèdre auquel appartient la face (l'extérieur est défini cor la portion de l'espace contenant les points à l'infini).

Le calcul se fait à partir de la formule suivante :

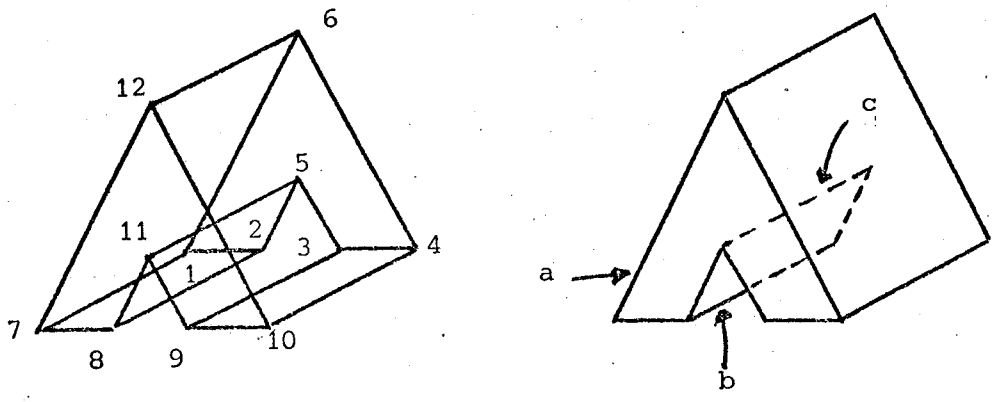
$$\vec{N}_f = 2\vec{u}\ |A(f)| = \sum_{i=2}^{i=p-1} \vec{s_1 s_i} \wedge \vec{s_1 s_{i+1}}$$

A aire de la face

L'idée de base de l'algorithme est d'étudier toutes les arêtes de la scène p rapport à toutes les faces, de manière à déterminer les portions vues ou cac

Dans un premier temps, on vérifie leur degré de visibilité (cf figure 3.9) :

- arête entièrement visible (a),
- arête partiellement visible (b),
- arête entièrement cachée (c).



polyèdre à étudier — après la première phase

figure 3.9 Traitement d'un polyèdre non convexe

Cette phase se décompose en deux temps :

- calcul des relations spatiales entre les arêtes et les faces,
- calcul des parties effectivement vues.

La propriété que l'on étudie est la suivante : un point M sera caché par la face f si et seulement si

- dans l'espace R^3 le segment de droite joignant le point M au point de vue F perce le plan de la face f,
- la projection du point de M appartient à l'intérieur de la projection de la face f sur le plan de l'image.

Si le point M ne vérifie pas l'une de ces deux conditions, il est visible.

Pour préparer les calculs, on étudie la position des extrèmités de chaque arê par rapport au plan de chaque face. On a alors les quatre cas suivants pour l'arête sp :

s	p	conclusion
même côté que F	même côté que F	arête potientiellement vis
même côté que F	côté opposé à F	étudier le segment Qp
côté opposé à F	même côté que F	étudier le segment sQ
côté opposé à F	côté opposé à F	étudier l'arête

remarque : un point appartenant à la face f sera considéré "du côté opposé à sauf dans le premier cas de figure (deux points appartenant à f)

On constitue ainsi, pour chaque arête, une liste des faces susceptibles de la masquer partiellement ou totalement, en conservant le point d'intersection av le plan de la face concernée.

Ces renseignements collectés, on étudie alors chaque arête point par point, d manière à produire le dessin final. Cette étude se fait en choisissant un pa pour parcourir les arêtes. Pour chacun des points obtenus, on détermine s'il est visible ou non, étant donné l'ensemble de faces suceptibles de le cacher. Il est clair que cette étude est extrêmement coûteuse, même si le pas choisi est relativement peu fin (par exemple de l'ordre d'un millimètre sur l'écran)

3.312 - Réduction de la complexité

Nous étudions sous cette rubrique les techniques qui visent à réduire le nom d'éléments étudiés lors de l'élimination des parties cachées.

Dans le cas de l'algorithme de GALIMBERTI-MONTANARI, le procédé essentiel vi à diminuer le nombre d'arêtes à étudier finement. L'idée de base consiste à chercher les arêtes qui sont forcément cachées. Cette recherche se fait en classant les faces de la scène en deux catégories :

- les faces dont la normale est tournée vers l'observateur (faces potentiellement visibles),

- les faces dont la normale est tournée dans le sens de visée (faces cachées)

On élimine alors toutes les arêtes appartenant à deux faces cachées. En effet, elles sont forcément invisibles pour l'observateur.

L'algorithme général se décompose donc en deux phases :

- une élimination des arêtes forcément cachées, phase très rapide et suffisant dans le cas des polyèdres convexes (cas rarissime dans la pratique),

- une étude point par point des arêtes subsistant après le premier pas. La figure 3.9(b) donne un exemple de polyèdre non convexe après la première phase

3.313 - Structure de données utilisée

Nous donnons simplement un schéma de la structure de données, afin de permettr une évaluation grossière de la place mémoire nécessaire pour mener à bien l'étude décrite ci-dessus.

nom du tableau	contenu	nombre d'éléments
* *X,Y,Z*	coordonnées des sommets	*3* NBSOM*
S,P	ensemble des arêtes, S début	
	P fin	*2*NBARETES*
FACES	description des faces	
	(couleur + 4**NBFACES* sommets + fin de liste)	*6*NBFACES*
PREMAR	pointeur début description des faces	*NBFACES*
PT1,PT2	faces associées à une arête	*2*NBARETES*
* *NORM*	composantes des normales aux faces	*3*NBFACES*
VISIBLE	face visible ou cachée	*NBFACES*
DEV	permet de noter, pour une arête	
	sa situation par rapport aux faces	*2*NBFACES*
NUMFACE	faces à étudier, pour une arête	*NBFACES*
* *LAMBDA*	point d'intersection d'une arête	
	avec les faces	*NBFACES*

Si on compte un mot (4 octets) par éléments, on voit que cet algorithme nécessite *3 * NBSOM + 31 * NBFACES* mots. Il s'agit d'une surestimation, puisque l'arrangement des tableaux peut être étudié de manière à gagner de la place. En particulier, seuls les tableaux notés d'une étoile (*) contiennent des réels. Tous les autres pourraient très bien utiliser des demi-mots (2 octets) pour représenter leurs informations.

3.314 - Critique

L'algorithme de GALIMBERTI-MONTANARI appartient à la famille des algorithmes dits "à résolution exacte". On trouve également dans cette catégorie les algorithmes de LOUTREL et APPEL. De nombreux produits commerciaux ont été développés à partir de ces algorithmes, parmi lesquels nous citerons EUCLID et EUKLID (!) [Eng 73] . Une liste d'un programme de cette famille est donnée dans [Enc 75] .

Les principales caractéristiques de cette famille sont :

- la grande précision de la solution calculée,
- une résolution dans l'espace utilisateur, sans tenir compte de la précision des dispositifs de visualisation.

La principale critique que l'on puisse faire à l'algorithme de GALIMBERTI et MONTANARI est qu'il s'adresse à une classe trop restreinte de polyèdres. Par exemple, il ne permet pas de représenter correctement un parallèlépipède ouvert.

De plus, l'obligation d'orienter les faces peut conduire à des difficultés :

- si les polyèdres sont définis par programme et la scène constituée par leur assemblage, il est alors très facile d'engendrer par calcul des faces orientées correctement (voir un exemple en 4.22) ;
- si les polyèdres sont donnés sans que la condition d'orientation soit observée, il est toujours possible de se ramener au cas où une arête appartient à deux faces seulement (quitte à rajouter des sommets et des arêtes fictives) mais le principal problème est alors de déterminer quelle orientation choisir de manière à ce que le calcul de la normale soit correct,

- si les polyèdres sont définis en mode dialogué, la vérification de l'orientation des faces fournies par l'utilisateur conduit à des procédures fastidieuses pour celui-ci.

La préparation des données est donc assez délicate.

Nous terminerons en cherchant à déterminer quels sont les éléments de la scène qui influent le plus sur les temps de calcul. Il est évident qu'u première approximation du coût de l'algorithme le plus simple va donner des temps de l'ordre de

$$C * NBFACES * NBARETES \sim C * 4 * NBFACES^2$$

Le but de la première phase des calculs vise à diminuer le nombre d'arêtes à traiter point par point en utilisant un test dont la complexité est relativement faible. Le traitement fait intervenir le calcul de la normale à chaque faces, ce qui donne un coût de l'ordre de

$$C_1 * 4 * NBFACES^2$$

où C_1 est beaucoup plus petit que C.

On suppose que $NBELIM$ arêtes sont éliminées. Il reste donc à traiter $NBRESTE = (NBARUTIL - NBELIM) = (3 * NBFACES - NBELIM)$ arêtes.

En ce qui concerne la deuxième phase, pour chaque arête restante on effectue

- le calcul de la position par rapport aux différentes faces

$$\sim C_3 * NBFACES$$

- la projection par rapport aux faces

$$\sim C_4 * NBFACES * NBARETES \sim C_4 * NBFACES^2 * 4$$

- étude point par point, qui dépend de la longueur L_i de l'arête étudiée et c pas employé

$$C_5 * L_i$$

Le coût global de traitement d'une scène est donc de l'ordre de

$$C * NBFACES^2 + C' * NBFACES + C'' * L$$

où L est la longueur totale des arêtes mesurée dans l'espace utilisateur. On peut donc constater que ce coût est grossièrement proportionnel au carré du nombre d'éléments formant la scène (décompté par rapport aux faces) et à l longueur totale des arêtes formant celle-ci. Si la première phase joue un rô important puisqu'elle élimine rapidement un certain nombre d'arêtes, en géné ce nombre est relativement restreint (de l'ordre du quart en moyenne). L'algorithme de MONTANARI-GALIMBERTI sera donc toujours relativement coûteux.

La figure 3.10 présente deux vues de la même scène :

- avant élimination des lignes cachées,
- après élimination des lignes cachées grâce à l'algorithme de GALIMBERTI-MONTANARI

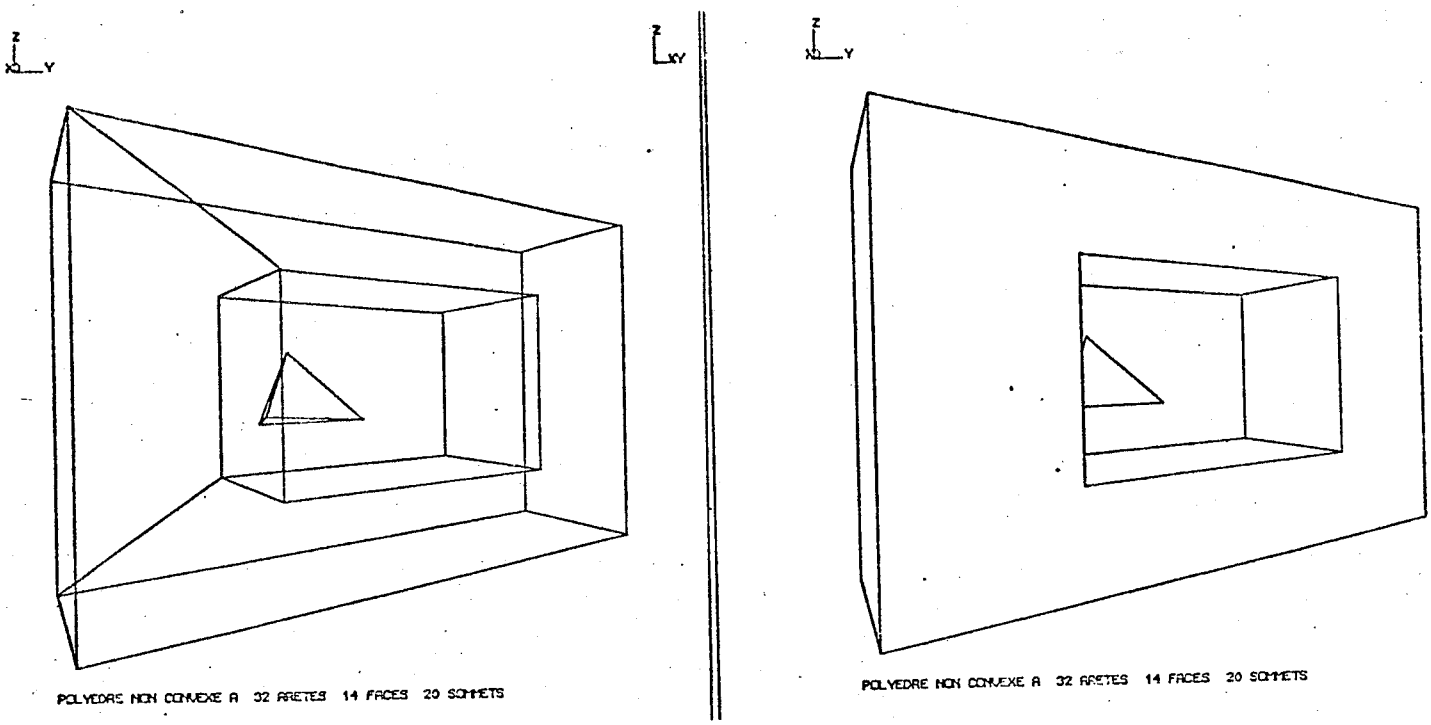


figure 3.14 Présentation d'une scène par la méthode de GALIMBERTI et MONTANARI

3.32 L'algorithme de WARNOCK

3.321 - Principe

L'idée de base de l'algorithme de WARNOCK est de construire le dessin non plus en comparant chaque arête avec l'ensemble des faces, mais en cherchant à dessiner le contour d'une face (ou le contour d'une portion de face). En effet, si l'on peut déterminer que dans une certaine partie de l'espace, une face est seule ou cache tout ce qui est derrière elle, on pourra dessiner directement l'image de cette face sur le plan de projection. On va donc comparer les faces entre elles pour établir quelle face est en avant de l'ensemble étudié. La portion d'espace choisie par WARNOCK est l'intérieur d'un parallèlépipède à section carrée (ou rectangulaire) perpendiculaire au plan de l'écran (cf figure 3.11) :

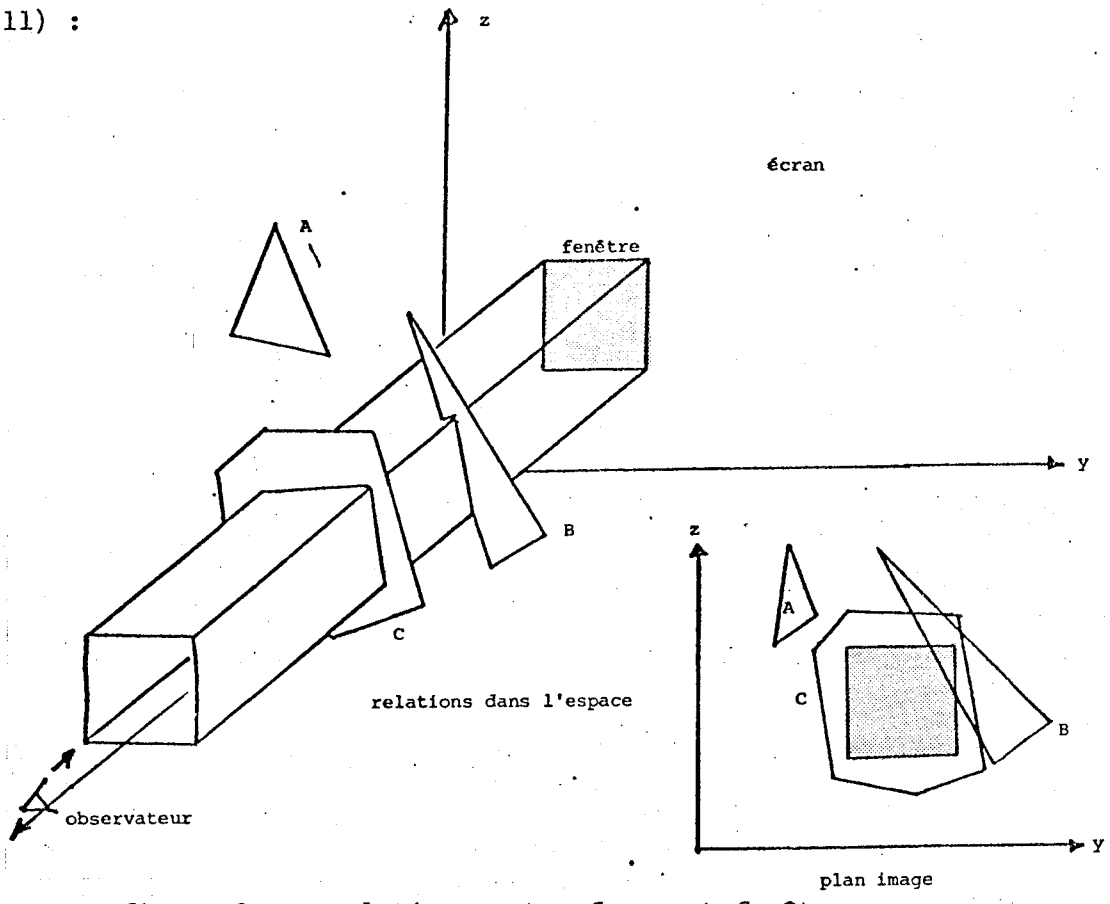


figure 3.11 Relations entre faces et fenêtre

On s'intéresse seulement aux faces susceptibles de modifier le dessin, à savoir les faces qui ne sont pas entièrement disjointes de la fenêtre. On cherche alors à établir les relations spatiales suivantes :

- faces entièrement en dehors du champ d'observation (exemple face A),
- faces entourant le champ d'observation (faces englobantes comme la face B),
- faces en partie intérieures au champ d'observation (faces intersectantes comme la face C).

Ces relations sont calculées en s'intéressant uniquement aux projections des faces sur le plan de l'écran.

Les faces de la première catégorie n'apportant pas d'éléments de dessin ne son pas étudiées. Les faces de la deuxième catégorie sont étudiées en vue de trouver s'il en existe une plus proche de l'observateur que les autres faces. En effet, elle cachera l'ensemble de la scène dans le champ d'observation et il n'y aura donc rien à dessiner. Nous parlerons de *polygone écran*.

Les faces de la troisième catégorie devront être étudiées afin de déterminer le dessin qui doit être fait.

Les diverses versions de l'algorithme de WARNOCK vont différer esser tiellement sur la manière d'étudier la portion de scène contenue dans la fenêt L'idée principale est de se ramener systématiquement à quelques cas de figure relativement simples. L'algorithme le plus simple que l'on puisse imaginer repose sur l'étude des trois situations suivantes :

- il n'y a rien à observer,
- il y a un écran en avant des autres faces, donc rien à dessiner,
- situation trop complexe, on ne sait pas dessiner.

Dans ce dernier cas, on cherche à réduire la surface à étudier. La solution la plus simple consiste à découper la surface observée en quatre parties identiques et réétudier le problème. Le découpage s'arrêtera lorsque la surface à examiner sera de la taille du point observable.

Il sera alors inutile de chercher à étudier la scène en dessous de la précisi atteinte, puisque quelle que soit la complexité de la scène, on pourra tout a plus afficher un point. Le coeur de l'algorithme de WARNOCK est donc la procédure suivante :

```
ETUDE  (entier GAUCHE, BAS, TAILLE) ;
début     {procédé de découpage pour l'algorithme de WARNOCK}
  si  TAILLE = 1
  alors POINT (GAUCHE, BAS)        {limite de résolution atteinte}
  sinon début
          CALCUL_DES_RELATIONS_SPATIALES_FACES_/_FENETRE ;
          ETUDE_DE_LA_CONFIGURATION ;
          si SUCCES
          alors AFFICHAGE_DE_CE_QUI_EST_VU
          sinon début     {découpage en quatre fenêtres filles}
                  ETUDE  (GAUCHE,BAS,TAILLE/2) ;
                  ETUDE   (GAUCHE+TAILLE/2, BAS, TAILLE/2) ;
                  ETUDE  (GAUCHE,BAS+TAILLE/2, TAILLE/2) ;
                  ETUDE  (GAUCHE+TAILLE/2,BAS+TAILLE/2,TAILLE/2)
                fin
        fin
fin ETUDE ;
```

appel : *ETUDE (0,0,1024,0) ;*

3.322 - Réduction de la complexité

L'algorithme précédent montre que les deux points clés sont le calcul des rel tions spatiales et l'étude de la configuration (c'est-à-dire la tentative de classement). En ce qui concerne le calcul des relations spatiales, la réducti de complexité joue surtout sur le nombre de faces à examiner. Deux remarques vont être utilisées :

- lorsque l'on divise la fenêtre en quatre fenêtres filles, certaines proprié calculées précédemment se conservent, à savoir la propriété d'être une face disjointe, ou une face englobante. On tiendra donc à jour un historique des faces, et il est alors clair qu'au fur et à mesure que l'on approchera de l limite de résolution de l'écran, le nombre de faces à examiner en détail diminuera.

- s'il existe une face englobante dans la liste des faces examinées, il est inutile de calculer les relations spatiales des faces situées derrière elle puisqu'elle joue le rôle d'un écran pour celles-ci. Plutôt que de calculer exactement ces positions relatives, un test extrèmement grossier est fait, reposant sur le principe suivant : on calcule la cote minimale en x*(XMINECR* des points appartenant à la face écran et la cote maximale en x*(XMAXFACE)* de chaque face à traiter.

L'observateur se trouvant à l'infini sur l'axe des x positifs toute face tel que *XMINECRAN* $\geqslant$ *XMAXFACE* sera située en arrière de l'écran et ne pourra inter nir dans le dessin (voir figure 3.12).

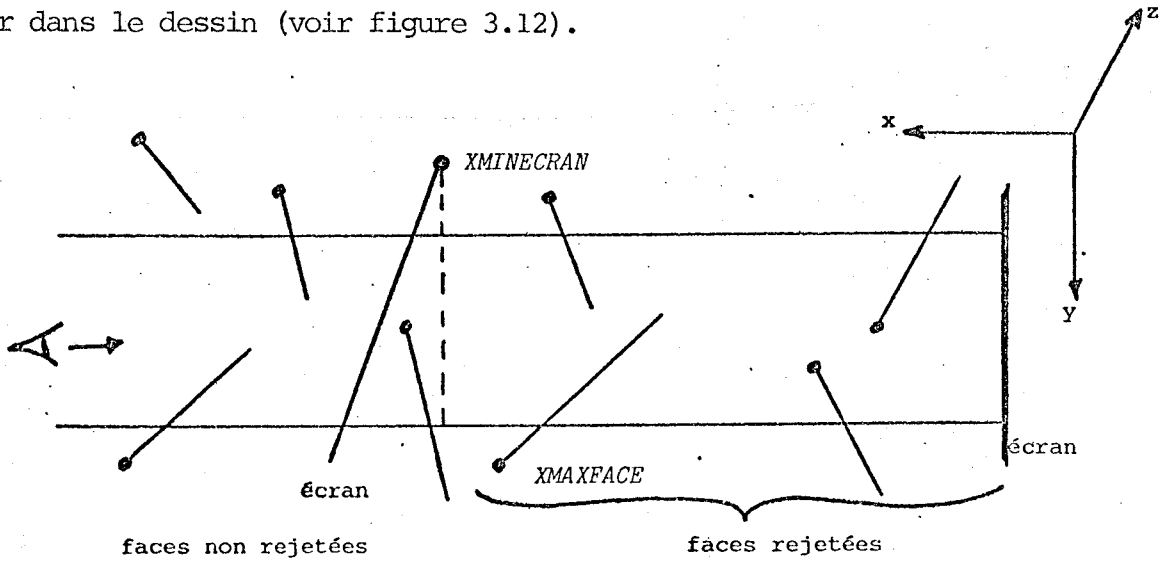


figure 3.12 Test des faces par rapport à un polygone écran

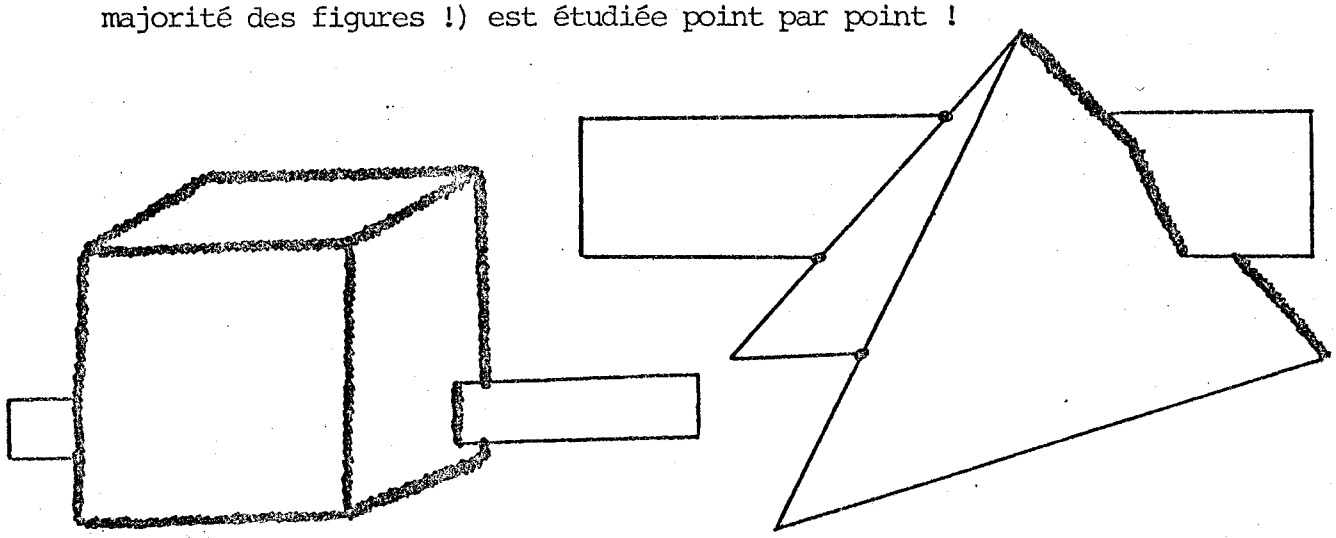
Dans tous les autres cas, on ne conclut pas. On voit que le test utilisé est suffisant, mais pas nécessaire. Seulement, étant rapidement mis en oeuvre, o s'en contente. Pour pouvoir l'utiliser, on effectue initialement un tri des faces sur leur cote la plus proche de l'observateur. Ensuite, lors du calcul relations spatiales, on conserve la cote minimale des polygones écrans : et arrête le calcul au premier élément de la liste des faces situé en arrière d cette cote. Ce test fait également diminuer le nombre de faces à traiter.

En ce qui concerne la recherche d'un polygone écran, une technique analogue est employée, mais on calcule la cote minimale des points d'intersection du polygone écran et des bords du parallèlépipède s'appuyant sur la fenêtre. La procédure de division assure que l'on parviendra à prendr une décision au bout d'un nombre fini d'opérations.

3.322 - Remarques sur la réalisation retenue

Le programme que nous avons écrit était issu, au départ, de celui publié dan [NeS 73]: Il s'est vite avéré que cette version était inexploitable, les tem de calcul étant largement prohibitifs. Une analyse rapide a montré que l'algorithme proposé était de très bas niveau, la figure 3.13 étant un témoi marquant de cette pauvreté.

On peut constater en effet que toute arête appartenant à deux faces (cas de la majorité des figures !) est étudiée point par point !



Les traits renforcés représentent les endroits où le tracé a été fait point par point

figure 3.13 Objet analysé par la méthode de WARNOCK

Nous avons donc apporté un certain nombre de modifications portant essentiellement sur les points suivants :

- organisation des données pour utiliser un tri des faces par segmentation (important, car de l'ordre de 300 ou 400 faces couramment),
- reconnaissance des cas suivants :
  - . rien dans la fenêtre,
  - . présence d'un polygone écran,
  - . présence de deux polygones écrans se coupant (on affiche l'arête d'intersection),
  - . une seule face à dessiner (ou portion de face), soit parce qu'il n'y a qu'elle, soit parce qu'elle est en avant d'un polygone écran,
  - . une seule arête traversant la fenêtre, ou plusieurs arêtes ayant même projection.

Ce cas est extrèmement fréquent, car il concerne aussi bien les arêtes commun à deux (ou plusieurs faces), que les arêtes contenues dans un même plan (ce plan dépendant de la projection choisie : perpendiculaire au plan de projecti ou passant par le point de vue).

Les améliorations résultant de ces modifications ont porté sur les temps de calcul et l'aspect du résultat. En effet, sur le plan esthétique, le nombre d divisions étant moins grand, les arêtes sont dessinées avec des morceaux de segments plus nombreux. La qualité visuelle est donc meilleure.

Une remarque doit être faite à propos de la procédure de découpage en découpant systématiquement en quatre parties égales, le calcul de l'emplacement de chaque fenêtre est très simple. Cependant, les points de div sion sont ainsi choisis sans référence à la scène elle-même, ce qui peut conduire à des découpages stupides (cf figure 3.14).

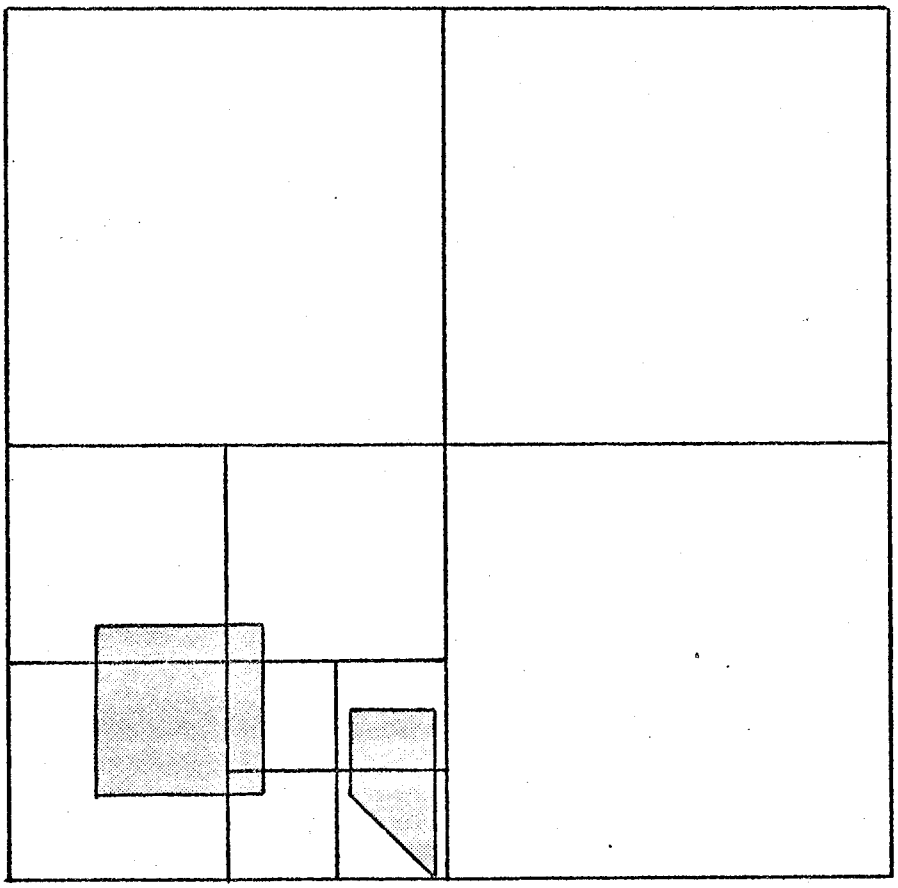


figure 3.14 : exemple de découpage arbitraire

De plus, le dessin obtenu sera différent suivant l'emplacement choisi sur l'écran !

Il est bien évident que l'on a tout intérêt à s'appuyer sur les projections des sommets pour choisir les points de division. En effet, l'aspect esthétiqu sera fortement amélioré et le nombre de divisions sera moins grand, puisqu'on s'appuiera sur des points où un changement d'aspect est probable. On touche ici à un point clé de l'algorithme, puisque toute une stratégie est à établir On peut choisir une solution très simple, consistant à construire un ensemble de fenêtres s'appuyant systématiquement sur tous les sommets. Une autre solution, plus audacieuse, consiste à construire des fenêtres de dimensions plus grandes, tout en cherchant à paver peu à peu l'écran tout entier. On peut par exemple utiliser le sommet le plus proche du point médian de la fenêtre que l cherche à découper. Ces solutions sont plus coûteuses du point de vue de la place mémoire, puisqu'elles exigent (pour ne pas trop alourdir les temps de calcul) que l'on ait une liste des sommets triés en x et une autre en y.

Une dernière remarque concerne l'aspect récursif de l'algorithme. Nous avons réalisé les versions suivantes en utilisant le langage de programmation ALGOLW :

- programmation récursive de l'algorithme,
- gestion explicite d'une pile des paramètres,
- élimination de la pile au profit d'un calcul direct des paramètres en fonct du niveau de découpage.

Dans les trois cas, aucune différence sensible de temps de calcul n'a été enregistrée.

3.324 - Structure de données utilisée

Nous donnons ci-dessous des indications tirées de notre réalisation.

nom du tableau	contenu	nombre d'éléments
* *X, Y, Z*	coordonnées des sommets	*3 * NBSOM*
ED1, ED2	tableaux de description des faces arête par arête	*4 * NBFACES * 2*
POLYLINK	ordre des faces sur Zmin croissant	*NBFACES*
* *POLYZMIN*	Zmin de chacune des faces, ordre croissant	*NBFACES*
POLYEDGE	pointeur sur première arête d'une face	*NBFACES*
NBCOTE	degré de chaque face	*NBFACES*
POLYLIST	listes des polygones écrans ou intersectants	*NBFACES*
HISTORIQUE	historique de chaque face par niveau	*NBFACES * 12*
* *POLYA, POLYB, POLY C, POLYD*	coefficients des équations des plans de chaque face	*NBFACES * 4*
* *COEFA, COEFB*	équation des projections des arêtes	*NBARETES * 2*

soit au total *3*NBSOM + 37 * NBFACES* mots.

3.325 - Critique

L'algorithme de WARNOCK appartient à la famille des algorithmes dits "à rés tion limitée sur une surface". Les principales caractéristiques de ces algo rithmes sont les suivantes :

- étude d'une scène en comparant les faces entre elles,

- calcul dans le plan image, en s'arrêtant à la limite de résolution de l'écr
- utilisation de divers procédés de réduction de la complexité.

Pour donner une idée de la différence fondamentale entre cet algorithme et l'algorithme précédent, on peut imaginer que la scène représente un ensemble d'immeubles en bordure d'une autoroute, ensemble protégé par un écran anti-br On calcule une vue de cet ensemble depuis l'autoroute. Dans le cas de l'algorithme de GALIMBERTI-MONTANARI (et tous ceux de cette famille), l'ensemble de la scène va être calculé dans l'espace utilisateur, puis on pro jettera ce qui est vu, c'est-à-dire ... rien. Dans le cas de l'algorithme de WARNOCK, le premier calcul des relations spatiales montrera qu'il y a un écra et le programme s'arrêtera.

Pour chercher à évaluer les éléments qui influent sur le coût de l'algorithme de WARNOCK, plaçons nous dans le cas d'un écran 1024x1024 points avec la procédure la plus simple possible (voir 3.321). On remarque alors que le temps de calcul est proportionnel à la longueur totale des traits formant le dessin. En effet, l'algorithme de base conduit à descendre systématiqueme au niveau de résolution de l'écran dès que quelque chose est à dessiner. Chaq point dessiné aura donc nécessité l'examen de 12 fenêtres. Si l'on continue d se placer dans le cas le plus simpliste, chaque examen de fenêtre consiste à calculer les relations spatiales des *NBFACES* de la scène, ce qui se fait en p courant une fois l'ensemble des arêtes. Le coût maximum de l'algorithme sera donc de l'ordre de

$$L * 12 * (C * 4 * NBFACES^2)$$

où L représente la longueur du dessin en unités écran. Il est donc clair que temps de calcul est ici proportionnel à la complexité de la scène visible (mesurée en longueur d'arêtes à représenter) et non plus à la complexité de l scène globale.

La formule précédente suggère deux axes d'optimisation :

- diminuer le nombre de faces examinées lors de l'étude d'une fenêtre,
- diminuer le nombre de fenêtres examinées.

En ce qui concerne le premier point nous avons donné en 3.322 quelques indications sur les procédés retenus : tenue à jour d'un historique des faces et élimination des faces situées derrière un polygone écran. Ces techniques introduisent alors les coûts suivants :

- tri du tableau des faces par x_{min} croissants. La réalisation que nous avons faite utilise un tri par segmentation. Les faces n'ayant aucune raison particulière d'être déjà ordonnées lors de la description, le temps de calcul est de l'ordre de

$$C_1 * NBFACES * log \quad NBFACES$$

- pour chaque fenêtre examinée :

  . vérification de l'historique

$$C_2 * NBFACES$$

  . il reste après cette étude *NBEXAM* faces à examiner. Pour chacune de celles-ci, le calcul des relations spatiales se fait en parcourant une fois l'ensemble des arêtes. Le temps de traitement est donc proportionnel à

$$\sim C_3 * 4 * NBEXAM^2$$

Nous ferons deux remarques :

- l'élimination des faces grâce à l'historique dépend des dispositions relatives des faces dans les directions des Y et des Z et donc (grossièrement) de leur surface (en projection) et de leur emplacement.

- l'élimination des faces par le test de disposition par rapport à un écran joue au contraire sur la profondeur en X. Plus la scène sera complexe en X (plus il y aura de faces cachées) et meilleures seront les performances de l'algorithme.

En ce qui concerne la procédure de décision, on voit que son effet es d'éviter des divisions en permettant de dessiner en une seule fois (ou plusieur trait, au lieu de le dessiner point par point. On cherche alors à repérer des situations simples à reconnaître, de manière à dessiner en une seule fois la représentation correspondante (exemple typique : reconnaître les fenêtres traversées par une seule arête ou par plusieurs arêtes ayant même projection). Cependant, il faut noter que le choix de ces situations peut à nouveau pénalise l'algorithme. En effet, les questions posées pour déterminer le cas de figure intéressant sont répétées pour toute fenêtre. Or, l'intérêt d'un test dépend du niveau de découpage auquel on est arrivé. En effet, pour qu'un test T_{C_i} complexe au niveau i soit plus intéressant qu'un test T_{S_i} plus simple conduisar à n divisions il faut que la complexité de T_{C_i} soit inférieure à n fois la complexité de T_{P_i}. Or le nombre de divisions évitées décroît avec la profondeu de récursion, puisque pour une fenêtre de niveau i, le nombre de divisions possibles est de 2^{12-i} $(0 \leqslant i \leqslant 12)$. Par exemple, si la figure à dessiner est la diagonale de la fenêtre, le nombre de division évitées est de $\sqrt{2}.2^{12-i}$. Cette remarque suggère que l'on mette en oeuvre des algorithmes qui testent de situations en fonction du niveau de découpage atteint.

Nous voyons en résumé que l'algorithme de WARNOCK prend en compte d'autres éléments de complexité de la scène que l'algorithme de GALIMBERTI-MONTANARI :

- nombre de faces et nombre d'arêtes,
- longueur des arêtes à dessiner dans l'espace écran (et non plus utilisateur)
- disposition relative des faces dans les trois directions, faisant ainsi inte venir emplacement, surface et profondeur de chaque face.

La figure 3.15 présente une scène traitée par la méthode WARNOCK. C remarquera dans le coin supérieur gauche de la face avant du "tore" la préser d'une arête liant le contour extérieur au coutour intérieur. Cette arête perm de décrire une face "trouée". Si l'on ajoutait la notion de transparence (ou d'opacité) d'une arête, elle pourrait être traitée correctement et ne pas être affichée.

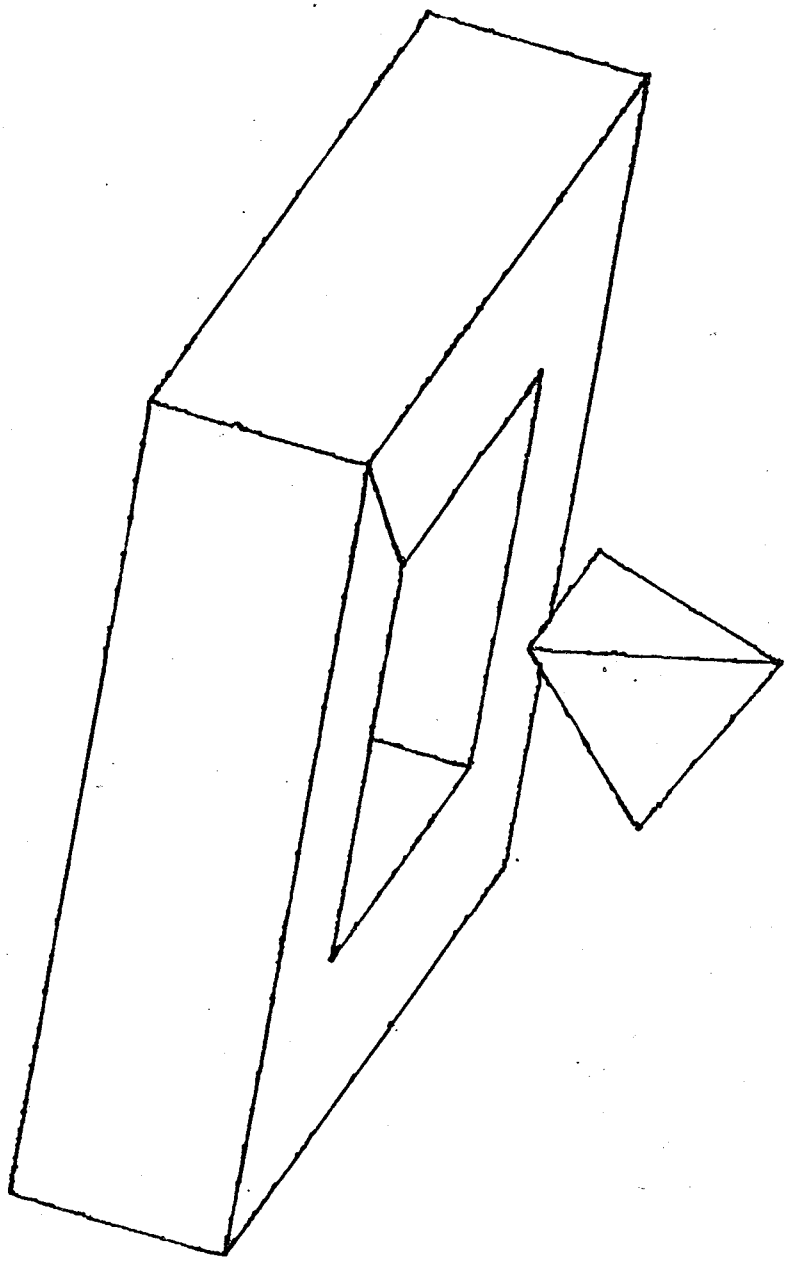


figure 3.15 Présentation d'une scène par la méthode de WARNOCK

3.4 ELIMINATION DE PARTIES CACHEES : COMPOSITION D'IMAGES

Nous nous intéressons dans ce paragraphe aux algorithmes qui présentent non p les arêtes visibles, mais les faces visibles (ou les portions de faces).

Nous étudierons deux algorithmes :

- l'algorithme de WATKINS, qui construit l'image ligne par ligne,
- l'algorithme de NEWELL, NEWELL et SANCHA, qui construit l'image point par p

3.41 L'algorithme de WATKINS

3.411 - Principe

L'idée première de l'algorithme de WATKINS est de passer du problème à trois dimensions à un problème à deux dimensions. La scène est découpée par une sér de plans parallèles au plan xOy, et on étudie les sections ainsi obtenues (vo figure 3.16).

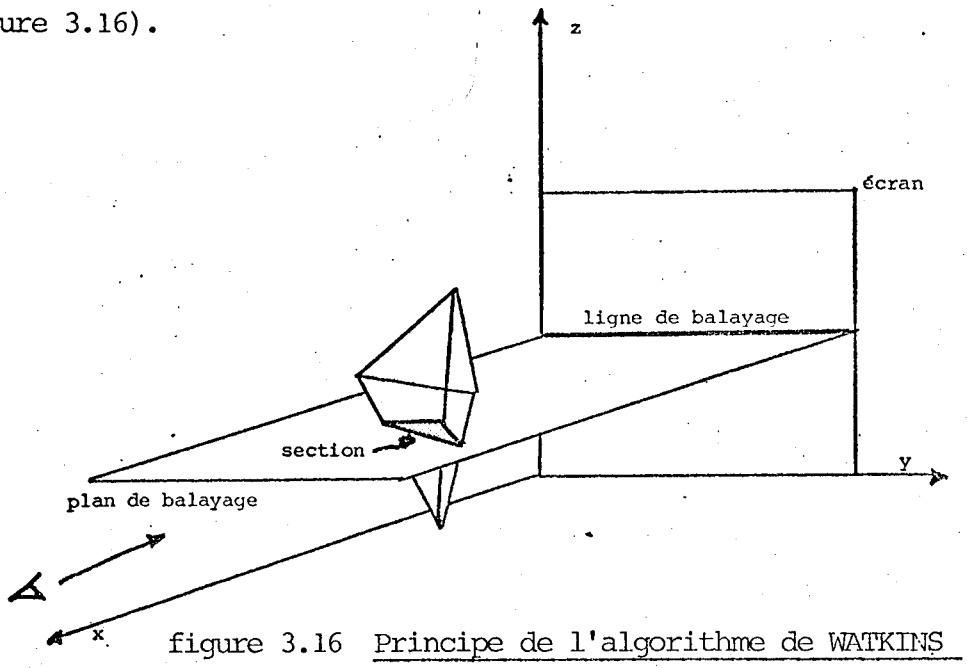


figure 3.16 Principe de l'algorithme de WATKINS

Pour chacun des plans de balayage, on cherche à déterminer quels sont les segments (ou les portions de segments) qui sont visibles. Ces segments sont projetés sur la ligne de balayage, et l'on obtient ainsi l'image sur une ligr

L'image est terminée lorsqu'on a construit toutes les lignes.

Le schéma de l'algorithme de WATKINS est très semblable à celui de WARNOCK. En effet, pour un plan de balayage donné, on cherche à comprendre la situati à partir d'un certain nombre de règles simples de dessin. Si la portion de p étudiée contient une situation trop complexe, on la partage en deux demi-pla que l'on va étudier séparément. Le schéma de l'algorithme de WATKINS est le suivant :

```
WATKINS ;
début    présentation d'une scène par découpage par des plans parallèles à x
| pour I:=1 pas 1 jusqua ZRESOLUTION faire
| début
| | CALCUL_DE_LA_SECTION_AU_PLAN_I ;
| | ETUDE(0, YRESOLUTION)
| fin
fin WATKINS ;

ETUDE  (entier GAUCHE,TAILLE) ;
début   étude d'un ensemble de segments contenu dans une bande
| si  TAILLE > 1
| alors début
|       | CALCUL_DES_RELATIONS_SPATIALES_ENTRE_SEGMENTS ;
|       | ETUDE_DE_LA_CONFIGURATION ;
|       | si SUCCES
|       | alors AFFICHAGE_DE_CE_QUI_EST_VU
|       | sinon début
|       |       | ETUDE  (GAUCHE, TAILLE/2) ;
|       |       | ETUDE  (GAUCHE+TAILLE/2,TAILLE/2)
|       |       fin
|       fin
fin ETUDE ;
```

La clé de l'algorithme se trouve dans la partie qui calcule les dispositions relatives des segments. Nous allons étudier quelques solutions à ce problème

3.412 - Algorithmes d'étude d'une suite de segments

Nous commencerons par étudier les techniques employées dans la version propo dans [NeS 73]. Nous avons écrit un programme directement inspiré par cette version, qui s'est avérée être d'excellente qualité, ce qui explique que l'é proposée ici s'appuiera essentiellement sur ce programme.

NEWMANN et SPROULL proposent un programme dont les éléments de base reposent sur la connaissance de quelques cas de figure simples à dessiner, qui sont au nombre de quatre (voir figure 3.17) :

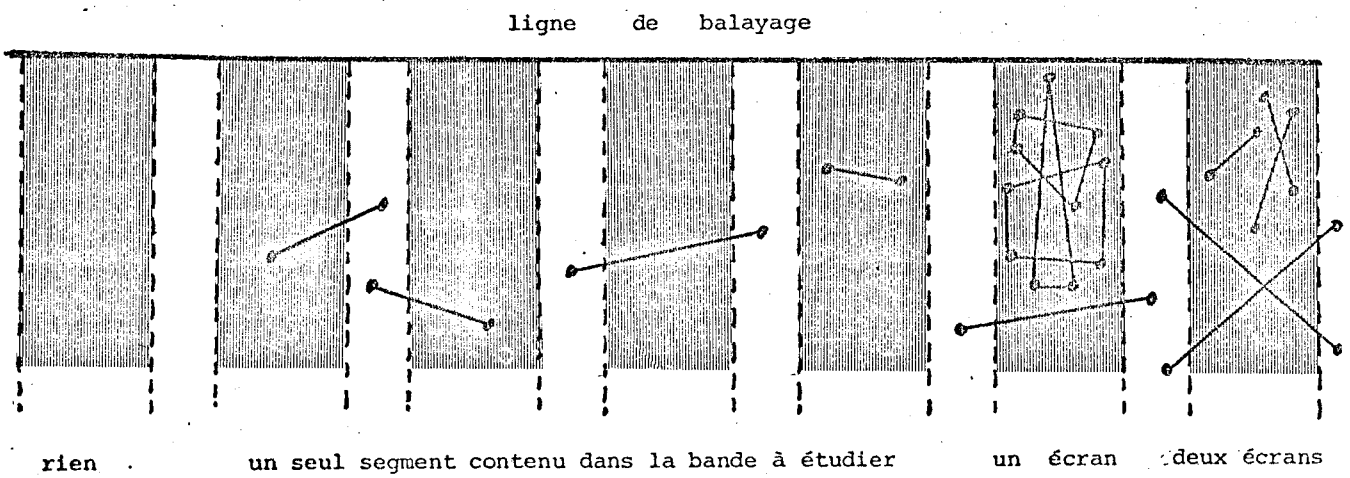


figure 3.17 Cas de figure simples à dessiner (WATKINS)

Tout le problème va donc consister :

- à découvrir si sur un domaine donné il existe une relation aussi simple,
- dans le cas contraire, à découper le domaine en question.

Le calcul des relations spatiales se fait à l'aide de tests relativement simp qui permettent de comparer le segment à classer par rapport à un ou plusieurs segments significatifs retenus. Les tests se font par rapport à une "boîte", qui est le rectangle à bords parallèles aux axes formé à partir des coordonné maximales et minimales des segments significatifs.

Les paramètres qui sont constamment tenus à jour sont les suivants :

- nombre de segments contenus dans la boîte *(NBSEG)*,
- coordonnées des sommets de la boîte,
- présence éventuelle de deux segments écrans et, dans ce cas, coordonnées du point d'intersection.

Au départ, *NBSEG* est égal à 0.

La suite des opérations se déroule de la façon suivante :

- *NBSEG=0* Les coordonnées des sommets de la boîte sont mises à jour en utilisant les coordonnées du segment que l'on vient d'étudier. *NBSEG = NBSEG+1*

- *NBSEG=1* Quatre cas de figure se présentent

  . Le nouveau segment cache entièrement la boîte. Celle-ci est ajustée su le nouveau segment, l'ancien étant oublié.

  . le nouveau segment est caché par la boîte. On l'oublie.

  . le nouveau et l'ancien segment traversent tous les deux l'empan considéré : on est en présence de deux segments écrans. On note l'intersect éventuelle, les sommets de la boîte sont ajustés sur les coordonnées minimales et maximales des deux segments *NBSEG = NBSEG+1*

  . dans tous les autres cas, on ajoute la boite. *NBSEG = NBSEG+1*

- *NBSEG > 1* Trois cas de figure sont pris en compte :

  . le nouveau segment cache la boîte. Le contenu de celle-ci est oublié et on ajuste une nouvelle boîte sur le nouveau segment. *NBSEG=1*

  . le nouveau segment est caché par la boîte : on l'oublie.

  . dans tous les autres cas, on ajuste la boîte de manière à englober le nouveau segment. *NBSEG = NBSEG + 1.*

Lorsque tous les segments inclus dans la bande ont été étudiés, les valeurs finales des paramètres permettent de déduire si l'on se trouve dans l'un des quatre cas de compréhension de la scène donnés plus haut.

La manière de calculer les positions relatives des différents segm ressemble aux tests de position de faces entre elles utilisée dans l'algorit de WARNOCK. En effet, le test est fait par rapport au rectangle englobant l'ensemble des segments significatifs. Si la condition testée est suffisante pour conclure au fait qu'un segment est vu ou caché, elle n'est cependant pa nécessaire, et dans certains cas des segments jouant le rôle d'écran ne sero pas détectés (voir figure 3.18).

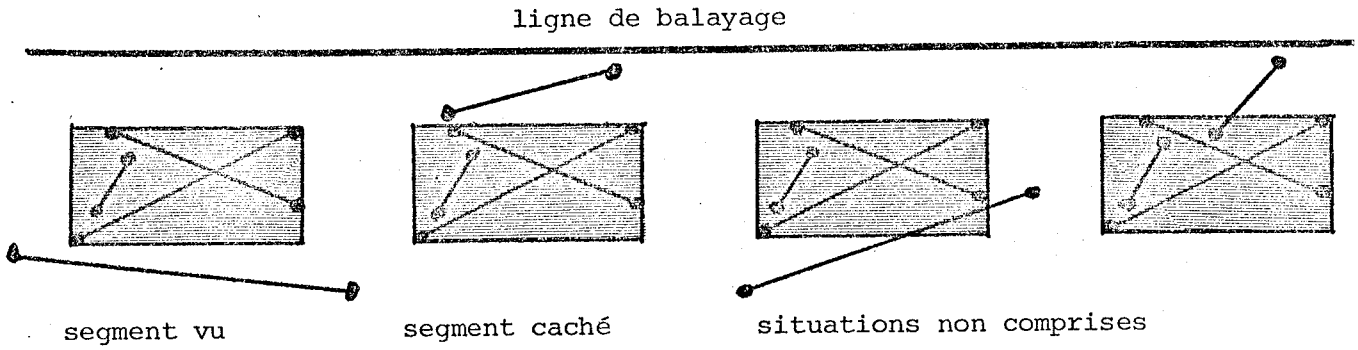


figure 3.18 Position d'un segment par rapport à un ensemble

Il y a de nouveau compromis entre la difficulté du test et le nombre de divisions à faire.

On voit donc que l'idée de base consiste à étudier ce qui se passe sur un esp plan donné. On peut partir d'une autre idée, consistant à construire la liste des segments visibles à un moment donné, et à comparer un segment à classer à cette liste. On se ramène à un problème de fusion de deux listes définies de la manière suivante :

- la première liste contient l'ensemble des segments formant la section à étudier. On supposera que cette liste est ordonnée de telle sorte que si $Si < S_j$ alors $y^1_i \leq y^1_j$ (liste ordonnée sur l'extrèmité gauche)

- la deuxième liste, appelée ligne de crêtes, sera constituée de segments ordonnés en ordre croissant sur l'extrémité gauche. Les projections de ces segments seront jointives. La propriété de cette liste est la suivante : si l'on mêne une droite x=cte, elle perce la ligne de crête soit en un seul pc (maximum des ordonnées des points d'intersection de la droite avec les segments de la première liste) ou en deux points en cas de discontinuité (maximum et second maximum des ordonnées).

L'opération de base consiste donc à comparer deux segments de droit en vue de les classer l'un par rapport à l'autre.

On étudie S_1 (x_1,y_1) (x'_1,y'_1) segment à classer $\in$ L1

S_2 (x_2,y_2) (x'_2,y'_2) segment de référence $\in$ L2

Le cas d'étude le plus simple est celui où

$$\begin{cases} y_1 = y_2 \\ y'_1 = y'_2 \end{cases}$$

Les cas de figure possibles sont alors simples (voir figure 3.19) :

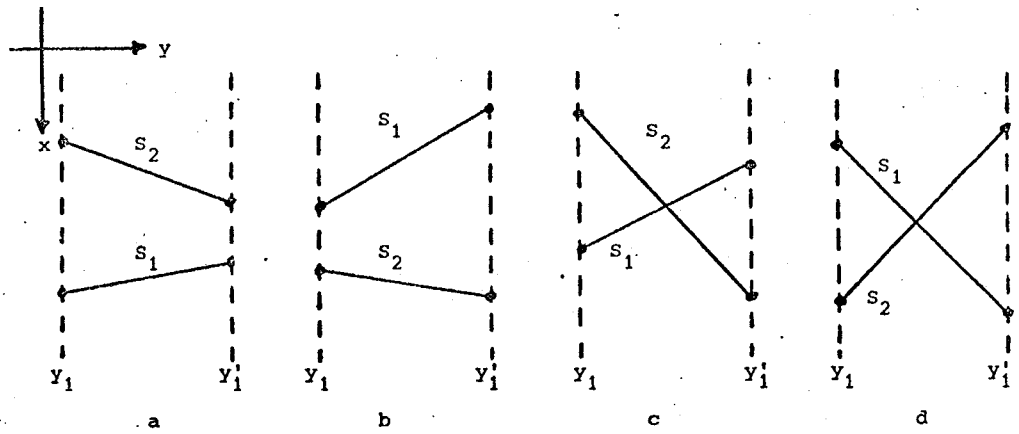


figure 3.19 Positions relatives de deux segments

et correspondent aux relations suivantes :

a) - $(x_2 > x_1)$ et $(x'_2 \geqslant x'_1)$ segment S_2 vu

b) - $(x_2 \leqslant x_1)$ et $(x'_2 \leqslant x'_1)$ segment S_2 caché

c) - $(x_2 > x_1)$ et $(x'_2 < x'_1)$

d) - $(x_2 < x_1)$ et $(x'_2 > x'_1)$ intersection des deux segments

Dans tous ces cas, on peut déterminer exactement ce qui est vu, et donc ce q est à dessiner

Le premier problème consiste à traiter correctement l'ensemble des segments éventuellement concernés par le segment à étudier. La situation se présente suivant le schéma de la figure 3.20 où n'est représenté que le domaine d'influence de chaque segment (amplitude en y).

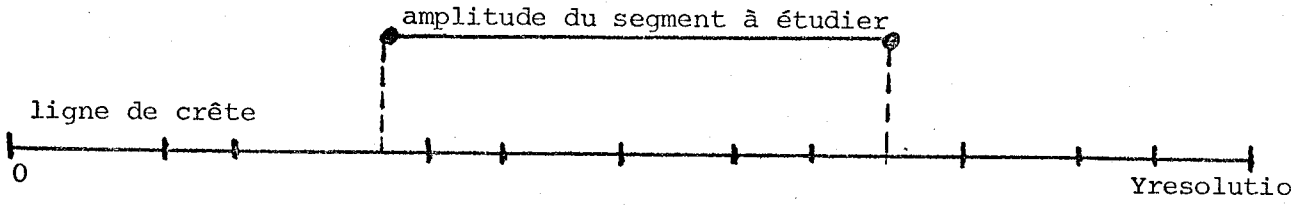
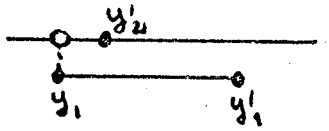


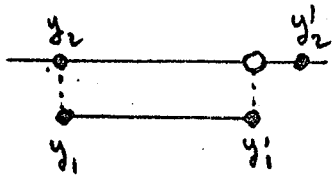
figure 3.20 Recouvrement d'un segment avec la ligne de crêtes

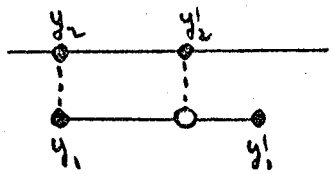
L'algorithme que nous présentons ci-dessous permet de comparer un segment (x_1, y_1, x'_1, y'_1) à la ligne de crête. Les difficultés à traîter concernent le découpage du segment par les segments de la ligne de crête avec lesquels il vient en concurrence et la création de cette ligne.

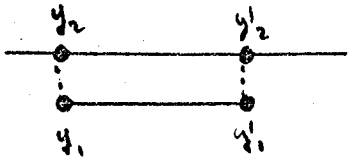
```
FUSION (entier X1,Y1,X'1,Y'1) ;
début  {fusion de la ligne de crête et d'un segment}
  PREMIER_SEGMENT ; {premier segment de la ligne de crête}
  tantque Y'2 < Y1 faire  SEGMENT_SUIVANT ; {recherche du début de zone concern
  si Y'2 > Y'1
  alors METTRE_A_JOUR (1) ;

  {dans tous les cas Y1 = Y2}
  ILYASEG := vrai ;
  tantque  ILYASEG  faire
  début
    si Y'2 > Y'1
    alors début
            METTRE_A_JOUR (2) ;
            CLASSER (X1,Y1,X'1,Y'1,X2,X'2,2) ;
            ILYASEG := faux
          fin ;
    si Y'2 < Y'1
    alors début
            METTRE_A_JOUR (3)
            CLASSER (X1,Y1,X'1,Y'1,X2,X'2,3) ;
            METTRE_A_JOUR (4) ;
            SEGMENT_SUIVANT
          fin ;
    si Y'2 = Y'1
    alors début
            CLASSER (X1,Y1,X'1,Y'1,X2,X'2,4) ;
            ILYASEG := faux
          fin
  fin
fin FUSION ;
```









Nous appellerons (α_1, β_1) les coefficients de l'équation de la droite portant l segment en cours d'étude et (α_2, β_2) les coefficients de l'équation de la droit portant le segment de la ligne de crête. La procédure *METTRE_A_JOUR* peut s'écrire comme ci-dessous :

```
METTRE_A_JOUR (entier TYPE) ;
début
  suivant TYPE faire
  début
    début {TYPE = 1}
      X2 := α2 * Y1 + β2 ;
      Y2 := Y1
    fin ;
    début {TYPE = 2}
      X'2 := α2 * Y'1 + β2 ;
      Y'2 := Y'1
    fin ;
    début {TYPE = 3}
      GARX := X'1 ;  {sauvegarde fin du segment}
      GARY := Y'1 ;
      X'1 := α1 * Y'2 + β1 ;
      Y'1 := Y'2
    fin ;
    début  {TYPE = 4}
      X1 := X'1 ;
      Y1 := Y'1 ;
      X'1 := GARX ;
      Y'1 := GARY
    fin
  fin
fin METTRE_A_JOUR ;
```

Les différents segments seront donc conservés sous la forme des coordonnées de leurs extrémités et des coefficients de l'équation de la droite les portant. C'est au cours du classement que ces opérations seront faites.

```
CLASSER (entier X1,Y1,X'1,Y'1,X2,X'2, TYPE) ;
début   {comparaison et traitement de deux segments définis sur un même interv
         le ; le rangement dépend du type, défini lors de l'opération de déco
         page}
  entier DELTA, XINT, YINT ;
  DELTA := X2 - X1 ;
  si DELTA × (X'2 - X'1) ⩾ 0
  alors début
          | si (DELTA < 0) ou (X'2 - X'1) < 0
          | alors {segment caché : rien à faire}
          | sinon RANGER (X2, Y2, X'2, Y'2, TYPE)
        fin
  sinon début
          | {calcul intersection}
          | XINT := X1 + (X'1 - X1) × (Y2 - Y1)/(Y2 - Y1 - Y'2 + Y'1) ;
          | YINT := Y1 + (Y'1 - Y1) × (XINT - X1) / (X'1 - X1) ;
          | si DELTA >0
          | alors début
          |        | RANGER (X2,Y1, XINT, YINT, TYPE) ;
          |        | RANGER (XINT, YINT, X'1, Y'1, TYPE)
          |       fin
          | sinon début
          |        | RANGER (X2, Y1, XINT, YINT, TYPE) ;
          |        | RANGER (XINT, YINT, X'1, Y'1, TYPE)
          |       fin
          |
        fin
fin CLASSER ;
```

remarques :

- le *TYPE* sert à indiquer le nombre de segments qui sont à ranger.
- on suppose que les segments y=constante ne sont pas conservés, car ils n'apportent rien au dessin.

L'étude complète de la section se fera en traitant l'ensemble de segments qui la forment par l'algorithme *FUSION*. Au départ, la ligne de crête sera constit par le segment *(0,0) - (0,YRESOLUTION)*.

une autre solution peut être proposée, reposant sur le principe suivant : on maintient à jour la ligne de crête point par point, en conservar pour chaque point de la ligne la hauteur trouvée et la couleur associée au se ment qui était visible en ce point. Pour chaque segment, on calcule point par point et on compare directement à la hauteur déjà présente. La mise à jour es évidente.

Au départ, pour chaque ligne de balayage, on donne la valeur 0 à chaque éléme des tableaux *COULEUR* qui contiendra l'indication de couleur en chaque point
CRETE qui contiendra l'indication d'altitude maximale atteinte.

On peut utiliser l'algorithme de dessin d'un segment point par point que nous avons proposé en 1.213. La procédure *AFFICHER* sera alors remplacée par la pro cédure *RANGER* que nous définissons ci-dessous.

```
RANGER (entier X,Y,COLORIS) ;
début
| si X > CRETE(Y)
| alors début
|       | CRETE(Y) := X ;
|       | COULEUR(Y) := COLORIS {couleur associée au segment}
|       fin
fin RANGER ;
```

Si l'on cherche à choisir entre ces trois algorithmes, plusieurs critères peuvent être retenus :

- simplicité de la programmation. Il est alors clair que la troisième soluti est de loin la plus simple à mettre en oeuvre, les deux autres impliquant gestion relativement complexe de listes de segments.

- efficacité ; c'est-à-dire rapidité de traitement. Si nous étudions la troisième solution proposée, il est évident que le facteur déterminant est la longueur totale des segments à étudier. Les deux autres solutions sont fond sur un essai de réduction du nombre de points à comparer, en traitant des segments de droite (ou des portions de segments de droite) en une seule fo et non point par point.

Il faut donc pouvoir démontrer que les dispositions relatives des différents segments sont telles que les opérations de découpage, classement et gestion de différentes listes sont en général (sinon toujours) moins coûteuses que les opérations de comparaison point par point. En fait, la complexité de la sectic à étudier dépend directement de la complexité de la scène initiale, les éléments déterminants étant le nombre de faces et leurs dispositions relatives dans l'espace. Il n'existe pas à l'heure actuelle d'étude théorique sur ce sujet, ce qui rend tout essai de comparaison relativement vain.

3.413 - Réduction de la complexité

Les points clés de l'algorithme de WATKINS sont le calcul des sections pour chaque plan de balayage et le calcul des relations spatiales. La procédure de décision est elle relativement simple et n'apporte pas beaucoup au temps de calcul final. En ce qui concerne le calcul des sections, la remarque fondamentale qui a été faite est que l'image à traiter est en fait constituée de larg zones cohérentes et continues (c'est-à-dire où la situation d'une ligne à l'autre change peu) et de quelques zones de discontinuité (lors de l'appariti d'un sommet ou lors de sa disparition) (cf figure 3.21).

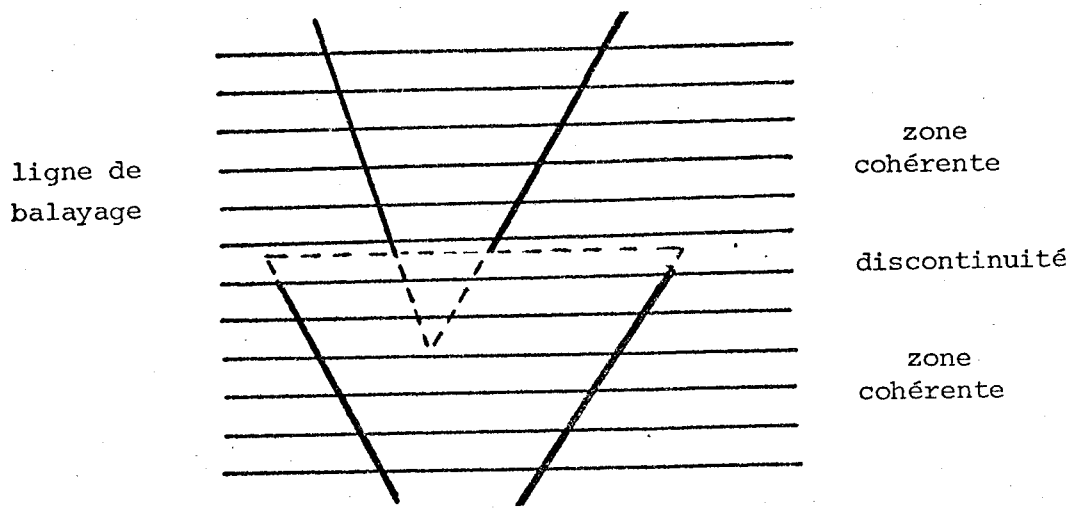


figure 3.21 Continuité de l'image

A partir de cette remarque, la stratégie consiste à préparer le calcul des sections de la manière suivante :

- faire un tri des faces par ordre d'entrée en scène, c'est-à-dire sur leur côte en z, de manière à ne s'intéresser à une face que lorsqu'elle commenc à apparaître dans le plan de balayage,

- tenir à jour une liste des segments actifs, c'est-à-dire constituant une s tion.

Cette liste de segments comporte entre autres renseignements les pas en x,y permettant de calculer par simple addition les coordonnées du point d'intersection d'une arête avec le plan i+1, connaissant les coordonnées au i.

Le calcul d'une section revient alors à effectuer les opérations suivantes :

- parcourir la liste des segments actifs du plan de balayage précédent et me tre à jour les coordonnées des extrémités des segments. Eliminer tous les segments liés à un (plusieurs) sommet disparaissant de la zône étudiée.

- ajouter à la liste de segments actifs les segments crées par apparition d' sommet à ce niveau du balayage.

Le calcul de 512 sections, qui serait prohibitif, est ainsi remplacé par des opérations de création et mise à jour de listes nettement moins coûteuses.

En ce qui concerne le calcul des relations spatiales, le choix du mode de dé page est fondamental. En effet, l'algorithme que nous avons décrit en 3.411 possède les mêmes défauts que celui de WARNOCK : le découpage est fait arbit rement, sans tenir compte de la scène. On peut alors, pour pallier cet incon nient, utiliser une stratégie consistant à appuyer le découpage sur les somm des segments à examiner, le balayage du plan se faisant de gauche à droite (cf figure 3.22).

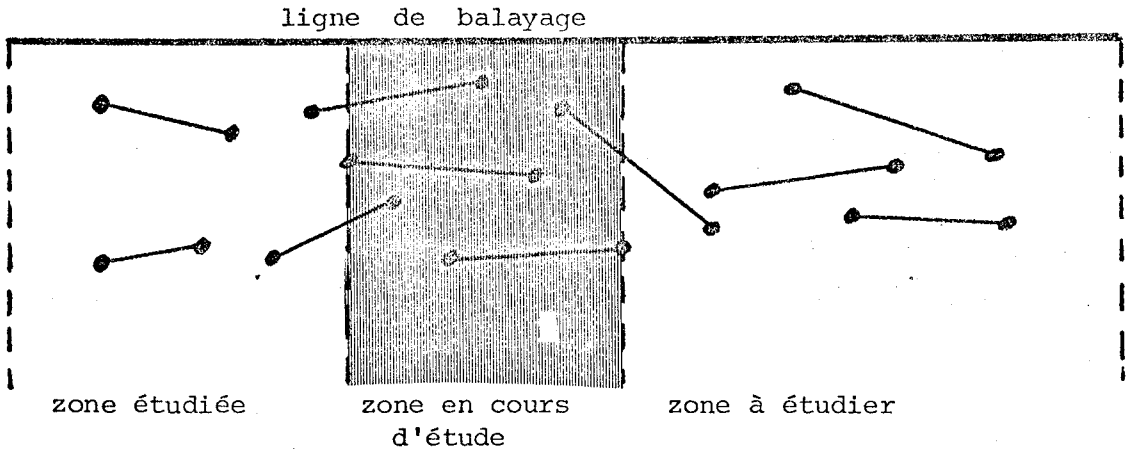


figure 3.22 stratégie d'étude d'un plan de balayage

Les segments formant la section se répartissent en trois catégories, en fonction de la zone de traitement considérée :

- un ensemble de segments situés entièrement à gauche de cette zone. Ils ne peuvent plus jouer de rôle pour la suite de l'étude, et disparaissent de la liste des segments à étudier.

- un ensemble de segments situés entièrement à droite de la zone étudiée. Ils ne jouent aucun rôle sur celle-ci mais seront étudiés par la suite.

- un ensemble de segments entièrement contenus dans la zone à traiter, ou à cheval sur celle-ci. Pour ces derniers, seuls ceux qui ont une extrémité située à droite de la zone seront conservés pour l'étude ultérieure. Tous les autres seront éliminés.

Au départ, on étudie l'ensemble des segments actifs. Il est clair qu'au fur et à mesure que le balayage progresse vers la droite le nombre de segments examinés diminue.

De plus, une liste est tenue des points de découpage ayant conduit à un succès. Cette liste est mise à jour par simples additions aux coordonnées des points retenus (tant que la zone est cohérente). La recherche de points de division est ainsi accélérée.

3.414 - Structure de données utilisée

La structure décrite ici est celle du programme de NEWMAN et SPROUL

nom des tableaux	contenu	nombre d'éléments
* *X, Y, Z*	coordonnées des sommets	*3 * NBSOM*
V1,V2	arêtes réellement utilisées	*2 * NBARUTIL*
P1,P2	faces associées aux arêtes	*2 * NBARUTIL*
COULEUR	couleur de chaque face	*NBFACES*
YENTER	départ des listes de sommets pour une ligne	*512*
ENTERLIST	liste des arêtes par ligne de balayage	*NBARETES*
CHANGING	liste des faces actives	*NBFACES*
SEGMENTLIST	têtes des listes de segments par face	*NBFACES*
POLYSEGMENT	listes des segments attachés à une face	*MAXSEG*
ACTIVE	liste des segments actifs	*MAXSEG*
XSORTRIGHT,XSORTLEFT	pointeurs de la liste de segments triée sur abscisse de gauche (liste double)	*2 * MAXSEG*
par segment 11 mots (6 coordonnées des extrémités + pas en x et en y (4 mots) + face de rattachement) dont 4 réels		*11 * MAXSEG*
SAMLINK	pointeurs de la liste des points d'empan	*MAXSEG*
* *SAMX*	liste des points d'échantillonnage	*MAXSEG*
VISSEG	couleur associée au segment à afficher	*256*
VISPOS	description de la ligne de balayage	*256*

Nous prenons pour *MAXSEG* l'hypothèse que toutes les faces sont coupées et convexes. La place demandée est alors de l'ordre de

*3 * NBSOM = 16 * NBFACES + 17 * MAXSEG + 1024 = 3 * NBSOM + 33 * NBFACES + 10*

mots.

3.415 - Critique

L'algorithme de WATKINS appartient à la catégorie des algorithmes à résolutio limitée travaillant par balayage. On trouve dans cette famille les algorithme de ROMNEY, BOUKGNIHT. De très nombreuses versions et variantes ont été réalisées. En particulier, l'université de Utah a réalisé une version cablée de c algorithme, autorisant la présentation d'images complexes en temps réel.

Cherchons à identifier les facteurs déterminants du coût de cet algorithme.

- première étape : tri des arêtes sur leur cote en Z

$$C_1 * NBARETES \cap\cup C_1 * 4 * NBFACES$$

- pour chaque ligne de balayage :

  . calcul de la section (mise à jour de la liste des segments courants, i sertion de nouveaux segments, tri de la liste). Comme il y a peu de ch gements en général d'une ligne à l'autre, le traitement peut globaleme être estimé de l'ordre de

$$C_2 * NBSEG$$

  où NBSEG représente le nombre de segments déterminé par la section de la scène,

  . construction de la ligne de balayage. Un majorant consiste à identifie celle-ci à un tri, ce qui donne un temps proportionnel à

$$C_3 * NBSEG^2$$

Le coût global de traitement d'une scène affichée sur un écran 512×512 sera donc de l'ordre de

$$C_1 * 4 * NBFACES + 512 * (C_2 * NBSEG + C_3 * NBSEG^2)$$

On voit ici que l'élément crucial est le nombre de segments à traiter pour chaque section. Ce nombre de segments dépend :

- du nombre de faces présentes au niveau de la coupe. Cet élément de complexi fait intervenir l'amplitude en hauteur des faces de la scène ainsi que leur disposition en Z,

- de la forme des faces. Par exemple, si l'on traite des faces convexes, il n aura qu'un seul segment déterminé par l'intersection d'une telle face avec plan de balayage. Une face non convexe pourra déterminer plusieurs segments à étudier.

Nous remarquerons d'autre part que les algorithmes que nous avons exposés en 3.412 visent à donner des temps de traitement qui ne soient plus de l'ordre d carré du nombre de segments à traiter, mais d'un ordre inférieur. Une étude f de ces algorithmes serait tout à fait nécessaire pour permettre de mieux comprendre le comportement de l'algorithme de WATKINS.

Le coût de cet algorithme est clairement fonction du nombre d'éléments de la scène à représenter :

- nombre de faces et nombre d'arêtes,
- hauteur des faces et disposition relative de celles-ci en Z,
- forme des faces.

Cet algorithme se rapproche donc de celui de GALIMBERTI et MONTANAI puisqu'il ne tient pas compte des particularités de la scène comme dans le ca de WARNOCK. Cependant, une différence essentielle apparaît, qui vient du fait que l'étude point par point des arêtes (section par section) se fait avec une résolution limitée dépendant de l'écran, et indépendante du système de coordo nées de l'utilisateur. La solution obtenue est donc moins précise que celle-calculée par la méthode de GALIMBERTI et MONTANARI.

La figure 3.23 présente une scène traitée par la méthode de WATKINS. L'image a été obtenue sur une surface à dessin au trait, en utilisant quatre "couleurs" différentes.

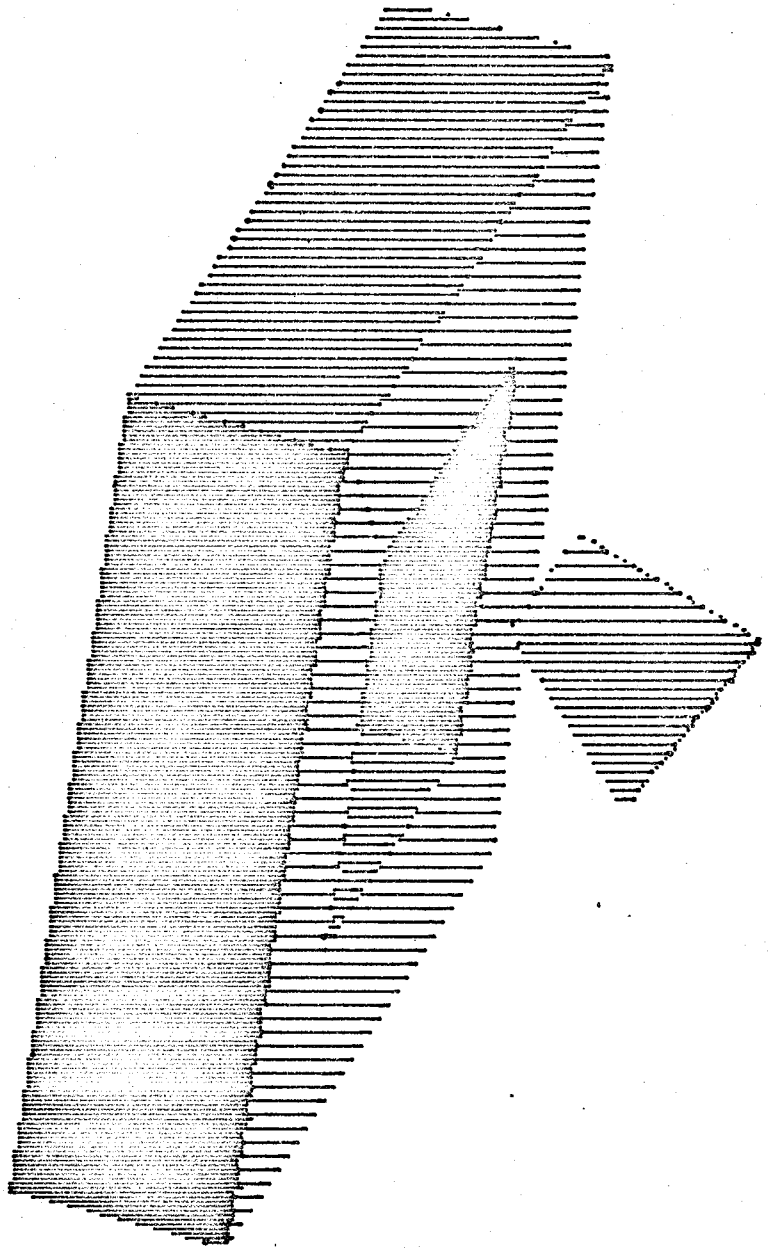


figure 3.23 Présentation d'une scène par la méthode de WATKINS

3.42 L'algorithme de NEWELL, NEWELL et SANCHA

3.421 - Principe

L'algorithme de NEWELL, NEWELL et SANCHA permet d'obtenir une représentation sous forme d'image à partir du principe suivant:

- dans un premier temps, on classe les faces en fonction de leur distance à l'observateur, de manière à pouvoir déterminer un ordre permettant de passe progressivement de la plus lointaine à la plus proche.

- on projette ensuite ces faces sur le plan de l'écran, en commerçant par la plus proche de l'écran (la plus lointaine pour l'observateur), et en se rap prochant progressivement de l'observateur. Chaque fois qu'une face va en ca cher une autre, elle sera forcément traitée après la face cachée, et sa pro jection viendra recouvrir la projection précédente. Lorsque la dernière fac a été projetée, l'image définitive est obtenue.

La clé de l'algorithme réside bien entendu dans la procédure de cla sement. Un premier critère de classement utilise le point de chaque face le plus proche de l'observateur. Cependant, un tri sur ces coordonnées ne permet d'obtenir qu'un ordre partiel, des difficultés subsistant lorsque des faces s coupent ou se chevauchent (voir figure 3.24)

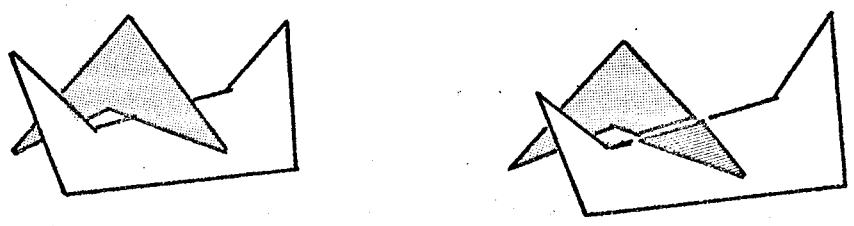


figure 3.24 Découpage de faces se chevauchant

Dans ces cas là, on découpe une face par le plan contenant l'autre face, ce qui permet alors d'obtenir un ordre total. Le découpage peut être réalisé par l'algorithme décrit en 3.22.

Une fois le classement établi, la projection peut se réaliser en utilisant la procédure décrite en 1.213.

3.422 - Structure de données utilisée

Nous prenons ici comme hypothèse que l'on réserve pour les sommets et faces crées par le découpage une place égale à celle utilisée dans les données initiales. Nous nommons *NBSOMFUT* le nombre de nouveaux sommets, et *NBFACFUT* le nombre de nouvelles faces.

nom des tableaux	contenu	nombre d'éléments
* *X, Y, Z*	coordonnées des sommets	*3 * (NBSOM + NBSOMFUT)*
FACES	liste des faces	*6 * NBFACES + 6 * NBFACFUT*
POINTFACES	pointeurs vers les équations des plans et la liste des sommets	*2 * (NBFACES + NBFACFUT)*
MARQUE	auxiliaire pour le classement	*NBFACES + NBFACFUT*
* *ZMIN, ZMAX*	coordonnées Zmin et Zmax des faces	*2 * (NBFACES + NBFACFUT)*
PT	liste des faces par ordre de Zmin	*NBFACES + NBFACFUT*
COULEUR	tableau des couleurs par faces	*NBFACES + NBFACFUT*
* *COEFA, COEFB, COEFC, COEFD*	coefficients des équations des plans des faces	*4 * (NBFACES + NBFACFUT)*
IMAGE	tableau représentant l'image	*512 × 512 octets*

Le coût de cette structure est donc donné par

*3 * NBSOM + 17 * NBFACES + 3 * NBSOMFUT + 17 * NBFACFUT + 512 × 512 mots*

Nous ferons deux remarques :

- nous ne disposons pas de statistiques sur le nombre de découpages de chaq face en fonction de la complexité de la scène. Il faut noter que du poin de vue de la place mémoire, le découpage est très coûteux. Si on admet pa exemple que chaque nouvelle face conduit à créer deux nouveaux sommets, a *NBSOMFUT* ∩ *2* ⁎ *NBFACFUT*. Si l'on imagine une situation qui conduirait cha face à être découpée une fois, le coût serait donc de
3 ⁎ *NBSOM + 40* ⁎ *NBFACES + 512 × 512 octets*

- le tableau *IMAGE* peut être soit réduit de taille (en n'étudiant qu'une po tion d'image à chaque fois), soit être remplacé par une étude ligne par l gne telle qu'elle est faite dans le cas de l'algorithme de WATKINS.

3.423 - Critique

L'algorithme de NEWELL et SANCHA fait partie de la famille des algorithmes d "à listes de priorité". Ces algorithmes se décomposent en deux phases :

- étude de la scène pour établir des priorités entre les faces (ou les grou de faces dans le cas de SCHUMACKER), étude se faisant dans l'espace utili teur,

- calcul de l'image en fonction des priorités précédentes, calcul se faisant dans le système image.

Cet algorithme est très utilisé, en particulier parce qu'il permet de réali des effets intéressants du point de vue du coloriage des faces. En effet, lorsqu'il y a recouvrement, on peut soit purement et simplement remplacer la couleur inscrite par une nouvelle couleur, soit faire une combinaison que conque entre ces couleurs. Cette dernière technique permet en particulier de rendre des effets de transparence et de reflets très réalistes [Bln 76] .
Le coût d'un tel algorithme peut être évalué de la manière suivante :

- première phase, tri des faces. Il est clair qu'une borne maximale est de l'ordre de C_1 ⁎ *(NBFACES + NBFACFUT)*^{2}. En général, les techniques mises en oeuvre permettent de descendre en dessous de cette valeur.

- deuxième phase : inscription des faces. Le temps d'inscription d'une face proportionnel à sa surface. Si l'on utilise l'algorithme d'inscription donn en 1.232, alors le traitement peut se détailler en

. inscription du contour	$C_2 * L$	longueur totale des arêt
. remplissage	$C_3 * SURFACE$	surface totale de la pro jection des faces

L'algorithme de NEWELL et SANCHA aura un comportement dépendant essentiellem de la surface des faces à traiter (et donc de la longueur des arêtes).

La figure 3.25 représente une scène traitée par la méthode de NEWELL et SANCI La sortie a été réalisée sur imprimante. L'image est constituée d'une matric 40 ×60

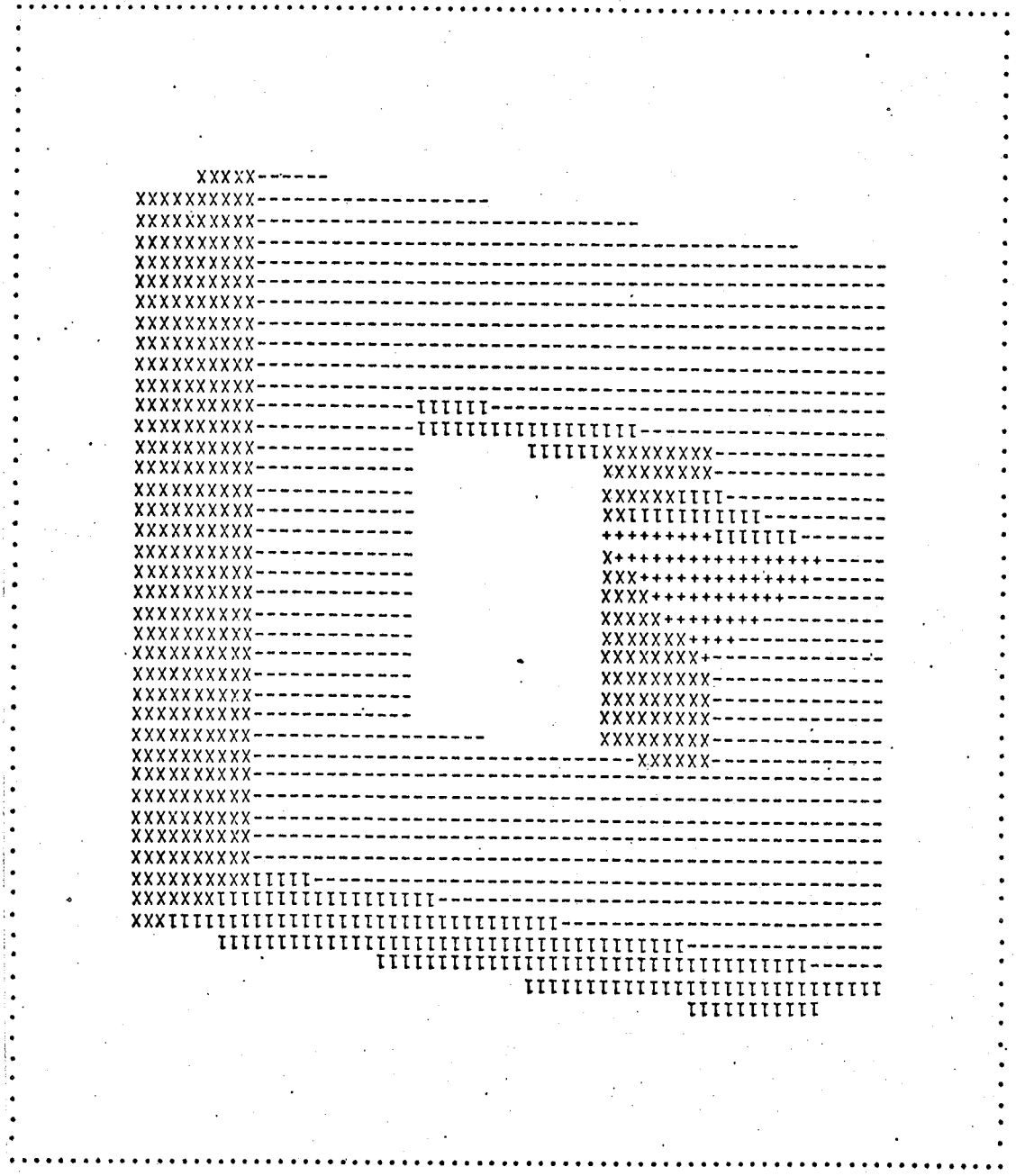


figure 3.25 Présentation d'une scène par la méthode de NEWELL, NEWELL et SANCHA

3.43 Coloriage des images

La technique la plus élémentaire (qui est employée dans les exemples présent dans ce document) consiste à attribuer une couleur arbitraire à chaque face à utiliser cette couleur lors de la présentation des faces visibles. Cette t nique ne conduit pas à des images très réalistes. En particulier, l'oeil a tendance à interpréter les degrés lunimeux en terme d'ombrage, et l'image présentée peut être difficile à saisir si la répartition des couleurs est aléatoire.

Une première amélioration consiste à moduler la couleur des faces (ou à attr buer une couleur) en supposant la présence d'une source lumineuse. On défini alors une fonction qui permet de tenir compte de la position de cette source à partir de l'angle fait par la normale à la face considérée avec la droite joignant le centre de la face et la source lumineuse. On obtient ainsi des images plus facilement interprétables, mais peu satisfaisantes car ne préser tant pas suffisamment de dégradés.

GOURAUD [Gou 71] a montré que l'image était fortement améliorée lorsqu'on calculait une luminosité en chaque point, luminosité obtenue par combinaison linéaire des intensités des faces voisines. Les images ainsi obt nues présentent un certain modelé, avec des défauts sur les arêtes, dûs au f que seule la dérivée première de la luminosité est traitée et que l'oeil est sensible aussi à la dérivée seconde.

BUI TUONG PHONG ([Pho 73] ,[Pho 75]) a proposé des techniques per mettant de réduire ces défauts, tout en tenant compte de transparences et de reflets sur les objets présentés.

BLINN et NEWELL ([BlN 76],[Bli 77]) ont proposé des techniques de calcul d'images très réalistes permettant de tenir compte de plusieurs sourc lumineuses, de transparences et de reflets d'autres objets en fonction de la texture de l'objet étudié. Il est très difficile de distinguer de telles im ges de photographies d'une scène réelle.

Signalons pour terminer des techniques nouvelles permettant de dessiner les ombres portées ([Cro 77], [Dal 77]).

3.5 ANALYSE COMPARATIVE DES ALGORITHMES D'ELIMINATION DE PARTIES CACHEES

3.51 Analogies et différences

Les quatre algorithmes que nous avons étudié présentent des similitudes plu ou moins importantes. Les deux qui semblent les plus proches sont les algori thmes de WARNOCK et de WATKINS. Les schémas que nous avons donné en 3.321 et en 3.411 montrent à l'évidence la similitude de structure des deux programme

- une boucle générale permettant de couvrir l'ensemble de l'image, fenêtre p fenêtre ou ligne par ligne,

- une procédure de calcul des relations spatiales, soit par rapport à une fe nêtre, soit par rapport à une bande d'un plan de balayage.

- une procédure de décision, permettant de savoir si la zone inventoriée éta facilement représentable ou non,

- une procédure de découpage soit dichotomique, soit s'appuyant sur les somm présents dans la zone à découper.

Les différences essentielles entre ces deux algorithmes portent su les points suivants :

- l'algorithme de WARNOCK traite des objets appartenant systématiquement à u univers ayant une dimension de plus que WATKINS : l'étude se fait par rapp à des portions d'espace, alors que dans le cas de WATKINS l'étude se fait dans un plan.

- par contre, ce traitement permet de mieux tenir compte du facteur essentie dans tous ces calculs, à savoir la présence de faces cachant une portion d la scène. En effet, la face étant l'élément de base des calculs chez WARNO il est très facile de conserver des renseignements globaux sur les positio relatives entre faces, alors que dans le cas de WATKINS la notion de face remplacée par celle de section par un plan.

Les deux algorithmes utilisent des propriétés de cohérence de la scène pour réduire la complexité des calculs. La même différence se retrouve continuité sur les surfaces chez WARNOCK, continuité sur les arêtes chez WATKINS.

Cependant, nous pouvons trouver d'autres analogies, moins évidente Par exemple, le fait d'étudier successivement les sections dues aux plans de balayage peut aussi s'interpréter comme une étude point par point des arêtes formant les faces des polyèdres. En effet, la gestion des segments contenus dans une section se fait essentiellement en observant le comportement des points d'intersection de chaque arête avec les plans de balayage. Or l'algorithme de GALIMBERTI MONTANARI repose également sur une analyse des ar tes point par point. La différence essentielle repose ici sur le fait que da le cas de GALIMBERTI MONTANARI cette étude est faite dans l'espace utilisate ce qui ne permet pas de tenir compte de la résolution de l'écran, alors que dans le cas de WATKINS ce calcul est fait en tenant compte de la résolution l'écran puisque le nombre de plans de balayage dépend de la résolution en y.

Cette différence entre calculs exacts et calculs approchés permet d'opposer les algorithmes de GALIMBERTI et de NEWELL aux algorithmes de WARNOCK et WATKINS.

Une autre analogie peut être trouvée si, dans l'algorithme de WATKINS, on utilise l'algorithme de calcul des dispositions relatives des se ments par projection point par point. En effet, le principe proposé en 3.412 est le même que celui retenu pour l'algorithme de NEWELL SANCHA, sauf que l' s'intéresse dans le premier cas à une étude de segments de droite et dans le second cas à une étude de surfaces. Il y a de nouveau une dimension supplémentaire qui est traitée.

Nous terminerons en présentant une extension faite par CATMULL [Ca pour la présentation point par point de surfaces analytiques du type "surfaces de BEZIER". L'algorithme utilisé est une combinaison de l'algorith de WARNOCK et du procédé d'étude de segments que nous avons proposé en 3.412

- l'idée de base est de maintenir en tout point de la surface de visualisati la hauteur maximale atteinte lors de la projection d'un élément de surface

- pour faciliter ce calcul, chaque carreau élémentaire est découpé autant de fois qu'il est nécessaire jusqu'à atteindre la taille du point élémentaire de la surface de visualisation. Le découpage se fait en partageant en quat chaque carreau trop complexe.

Ce procédé a été étendu pour ne dessiner que les arêtes visibles [Gri 75] Il est cependant fort coûteux du point de vue temps de calcul (plusieurs diz nes de minutes pour les exemples habituellement présentés).

3.52 Passage dessin-image ou image-dessin

Nous étudions dans ce paragraphe la possibilité d'étendre les algorithmes qu nous venons d'étudier soit à la composition d'images, soit à la composition de dessins.

En ce qui concerne l'algorithme de GALIMBERTI-MONTANARI, le princi même de l'algorithme interdit d'obtenir une image, puisque la notion de face est perdue au profit de l'étude arête par arête.

En ce qui concerne l'algorithme de WARNOCK, il est possible de col rier le dessin. En effet, on se trouve dans les cas suivants :

- soit on a trouvé un polygone écran. Il suffit de remplir la fenêtre avec la couleur du polygone en question,

- soit on a à dessiner une portion de face s'inscrivant dans la fenêtre. Une légère modification de l'algorithme de remplissage de taches donné en 1.21 permet de réaliser ce coloriage.

En ce qui concerne l'algorithme de WATKINS, la modification de l'algorithme consiste à tenir une trace de la nature visible ou invisible d' arête. Lors du début de traitement d'une arête (apparition d'un sommet au ni veau d'un plan de balayage), on note si le premier point est visible ou non.

Le principe de tracé consiste alors à s'assurer, à chaque plan de balayage, si une arête en cours de traitement ne change pas de nature :

- si elle passe de cachés à visible, on note ce changement d'état et les coordonnées de ce point ;

- si elle passe visible à cachée, on trace le segment de droite joignant le point conservé précédemment au point auquel on est arrivé.

On voit donc ici ressortir nettement le caractère d'étude point par point de arêtes.

En ce qui concerne l'algorithme de NEWELL, NEWELL et SANCHA, l'idé de base consiste à conserver, lors de la projection des faces, une trace des contours. Une fois la projection terminée, les contours apparents seront inscrits dans la matrice ayant servi de modèle et il n'y aura plus qu'à les afficher. Un exemple est donné en figure 3.26

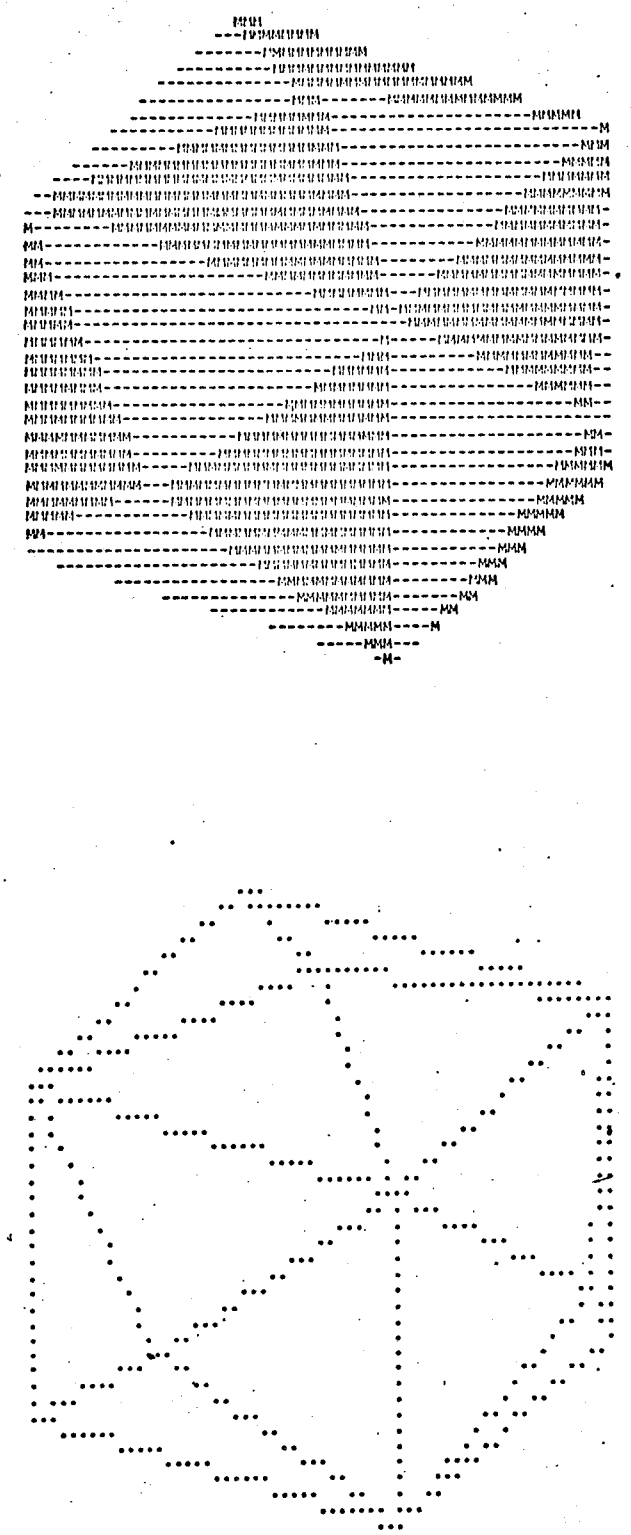


figure 3.26 Présentation d'une scène par la méthode de NEWELL, NEWELL et SANCHA

Cependant, l'aspect esthétique des traits n'est pas très bon, à cause de l'approximation utilisée pour engendrer les segments de droite lors de l'inscription des contours. On pourrait envisager d'améliorer le dessin en cherchant à remplacer un ensemble de points par un segment de droite. On aur le schéma suivant :

modèle au trait → *passage image* → *amélioration image* → *modèle au trait*

On peut se demander si une telle opération est réellement rentable.

En résumé, seul l'algorithme de GALIMBERTI-MONTANARI ne permet qu'un affichage sous la forme d'un dessin. Les autres algorithmes permettent d'obtenir l'une ou l'autre des présentations. Cependant, si l'on retient un critère de qualité esthétique, l'algorithme de NEWELL-SANCHA ne permet pas d'obtenir simplement des dessins de bonne qualité.

3.53 Traitement des faces se coupant

On peut s'intéresser à des scènes où les faces se coupent ailleurs que par u arête commune.

L'algorithme de GALIMBERTI-MONTANARI ne permet pas de traiter ce t de faces.

Dans le cas de l'algorithme de WARNOCK, la version standard fournit une visu lisation point par point de la droite d'intersection. En effet, il n'y aura jamais possibilité de classer les deux faces l'une par rapport à l'autre, sa lorsqu'un bord de la fenêtre sera exactement sur la trace de la droite d'intersection. Il est cependant possible de détecter dans certains cas la p sence de ces deux faces et de les dessiner, en particulier, lorsque l'on a d faces jouant le rôle d'écran et se coupant. Il est alors très facile de calc l'intersection et de la dessiner. Ce calcul est d'ailleurs le pendant du cas de deux segments écrans se coupant dans l'algorithme de WATKINS.

En ce qui concerne l'algorithme de WATKINS, le traitement des inter sections se fait de manière extrèmement simple. En effet, la détection d'un t cas de figure se fait lors de l'étude des positions relatives des segments. I y a alors création d'une arête fictive qui est traitée comme les autres arête Dans le cas de la production d'images, la droite d'intersection ne sera visib que si les facettes se coupant ont des couleurs différentes. Dans le cas de l production d'un dessin, la droite d'intersection sera dessinée correctement.

En ce qui concerne l'algorithme de NEWELL et SANCHA, les faces ayar une intersection commune seront découpées automatiquement lors du classement. Lors de l'affichage, la droite d'intersection n'apparaîtra que si les deux fa cettes ont des couleurs différentes. Par contre, dans le cas où on cherche se lement le contour, une gestion un peu délicate doit se faire. En effet, il fa dra différencier les découpages faits pour le besoin du classement et ceux dû à une intersection, de manière à pourvoir afficher correctement les segments représentatifs.

En résumé, seul l'algorithme de GALIMBERTI et MONTANARI ne traite pas les fac se coupant. Ce type de configuration se rencontre pourtant relativement souve en particulier pour des raisons de commodité de description : l'opérateur déc les faces sans se soucier de leur intersection. La figure 3.27 donne un exem de cubes imbriqués, présentés grâce à l'algorithme de WATKINS.

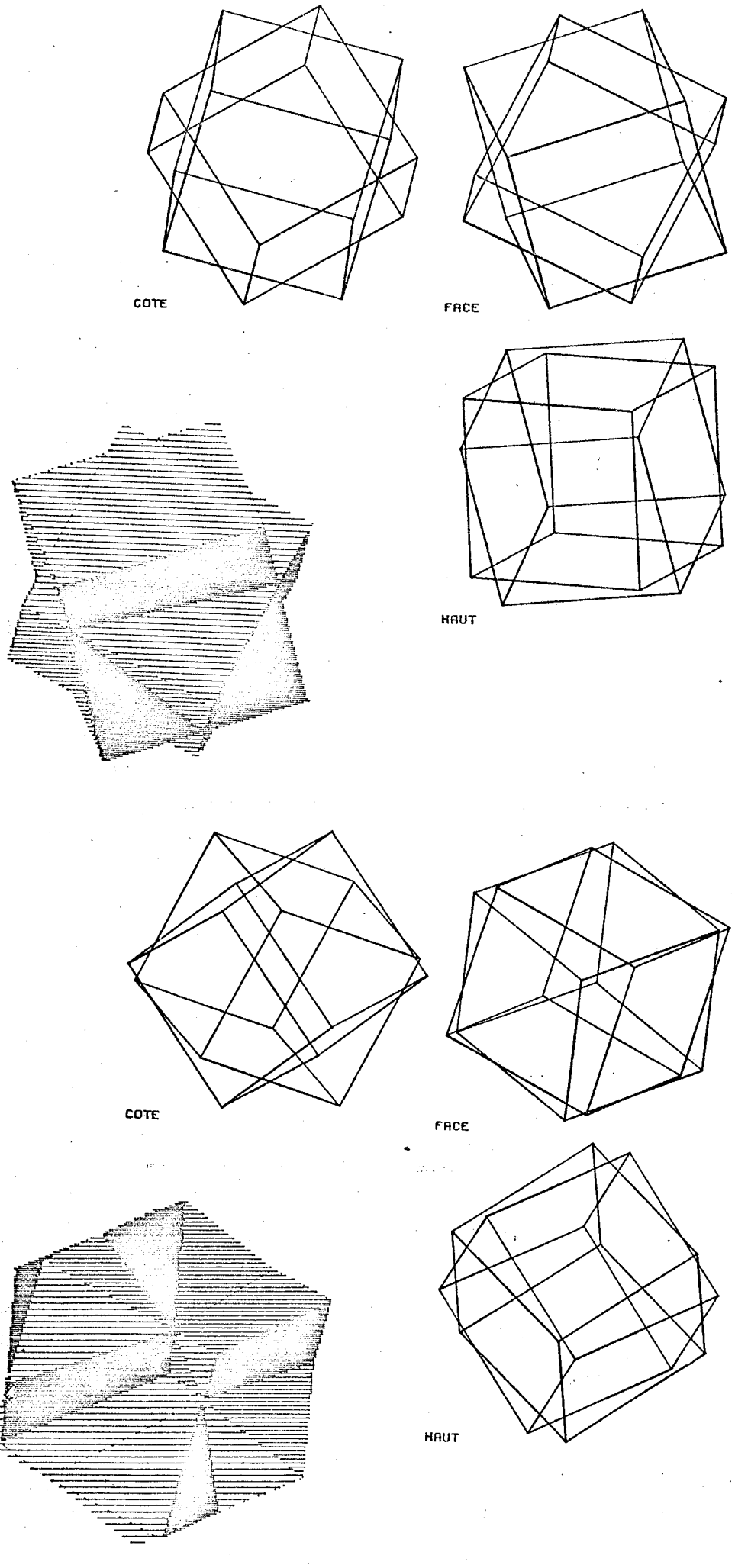


figure 3.27 Présentation de l'intersection de faces (WATKINS)

3.54 Récapitulation.

Le tableau 3.2 regroupe les principales caractéristiques des algorithmes étu

	GALIMBERTI MONTANARI	WARNOCK	WATKINS	NEWELL-SANCHA
éléments de la scène	solides *arêtes communes à deux faces au maximum*	faces polygonales planes	faces polygonales planes *arêtes communes à 2 faces au maximum*	faces polygonales planes *quadrilatère*
traitement de faces se coupant	non	oui	oui	oui (pas de dessi
précision des calculs	quelconque	dépend de l'écran (1024×1024)	dépend de l'écran (512×512)	dépend de l'écran (512×512)
esthétique du dessin	très bonne	moyenne (dépend du niveau des tests	bonne	mauvaise
de l'image		moyenne	bonne	très bonne
complexité de la programmation	moyen	moyen	très complexe	moyen
taille mémoire (en mots)	*3*NBSOM +31*NBFACES*	*3*NBSOM +37*NBFACES*	*3*NBSOM +33*NBFACES +1024 octets*	*3*NBSOM + 40*NBFACES + 512×512 octets*

tableau 3.2 Caractéristiques de quatre algorithmes

Au vu de l' ensemble de ces caractéristiques, on peut chercher à déterminer quel algorithme doit être choisi, de préférence aux autres. Si l'on choisit critère de bonne qualité du dessin (ou de l'image), nous pouvons faire les r marques suivantes :

- s'il s'agit de réaliser des dessins très précis (du type mécanique), il fa s'orienter vèrs une solution de type GALIMBERTI-MONTANARI. Cependant, il y aura tout intérêt à n'utiliser cet algorithme que pour les sorties définit sur traceur de courbes, et utiliser un autre algorithme pour la préparatio du dessin à l'écran.

- si l'on se contente de dessins au trait, l'algorithme de WARNOCK est facil à mettre en oeuvre. Il sera d'autant plus intéressant que le degré de comp xité de la scène sera élevé.

- l'algorithme de WATKINS est intéressant dans la mesure où il permet d'obte des dessins d'assez bonne qualité, aussi bien que des images très réalist Il est donc un excellent élément de base. Sa programmation est cependant délicate.

- l'algorithme de NEWELL et SANCHA est intéressant de par la simplicité de l programmation. Il permet surtout de traiter très facilement les problèmes reflets ou de transparence qui sont parfois cruciaux (exemple de l'industr du verre). D'autre part, son extension possible à des surfaces analytiques est un bon atout.

Si l'on cherche à discuter de critères de réalisation informatique, nous fer d'abord remarquer que ces algorithmes exigent une taille de mémoire importan A titre d'exemple, la scène représentée page 110 nécessite au moins 4K mots pour sa représentation. Sachant que l'utilisation croissante des petits calculateurs conduit à utiliser des tailles mémoire limitées avec apport d'u mémoire secondaire, il est alors évident que les algorithmes réalisés sur un petite machine seront pénalisés du point de vue de leurs performances, à cau d'échanges fréquents entre mémoires principales et secondaires.

En ce qui concerne l'efficacité des algorithmes étudiés, nous avons renoncé présenter un tableau comparatif des performances pour les deux raisons suiva tes :

- pour que les comparaisons aient un sens, il faut que les programmes étudié soient d'un même niveau technique. Nous ne parlons pas seulement d'utilise un même langage de programmation, ni même de la nécessité de prouver que l programmation est faite dans tous les cas avec un soin parfait.

Nous évoquons surtout le fait de disposer de programmes correspondant à des niveaux d'algorithmes analogues, pour ne pas comparer, par exemple, la versi la plus basse de l'algorithme de WARNOCK avec la version la plus évoluée de l'algorithme de WATKINS. Il est évident que ce type de comparaison est très difficile à réaliser objectivement.

- pour effectuer des mesures, il faut disposer de jeux d'essais qui ne fausse pas les résultats attendus. Nous avons montré que le coût des différents algorithmes dépendait de facteurs de la complexité de la scène, facteurs tr variables. Ne disposant pas d'une étude de cette notion de complexité, nous n'avons pu établir un tel jeu d'essai.

Cette dernière constatation nous conduit à proposer deux réflexions contradic toires :

- les techniques de traitement d'objets de l'espace à trois dimensions sont suffisamment bien maîtrisées à l'heure actuelle pour que l'on cherche à développer des applications traitant de la conception de formes tridimensionnelles,

- il faut étudier la notion de complexité d'une scène, de manière à pouvoir établir si certains éléments de cette complexité sont déterminants sur les calculs d'élimination de parties cachées. Le choix définitif d'un algorithme pourra se faire qu'une fois cette connaissante théorique acquise. C'est alors seulement que le problème de l'élimination des parties cachées pourra être considéré comme vraiment résolu.

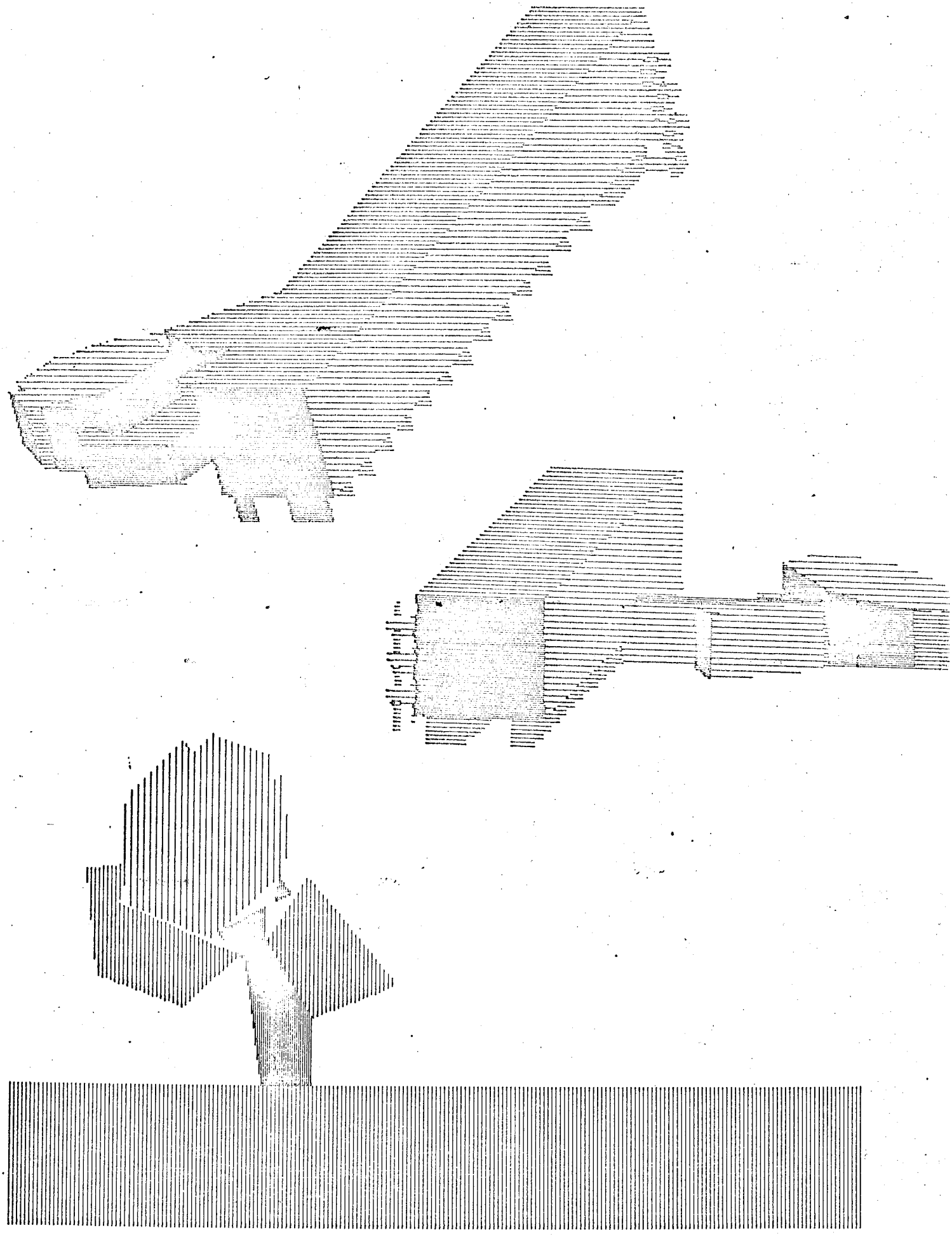
3.6 CONCLUSION

Nous avons présenté dans ce chapitre les éléments de base qui ont permis de réaliser un logiciel interactif de visualisation de polyèdre. Nous avons ains montré un exemple de couche de programmes que l'on peut superposer à un logic graphique de base, en l'occurence GRIGRI.

Nous remarquerons simplement, par rapport à la proposition de norme américain que les fonctions utilisées pour réaliser ce logiciel (en particulier les tra formations géométriques et les projections) pourraient parfaitement être isol du contexte VISU3D pour devenir des primitives d'un logiciel de visualisation de scènes à trois dimensions passif. Nous aurions dès lors l'équivalent de l'ensemble des primitives de préparation à la visualisation proposées dans l norme. Il resterait encore à ajouter des primitives de description. L'étude de quelques exemples d'application utilisant VISU3D montrera en particulier l diversité des primitives nécessaires. Nous réserverons alors leur place au ni veau d'un logiciel de description géomètrique, venant former une nouvelle cou che au dessus de VISU3D.

CHAPITRE 4

QUELQUES EXEMPLES DE PROGRAMMES D'APPLICATION



IMAG bâtiment d (ENSIMAG)

4. QUELQUES EXEMPLES DE PROGRAMMES D'APPLICATION

4.1 CLASSIFICATION DES APPLICATIONS

Nous reprenons ici la classification proposée dans [Luc 73] . Les différents logiciels de production de dessins (ou d'images) peuvent être regroupés en quatre grandes catégories:

- Les logiciels produisant des dessins ou images statiques

  . logiciels à vocation essentiellement graphique
  Il s'agit d'applications dans lesquelles le dessin est non seulement le support de réflexion, mais surtout le produit final attendu des calculs. Les principaux domaines d'application sont la conception de formes artistiques, la création de motifs pour les industries textiles ou pour la décoration, la composition de textes ou d'affiches, la cartographie.

  . logiciels pour la conception assistée par ordinateur
  Nous classons dans cette catégorie les applications pour lesquelles le dessin est un support de réflexion et de communication. Le dessin est une représentation des données qui sont traitées, toute modification de ces données étant répercutée sur la représentation graphique et vice-versa. Les principaux domaines de la conception assistée sont la mécanique, le génie civil, l'électronique, les industries automobile, aéronautique et navale, l'architecture, l'urbanisme.

- Les logiciels produisant des dessins ou images dynamiques
  La succession des images dans le temps est un paramètre supplémentaire pour ce type de logiciels. Deux catégories peuvent être distinguées:

  . logiciels où le temps est simulé
  L'enchaînement de la succession des dessins est géré par programme. Nous rangeons dans cette catégorie les systèmes de production de dessins animés et les logiciels d'illustration de programmes.

. logiciels en temps réel

L'enchaînement des dessins se fait en temps réel, commandé par des dispositifs extérieurs au calculateur. Les exemples les plus classiques se trouvent dans le contrôle de processus industriels, le contrôle de trafic urbain, aérien ou maritime.

Cette classification nous permettra surtout de mettre en valeur certaines caractéristiques propres à ces catégories de logiciels. Il est bien évident que certains systèmes utilisent des techniques appartenant à plusieurs de ces catégories. Par exemple, il n'est pas rare qu'un système de conception assistée propose, à un moment donné de la phase de conception, l'établissement de documents graphiques qui peuvent alors être considérés comme le produit d'un logiciel de la première catégorie.

Nous présentons dans la suite du chapitre des exemples d'application appartenant aux trois premiers types que nous avons définis. Nous n'avons aucune expérience en ce qui concerne la production de dessins en temps réel, ce qui explique que nous n'en parlerons pas.

4.2 APPLICATIONS ESSENTIELLEMENT GRAPHIQUES

4.21 Etude de la répartition des chromosomes humains lors de la métaphase

Dans cette étude, traitée en commun avec le Laboratoire de Cytologie du Centre Hospitalier de Grenoble et l'équipe de Statistiques de l'IMAG, le problème consistait à vérifier si la disposition des chromosomes humains, lors de la métaphase, obéit à une loi de probabilité normale, ou si certains couples de chromosomes ne montrent pas une certaine attirance (ou répulsion). La première partie de ce travail a reposé sur l'utilisation d'une tablette graphique pour obtenir les coordonnées des chromosomes. Nous avons étudié quatre séries de cent photos, chaque photo étant obtenue après une préparation spéciale visant à mettre en valeur certaines caractéristiques des chromosomes (en particulier des bandes de couleurs différentes), permettant leur identification couple par couple (voir photo ci-dessous).

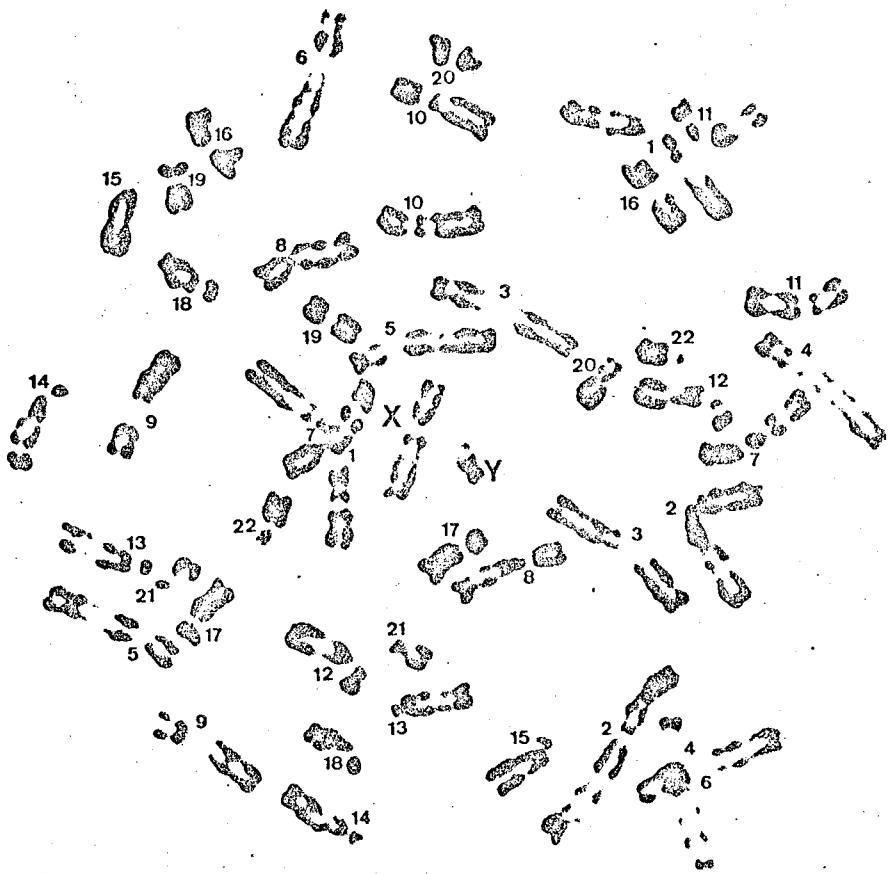


Photo X., Laboratoire de Cytologie du CHR de Grenoble

Pour ne pas être tributaires des déformations dûes au tirage des photos (en particulier différences d'échelle), nous avons remplacé les coordonnées (ξ_i, η_i) des centromères par

$$\left\{ x_i = \frac{\xi_i - \bar{\xi}}{\sigma_\xi} , \quad y_i = \frac{\eta_i - \bar{\eta}}{\sigma_\eta} \right\}$$

où (ξ_i, η_i) sont les coordonnées mesurées pour le chromosome i

$(\bar{\xi}, \bar{\eta})$ sont les coordonnées du barycentre des 46 centromères

$(\sigma_\xi \quad \sigma_\eta)$ sont les variances empiriques des observations ξ_i et η_i

(x_i, y_i) sont les coordonnées utilisées pour les calculs.

Notre étude a porté sur deux points:

- étude des distances au barycentre [SJL76] ,

- mise en évidence de certaines stabilités dans les voisinages des centromères [SJL 77] .

Nous évoquerons ici le deuxième point. Partant de l'hypothèse que les 46 points (x_i, y_i) de chaque plaque pouvaient être considérés comme un échantillon de taille 46 d'une loi normale centrée réduite dans R^2, nous avons cherché à établir si, dans l'ensemble des plaques d'une série, certains chromosomes ne présentaient pas des affinités se traduisant par des distances anormalement prôches.

La méthode utilisée a consisté à déterminer, pour un nombre $\alpha \in]0,1[$ et pour tout point M du plan, un carré $\Gamma(M)$ de centre M, de côtés parallèles aux axes, tel que la probabilité de $\Gamma(M)$ pour la loi normale $G([\begin{smallmatrix}0\\0\end{smallmatrix}], [\begin{smallmatrix}1&0\\0&1\end{smallmatrix}])$ dans le plan soit égale à α.

Pour chaque centromère, nous avons calculé le carré correspondant et dressé la liste des chromosomes situés à l'intérieur du carré (voir figure 4.1).

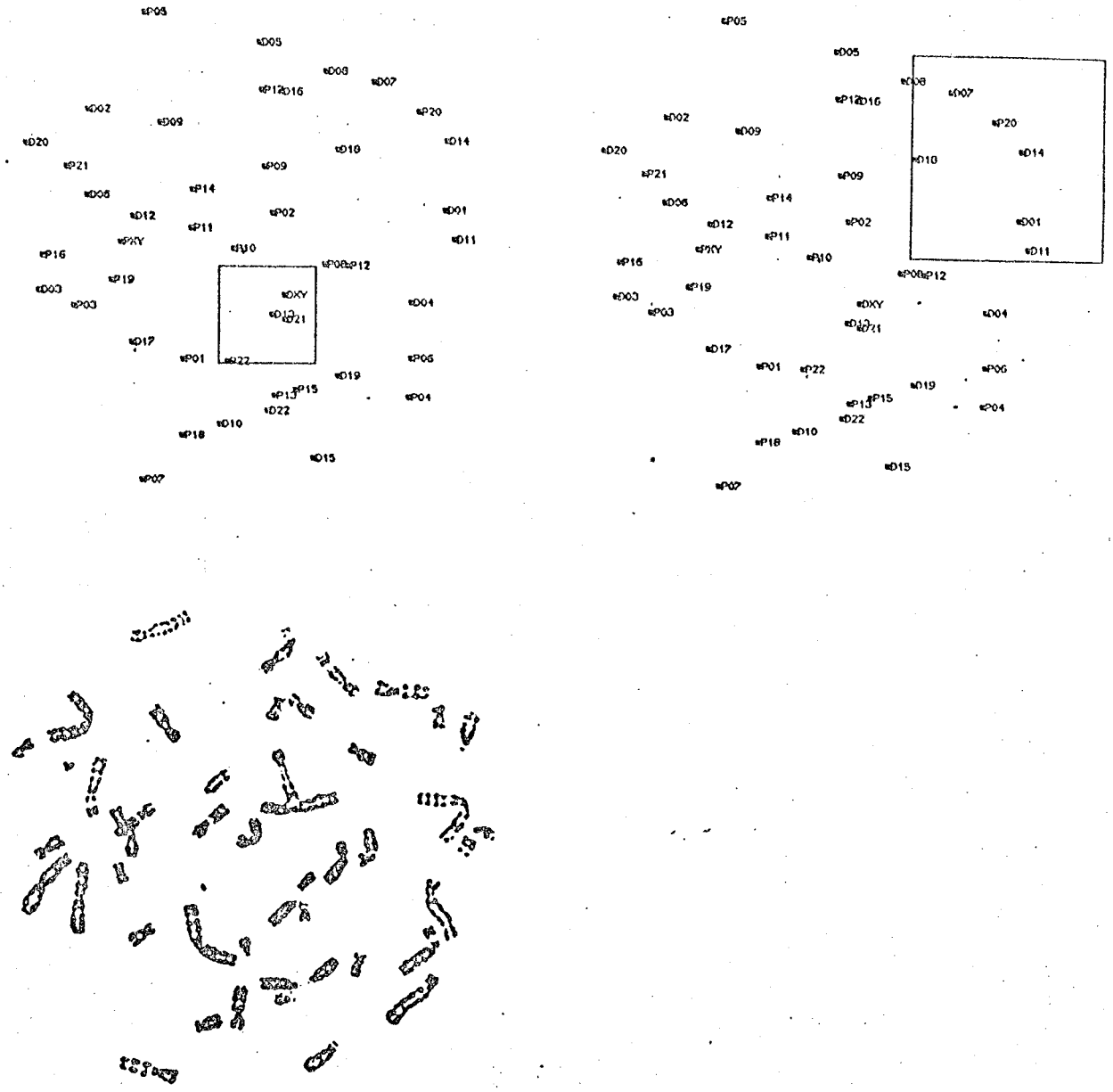


figure 4.1 test de proximité

Le résultat de ces calculs est une matrice de proximité (46x46) contenant, pour un couple (i,j), le nombre d'apparitions pour cent plaques du chromosome j dans le carré centré sur le chromosome i. Aucun moyen de discerner les deux chromosomes d'une même paire à partir des préparations étudiées n'étant fourni, nous avons en fait étudié une matrice 23x23 pouvant s'interpréter comme la fréquence de proximité d'une paire de chromosomes vis-à-vis d'une autre paire de chromosomes. La figure 4.2 donne un exemple d'une telle matrice (les valeurs de la diagonale principale ont été remplacées par des zéros).

SERIE 100XV02

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	0	50	37	29	35	45	40	24	36	35	33	38	41	31	33	47	40	40	40	36	42	46
2	45	0	46	29	35	35	38	38	28	46	46	41	33	39	45	38	33	32	30	36	38	37
3	32	55	0	42	42	29	51	40	30	35	47	45	40	53	33	38	33	34	39	42	32	40
4	39	32	39	0	44	39	37	44	34	34	36	33	42	36	50	34	42	46	45	30	41	46
5	34	52	36	41	0	34	36	49	35	36	40	37	38	35	41	38	48	49	34	36	48	35
6	41	38	29	42	39	0	39	39	43	40	38	45	43	42	43	35	38	44	45	46	47	42
7	37	36	52	47	32	37	0	38	36	32	49	36	43	42	37	41	38	36	43	45	35	38
8	21	40	36	41	44	43	33	0	50	37	44	32	37	38	33	30	45	41	34	32	33	48
9	51	23	31	34	38	42	34	57	0	43	26	25	26	40	42	45	40	26	55	40	40	41
10	28	43	23	38	40	36	28	37	39	0	47	34	38	41	41	38	40	43	30	42	36	37
11	45	40	48	36	33	31	42	44	27	44	0	40	36	39	42	48	48	35	51	43	42	36
12	39	37	37	38	33	43	40	34	41	42	41	0	44	47	50	40	40	42	50	47	28	36
13	36	38	38	38	35	38	45	37	29	41	38	41	0	60	54	46	31	37	46	27	46	51
14	30	34	26	42	29	42	42	43	43	41	34	44	55	0	52	39	29	37	43	41	50	56
15	27	39	27	40	36	44	29	50	44	41	36	47	47	51	0	30	36	45	49	40	50	50
16	31	39	42	37	40	28	35	29	48	42	50	33	49	44	33	0	36	49	46	37	43	43
17	35	36	29	44	47	26	31	50	40	30	38	37	31	36	35	38	0	35	37	50	36	50
18	41	26	49	37	45	33	33	42	27	53	31	41	45	37	40	35	37	0	43	55	34	37
19	36	31	39	41	37	36	35	25	52	31	45	41	43	39	51	40	35	42	0	45	39	47
20	26	52	38	30	34	44	36	32	41	38	26	44	32	43	30	33	40	50	43	0	49	44
21	38	26	31	33	47	43	30	31	46	30	43	28	38	51	50	37	34	37	37	49	0	34
22	41	26	34	34	31	30	36	53	40	34	32	37	43	48	51	47	51	37	43	49	51	0
23	36	28	44	48	30	30	33	45	56	44	39	30	38	36	38	43	39	36	41	37	41	33

figure 4.2 matrice de proximité

Il est évident que ce qui nous intéresse dans une telle matrice est l'existence et la répartition de fortes valeurs, indiquant une fréquence de proximité anormalement élevée. Pour guider notre recherche, nous avons d'abord utilisé une représentation permettant de voir rapidement si des zones intéressantes existaient. Pour ce faire, nous avons visualisé les éléments dont la valeur était supérieure à un certain seuil à l'aide d'une étoile. La figure 4.3 montre un exemple de cette représentation, le seuil étant fixé à un minimum de 42 apparitions.

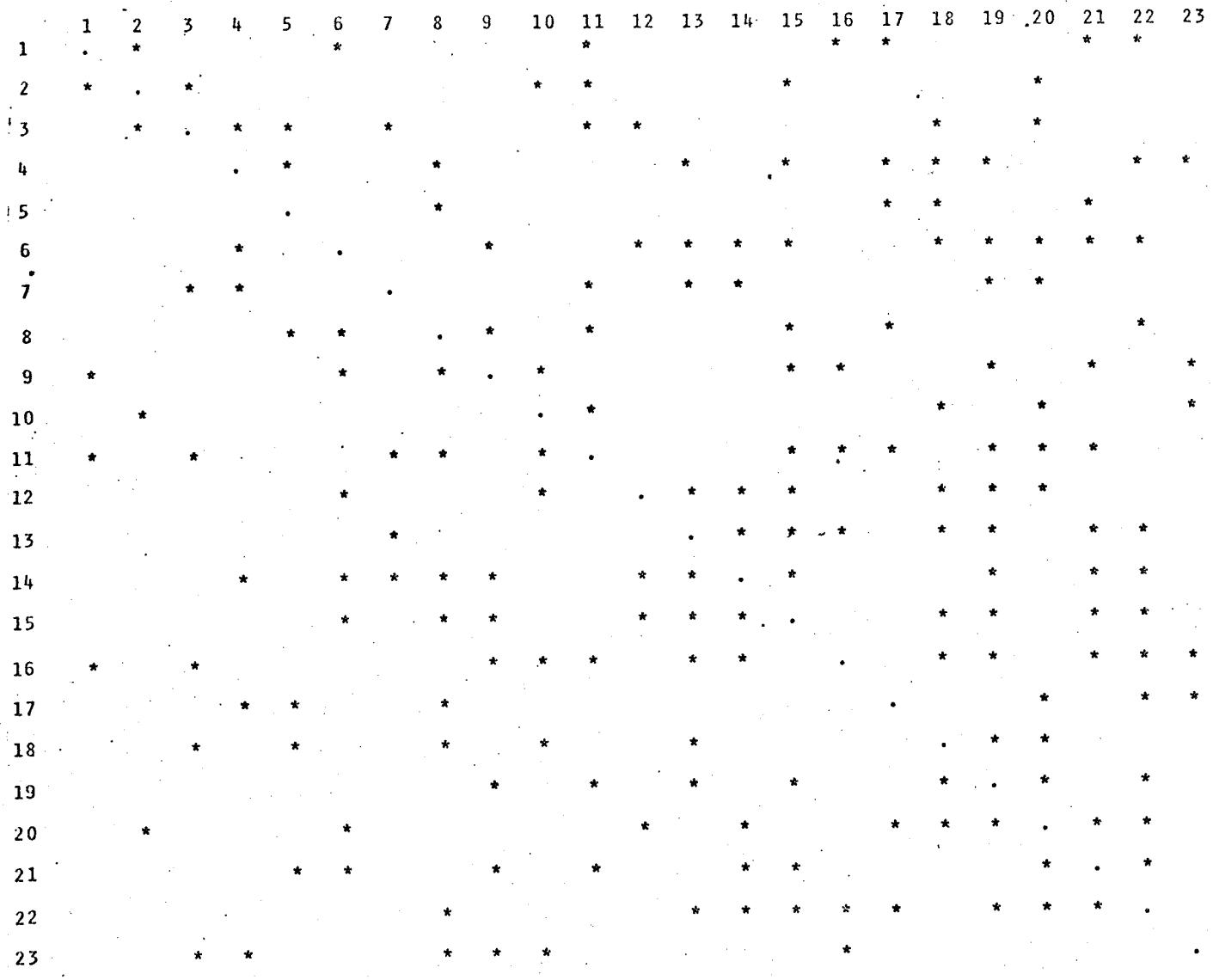


figure 4.3 Proximité des couples de chromosomes

Cependant, cette représentation ne donnait aucune indication quant aux différences d'amplitude. Nous aurions pu utiliser des luminosités différentes, utilisant la technique exposée en 1.214. Cependant, l'oeil n'étant réellement sensible qu'à de forts contrastes, nous avons préféré utiliser un logiciel permettant d'obtenir une représentation tridimensionne de matrices [AnL 74].

La figure 4.4 donne plusieurs exemples, correspondant :

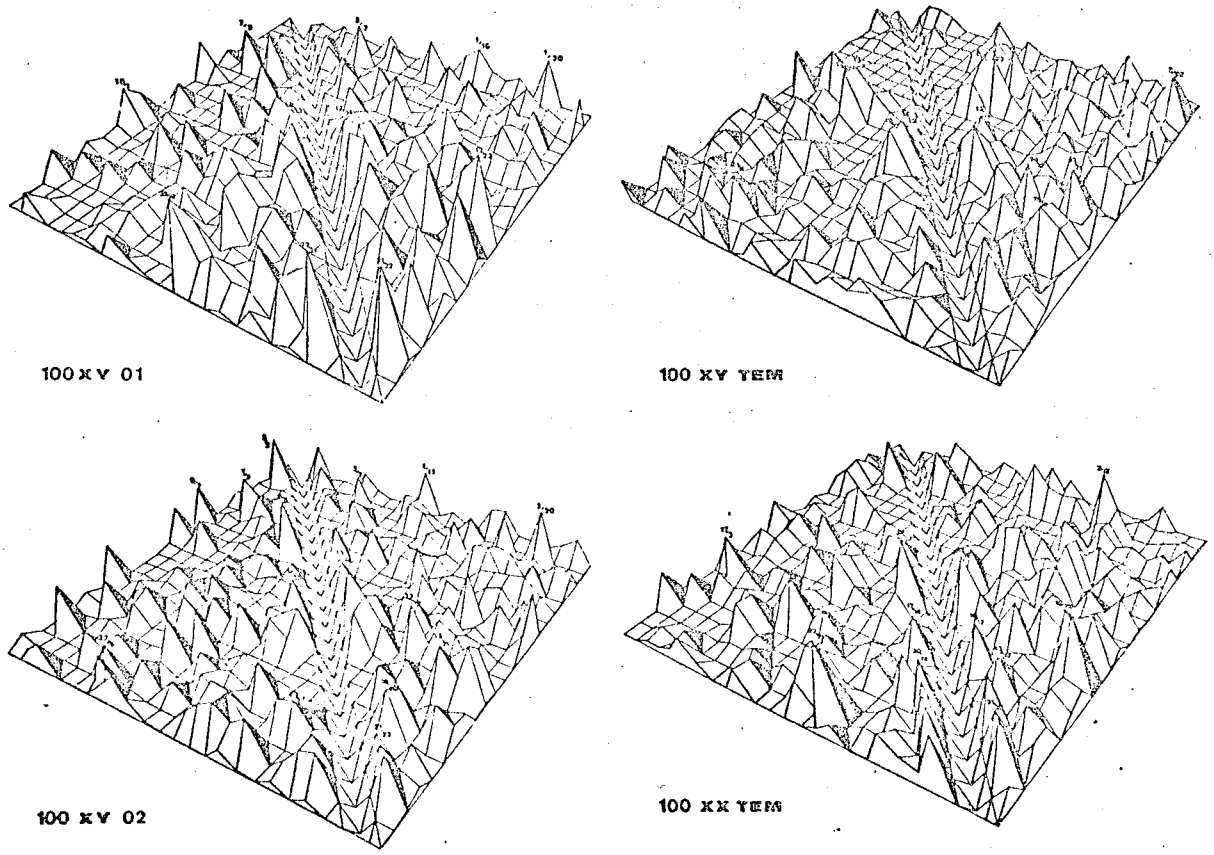
- à la présentation des résultats des quatre séries, telles que les matrices étaient calculées,

- à la présentation des résultats des quatre séries, en privilègiant les les valeurs situées au dessus d'un certain seuil, afin de mettre en valeur les pics les plus élevés.

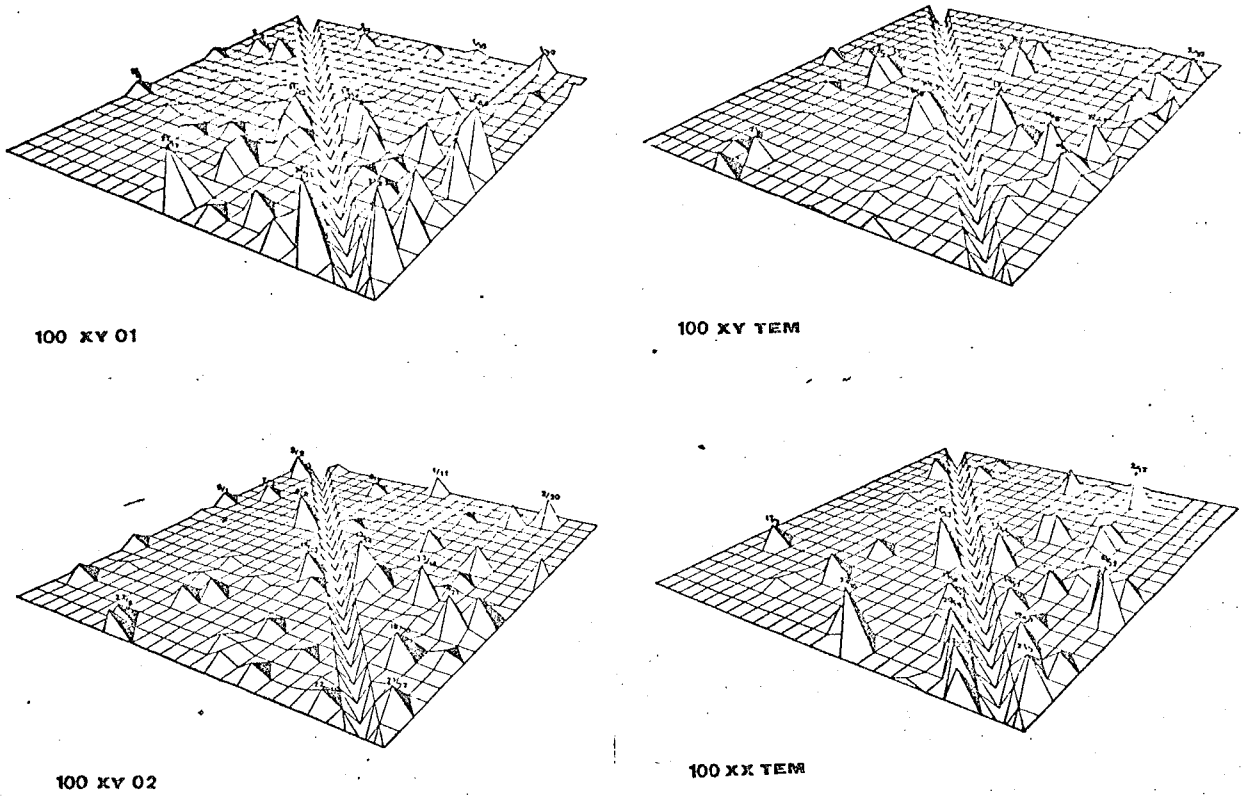
Nous avons pu ainsi vérifier des associations préférentielles entre couples de chromosomes, tels que les couples 13-14, 13-15 et 21-22. Les proximités des groupes 13, 14, 15 et 21, 22 ont été également mises en évidence.

Nous ferons deux remarques à propos de cet exemple:

- nous avons présenté un exemple classique d'utilisation de graphiques tridimensionnels, apportant des renseignements essentiellement qualitatif L'interprétation de ces graphiques permet de choisir la zone qui est intéressante pour les calculs, et donc de guider interactivement le programme d'application. Cette situation est très fréquente en analyse numérique, où les techniques graphiques interactives permettent de contrôler efficacement le déroulement d'un algorithme, par exemple par le choix des valeurs initiales.



présentation des matrices initiales



présentation avec un seuil à 52 apparitions

figure 4.4 Présentation tridimensionnelle de matrices de proximité

- la visualisation de fonctions de deux variables (l'interprétation d'une matrice comme fonction de deux variables étant un sous-produit) se résoud classiquement en découpant la surface par des plans parallèles et en étudiant la scène plan par plan en construisant une ligne de crête. Les algorithmes les plus utilisés sont celui de WILLIAMSON [Wil 72] qui utilise une résolution dans l'espace utilisateur et celui de WRIGHT [Wri 72] qui utilise une résolution dans l'espace écran. Nous voudrions noter ici la similitude entre la solution de WATKINS et celles-ci, puisque dans tous les cas la scène à étudier est découpée par des plans. L'algorithme de WILLIAMSON a une structure proche de la première solution que nous avons suggérée (fusion de deux listes de segments), alors que l'algorithme de WRIGHT a une structure proche de la de xième solution que nous avons suggérée (projection des segments et tenue d'un tableau des hauteurs). Ces algorithmes sont donc en.fait des algorithmes de WATKINS adaptés au traitement particulier des fonctions de deux variables.

Nous noterons pour conclure que le programme graphique n'utilise pas ici de logiciel de description, puisque le résultat même des calculs est une scène à trois dimensions parfaitement décrite. Dans ce cas particulier, la partie graphique se compose donc d'un logiciel de préparation à la visualisation (projection et présentation de surfaces) et du logiciel graphique de base.

4.22 Construction de quasi-hélices par itération de deltaèdres

Cette étude, conduite dans le cadre de recherches morphologiques ([OdA 76], [Odi 75]), vise à trouver des structures géométriques possèdant les caractéristiques suivantes:

- structures hélicoïdales présentant une certaine stabilité, donc des critères mécaniques de rigidité,

- utilisation de volumes élémentaires permettant d'engendrer par une ou plusieurs itérations plusieurs hélicoïdes.

Ces structures sont étudiées car elles semblent présentes dans beaucoup de constructions biologiques (l'exemple le plus connu est celui de la double hélice de l'ADN).

Notre étude a porté sur l'utilisation d'isoèdres congruents du type bipyramide à base pentagonale régulière, entièrement décrits par la donnée de la longueur de l'arête. Au départ, on dispose simplement de la description d'une bipyramide. L'opérateur donne le numéro d'une face, numéro compris entre 2 et 10. Par convention, cette face sera accolée à la face numéro 1 de la bipyramide initiale. Les deux faces ainsi sélectionnées possèdent chacune un sommet qui est commun à cinq faces distinctes (sommet de la bipyramide) et deux sommets appartenant au contour pentagonal. Le collage se fait en faisant correspondre les sommets des deux faces de telle sorte qu'un sommet de chaque bipyramide corresponde à un point de contour de l'autre (on ne colle pas les sommets ensemble).

De manière générale, pour accoler un solide S à un solide R fixé, il suffit d'amener trois points (A,B,C) non alignés de S en coïncidence avec trois points (A',B',C') de R (voir figure 4.5).

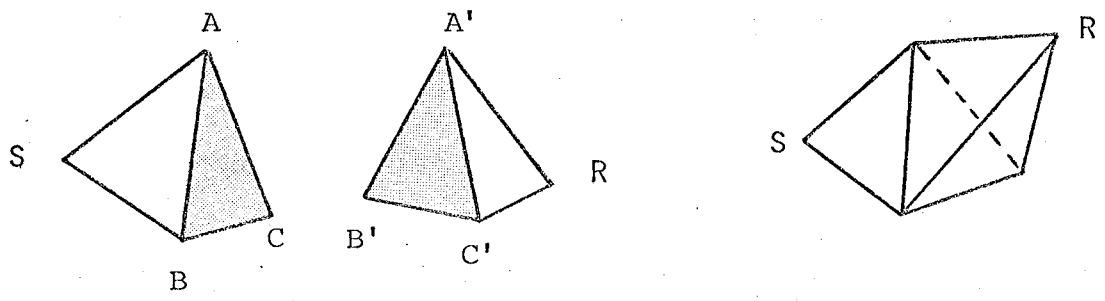


figure 4.5 Collage de deux solides

Il faut donc trouver un déplacement δ tel que

$$\delta(A) = A' \quad \delta(B) = B' \quad \delta(C) = C'$$

sachant que les triangles ABC et A'B'C' sont égaux dans notre cas.

Pour celà, on décompose δ en une rotation ρ (centrée à l'origine) suivie d'une translation τ ($\delta = \tau \circ \rho$).

De manière analytique, l'image M' d'un point M est donnée par

$$M' = \Omega M + T$$

où Ω est la matrice relative à ρ et T le vecteur relatif à τ.

Sachant que
$$A' = \Omega A + T$$
$$B' = \Omega B + T$$
$$C' = \Omega C + T$$

on déduit
$$U' = \Omega U$$
$$V' = \Omega V$$
$$W' = \Omega W$$

avec $U = B - A$ $\quad V = C - A$ $\quad W = U \wedge V$

$U' = B' - A'$ $\quad V' = C' - A'$ $\quad W' = U' \wedge V'$

Si l'on note Θ la matrice (U,V,W) et Θ' la matrice (U',V',W')
alors $\Theta' = \Omega\Theta$

ce qui permet d'obtenir $\Omega = \Theta' \Theta^{-1}$
et par suite $T = A' - \Omega A$

remarque : Ω est traitée a priori comme la matrice d'une application affine, mais est en définitive une matrice orthogonale du fait de l'hypothèse d'égalité des triangles ABC et A'B'C'.

La structure de données utilisée comporte les éléments suivants:

- pyramide de référence	*X, Y, Z*	coordonnées des sommets de la dernière pyramide générée
	FACES	description des faces (10)
	ARETES	description des arêtes (15)
	CORSOM	correspondance entre les sommets de la pyramide initiale et ceux de la pyramide courante
- description de l'hélice	*XHELICE, YHELICE, ZHELICE*	coordonnées des sommets de l'hélice ($4 \times NBBIP + 3$)
	AHELICE	arêtes de l'hélice ($12 \times NBBIP + 3$
	FHELICE	faces de l'hélice ($8 \times NBBIP + 2$
	NBBIP	nombre de pyramides de l'hélice.

L'addition d'une bipyramide se fait grâce à la suite d'opérations suivante:

- calcul de la matrice de transformation,
- transformation des coordonnées des sommets de la pyramide de référence,
- recopie des coordonnées des quatre nouveaux sommets et mise à jour du tableau de correspondance des indices,
- recopie des faces. La face 1 est ignorée, la face 2 remplace la face sur la quelle on a collé la nouvelle bipyramide, les autres faces sont recopiée à la suite,
- on recopie les arêtes, sauf celles qui correspondent à la face 1. Les indices recopiés sont mis à jour par l'intermédiaire du tableau des correspondances.

On peut montrer qu'il existe seulement quatre séries d'hélices (à un sens de rotation près) :

- en laissant une seule face libre à chaque pas, la structure hélicoïdale n'est ébauchée que jusqu'à 5 pyramides, au delà il y a recouvrement et donc impossibilité de poursuivre (voir figure 4.6).

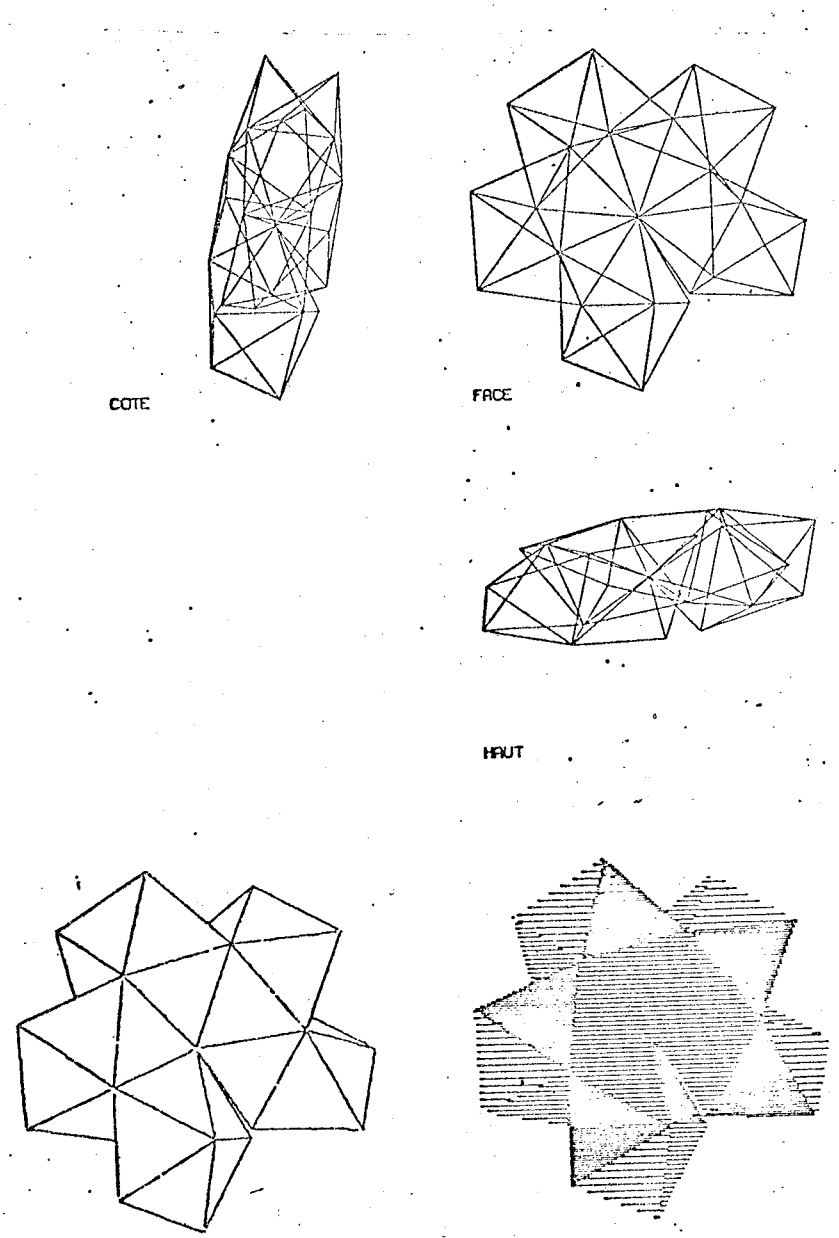


figure 4.6 Structure ne conduisant pas à une quasi-hélice

- en laissant deux faces libres, un isoèdre quasi-hélicoïdal de pas 10 peut être construit (voir figure 4.7).

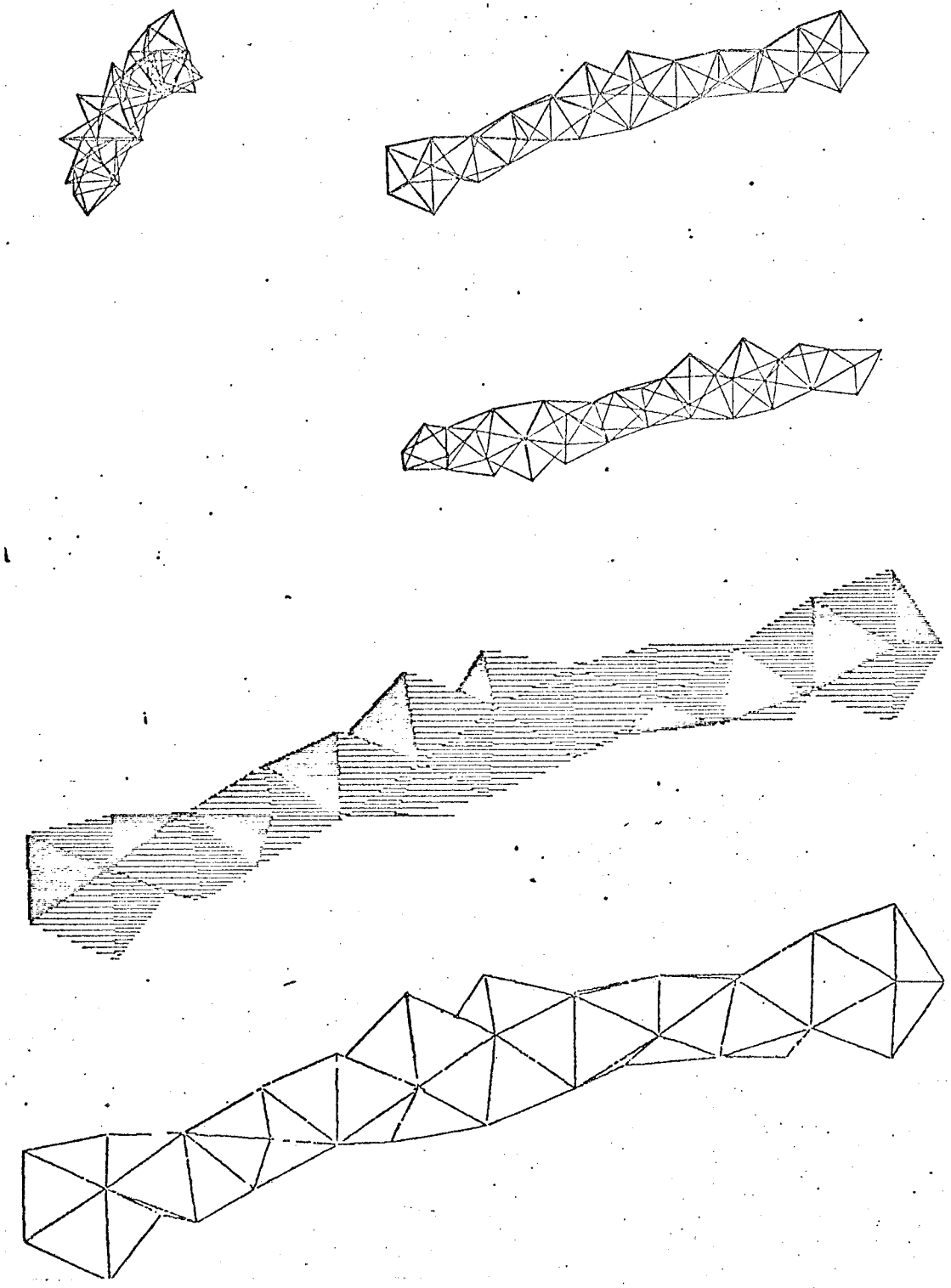


figure 4.7 Quasi-hélice de pas 10

- en laissant trois faces libres, un isoèdre quasi-hélicoïdal de pas 4 peut être construit (voir figure 4.8).

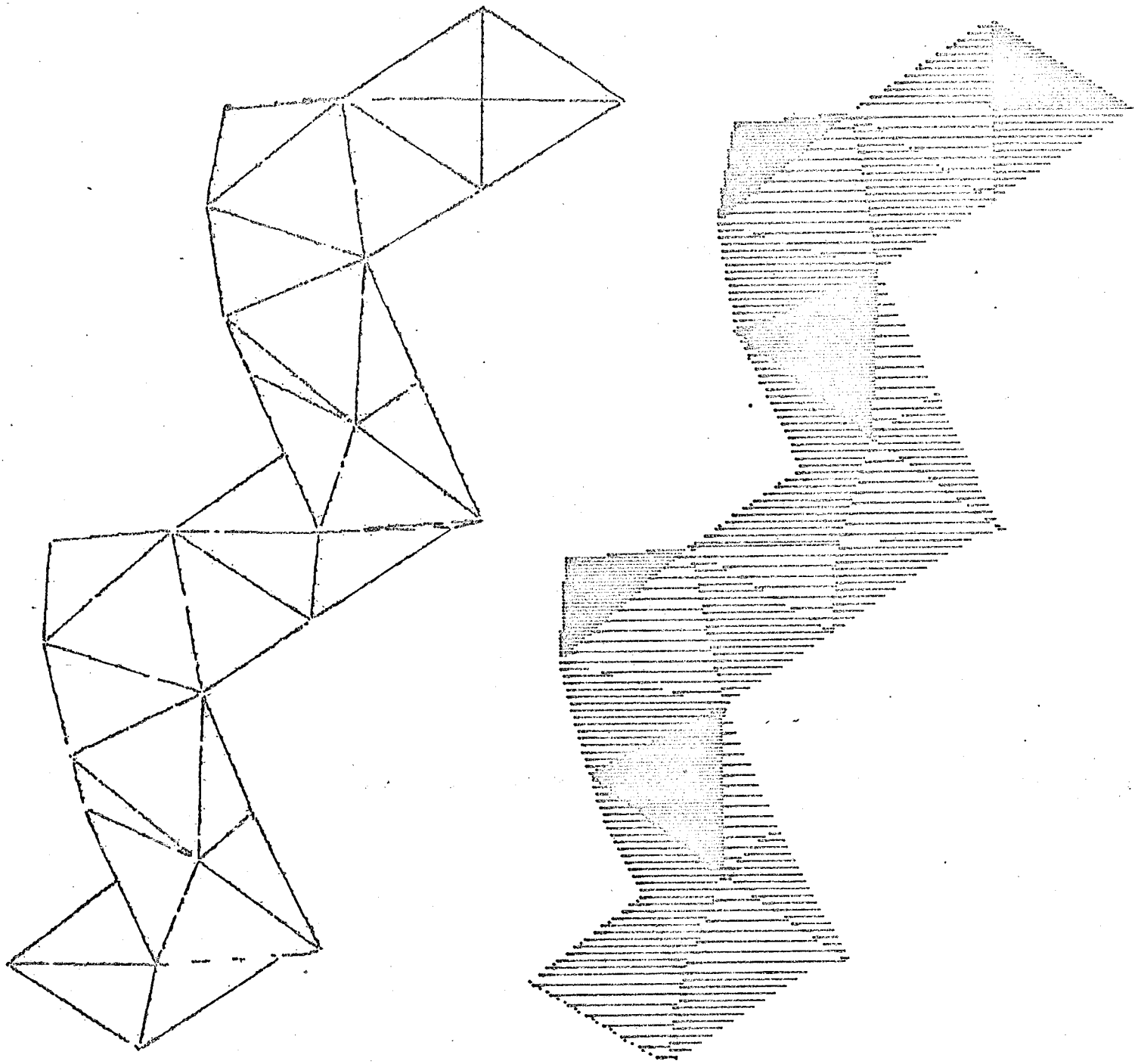


figure 4.8 quasi-hélice de pas 4 (deux périodes)

- en laissant quatre faces libres, un isoèdre quasi-hélicoïdal de pas 9 peut être construit (voir figure 4.9).

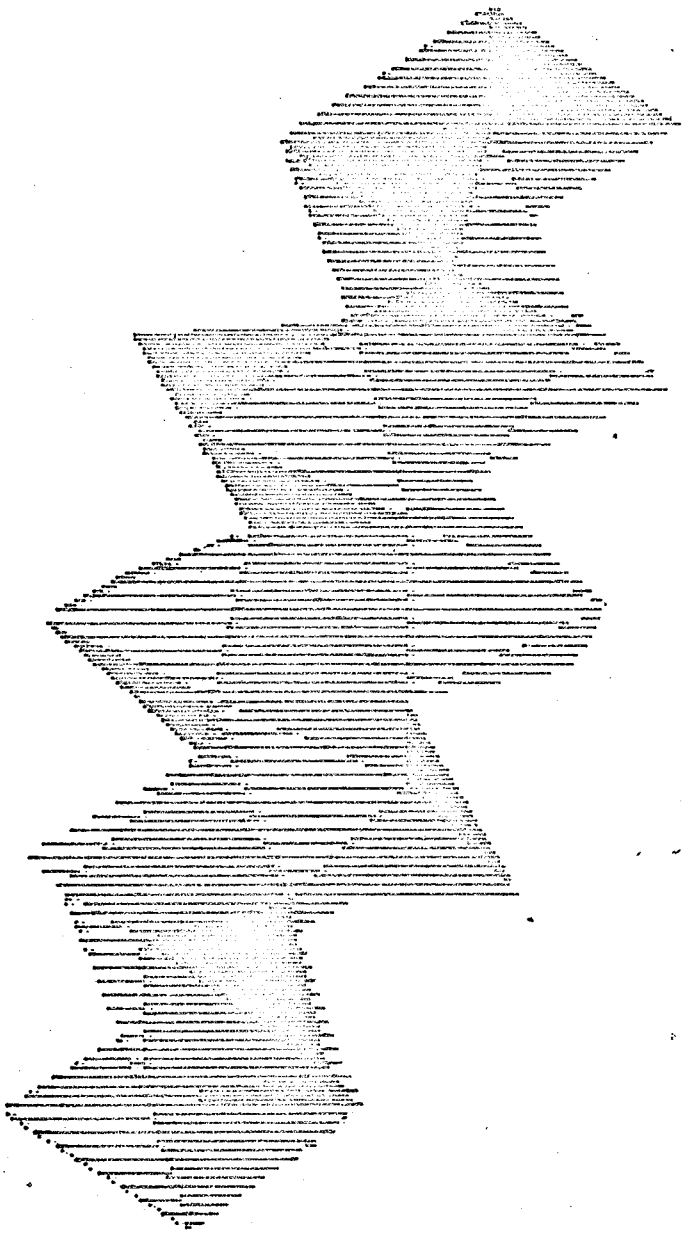


figure 4.9 Quasi-hélice de pas 9

Les dessins des figures précédentes ont été obtenus en utilisant VISU3D (voir le chapitre 3). Nous ferons deux remarques:

- les couleurs ont été attribuées de façon arbitraire à chacune des faces, et ne tiennent pas compte d'un ombrage possible,

- l'application proposée est un exemple typique de polyèdres engendrés par calcul où l'on peut choisir une orientation des faces correspondant au critère exigé par l'algorithme de GALIMBERTI et MONTANARI. Ceci nous permet en particulier d'éliminer les faces orientées vers l'arrière avant d'entame le calcul des parties cachées par un algorithme du type WARNOCK ou WATKINS.

Une extension de cette étude est en cours de réalisation, avec pour polyèdres de base des icosaèdres. Nous voudrions souligner ici l'intérêt d'un modèle sur calculateur par rapport à un modèle fait à la main (en papie par exemple). La rapidité de calcul permet d'étudier en un temps très bref différentes possibilités de construction, encourageant l'expérimentateur à tenter des combinaisons très variées et complexes. De plus, les différents types de présentation utilisés permettent de mieux saisir la nature de l'ensemble. Par exemple, la présentation sans élimination de parties cachées permet de voir des "mouvements" de lignes insaisissables sur un modèle en papier (voir par exemple la figure 4.7).

Pour conclure, nous noterons la simplicité du logiciel de description, constitué de la description formelle du polyèdre élémentaire, de la description de la scène et des algorithmes de calcul de la transformation et de mise à jour de la structure de données.

4.23 Construction de scènes à partir d'ordres donnés en français

Cette étude, conduite en collaboration avec l'équipe de Traduction Automatique du GETA et l'équipe d'Intelligence Artificielle de Marseille Luminy visait à réaliser une expérience de "robotique" pour construire des scènes tridimensionnelles par étapes successives. Les commandes sont données en français, en utilisant un vocabulaire limité et une structure syntaxique suffisamment riche pour décrire des scènes relativement complexes [Laf 76] .

Les scènes sont composées à l'aide de corps élémentaires:

- des prismes à base rectangulaire,

- des pyramides à base rectangulaire,

- des parallèlépipèdes rectangles (dénommés blocs par la suite).

Les bases sont horizontales, à bords parallèles aux axes Ox et Oy. Ainsi, pour définir entièrement un objet de base, il suffit de donner ses dimensions extérieures (longueur, largeur et hauteur) ainsi que les coordonnées d'un point servant d'origine.

Une scène est décrite par une suite de phrases, dont nous donnons un exemple ci-dessous:

Créez un bloc B plus large que long,
Créez un prisme aussi long et large que B, sur le bloc,
Créez un cube C et un cube P aussi haut que C,
Augmentez la hauteur de C et transformez P en pyramide,
Mettez C à gauche de B, contre lui, les faces avant de C et B étant alignées,
Posez la pyramide sur le bloc le plus haut.

Cette suite est analysée grâce au système CETA [Cha 74] , puis traduite en ordres de description. Pour ce faire, chacune des phrases est transformée en une expression logique faisant apparaître une conjonction de

tous les prédicats élémentaires avec leurs arguments. Cette première phase se décompose en deux temps :

- une phase morphologique, visant à retrouver la structure des mots et qui produit une première représentation dont nous donnons un exemple figure 4.10

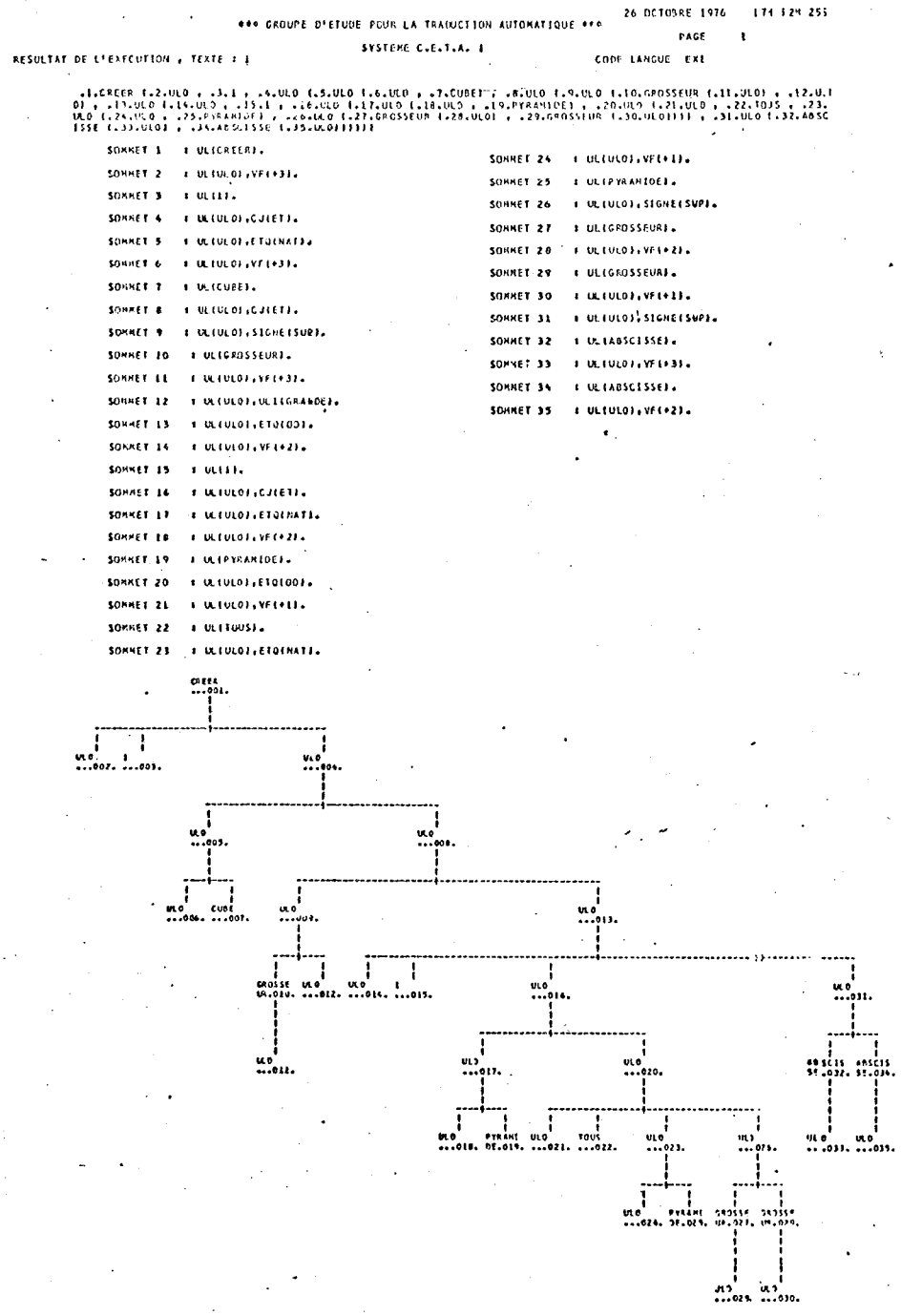


figure 4.10 Résultat de l'analyse morphologique (d'après [Laf 76])

- analyse de la structure morphologique pour calculer les propriétés syntaxiques et sémantiques de la phrase. Le résultat de cette analyse est ce que l'on nomme la structure profonde de la phrase. Un exemple de structure profonde est donné figure 4.11.

```
                                                                    26 OCTOBRE 1976    17H 12M 25S
                    *** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***                PAGE     1
                                        SYSTEME A.T.E.F.
RESULTAT DE L'EXECUTION , TEXTE 1 1                                  CODE LANGUE  EXI

 .1.ULTXT (.2.ULFRA (.3.ULOCC (.4.CREER) , .5.ULOCC (.6.UN) , .7.ULOCC (.8.GROSSEUR) , .9.ULOCC (.10.CUBE) , .
11.ULOCC (.12.ABSCISSE) , .13.ULOCC (.14.LA , .15.LA) , .16.ULOCC (.17.PLUS) , .18.ULOCC (.19.GROSSEUR) , .20.
ULOCC (.21.ULNCP (.22.PYRAMIDE , .23..)))

     SOMMET 1   : UL(ULTXT).

     SOMMET 2   : UL(ULFRA).

     SOMMET 3   : UL(ULOCC).

     SOMMET 4  CREER: UL(CREER),SEG(B),CVI(IOPR1,IOPR3,IOPR4,(NPR1,(NP33),NBR(PLU),CAT(VB),MODE(IOPR,(NPR1,PRE
         (2),ARG2(N),AUX(A).

     SOMMET 5   : UL(ULOCC).

     SOMMET 6  UN: UL(UN),SEG(INVA),GNR(MAS),NBR(SIN),CAT(DET),SOUCAT(ARTI).

     SOMMET 7   : UL(ULOCC).

     SOMMET 8  GROS: UL(GROSSEUR),SEG(B),GNR(MAS),NBR(SIN,PLU),CAT(ADJ),SEM(TAILLE),UL1(GRANDE),SIGNE(SUP).

     SOMMET 9   : UL(ULOCC).

     SOMMET 10  CUBE: UL(CUBE),SEG(B),GNR(MAS),NBR(SIN),CAT(NMC),OBB(1).

     SOMMET 11   : UL(ULOCC).

     SOMMET 12  ABS: UL(ABSCISSE),SEG(INVA),CAT(PRE),SEM(PLACE),UL1(GRANDE),SIGNE(SUP).

     SOMMET 13   : UL(ULOCC).

     SOMMET 14  LA: UL(LA),SEG(INVA),GNR(FEM),NBR(SIN),CAT(PRP),PRS(3),MODE(OBJ).

     SOMMET 15  LA: UL(LA),SEG(INVA),GNR(FEM),NBR(SIN),CAT(DET),SOUCAT(ARTD).

     SOMMET 16   : UL(ULOCC).

     SOMMET 17  PLUS: UL(PLUS),SEG(INVA),CAT(ADV),OPI(ADD,PROD),ARG(AJC),SIGNE(SUP).

     SOMMET 18   : UL(ULOCC).

     SOMMET 19  GROSSE: UL(GROSSEUR),SEG(B),GNR(FEM),NBR(SIN),CAT(ADJ),SEM(TAILLE),UL1(GRANDE),SIGNE(SUP).

     SOMMET 20   : UL(ULOCC).

     SOMMET 21   : UL(ULNCP).

     SOMMET 22  PYRAMIDE.: UL(PYRAMIDE),SEG(B),GNR(FEM),NBR(SIN),CAT(NMC),OBB(1).

     SOMMET 23  PYRAMIDE.: UL(.),CAT(PCT),CJ(ET),NC(1).
```

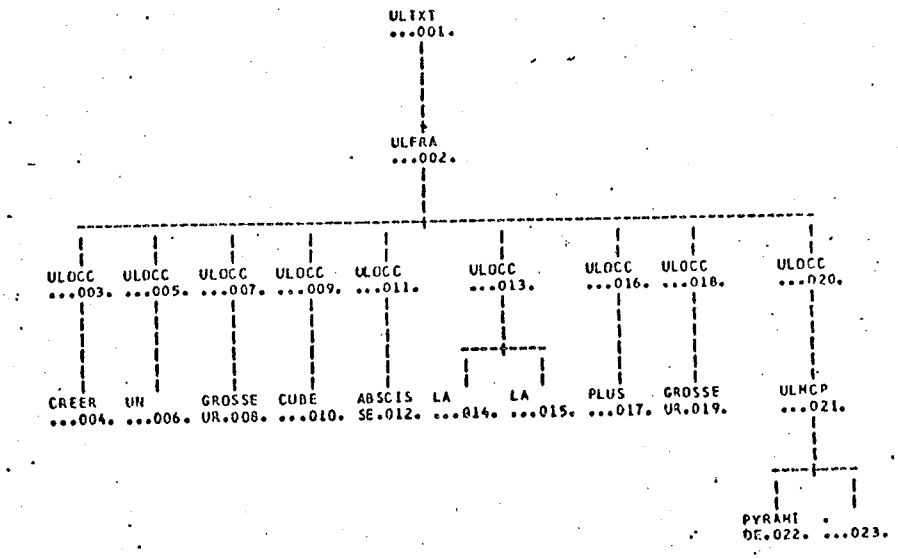


figure 4.11 Exemple de structure profonde (d'après [Laf 76])

Le résultat de cette première phase est alors analysé par le système PROLOG [Col 73] qui permet de calculer l'enchaînement des programmes graphiques. On obtient une suite de commandes s'adressant à un logiciel de description graphique. Les commandes élémentaires sont [Lal -] :

- *"C" nom type coordonnées de l'origine longueur largeur hauteur*
  création d'un volume élémentaire. Le *nom* permet de le repérer dans la scène, le *type* définit quel objet doit être utilisé.

- *"S" nom*
  destruction de l'objet

- *"I"*
  visualisation de la scène courante

- *"M" "chaîne de caractères"*
  affichage d'un message permettant de repérer la phase de construction

- *"F"*
  fin d'enchaînement des ordres

Un exemple typique de suite d'ordres est donné ci-dessous:

```
"C"  1  "BLOC"       500  500  500  100  100  100
"C"  2  "BLOC"       500  500  500  100  100  100
"C"  3  "BLOC"       500  500  500  400  100  100
"S"  1
"I"
"C"  4  "BLOC"       500  500  600  100  100  300
"S"  2
"I"
"C"  5  "BLOC"       650  400  600  100  100  300
"S"  4
"I"
"C"  6  "BLOC"       640  425  600  100   50  300
"S"  5
"I"
"C"  7  "PYRAMIDE"   650  425  825  100   50  150
"C"  8  "PRISME"     500  500  600  400  100  100
"C"  9  "BLOC"       650  575  600  100   50  300
"C" 10  "PYRAMIDE"   650  575  825  100   50  150
"C" 11  "PYRAMIDE"   650  425  875  100   50  250
"I"
"S"  7
"I"
"C" 12  "BLOC"       650  575  650  100   50  400
"S"  9
"I"
"C" 13  "BLOC"       650  575  650  100  100  400
"S" 12
"I"
"C" 14  "PYRAMIDE"   650  575  825  100  100  150
"S" 10
"I"
"C" 15  "PYRAMIDE"   650  575  875  100  100   50
"C" 16  "PYRAMIDE"   650  575  875  100  100  150
"S" 14
"I"
"F"
```

La structure de données du logiciel de description est constituée des éléments suivants (voir figure 4.12) :

- objets de référence	*DBLOC*, *DPRIS*, *DPYRAM*	faces des solides élémentaires
- description de la scène	*X*, *Y*, *Z*	sommets des polyèdres
	FACES	faces des polyèdres
	NUFIG	corrélation avec le nom de chaque objet
	TYPE	type de l'objet créé.

Les tableaux *NUFIG* et *TYPE* permettent de retrouver les faces associées à un objet dont le *nom* est donné.

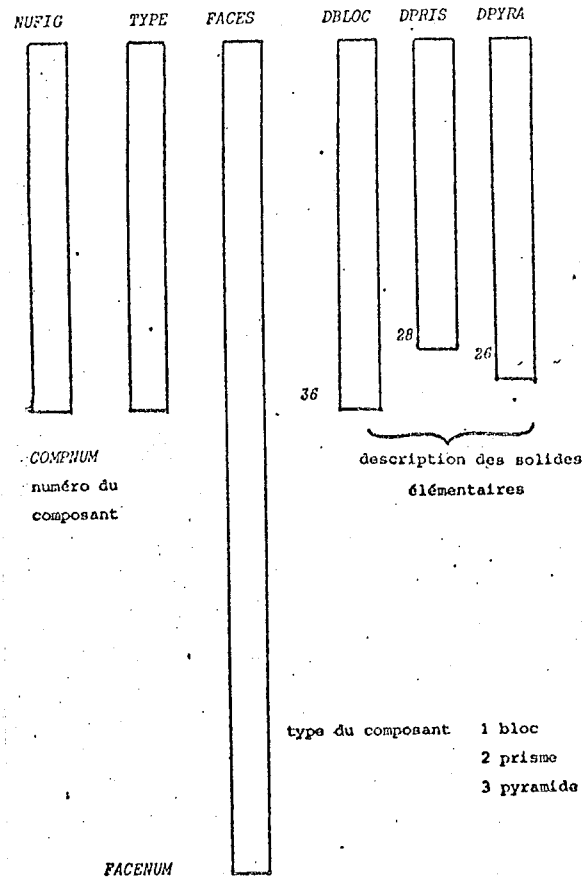


figure 4.12 Structure de données du logiciel de description

Les opérations effectuées sur cette structure sont les suivantes:

- création: calcul des coordonnées du volume élémentaire en fonction du type, addition de ces coordonnées dans les tableaux *X*, *Y*, *Z* , addition de la description des faces dans *FACES*, en mettant à jour les indices de somme mise à jour de *NUFIG* et *TYPE*.

- suppression: recherche à l'aide de *NUFIG* et de *TYPE* de l'emplacement des sommets et des faces associées au *nom* passé en paramètre. Suppression par retassement des tableaux concernés.

Nous concluerons la description de cette application par les deux remarques suivantes:

- on peut constater la complexité de la représentation de la scène, puisqu'en fait cinq codes différents sont utilisés.

- la description graphique elle-même présente une structure très classique du point de vue des codes géométriques. On dispose d'un ensemble de primitives qui décrivent formellement les objets élémentaires. La description d'une scène consiste alors à utiliser ces primitives avec des paramètres appropriés, ce qui conduit à l'obtention de la scène réelle. Il est clair que, comme dans le cas des logiciels graphiques interactifs de base, le choix des primitives est très important. On trouvera en général deux types de primitives:
  . des primitives décrivant des volumes élémentaires,
  . des primitives permettant de faire leur assemblage (transformations géométriques par exemple).

Ces logiciels sont en fait très dépendants des applications les utilisant. On rapprochera la structure utilisée de celle exposée en 4.22.

La figure 4.13 présente plusieurs vues de la scène finale décrite par la suite de commandes donnée page 206.

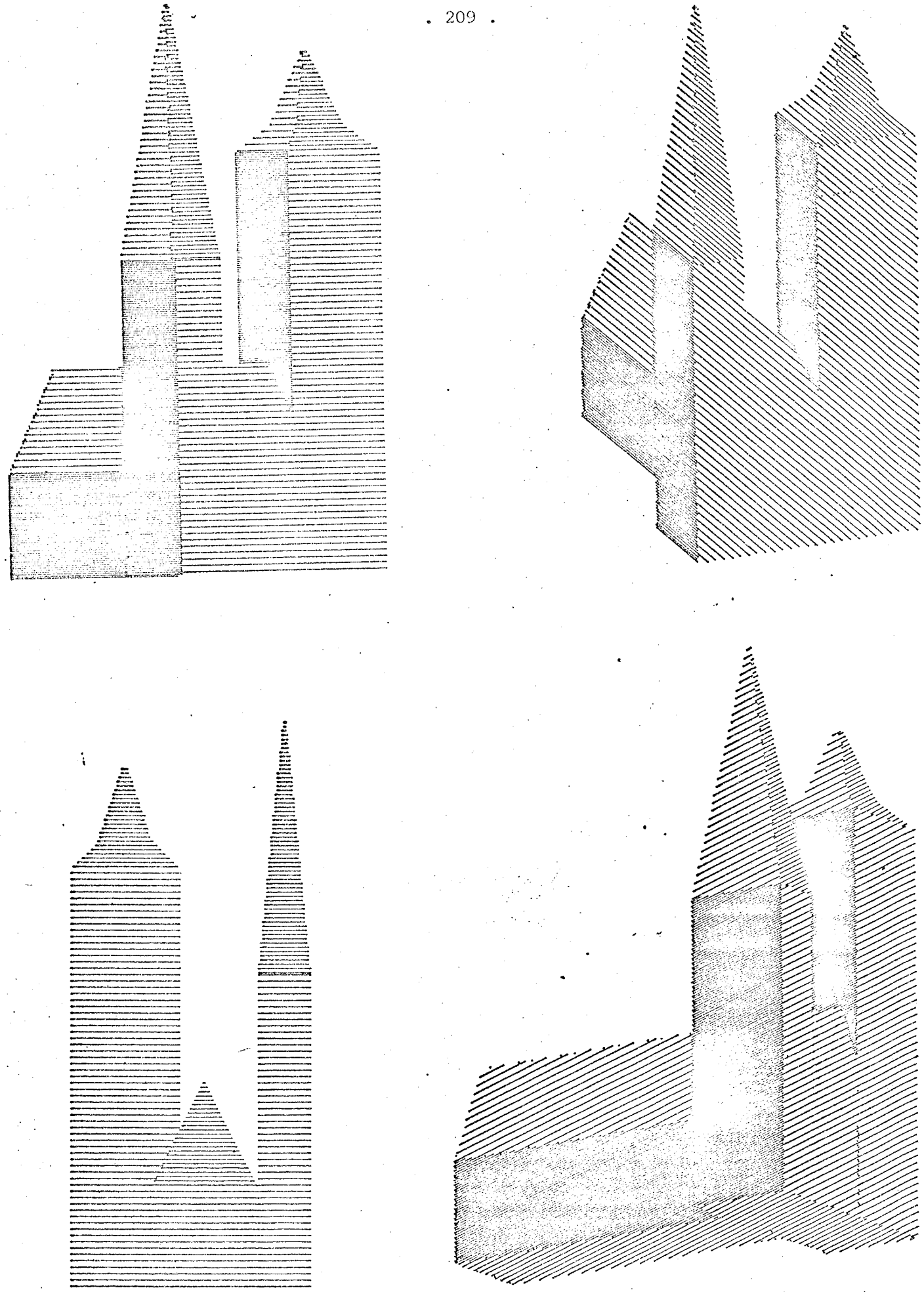


figure 4.13 : Exemple de scène sous divers angles

4.3 APPLICATIONS DE CONCEPTION ASSISTEE PAR ORDINATEUR

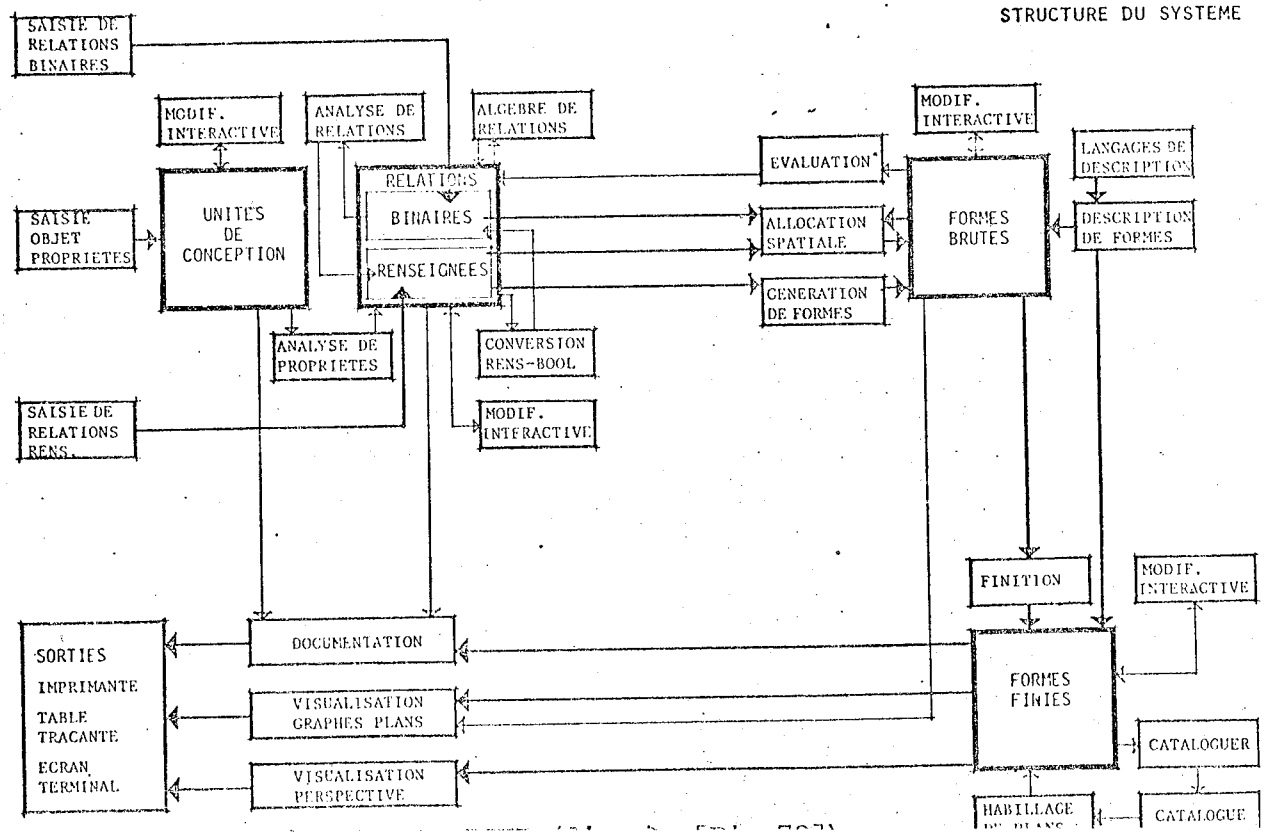
4.31 Aide à la conception architecturale : le système SIGMA-ARCHI

Le système SIGMA-ARCHI est un Système Interactif Graphique pour Méthodes d'Aides à la conception ARCHItecturale. Développé par l'équipe de Méthodologie de la Conception Assistée par Ordinateur, il constitue une application du système SIGMA-CAO pour la construction de système intégrés [Dav 75] . Le domaine de l'architecture a été choisi à cause de la diversité des problèmes à traiter. Les concepts fondamentaux [Riv 77] de ce système sont les suivants:

- une base de données du problème comportant, au niveau le plus élevé, les renseignements concernant les quatre types d'informations nécessaires à la représentation de problèmes de conception architecturale à un seul niveau: unités de conception, relations, formes brutes et formes finies.

- un réseau de traitements permettant de faire progresser le projet selon les différentes directions du domaine de conception.

Le schéma général du système est donné figure 4.14



L'aspect externe du système se compose, aux yeux de l'utilisateur, d'un ensemble de programmes couvrant une large partie des méthodes de conception:

- saisie de données à l'aide de programmes graphiques interactifs permettant de définir aussi bien les objets du problème (définitions purement graphiq que leurs propriétés et relations (définitions pouvant être verbales),

- manipulation de données, permettant de créer, combiner, pondérer des graph de relation,

- analyse de données, permettant de créer la hiérarchie des sous-problèmes afin de contrôler leur connectivité,

- génération de formes, permettant d'engendrer diverses configurations à partir de contraintes de surface et de proximité,

- allocation spatiale, permettant de rechercher des répartitions optimales en fonction de la complexité des objets et le type des contraintes qui leu sont appliquées,

- évaluation des configurations, donnant une représentation graphique de la satisfaction des contraintes,

- description de la configuration en deux ou trois dimensions,

- édition et habillage de plans par utilisation de catalogues de solutions techniques et d'équipements,

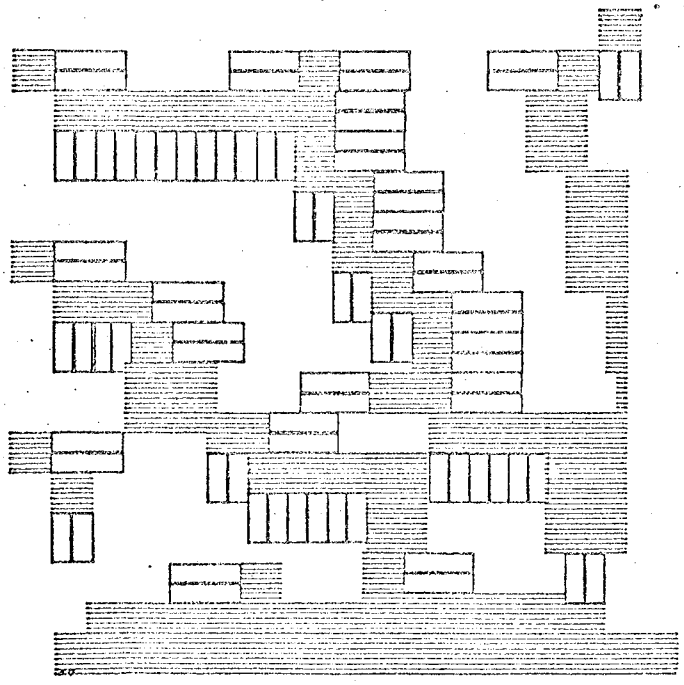
- évaluations techniques, donnant des calculs de surface et de volumes, des estimations de coût, des longueurs de circulation, etc ...

Nous n'insisterons que sur les aspects suivants:

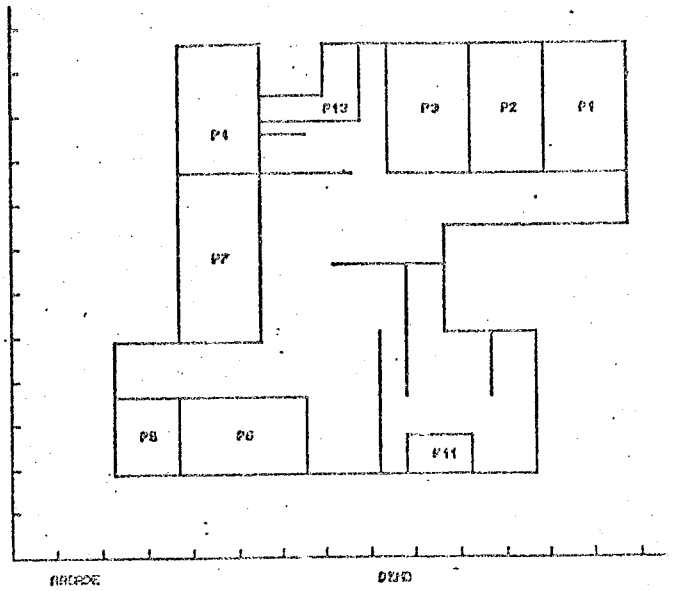
- présence des techniques graphiques à tous les niveaux de conception, conduisant à des représentations extrêmement diverses dont la figure 4.15: donne quelques exemples,

- utilisation de plusieurs logiciels de description spécialisés (plans, plan masses, éditions de graphes, etc ...) puisant leurs renseignements dans une base de données unique.

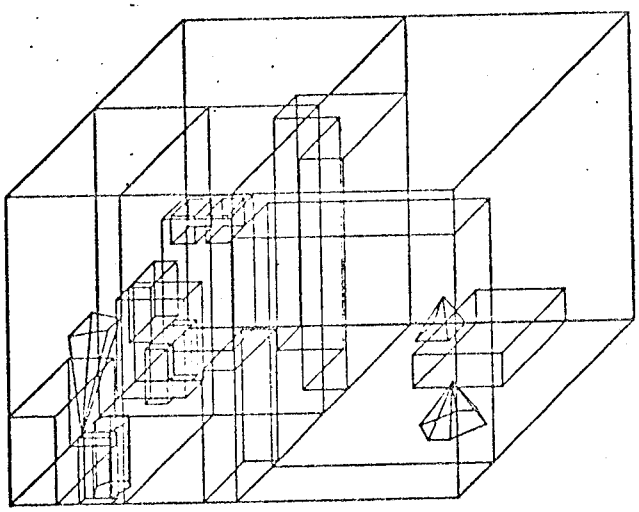
Cette structure est extrêmement représentative de programmes intégrés pour la Conception Assistée par Ordinateur.



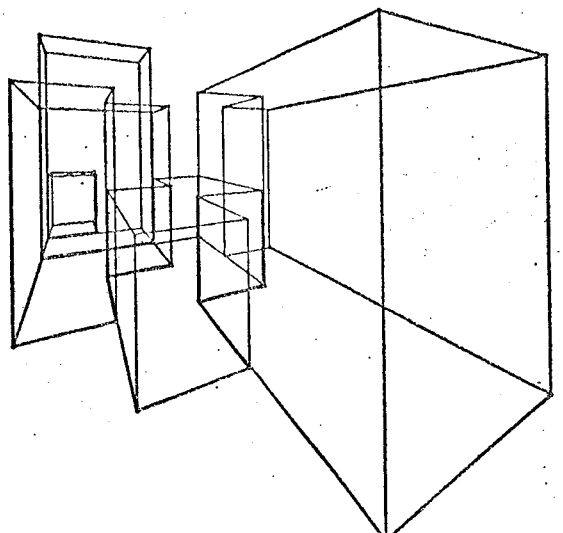
étude de répartition d'immeubles sur un site



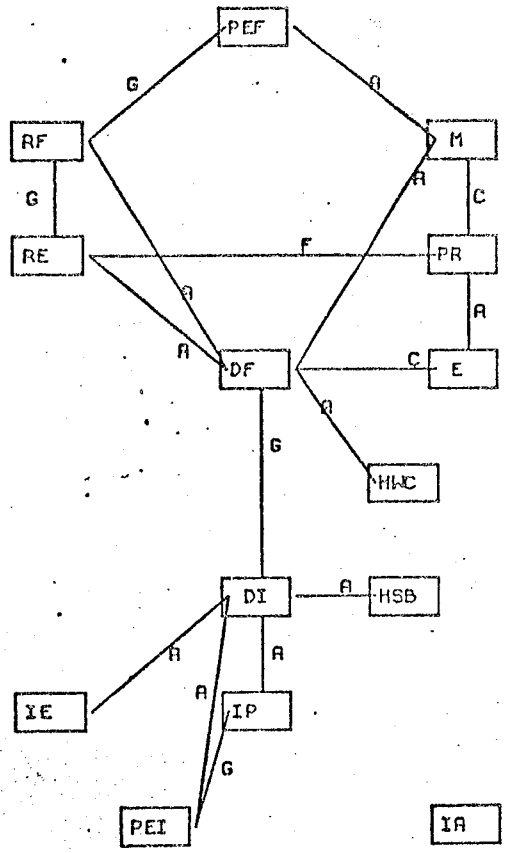
plan d'appartement



visualisation schématique d'une
cellule d'habitation



vue perspective d'un ensemble
d'immeubles



graphe de définition fonctionnelle
d'un appartement

figure 4. Présentations obtenues grâce à SIGMA-ARCHI

4.32 Aide à la conception de mécanismes

Nous ne pouvons décrire en détail l'application que nous avons traitée, pour des raisons de propriété industrielle. Nous soulignerons simplement la démarche suivie, qui est typique d'un grand nombre d'applications analogues. Le point clé est que l'opérateur souhaite obtenir, à partir d'un certain nombre de données initiales, différentes présentations correspondant à des étapes diverses de la conception. Le logiciel graphique doit donc permettre d'interpréter les données initiales pour fournir les dessins nécessaires. Nous donnerons deux exemples de présentation:

- une épure, qui permet de repérer les grands axes du mécanisme et de choisir (ou placer) certains points importants. La figure 4.15 donne un exemple d'épure après calcul. L'épure a été complétée par sélection d'une position par l'opérateur (point étiquetté C), position servant d'appui à deux pièces

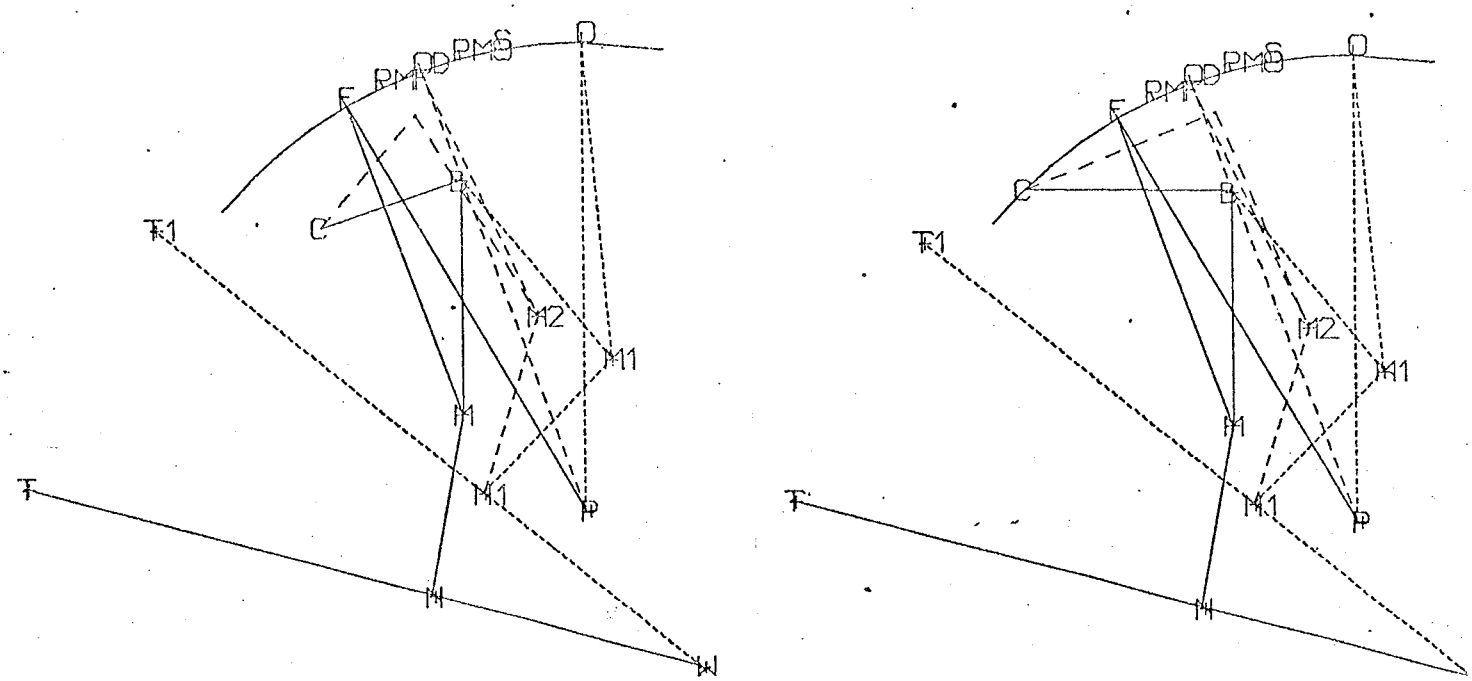


figure 4.16 Epure d'un mécanisme

Une fois l'épure réalisée, il faut vérifier que les pièces s'appuyant sur l axes choisis peuvent effectivement fonctionner, et qu'il n'y a pas (par exemple) de problèmes d'encombrement.

- un habillage du schéma précédent, qui permet de mettre en valeur les dispositions relatives des pièces du mécanisme, soit par simple délimitation du contour apparent en deux dimensions, soit en trois dimensions pour des pièces complexes. La figure 4.17 donne un exemple d'habillage en deux dimensions de l'épure précédente.

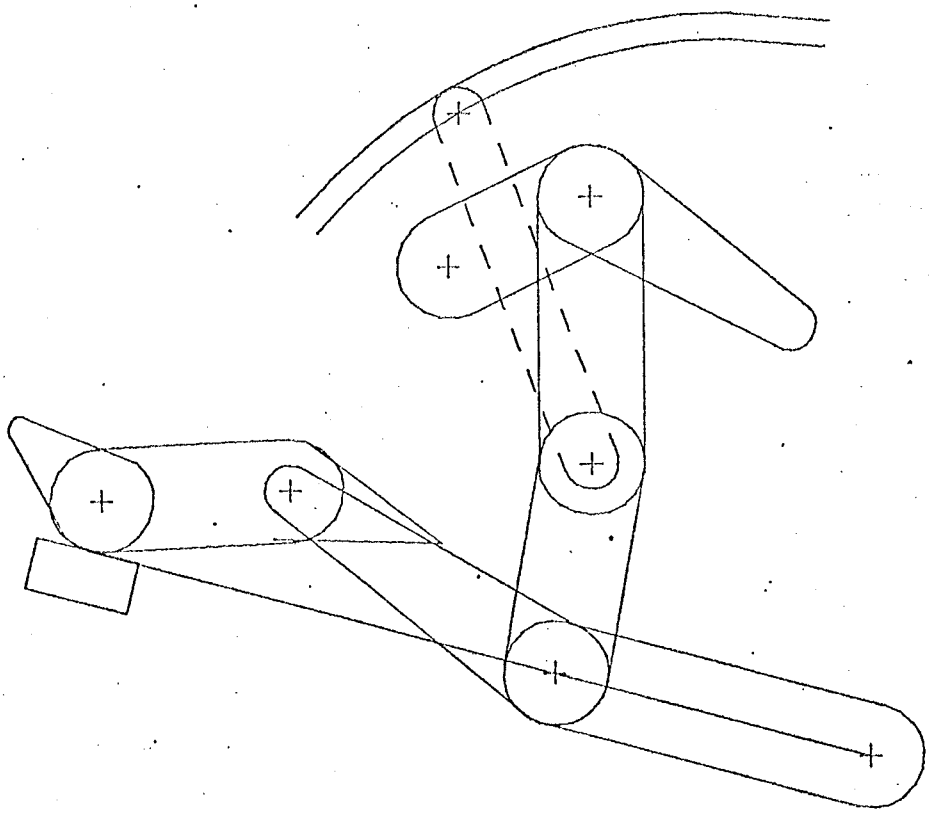


figure 4.17 Habillage d'une épure

Un des aspects les plus intéressants de ce type de présentation est de permettre de vérifier rapidement des séquences de fonctionnement d'un mécanisme, en visualisant les positions des pièces lorsque les valeurs des paramètres définissant le mouvement sont étudiées pas à pas.

Nous noterons donc de nouveau ce besoin de présenter diversement un même ensemble de données, impliquant que le logiciel de description soit à même de répondre à plusieurs fonctions.

4.33 Contrôle de programmes de calcul d'éléments finis

Les méthodes de calcul aux éléments finis se répandent de plus en plus à l'heure actuelle pour deux raisons:

- les méthodes numériques sont mieux connues et mieux adaptées aux problèmes traités,

- la visualisation graphique permet d'intervenir efficacement au niveau des algorithmes, en autorisant un contrôle immédiat des résultats du calcul.

Nous ne traiterons pas ici des algorithmes de calcul aux éléments finis, mais donnerons simplement quelques remarques sur l'apport de la visualisation

4.331 - Eléments finis bidimensionnels

Nous prendrons comme exemple, pour le calcul d'éléments finis bidimensionnels, le système DELTA ([LeP -], [Pon 74]), développé au sein de l'équipe d'analyse numérique de l'IMAG. Ce code permet, en mode graphique, d'assurer la résolution d'équations aux dérivées partielles puis de visualiser les résultats approchés. Trois étapes essentielles interviennent au cours du calcul:

- saisie du domaine, qui peut se faire à l'aide d'une tablette graphique,

- triangulation automatique du domaine. La complexité des domaines considéré justifie la visualisation des triangles au fur et à mesure de leur construction. L'opérateur peut en particulier intervenir pour corriger certaines triangulations, permettant une meilleure efficacité de l'algorithme. La figure 4.18 donne un exemple de triangulation d'un domaine représentant la Manche.

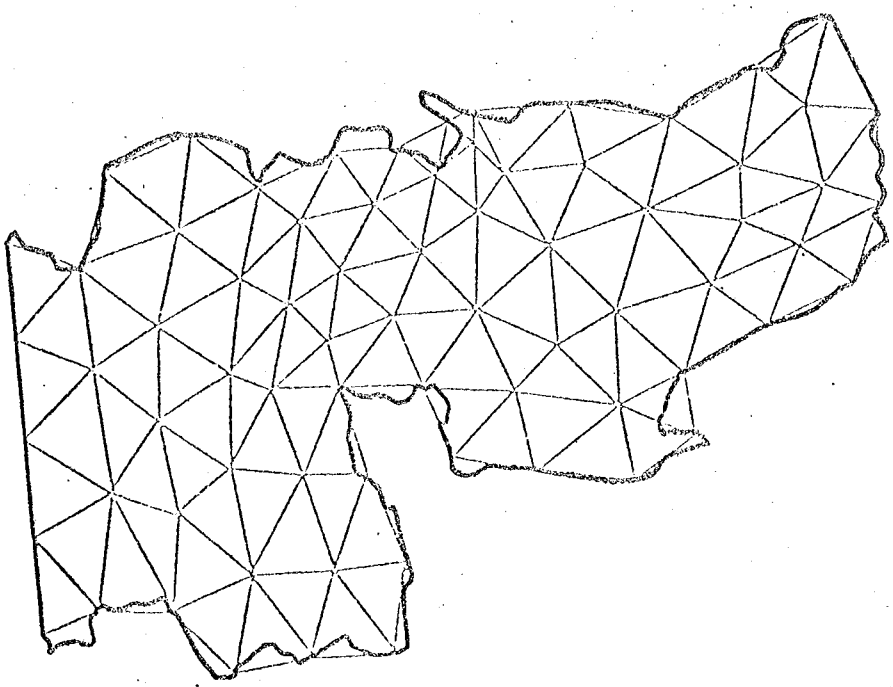


figure 4.18 Triangulation pour une étude de marée

- visualisation des résultats approchés. A partir des calculs précédents, plusieurs types de présentations peuvent être utilisés par l'opérateur:
  . présentation de courbes de niveau,
  . présentation de vecteurs dérivés, représentant des gradients, des contraintes de courants,
  . présentation tridimensionnelle, identique à celle évoquée en 4.21

La figure 4.19 présente la rose des courants aux centres de gravité des triangles de la figure précédente.

figure 4.19 Visualisation de la rose des courants

4.332 - Eléments finis tridimensionnels

En ce qui concerne les éléments finis tridimensionnels, nous avons adapté VISU3D à un programme écrit en collaboration avec l'équipe d'analyse numérique de l'IMAG et une équipe du Laboratoire d'Electrotechnique de l'INPG. Nous ferons les remarques suivantes:

- le résultat direct des calculs est une description de l'objet à étudier, sous forme de coordonnées de points dans l'espace et de solides de base servant au découpage (tétraèdres, cubes ou parallèlépipèdes rectangles par exemple). Le programme d'application se comporte donc comme un logiciel de description, et utilise directement les services du logiciel de préparation à la visualisation.

- la notion de découpage de la scène en sections (voir 3.1) s'est révélée très utile, permettant de lier certains ensembles de faces de manière à pouvoir les étudier séparément du reste de la scène. Par exemple, lors de l'utilisation de méthodes frontales, on lie les faces créées à chaque pas par un même numéro de section, ce qui permet ensuite d'étudier pas à pas la progression du front et de vérifier si aucune erreur ne s'est produite.

- la présentation avec élimination de parties cachées s'est avérée être indispensable, les scènes étant particulièrement complexes du point de vue du nombre de faces et du nombre d'arêtes. Deux remarques importantes peuven être faites:
  . Nous avons aménagé les algorithmes écrits pour pouvoir tenir compte de caractéristiques intéressantes, telles que : faces uniquement triangulair ou quadrilatèrales, faces ne se coupant jamais.
  . le programme de génération des éléments finis permet de tenir compte de renseignements importants, comme le fait de savoir si une face est sur le pourtour extérieur de l'objet à étudier, ou à l'intérieur. Cette connaissance permet d'éliminer a priori, lors de l'étude des parties cachées, toutes les faces internes.

La figure 4.20 donne un exemple de présentation d'un objet étudié dans le cadre de tests du logiciel . La coloration " en damier " a été choisie pour bien différencier la succession des faces. La structure de cette scène est donnée figure 3.4 sous la forme de plusieurs vues.

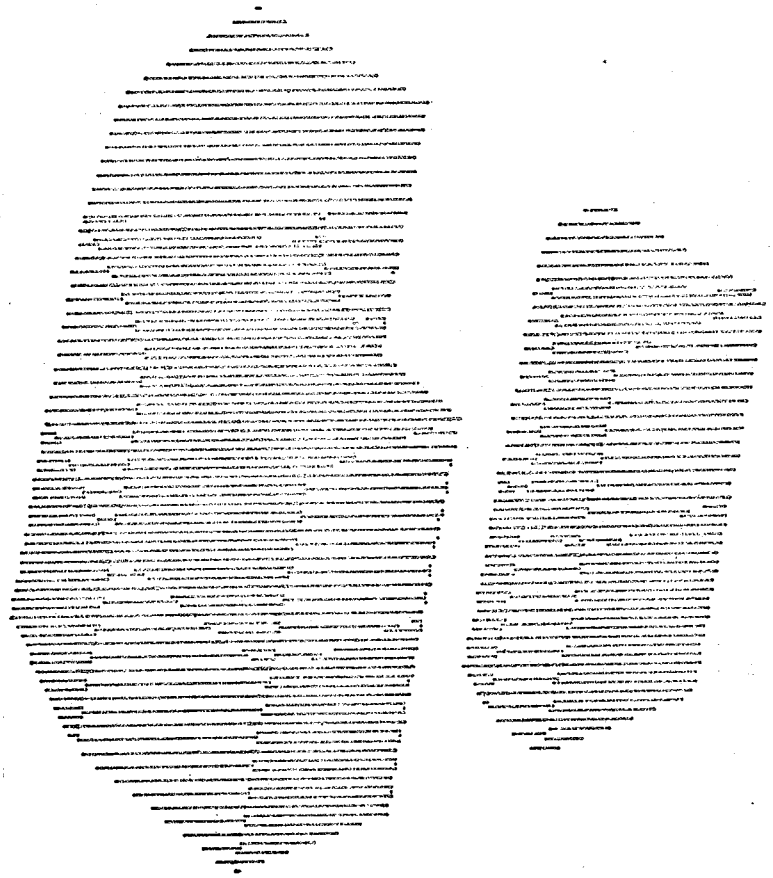


figure 4.20 Exemple de présentation d'un objet découpé

4.4 AIDE A LA CONCEPTION ET A LA REALISATION DE FILMS

4.41 Généralités

Les techniques informatiques ont été introduites dans de nombreux secteurs de l'industrie cinématographique, aussi bien pour faciliter le trava de gestion à différents niveaux (décors, prises de vues, montage, accessoire et matériels, élaboration des plans de travail, etc ...) que pour la réalisat des films eux-mêmes. De nombreux systèmes graphiques ont vu le jour, dont la production trouve des débouchés essentiellement dans les domaines de l'enseignement, la publicité et le divertissement.

Nous distinguons deux catégories de logiciels [Luc 75b] :

- les logiciels assurant l'animation d'objets dont la forme et l'évolution dans le temps sont entièrement commandés par le déroulement d'un programme écrit dans un langage de programmation quelconque et qui permet d'illustrer par exemple une méthode numérique, la simulation de mécanismes complexes, l'évolution d'un modèle représentant un phénomène physique, le traitement d'informations, etc... Nous parlerons dans ce cas d'*illustration de progran*

- les logiciels assurant l'animation d'objets dont la forme et l'évolution échappent à une représentation mathématique simple, mais que l'on peut décrire facilement à l'aide de dessins. Nous sommes alors dans le domaine du dessin animé proprement dit, qui se caractérise par la création de mouvements (déformations) imaginaires, par opposition aux présentations évoquées ci-dessus qui sont des présentations de la réalité. Nous parlerons de *dessin animé*.

Avant d'étudier ces deux types de logiciels, disons un mot du support du film. Nous distinguons trois types de réalisation:

- le support est un film, au sens ordinaire du mot. Le système graphique ne sert alors qu'à élaborer les images successives qui sont ensuite photographiées par une caméra ou obtenues directement sur film. Le temps de produc d'une image n'a pas d'importance, ce qui permet d'employer les calculs les plus complexes. Nous parlerons dans ce cas de systèmes d'animation en *différé*. Ces systèmes sont très courants et assurent des productions de qualité quasi professionnelle.

- le support est un écran de console de visualisation (à raffraîchissement ou à balayage télévision en général), permettant de visionner directement le film ou les séquences en cours de construction. Les calculs doivent être effectués suffisamment rapidement pour assurer une continuité satisfaisante pour l'oeil (24 images par seconde en général). Nous distinguons de nouvea deux types d'animation :
  . l'animation en *continu* , où les intervalles séparant deux images représen tent une unité de temps simulée, sans rapport avec l'écoulement réel,
  . l'animation en *temps réel* , où l'intervalle de temps séparant deux images représente le temps absolu qui s'est écoulé. Ce dernier type d'animation est surtout utilisé pour le contrôle de processus ou la simulation de phénomènes réels (apontage sur porte-avions, conduite automobile, etc...

Nous nous sommes intéressés essentiellement aux techniques d'anima tion en mode continu, estimant que leur intérêt était grand pour les deux typ de logiciels que nous avons définis:

- en ce qui concerne l'illustration de programmes, l'animation en continu permet de choisir les variations qui semblent intéressantes à un moment donné, par exemple dans un souci pédagogique, par opposition au film où tou est figé.

- en ce qui concerne le dessin animé, le contrôle sur l'écran permet de juger immédiatement de l'impact d'une séquence, et donc de modifier celle-ci sans attendre le montage de l'ensemble du film.

Nous étudierons successivement ces deux types de logiciels.

4.42 L'illustration de programmes

Disposant à un moment donné d'un programme dont on sait qu'il est au point, on souhaite, pour une raison ou une autre, obtenir une illustratior des différentes phases de son exécution. Lors de la mise au point de ce programme (ou simplement pour juger de ses résultats), un certain nombre d'instructions d'écriture ont été utilisées, caractérisées par les points suivants

- donnée d'une liste de variables dont on désire connaître la valeur,

- utilisation d'un mode d'impression du contenu de ces variables, par le biai d'instructions de mise en page implicites (ALGOL W) ou explicites (formats en FORTRAN).

Le programmeur utilise donc des primitives de haut niveau, laissan au système d'exploitation le soin de réaliser les opérations d'impression. Le schéma d'un programme est alors celui de la figure 4.21:

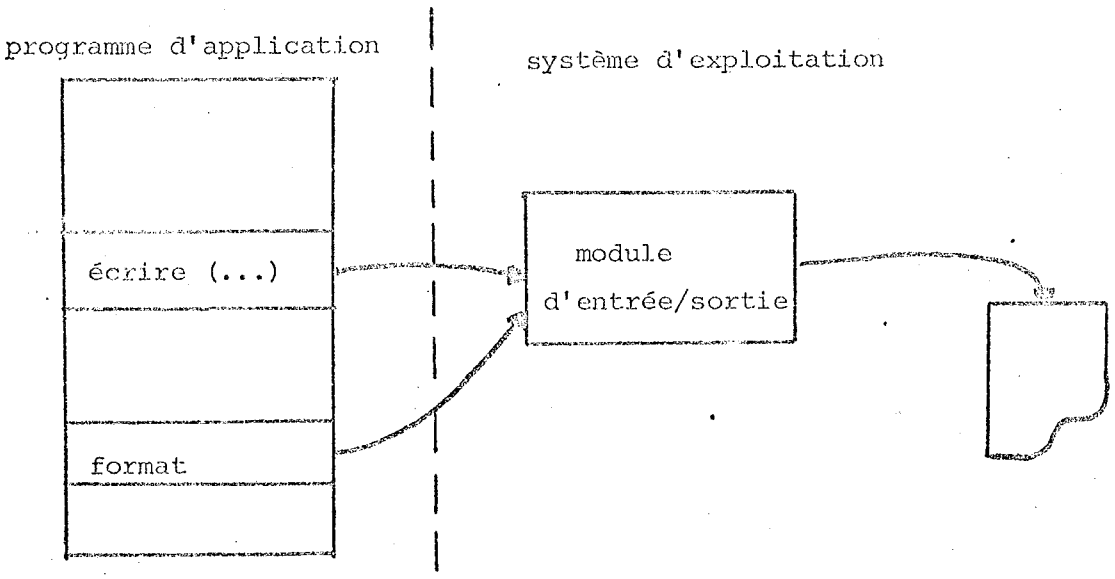


figure 4.21 Schéma des instructions d'impression

La technique classique pour illustrer graphiquement un tel progra consiste à le réécrire entièrement, en utilisant les primitives de logiciels graphiques de niveaux différents, dépendant de la complexité de l'illustratic à réaliser. Ce procédé présente un certain nombre d'inconvénients, dont nous citons les plus graves:

- mise en cause de la structure logique du programme par insertion des primi ves graphiques, ce qui conduit souvent à de graves erreurs de programmatio

- difficulté de réalisation souvent disproportionnée avec l'illustration souhaitée.

- toute modification de la représentation choisie entraîne la réécriture du programme complet.

- difficulté de réutiliser les mêmes représentations graphiques pour différen programmes, à moins de se forger une bibliothèque personnelle dont la gestic présente de nouveau des difficultés majeures sans rapport avec le profit attendu.

Ayant constaté en particulier que beaucoup d'utilisateurs désirent une illustration relativement standard du fonctionnement de leurs programmes et que, de plus, ils ne sont pas disposés à consacrer beaucoup de temps à la réécriture de leurs programmes, nous avons été amenés à proposer la définitio d'un logiciel d'illustration dynamique de programmes.

L'idée de base est de fournir à l'utilisateur les moyens d'illustr son programme en le modifiant le moins possible. Le programme initial fournit un certain nombre de résultats imprimés qui représentent en fait les scènes successives qui permettent de comprendre l'évolution d'un phénomène. Nous proposons un mécanisme permettant de reprendre ces valeurs numériques afin de les interpréter graphiquement. L'interprétation graphique sera confiée à un logiciel spécialisé, agissant sous le contrôle du programme d'application, ma extérieur à celui-ci. L'appel à ce système se fait par une instruction d'affi chage ayant la même structure qu'un ordre d'écriture classique, la différence essentielle étant que le descripteur d'image se trouve au niveau du logiciel graphique et non dans le programme. Le schéma de base d'un logiciel d'illustration dynamique de programmes est donné figure 4.22:

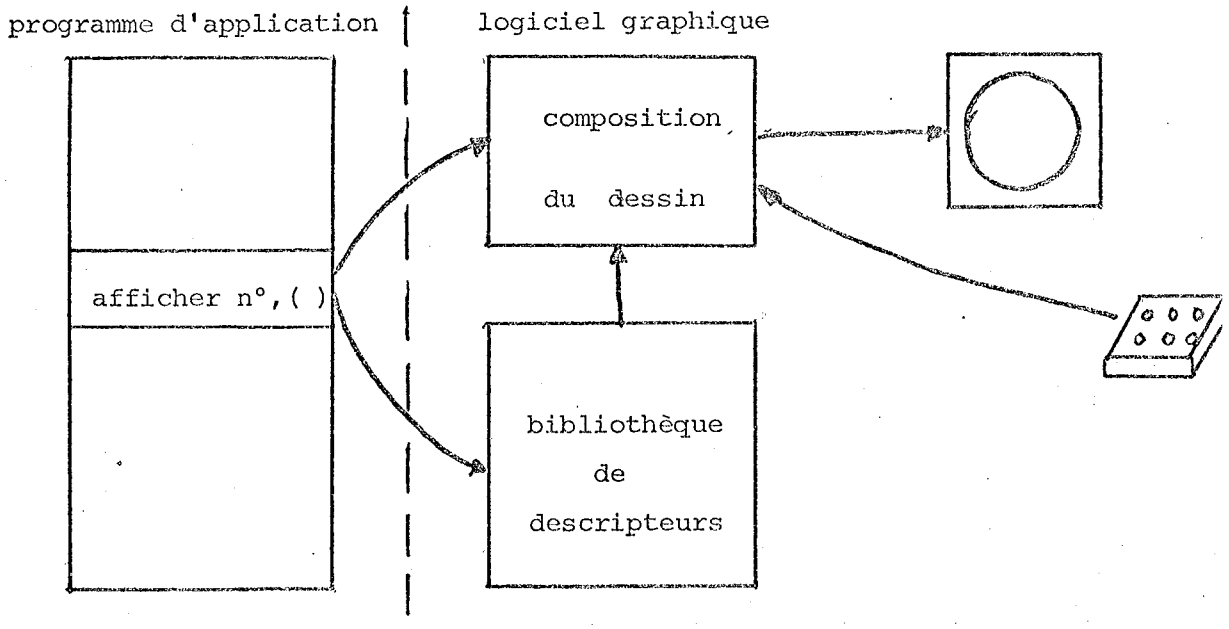


figure 4.22 Schéma d'un logiciel d'illustration dynamique de programmes

Pour illustrer un programme, l'utilisateur remplace les ordres d'écriture par des ordres d'affichage, choisissant les descripteurs qui l'intéressent et les associant aux ordres d'affichage par leur numéro d'identification. Lors de l'exécution du programme ainsi modifié, la rencontre d'un ordre d'affichage provoque l'appel du logiciel graphique qui compose le dessi à partir du descripteur (image formelle) et des valeurs passées en paramètres (création de l'image réelle). Le passage à l'image suivante (c'est-à-dire le retour au programme d'application) se fait soit sur commande de l'utilisateu soit automatiquement.

Les avantages d'un tel système sont les suivants:

- le programme à illustrer est très légèrement modifié, sa structure interne étant intégralement respectée.

- une certaine indépendance par rapport aux langages de programmation est obtenue puisque les ordres graphiques sont rejetés en dehors du programme d'application.

- le programme à illustrer n'est compilé qu'une seule fois, quel que soit le nombre d'essais faits pour choisir une représentation adéquate.

- si les descripteurs nécessaires existent dans la bibliothèque, le travail d'illustration est minime.

- une représentation ayant été choisie et introduite dans la bibliothèque, elle peut servir à illustrer une série de programmes, sans besoin de la recréer.

- la séparation entre programme d'application et illustration graphique permet d'envisager un partage de tâches entre un calculateur principal chargé d'exécuter le programme d'application et un calculateur satellite chargé de l'illustration.

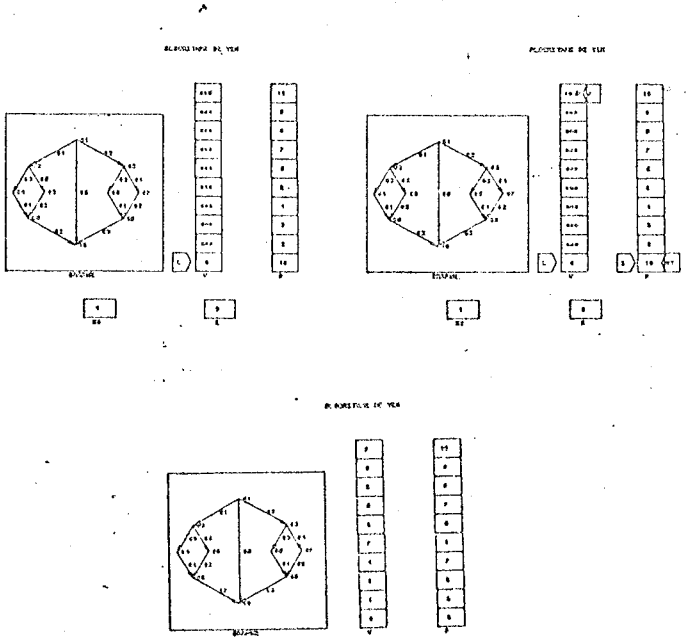
Un tel système a été réalisé à partir de ces principes ([Lau 76a] [Lau 76b]). Il réunit de plus les qualités suivantes:

- outils de création et de mise à jour de la bibliothèque de descripteurs, même en cours d'exécution du programme d'application. On peut ainsi, au vu d'une image, modifier la description ou créer un nouveau descripteur.

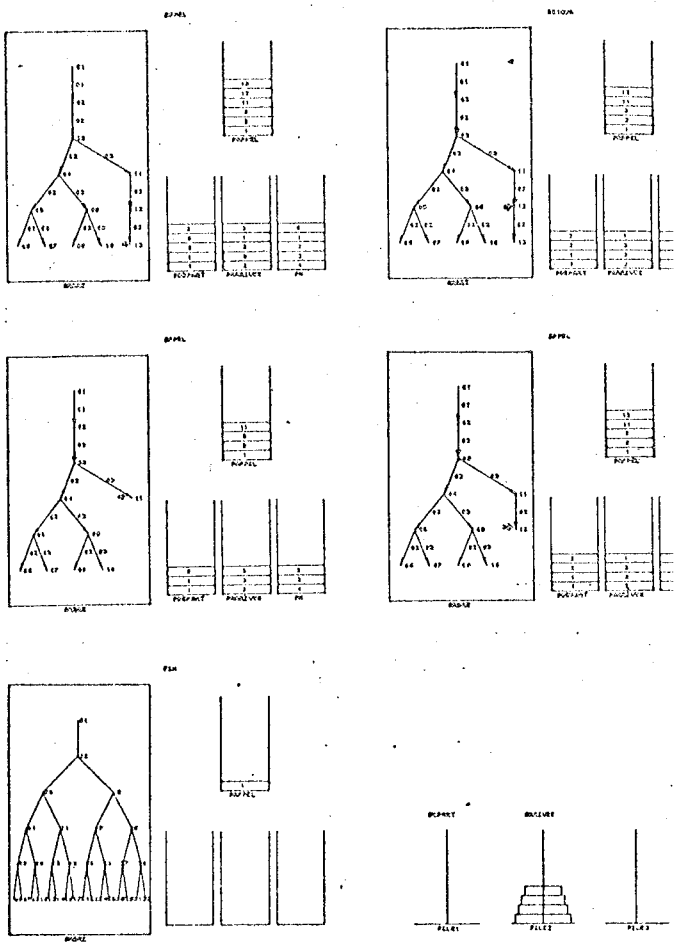
- mise en page automatique de l'ensemble du dessin, en utilisant des critères de répartition sur l'écran permettant d'obtenir une image visuellement homogène.

- existence d'un préprocesseur permettant de remplacer automatiquement certains ordres du programme d'application pour faciliter le suivi du déroulement. C'est ainsi que les commentaires du programme peuvent être affichés automatiquement sur l'écran, que l'on peut obtenir la trace des appels d'une procédure récursive, etc...

Nous sommes donc en présence d'un logiciel graphique interactif intégrant en logiciel de description que l'on peut enrichir à volonté et un logiciel de préparation à la visualisation de niveau élevé puisque capable d'assurer une mise en page automatique. L'ensemble constitue un programme très complexe et très volumineux. Les expériences que nous avons pu faire ont cependant montré son intérêt, en particulier dans le domaine de l'enseignement. La figure 4.23 donne quelques exemples d'illustrations obtenues grâce à ce système.

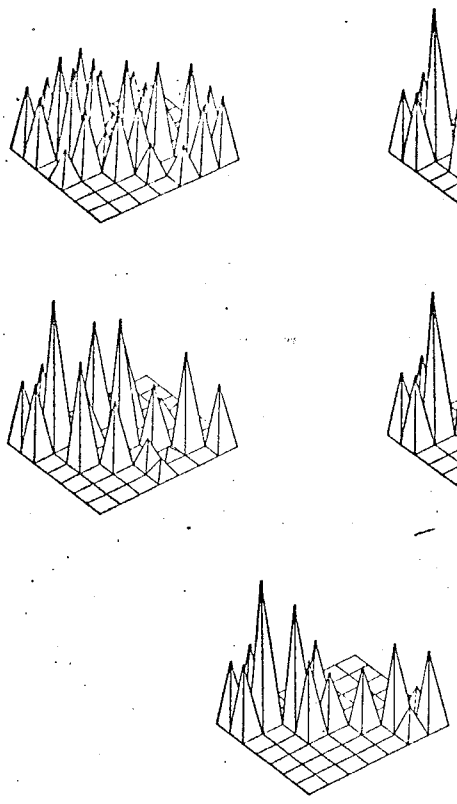


algorithme sur un graphe (YEN)

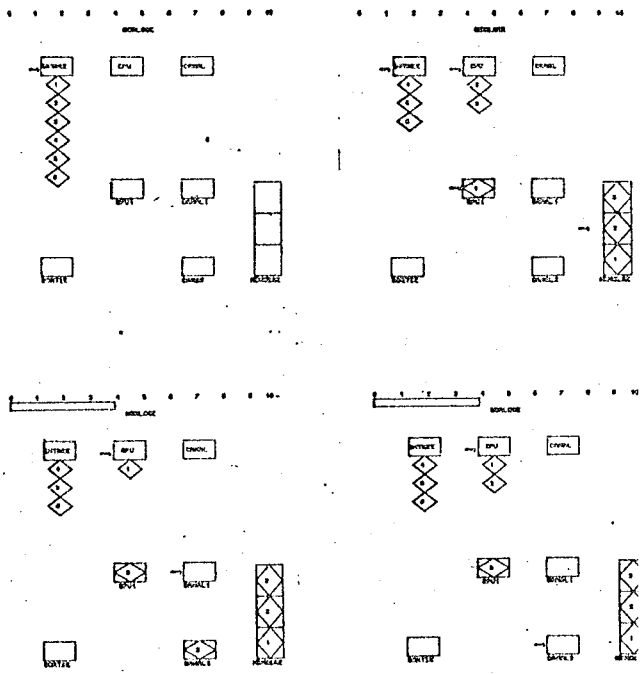


Tours de HANOI

(avec l'arbre des appels récursifs)



diagonalisation de matrice



simulation d'un système d'exploitatio

figure 4.23 Illustration d'algorithmes (d'après [Lau 76a])

4.43 Le dessin animé

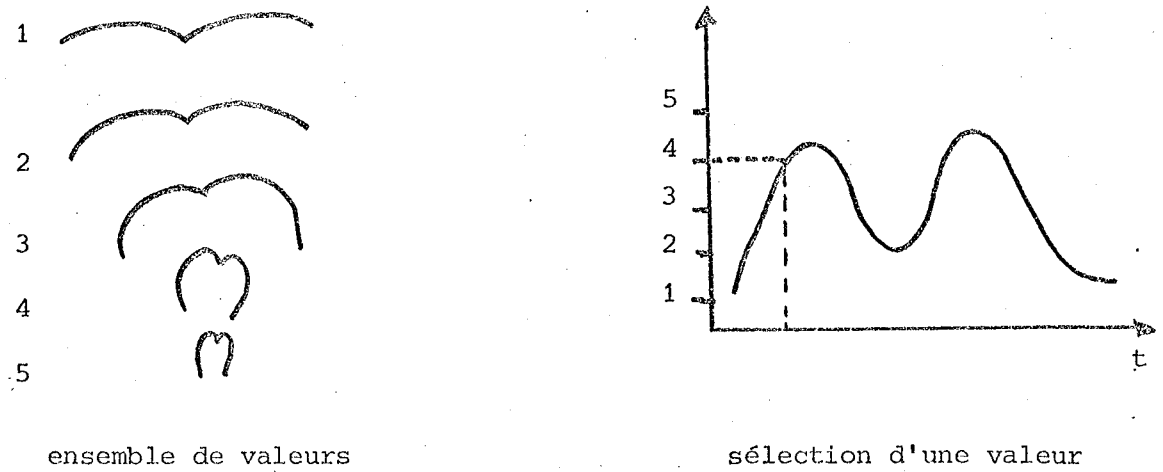
Les techniques de l'illustration de programme sont insuffisantes dans le cas de la création de dessins animés, puisque la principale caractéristique de ceux-ci est que l'évolution des différents éléments doit tout à l'imagination du réalisateur et qu'il est en général impossible de trouver des lois mathématiques (donc programmables) permettant d'assurer les différen transitions. La réalisation d'un dessin animé revient donc à créer un grand nombre de dessins qui, pris dans un ordre chronologique, diffèrent légèrement l'un de l'autre. L'animation est obtenue en jouant sur les différents élément caractéristiques d'un dessin tels que nous les avons énumérés en 1.1. A chacu de ces caractéristiques on peut associer une notion de *valeur*:

- pour le tracé, il s'agit de l'ensemble des points, traits et taches,

- pour le repère, il s'agit d'un couple de réels pour définir l'origine, d'un réel pour l'angle de rotation,

- pour la taille, il s'agit d'un réel (homothétie) ou d'un couple de réels (similitude).

Animer un dessin, c'est faire varier en fonction du temps ces caractéristiques. Par exemple, si l'animation résulte d'un changement de form on utilisera un ensemble de formes (ensemble de valeurs) et un descripteur permettant d'indiquer quelle valeur est choisie à un instant t donné. Cette animation peut être décrite à l'aide de dessins et de graphiques dont la figure 4.24 donne un exemple.



ensemble de valeurs sélection d'une valeur

figure 4.24 Animation par changement de forme

Ce mécanisme peut être étendu aux autres caractéristiques, ce qui permet dès lors de disposer d'un système d'animation qui tient compte globalement de l'ensemble des caractéristiques d'un dessin et pas seulement de la position et de la forme comme c'est le cas dans la plupart des logiciels existants. Une autre remarque peut être faite à partir de l'examen de la figure 4.25:

figure 4.25 Différentes interprétations d'un relevé

Nous voyons que le dessin de la mer joue trois rôles:

- dessin proprement dit des vagues,

- trajectoire du bateau (positions successives de son origine),

- orientation du bateau (il suit la pente des vagues).

Nous déduisons de cette observation quelques mécanismes de base des logiciels d'aide à la conception de dessins animés:

- le système doit fournir la possibilité de construire des *ensembles de valeu*

- le système doit permettre de *sélectionner* une valeur d'un ensemble en fonction du temps,

- le système doit permettre d' *interpréter* un ensemble de valeurs différemmer suivant la signification attribuée au moment de l'évaluation du dessin.

Un logiciel de production d'animation a été réalisé par Francis MARTINEZ [Maz 77b] à partir de ces réflexions. SAFRAN est un système qui présente de plus les caractéristiques suivantes:

- le dialogue animateur-système est essentiellement graphique,

- la mise en oeuvre est calquée sur les habitudes de travail des animateurs,

- la description des mouvements et des déformations est analogue à celle utilisée dans les ateliers d'animation.

Les éléments de base du système SAFRAN sont les suivants:

- le *relevé*, qui permet d'exprimer sous forme graphique l'ensemble des donné nécessaires à l'animation. Un relevé regroupe l'ensemble des renseignement associés à un trait introduit par le dessinateur à l'aide d'une tablette graphique. Les relevés serviront de base à l'interprétation qui en sera fa pour définir les différents ensembles de valeurs et les descripteurs.

- les *acteurs*, qui seront les composants de l'animation et sont dotés de tro rubriques:
  . le *repère*, permettant de définir la position, la taille, l'orientation d l'acteur,
  . la *trame*, constituée de *points-guides* qui permettent la manipulation de l'acteur au cours de l'animation,
  . les *états*, qui sont la description des différents aspects et formes que l'acteur est susceptible de prendre au cours de l'animation.

La description de l'animation se fait à l'aide de *fonctions d'animation*, définies par les éléments suivants:

- le *domaine d'évolution*, c'est-à-dire l'ensemble des valeurs associées à une caractéristique,

- la *fonction d'évolution*, qui permet de choisir à l'instant t une valeur.

Ce mécanisme permet d'utiliser les relevés aussi bien pour définir les domai d'évolution que les fonctions d'évolution.

Le point le plus important est l'utilisation d'un outil *d'interpolation* pour définir des ensembles de valeurs. En effet, il est classique de définir une suite de dessins à partir de deux dessins-clés, laissant le soin au calculateur de définir les intervalles. Notre proposition de ne considérer la forme que comme une caractéristique comme une autre permet d'appliquer le procédé d'interpolation à toutes les autres caractéristiques d dessin (couleur, aspect, positions). L'interpolation devient donc l'outil indispensable de tout système d'animation. Différents procédés d'interpolatio doivent être mis à la disposition de l'animateur, afin qu'il puisse choisir c qui lui semble le plus adapté à l'effet qu'il cherche à obtenir. On trouvera étude de quelques procédés dans [Maz 77a].

Le système SAFRAN a prouvé qu'il était possible de réaliser un système complet basé uniquement sur des manipulations graphiques. Cependant, des extensions peuvent être envisagées à ce type de système, portant sur deux catégories d'améliorations:

- une description de l'animation permettant d'obtenir des compositions de mouvements,

- la possibilité d'introduire des contraintes et des animations conditionnell

En ce qui concerne la composition de mouvements, nous avons propos ([Luc 72a], [Luc 72b]) la notion de *bloc d'animation* , qui permet d'emboîter des fonctions d'animation de même type. L'animateur décrit des mouvements ind duels et laisse le système les composer entre eux pour présenter le mouvement final. La notion de bloc d'animation permet d'exprimer, à l'aide d'une notati de type ALGOL, la portée d'une fonction d'animation. Les éléments appartenant à un bloc subissent l'action des fonctions utilisées dans ce bloc et dans tou les blocs englobants. Les mouvements définis dans deux blocs disjoints ne se composent pas entre eux. Nous avons réalisé un système d'animation qui a perm de tester ce concept sur des animations à base de translations. L'expérience a montré que ce concept était très intéressant pour tout ce qui concernait le mouvements (c'est-à-dire tout ce qui touche au repère). Pour les autres caract téristiques, le principal problème est de définir les lois de composition. Nc n'avons pas eu l'occasion d'étudier cette question et ne pouvons donc conclur sur l'intérêt pratique de cette généralisation.

La possibilité d'introduire des contraintes conduit à utiliser deux types de données:

- des *points d'appui* qui permettent de construire l'ensemble du dessin: c'est le rôle de la trame de SAFRAN,

- des *relations* entre points, relations s'exprimant en termes de distances, angles, etc...

On pourra dès lors réaliser des animations conditionnelles telles que:

- avancer tant que tel acteur n'a pas franchi telle limite,

- dessiner un acteur dont la taille est fonction de l'éloignement de tel autre acteur.

Certains essais ont été faits pour introduire ces notions, soit en développant de véritables langages de programmation ([Bae 69], [Pfi 75]), soit en étudiant des algorithmes permettant de tenir compte de certaines contraintes [Maz 75]. La leçon principale que nous avons tirée est que le développement de tels systèmes implique la définition de langages de commande plus ou moins sophistiqués, et que l'on ne peut plus se contenter de logiciels uniquement graphiq

4.5 CONCLUSION

Nous nous contenterons de faire deux remarques :

- les logiciels de description ont une structure extrêmement variée, allant de l'inexistence à une séquence de plusieurs codages. Ils sont très dépendants des applications qui les utilisent, et leur résultat est également très sensible au code utilisé. Il semble difficile d'envisager des logiciels de description généraux, même en ce qui concerne les logiciels de description géométrique. Nous en voulons d'ailleurs pour preuve le fait que tous les logiciels généraux de description que nous connaissons se voient tour à tour adaptés à la mécanique, à la cartographie, à l'aéronautique etc ..., les versions successives étant incompatibles entre elles.

- la notion d'interprétation des données est omniprésente, les données initiales pouvant conduire à des types de dessins très différents suivant les besoins du moment. Ceci est particulièrement vrai en conception assistée, les besoins de présentation variant en fonction de la phase de conception où l'on se trouve.

CONCLUSION

CONCLUSION

Nous voudrions souligner pour conclure la contradiction existant entre les propositions que nous avons faites au long de cette thèse et l'état actuel des travaux dans le domaine des techniques graphiques interactives. En effet, les progrès spectaculaires réalisés ces dernières années tant du point de vue du matériel que du point de vue du logiciel peuvent conduire à penser que les travaux de recherche sont terminés dans ce domaine. Nous pensons au contraire qu'ils ne font que commencer. En effet, s'il est indéniable qu'une certaine maîtrise des techniques a été atteinte, les bases conceptuelles qui ont conduit à cette maîtrise nous semblent à revoir. Elles ont une origine historique, liée à l'apparition des traceurs de courbe, puis à celle des dispositifs de communication. Tout s'est passé comme si une analyse ascendante des problèmes avait été faite, se traduisant par l'accumulation de couches de logiciel conçues au coup par coup, sans remise en cause des hypothèses de départ.

Nous proposons au contraire de reprendre les analyses en repartant cette fois des applications, sachant parfaitement que l'on saura réaliser les programmes permettant de travailler au plus bas niveau. Nous bâtirons notre conclusion à partir des principaux points de définition donnés dans cette thèse, proposant les voies de recherche qui nous semblent les plus importantes.

Nous avons d'abord proposé de faire une distinction entre les dessins et les images. Nous avons cependant montré que dans certains cas, cette distinction n'avait pas lieu d'être, puisque l'un ou l'autre type de présentatic peut être réalisé quel que soit le matériel employé. Cependant, si une bonne maîtrise des dessins au trait semble acquise, aussi bien au niveau de la description que du tracé, il n'en est pas de même en ce qui concerne les images. En particulier, les problèmes de description d'images complexes (par exemple formées de taches multichromes) ne sont pas résolus à l'heure actuelle. De plus, il serait utile de pouvoir disposer d'opérations de composition de taches (unior intersection, superposition etc...) permettant de composer ainsi simplement des images complexes.

En ce qui concerne le dialogue, la recherche de nouvelles actions de base permettrait en particulier de simplifier les problèmes d'acquisition d'un dessin ou d'une image. Il nous semble qu'ici un pont devrait s'établir avec les chercheurs qui travaillent dans le domaine du traitement des images. En effet, les utilisateurs ont un besoin urgent de techniques d'acquisition aussi bien en deux dimensions (entrée de plans, de schémas de cablage, etc...) qu'en trois dimensions (entrée d'objets à partir de photos). Nous noterons que la définition correcte de ce qu'est une image est ici aussi indispensable. De plus, les techniques de restauration et amélioration d'images commencent à être utilisées dans le domaine de la production d'images par ordinateur. Nous voyons donc que l'ensemble des techniques de traitement d'images s'appliquera au domaine des techniques graphiques interactives. Une collaboration intense entre les chercheurs de ces deux domaines est donc souhaitable.

Nous avons également proposé de distinguer trois nivéaux de logiciels graphiques. Cette distinction devrait conduire à des travaux important sur les problèmes suivants:

- lorsqu'on parle d'interaction, à quel niveau s'applique-t-elle ? faut-il ou non remonter systématiquement jusqu'au programme d'application pour modifie ce qui est sur l'écran ? si non, comment spécifier le niveau de travail ?

- faut-il disposer des trois (ou plus) structures de données différentes, ou peut-on au contraire faire des simplifications ? comment passer les différents noms à tous les niveaux de données aux diverses couches de logiciel ? faut-il utiliser plusieurs listes de noms, relatives chacune à un niveau de données, mais liées entre elles ?

- comment partager les différentes couches de logiciel entre deux (ou plusieu calculateurs (ou micro-processeurs)? Quelles sont les techniques à employer pour que le passage à travers les différentes couches de logiciel ne soit pas trop pénalisant ?

- quel degré d'indépendance par rapport à l'environnement de chacun de ces logiciels (matériel, langage de programmation, système d'exploitation, application) peut on espérer ?

En ce qui concerne la définition de logiciels graphiques interactif de base, nous avons proposé un découpage dont nous avons déduit un logiciel qui est en cours d'expérimentation. On peut bien entendu se poser la question de définir d'autres découpages, et donc de définir d'autres logiciels. Nous pensons qu'un effort important doit être porté sur les modes d'expression du dialogue. En effet, plusieurs voies sont offertes:

- définition de nouvelles primitives d'entrée plus adaptées au travail en mode graphique,

- définition de l'enchaînement des différentes commandes par l'opérateur. Cette définition peut se faire à l'aide de langages spécialisés, à l'aide de diagramm d'état ou autres techniques infomatiques. Il nous semble important de chercher à introduire dans ces diverses techniques une notion d'apprentissage, qui permettrait ainsi au logiciel graphique de prendre en charge peu à peu l'enchaînement de séquences d'opérations, soulageant ainsi aussi bien le programmeur d'application que l'opérateur. De plus, une meilleur sécurité opératoire en découlerait, puisqu'un contrôle pourrait être fait sur les différentes phases de la conception.

- intégration de la description du dialogue au programme d'application, ou au contraire expression de ce dialogue en dehors du programme d'application. Le point soulevé ici correspond en fait à des problèmes de compilation, puisqu'il s'agit en fait de savoir si l'on peut définir et compiler des portio de programmes séparées, portions destinées éventuellement à ne pas être exécutées sur les mêmes machines.

Ce dernier point est également prôche d'un problème que nous n'avons pas abordé, celui de l'intégration des logiciels graphiques à un langage de programmation. Le point d'étude intéressant est de regarder quelle peut être l'influence d'un type de alngage de programmation sur la réalisation des primitives graphiques. En effet, il est probable que l'on ne programme pas de la même manière si on a " l'esprit FORTRAN " ou si on a " l'esprit APL ". Cette étude pourrait conduir à la définition de normes différentes suivant les types de langages.

En ce qui concerne la définition de logiciels de préparation à la visualisation, beaucoup de travail reste à faire, en liaison avec les différents types de codes que l'on peut utiliser. Nous pensons cependant que des logiciels de préparation à la visualisation pourraient être normalisés, en particulier en ce qui concerne les opérations de mise en page. Cependant, il serait important de disposer d'études théoriques sur la notion de complexité d'une scène, afin de pouvoir mieux adapter les logiciels de préparation à la visualisation. C'est ainsi que nous avons présenté quelques algorithmes d'élimination de parties cachées. S'il est indéniable que les plus grands progrès ont été accomplis dans ce domaine ces dernières années, si des images extrêmement spectaculaires et réalistes peuvent être obtenues, il est tout aussi frappant de constater que cette apparente maîtrise s'appuie sur une méconnaissance profonde des caractéristiques principales des scènes tridimensionnelles, fussent-elles composées uniquement de polyèdres. Il est indispensable que des études théoriques soient faites sur ce point, qui permettraient ensuite de choisir les algorithmes de base les meilleurs pour réaliser l'élimination de parties cachées. De plus, des comparaisons entre les performances des différents algorithmes pourront alors être faites de manière un peu moins empirique, ce qui permettrait peut être de conclure définitivement quant au meilleur algorithme.

Une autre voie de recherche concerne les différents domaines d'application. Nous sommes persuadés que des logiciels de description s'adaptant à de larges classes d'applications pourraient être définis. Ceci implique que l'on se pose des questions dans les domaines suivants:

- étudier de nombreuses applications pour essayer de trouver certaines caractéristiques communes, conduisant à définir avec précision une classe,

- étudier les problèmes de méthodologie de programmation et de mise en oeuvre des programmes d'application, afin de voir l'influence que peuvent avoir les différents niveaux de conception de logiciel que nous avons définis sur la réalisation d'un programme graphique interactif,

- déduire de certaines applications les primitives qui leur sont adaptées, quitte à ce que ces langages de commande reposent en fin de parcours sur un logiciel graphique classique. Le point soulevé ici revient à voir si l'on ne pourrait pas trouver des méthodes permettant de "personnaliser" un logiciel graphique par le biais d'extensions linguistiques adaptées aux habitudes de pensée de l'utilisateur.

Une autre voie de recherche très importante concerne le traitement des images dynamiques. Nous avons déjà évoqué au chapitre 4 les problèmes de description du dynamisme. Cependant, des travaux fructueux pourraient être entrepris au niveau purement algorithmique, si l'on cherchait à améliorer les temps de calcul lorsque l'on passe d'une image à une autre. En effet, la notion de "continuité d'une image" qui est utilisée dans le cadre des algorithmes d'élimination de parties cachées pourrait être reprise ici au niveau de la suite d'images (ou de dessins). C'est ce qui se fait déjà lorsque l'on engendre des séries de dessins à partir de dessins-clés. Cependant, une attention insuffisante a été portée à l'utilisation systématique de techniques de réduction de la complexité des calculs lorsque l'on passe d'une image à une autre. En particulier, dans le cas de la présentation d'images tridimensionnelles, de nombreuses simplifications pourraient être apportées, si l'on était cacpable par exemple de ne recalculer qu'une partie de l'image, le reste étant inchangé, ou de prévoir des positions clés telles que l'image n'est profondément modifiée que si on les atteint (cas de la rotation de polyèdres : les faces apparaissent en disparaissent en fonction du point de vue).

Un dernier sujet d'intérêt concerne la réalisation de ces différen algorithmes à l'aide de techniques caclées ou micro-programmées. Nous n'avons pas évoqué dans cette thèse les problèmes matériels. Cependant, il est évident que les consoles de visualisation seront entourées à l'avenir d'une multitude de micro-processeurs prenant en charge une grande partie des différentes transformations entre la donnée d'application et le dessin ou l'image. Dès maintenant, on assiste à une floraison de réalisations, caractérisées par une diversité des niveaux d'impact proprement incroyable. Il serait très important de faire un travail de synthèse sur une normalisation de ce qui doit être absolument réalisé technologiquement et ce qui n'a aucun intérêt.

En conclusion, nous pensons que notre travail permet de jeter les bases de recherches nouvelles sur les problèmes de conception des logiciels graphiques interactifs. Nous espérons que de nombreuses équipes de recherche pourront se servir de cette thèse comme point de départ de leurs travaux, permettant ainsi de passer du graphique à l'age de FORTRAN au graphique à l'age d'ALGOL ou APL.

ANNEXE 1

EXEMPLES DE CODES UTILISÉS POUR LA DESCRIPTION DE SCÈNES

Nous présentons ci-après quelques exemples de codes particulièrement utilisés :

- le code de FREEMAN, qui est destiné essentiellement à conserver et à manipuler facilement un dessin au trait destiné à une surface à pointillage,

- les codes syntaxiques, qui permettent de décrire des dessins et des images à l'aide d'un vocabulaire et de règles de grammaires définissant de véritables langages graphiques,

- les codes géométriques, qui permettent de décrire les futurs dessins en termes d'objets mathématiques.

A1.1 LE CODE DE FREEMAN

Il permet de représenter un dessin au trait discrétisé sur une surface à pointillage. Cette discrètisation conduit à décrire une courbe par une liste de petits vecteurs élémentaires orientés suivant huit directions (cf figure A1.1) :

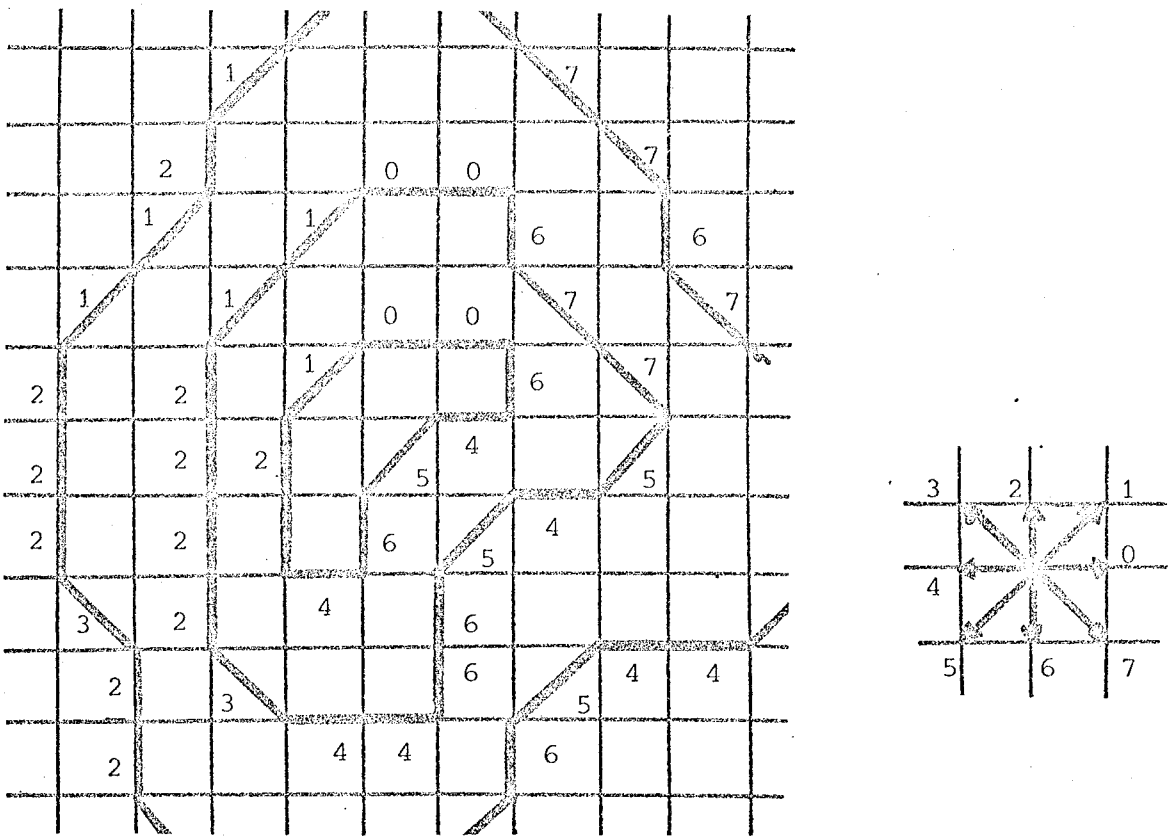


figure A1.1 : Principe du code de FREEMAN

Une chaîne est une suite ordonnée de liens. Un lien est un vecteur élémentaire de longueur $T(\sqrt{2})^p$ formant un angle $a_i \times 45°$ avec l'axe des x, a_i étant un entier compris entre 0 et 7, p étant la valeur de a_i modulo 2, T la largeur de la maille.

Pour diminuer la longueur de la chaîne de codes, il est introduit des opérateurs de répétition d'un même vecteur élémentaire. D'autres opérateurs permettent d'introduire des transformations élémentaires, des informations qualitatives (nom) ou quantitatives (nombre de vecteurs passant par un noeud donné). Nous donnons ci-après la liste des codes opération proposée par FREEMAN [Fre 7

0400	fin de chaîne
0401	la chaîne qui suit ne doit pas être tracée (chaîne invisible)
0402	la chaîne qui suit doit être tracée (opposé de 0401)
0403	sert pour indiquer que le reste du mot mémoire est vide
0404	la chaîne contient la séquence de liens inverses 04
0405XYZ	XYZ est un identificateur octal (000 à 777) que l'on associe à certains points de la chaîne
0407UV	la suite de UV digits apparaissant après ce code est un identificateur numérique, utilisé par exemple pour repérer une chaîne
0410WXYZ	suite de WXYZ digits ne faisant pas partie de la chaîne de codag Permet d'insérer des commentaires dans la description
04111	même description que pour le code précédent, sauf que la suite e terminée par un code spécial, qui permet d'éviter de donner a priori la longueur de la zone commentaire
04127777	fin de la zone de commentaires
0413UWXYZ	WXYZ est un identificateur d'un noeud de grille. Ce code permet de spécifier le nombre de chaînes passant par ce noeud (U). Si U = 0, le nombre est inconnu
0414UVWXYZ	indicateur de rotation. La chaîne qui suit a subi une rotation de U.VWXYZ radians
0415TUVWXYZ	indicateur de changement d'échelle. La chaîne qui suit a subi une modification d'échelle de facteur UVXYZ. Les autres digits permettent de spécifier exactement la transformation
0417UXYZ	répétition du lien U un nombre de fois égal à XYZ
0420UN	répétition du lien U un nombre de fois égal au nombre représenté sur les (N + 4) digits qui suivent. Sert pour les longues chaînes
0421TUVWXYZ	répétition du groupe de TUV digits qui suivent WXYZ fois
0422U	indication de couleur
0423WXYZ	sert pour les courbes de niveau : WXYZ est l'altitude de la chaî qui suit
0424XYZ	sert à représenter des intensités lumineuses sur trois digits
0425U	point de contrôle pour une somme logique
0426VWXYZ	forcer l'abscisse x à la position VWXYZ
0427VWXYZ	forcer l'ordonnée y à la position VWXYZ

Ces deux dernières instructions permettent de fixer les coordonnées de départ d'une chaîne, le reste de la description étant constitué de composantes de ve teurs (coordonnées relatives) et non de coordonnées absolues.

Ce code permet de réaliser de façon relativement simple un certain nombre de calculs sur les chaînes à traiter : longueur, rectangle englobant, inversion du sens de parcours, calcul de la surface enclose dans une courbe fermée, moments d'inertie par rapport aux axes ou aux diagonales, détermination de la plus courte chaîne joignant le début et la fin d'une chaîne donnée et telle que les codes des liens soient rangés en valeur croissante, distance entre deux points, chaîne miroir. Ces opérations sont extrêmement utilisées dans certains traitements d'images en vue de reconnaissance de formes. FREEMAN donne dans son article plusieurs exemples de traitement en vue par exemple de résoudre des problèmes de Puzzles, d'ajustement de contours de continents ou de découpage d'une plaquette de métal.

Nous remarquerons que ce code est très bien adapté au cas des contours très accidentés (avec de nombreux points d'inflexion et des rayons de courbure très faibles). Il est malheureusement trop simpliste pour la plupart des applications que l'on traite en infographie interactive. En particulier, il ne permet pas de décrire simplement des images, ses primitives étant de trop bas niveau. Cependant, de nombreuses applications de traitement d'images en mode interactif utilisent ce code (ou un dérivé).

A1.2 LES CODES SYNTAXIQUES

Les dessins sont considérés comme des graphes de R^2. Les codes doivent permettre non seulement de retrouver tous les composants (sommets, arêtes), mais aussi de conserver une trace de la structure de description du graphe. Pour ce faire, on utilise une grammaire G, produisant un langage L(G). Les propriétés attendues de telles grammaires sont les suivantes :

- permettre de décrire avec un ensemble fini de signes et de règles une famille de graphes,

- permettre de donner une structure au graphe décrit à travers les règles de génération,

- permettre un codage économique du graphe,

- donner des moyens efficaces pour la manipulation, l'analyse et la génération de graphes.

On peut classer les modèles de grammaires décrivant une famille de graphes en trois catégories [Aze 75] :

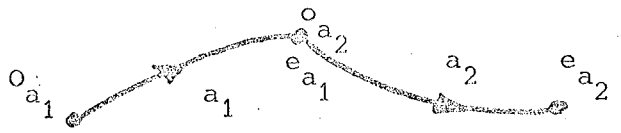
- les grammaires de chaînes,
- les grammaires de structure,
- les grammaires de graphes,

Les grammaires de chaînes produisent des expressions linéaires dé vant le graphe au moyen d'opérateurs unaires et binaires. Soit G un graphe G = (S, A) où S est l'ensemble des sommets et A ⊂ S × S l'ensemble des arête Si a ∈ A, on a a = (o_a, e_a) où o_a est l'origine de a et e_a l'extrémité de A toute arête a du graphe on associe v(a) ∈ V, ensemble des valeurs associée aux arêtes.

Pour décrire le graphe, on définit un certain nombre d'opérations appelées concaténation, qui permettent de définir des lois de composition in quant, pour chaque objet concaténé, la nouvelle origine et la nouvelle extr mité. On trouvera dans ([Moh 71], [Moh 73] une étude algébrique des opérateu de concaténation. Nous donnerons simplement ici l'exemple le plus connu, uti lisé par A.C. SHAW [Sha 67]. Les règles retenues sont les suivantes :

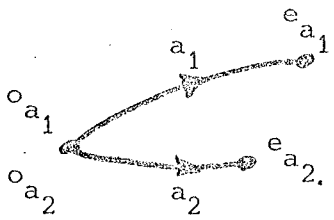
- $a_1 + a_2$

e_{a_1} et o_{a_2} sont confondues, la nouvelle origine est o_{a_1}, la nouvelle extrémité est e_{a_2}.



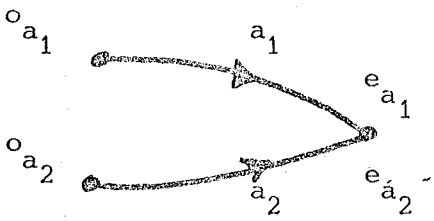
- $a_1 \quad a_2$

o_{a_1} et o_{a_2} sont confondues, la nouvelle origine est o_{a_1} (ou o_{a_2}) et la nouvelle extrémité est e_{a_1}



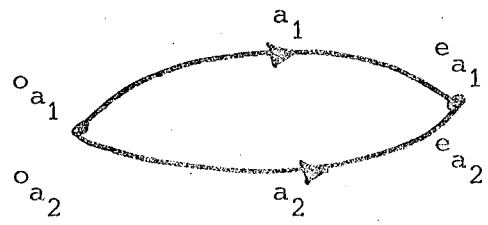
- $a_1 - a_2$

e_{a_1} et e_{a_2} sont confondues, la nouvelle origine est o_{a_1} et la nouvelle extrémité e_{a_1} (ou e_{a_2})



- $a_1 * a_2$

o_{a_1} et o_{a_2} sont confondues, ainsi que e_{a_1} et a_{a_2}. La nouvelle origine est o_{a_1} (ou o_{a_2}) et la nouvelle extrémité e_{a_1} (ou e_{a_2})



- $/a_1$

cet opérateur échange l'origine et l'extrémité.

La chaîne suivante

$$(((((\ /a_2) + a_1) \circledast a_3) \circledast (\ /a_1)) \circledast (\ / (((a_1 + a_2) \circledast (\ /a_3\ )) + a_2)))$$

représente le graphe de la figure A1.2

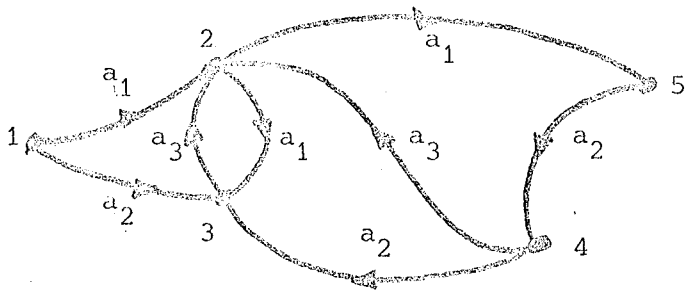
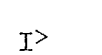


figure A1.2 : Exemple de graphe

Pour un graphe donné, la représentation n'est pas unique, mais on montre que tout graphe peut être décrit par une telle expression. Il suffit d'ailleurs de disposer des opérateurs /, + et (voir [Moh 71], [Moh 73]). Les autres opérateurs ont été introduits pour faciliter le traitement habituellement associé à ce type de représentation, à savoir l'analyse et la reconnaissance de formes. C'est ainsi que SHAW a utilisé son système pour traiter des photographies de chambres à bulles, permettant de détecter automatiquement les chemins suivis par les particules à décompter [Sha 70].

Les grammaires de structure indiquent non seulement la façon de décrire le graphe, mais fournissent de plus, dans les règles de production, la manière de construire le graphe. Les primitives utilisées ne sont plus ass milables aux arêtes du graphe, car au lieu de disposer simplement de la notic de début et d'extrémité, on dispose maintenant d'autant de points de connexic que nécessaire. Dans le formalisme des grammaires de structure, les symboles terminaux représentent les primitives et les symboles non terminaux les parties de dessin de plus haut niveau. La description des concaténations des dif férentes parties du dessin entre elles est donnée par une extension des règle de production : à la suite des symboles terminaux et non terminaux, en partic droite et en partie gauche, on adjoint une liste de points de connexion ainsi qu'une table de correspondance entre les points de connexion de la partie droite et ceux de la partie gauche.

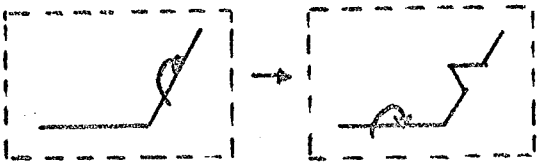
L'exemple le plus connu est celui de la description de l'ensemble de l'alphabet manuscrit, donnée par NARASHIMAN [Nar 69], pour un système de reconnaissance de caractères manuscrits. Nous étudierons un autre exemple, tiré de GIPS [Gip 75] qui a présenté un ensemble de travaux portant sur la génération de dessins pour l'art non figuratif ou la reconnaissance de scènes tridimensionnelles à partir d'une présentation bidimensionnelle. Il utilise pour ce faire ce qu'il appelle les grammaires de forme ("shape grammars"), dont nous étudions l'exemple suivant qui permet de produire la courbe dite du flocon de neige. Pour engendrer cette courbe, GIPS donne deux grammaires, l'une classique, où la courbe est obtenue en substituant les règles une par une, séquentiellement, l'autre exigeant une substitution en parallèle de différentes règles. Le dessin ne se construit donc plus en une seule portion du plan de visualisation, mais simultanément en plusieurs endroits.
La grammaire donnant une construction séquentielle du dessin est la suivante :

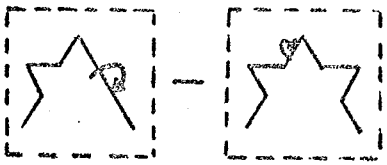
$SG7 = \langle V_T, V_M, R, I \rangle$ $\quad V_T =$  $\quad V_M =$

Règles

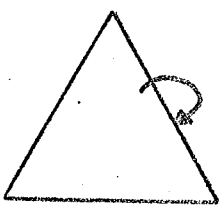
règle 1 

règle 2 

règle 3 

règle 4 

règle 5 

règle initiale 

le symbole permet d'indiquer les éléments non terminaux.

Un exemple de substitution est présenté dans la figure A1.3

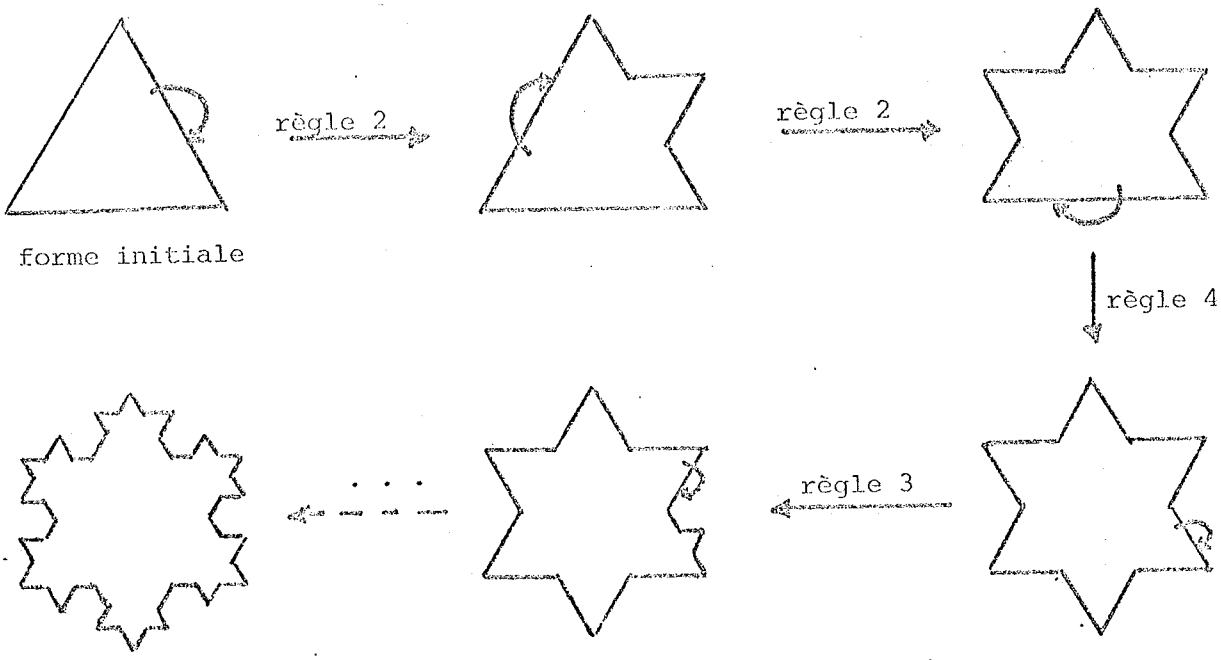


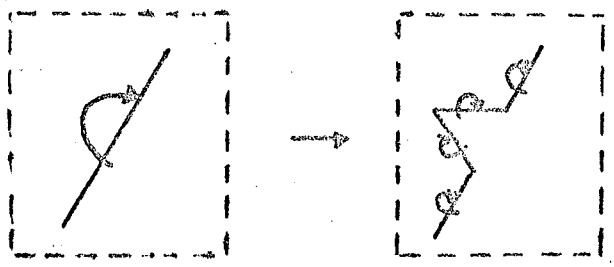
figure A1.3 : Exemple de substitution séquentielle
(d'après [Gip 75])

La grammaire parallèle est la suivante :

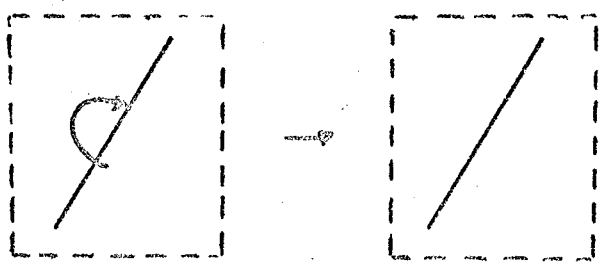
PSG6 = $<V_T, V_M, R, I>$ V_T = { ⟶ } V_M = { ↷ }

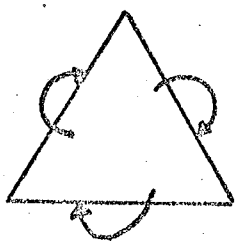
Règles

règle 1



règle 2





règle initiale

Un exemple de substitution est présenté dans la figure A1.4

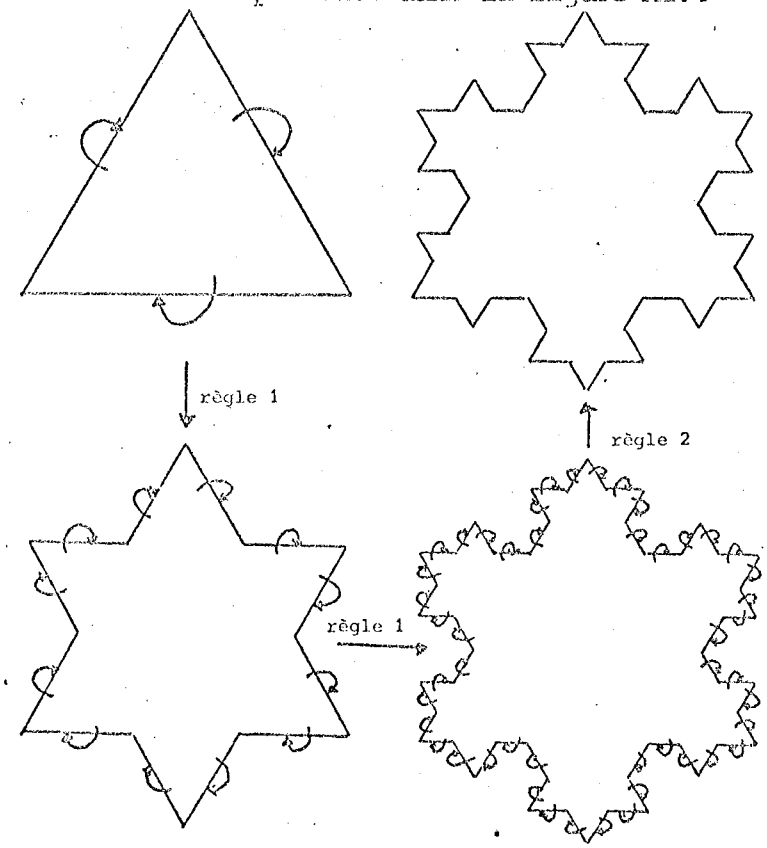


figure A1.4 : Exemple de substitution parallèle
(d'après [Gip 75])

Les grammaires de structure (ou grammaires de forme) sont des outils très intéressants dans la mesure où la description des graphes est plus synthétique (puisque les primitives choisies sont de plus haut niveau) et où la grammaire donne exactement la façon de construire le graphe, ce qui n'était pas le cas pour les grammaires de chaîne.

Un dernier type de grammaire est la grammaire de graphes, permettan de produire directement des graphes, l'application d'une règle de production consistant à remplacer un sous-graphe par un autre sous-graphe. L'axiome de départ est un ensemble de graphes initiaux. Chaque graphe a ses arêtes et ses sommets étiquetés, les étiquettes étant des éléments du vocabulaire terminal et du vocabulaire non terminal.

Dans une règle de production il y a :

- d'une part deux graphes, le graphe de la partie gauche devant avoir au moins un sommet ou une arête étiquetée par un symbole non terminal,

- d'autre part un ensemble de règles d'imbrication et de conditions d'application de la règle.

Nous donnerons en exemple la grammaire permettant de produire toutes les représentations de formules de molécules de chimie, exemple donné par AZEMA dans [Aze 75].

Les éléments du vocabulaire terminal sont d'une part les symboles atomiques (étiquettes du graphe), d'autre part des étiquettes désignant le degré de la liaison (1, 2, 3). Les éléments du vocabulaire non terminal ne sont que des étiquettes de sommets : M_1, M_2, M_3, M_4, M_o

Le graphe initial est M_o

Les règles de production sont les suivantes :

1) M_i — 1 — M_1 $\quad o \leqslant i \leqslant 3$

2) M_i — 2 — M_2 $\quad o \leqslant i \leqslant 2$

3) M_i — 3 — M_3 $\quad o \leqslant i \leqslant 1$

Pas de conditions d'application, mais des règles d'imbrication :

- le sommet de partie droite étiqueté M_{i+1} ou M_{i+2} ou M_{i+3} est le même que l'unique sommet de partie gauche

- le sommet étiqueté M_1, M_2 ou M_3 est un nouveau sommet.

4) M_i M_j — 1 — M_{i+1} M_{j+1} $\quad 1 \leqslant i \leqslant 3$

5) M_i M_j — 2 — M_{i+2} M_{j+2} $\quad 1 \leqslant i \leqslant 2$

6) M_1 M_1 — 3 — M_4 M_4

La condition d'application de ces trois règles est que les sommets de la partie gauche ne doivent pas être connexes dans le graphe à dériver. Les règles d'imbrication montrent que l'on crée une nouvelle arête entre les deux sommet et que leurs étiquettes deviennent celles de la partie droite.

Les dernières règles sont les suivantes :

7)	M_1	H/Li/Na/F/Cl ...	atomes de valence 1
8)	M_2	Be/Mg/Ca/O/S ...	atomes de valence 2
9)	M_3	B/Al/Ni/P ...	atomes de valence 3
10)	M_4	C/Si/Ti ...	atomes de valence 4

Un exemple de dérivation est présenté figure A1.5

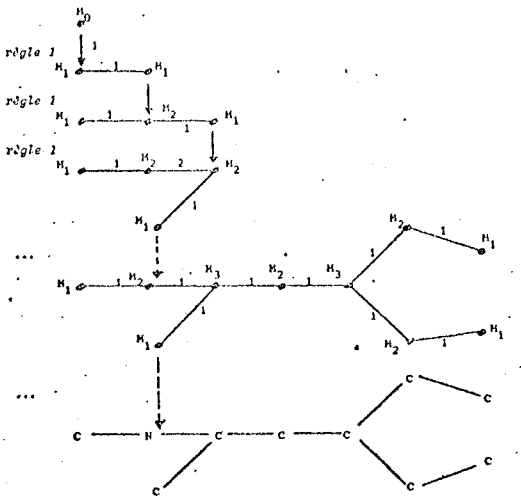


figure A1.5 : Exemple de dérivation d'une formule chimique (d'après Aze 75)

Les grammaires de graphe ont été très utilisées pour développer des algorithmes permettant en particulier de traiter certaines propriétés de graphes : reconnaissance ou production de types particuliers de graphes, problèmes de planéarité ou d'isomorphisme, codage de cartes.

La conclusion que nous voulons tirer de cette rapide étude est que les grammaires de graphes sont en général bien adaptées pour un certain type d'application. Elles sont plus intéressantes que le code de FREEMAN, mais de nouveau insuffisantes pour établir des logiciels généraux : il faudrait en effet proposer des logiciels qui permettent de construire des grammaires dépendant de l'application concernée, et ceci le plus automatiquement possible. Les techniques correspondantes n'étant pas encore disponibles, la grande majorité des systèmes généraux utilise un codage géométrique que nous étudions au paragraphe suivant.

A1.3 LES CODES GEOMETRIQUES

Nous regroupons sous cette appellation les codages d'images qui se font à l'aide de primitives géométriques élémentaires et d'opérations de transformations géométriques telles que la translation, la rotation, le changement d'échelle, le changement de système de coordonnées etc ... L'idée de base est que toute scène à représenter sera décomposée en éléments géométriques dont on indiquera, pour chaque composant, le type, la position, la taill et divers autres attributs qui serviront plus tard à réaliser le tracé. On distingue en général les codes dans le plan et les codes dans l'espace.

En ce qui concerne le plan, les primitives généralement employées sont les points, les segments de droite, les coniques et arcs de conique, les polynômes de degré quelconque. Les codes diffèrent essentiellement par le nombre , la paramétrisation et le niveau d'abstraction des primitives. Les opérations généralement autorisées concernent la définition de systèmes de coordonnées, la translation, la rotation et le changement d'échelle.

En ce qui concerne l'espace à trois dimensions, les codes diffèrer par le type des objets que l'on peut définir. Citons :

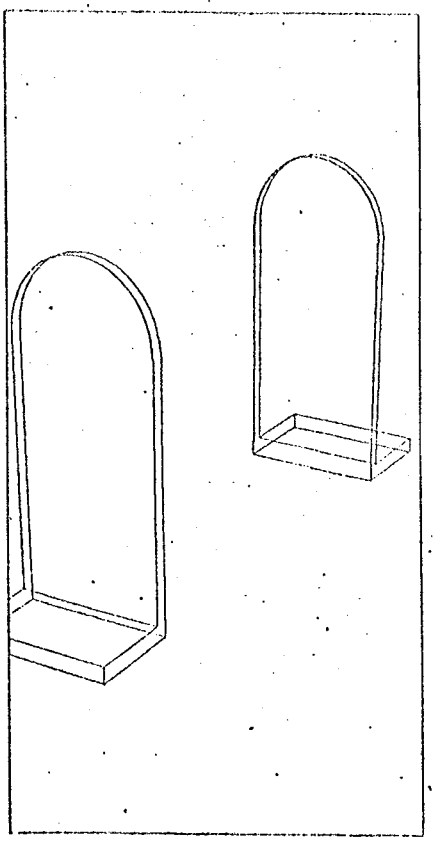
- utilisation de segments de droites de l'espace, permettant de définir des graphes de R^3 par la donnée de sommets et d'arêtes. La figure A1.6 donne un exemple de programmation à l'aide de FORTRAN 3-D ([Duc 74], [IRI 76]),

```
C DEFINITION D'UNE FACE
      SUBROUTINE FACE
      LINE FROM 5-150.,0.,0.) TO (-150.,470.,0.);
      REAL TC(2),TP(4)
C SPECIFICATION DU CENTRE DU CERCLE
      TC(1) = 0.
      TC(2) = 470.
C SPECIFICATION DES POINTS DE DEPART ET D'ARRIVEE DE L'ARC
      TP(1) = 150.
      TP(2) = 470.
      TP(3) = -150.
      TP(4) = 470.
C TRACE DE L'ARC DANS LE PLAN Z=0.
      CALL ARC2P (TC,TP)
      LINE FROM (150.,470.,0.) TO (150.,0.,0.);
      RETURN
      END

C DEFINITION DES AXES. OZ ORIENTE VERS L'ARRIERE EN FORTRAN 3D
      SUBROUTINE AXES
      LINE FROM (0.,0.,0.) TO (200.,0.,0.);
      MOVE MARK (' X',2,0.,1.);
      LINE FROM 50.,0.,0.) TO (0.,800.,0.);
      MOVE MARK (' Y',2,0.,1.);
      LINE FROM (0.,0.,0.) TO (0.,0.,-240.);
      MOVE MARK (' Z',2,0.,1.);
      RETURN
      END

C DEFINITION D'UN SOCLE DE LARGEUR A ET DE LONGUEUR 300 DANS XOZ
      SUBROUTINE SOCLE (A)
      REAL A
      LINE OF 5-150.,0.,0.) OF (0.,0.,-A) OF (300.,0.,0.)
      OF (0.,0.,A) OF (-150.,0.,0.)
      RETURN
      END

C DEFINITION D'UN SERRE-LIVRES COMPOSE DE DEUX SOCLES, DEUX FACES
C ET QUATRE SEGMENTS
      SUBROUTINE SERRE
      CALL FACE AT (0.,30.,0.) DIM(3);
      CALL FACE AT (0.,30.,-25.) DIM(3);
      CALL SOCLE (200.) AT (0.,0.,0.) DIM(3);
      CALL SOCLE (175.) AT (0.,30.,-25.) DIM(3);
      LINE FROM (-150.,0.,-200.) OF (0.,30.,0.);
      LINE FROM (150.,0.,-200.) OF (0.,30.,0.);
      LINE FROM (150.,0.,0.) OF (0.,30.,0.);
      LINE FROM (-150.,0.,0.) OF (0.,30.,0.);
      RETURN
      END
C DEFINITION DE L'IMAGE IMAG1 FORMEE D'UN SERRE-LIVRES ET DES AXES
      SUBROUTINE IMAG1
      CALL SERRE AT (0.,0.,0.) DIM(3);
      CALL AXES AT  (0.,0.,0.) DIM(3);
      RETURN
      END
```



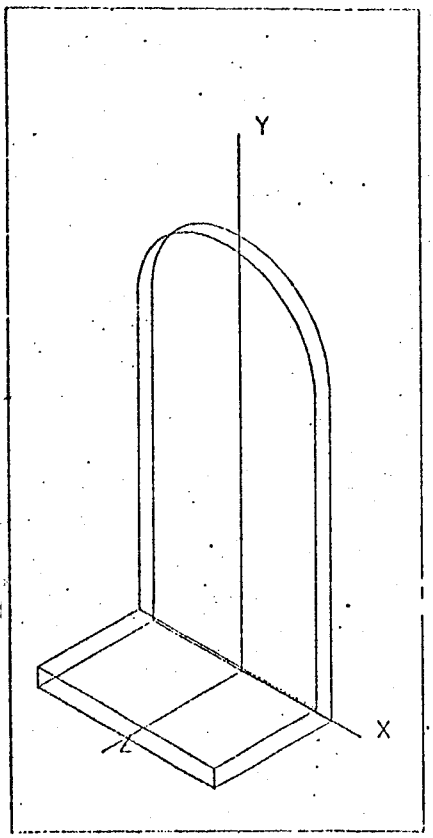
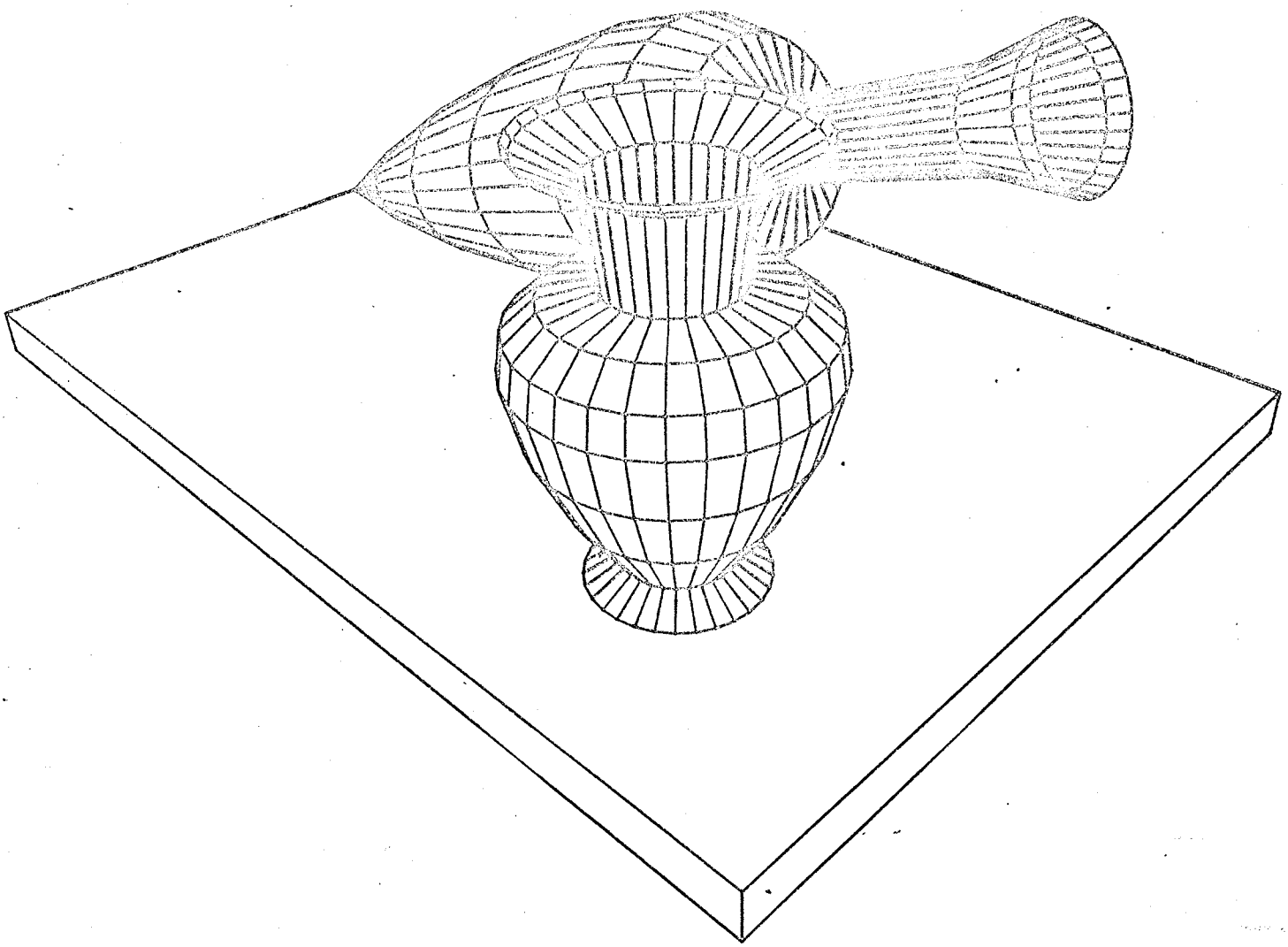


figure A1.6 Exemple de programmation en FORTRAN 3D

(avec l'aimable autorisation de l'IRIA)

- utilisation de polyèdres, le nombre de facettes étant d'autant plus élevé que l'objet à représenter est complexe. C'est à l'heure actuelle le type d'objets le mieux connu et nombre de logiciels ont été développés. Citons par exemple EUCLID [The 72], développé au LIMSI, qui permet de décrire des formes complexes avec un langage de commande dont la figure A1.7 donne un exemple.



nature morte

```
VASE = REVOL(30,11)
CRATER = DPAFFZ(REVOL (30,11), 2.5)
CRATER = DPTRAN( DPROTX ( CRATER, -70), -10, -30, 0)
SOCLE  = DPTRAN( BOITE ( 32, 45, 2), -24, -35, -2)
NATMOR = FIGURE( SOCLE, VASE, CRATER)
OEIL   = POINT ( 30, 30, 25)
CALL PERS ( NATMOR, OEIL, POINT( 0, 0, 10), 35)
```

figure A1.7 : Exemple de programmation EUCLID

(avec l'aimable autorisation du LIMSI)

- définition de solides à l'aide de volumes élémentaires. Par exemple, BRAID ([Bra 74], [Bra 75]) utilise six formes élémentaires pour définir des formes complexes.
  La figure A1.8 présente les formes élémentaires utilisées.

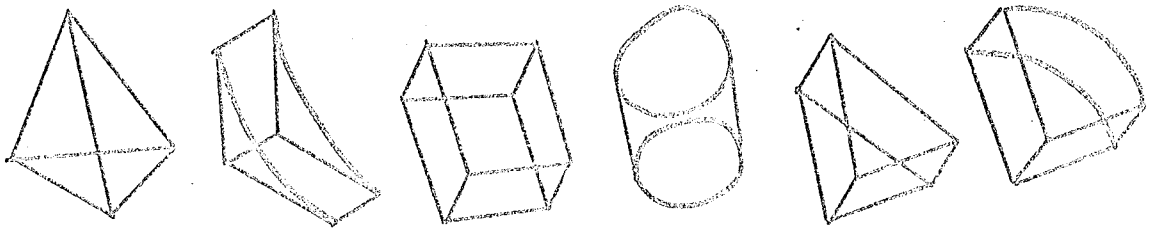
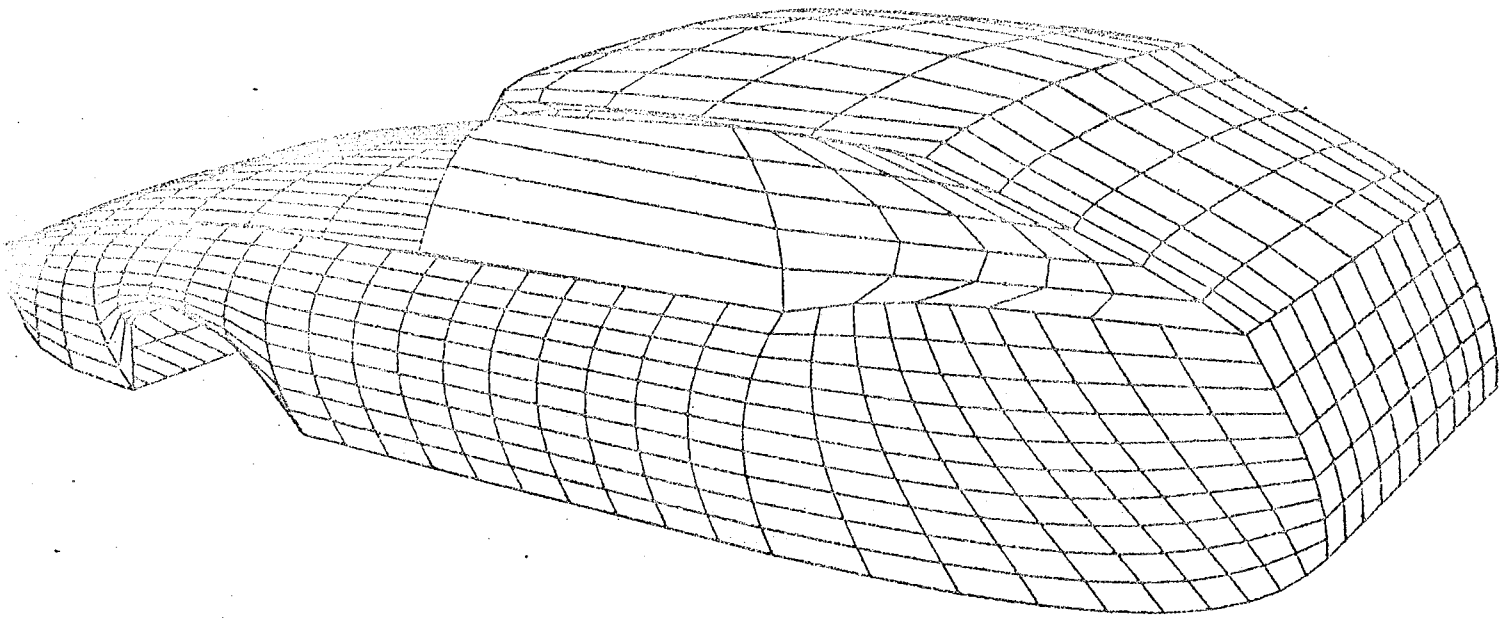


figure A1.8 : Formes élémentaires
(d'après [Bra 75])

Les formes élémentaires sont combinées à l'aide d'opérations d'addition, de soustraction et d'union.

- utilisation de quadriques et d'inégalités quadratiques ([Mah 70], [Lev 76]). Cette approche est satisfaisante pour certains types de pièces (pièces à usiner), mais ne suffit plus pour des formes un peu complexes.

- utilisation de surfaces définies par des approximations polynomiales, telles que les surfaces de COONS [Coo 67], BEZIER [Bez 77] et les b-spline [Rie 73]. Ces surfaces offrent de plus grands degrés de liberté et se prêtent donc mieux à la conception de formes tourmentées. De plus, les paramètres définissant les formes sont choisis de telle manière que la conception puisse se faire de manière interactive (voir figure A1.9).



Cette surface a été décrite puis présentée grâce au logiciel EUCLID

Figure A1.9 Surface de BEZIER

(avec l'aimable autorisation du LIMSI)

ANNEXE 2

PRIMITIVES DE QUATRE LOGICIELS GRAPHIQUES INTERACTIFS DE BASE

A2.1 LES PRIMITIVES DU LOGICIEL PROPOSE PAR LE GROUPE GSPC

A2.11 Les primitives d'affichage

A2.111 - Les primitives de tracé

- primitives permettant de mettre le point courant en un endroit voulu :

MOVE_ABS_2 (x, y)
MOVE_ABS_3 (x, y, z)
MOVE_REL_2 (dx, dy)
MOVE_REL_3 (dx, dy, dz)

- primitives de tracé de segments de droite

LINE_ABS_2 (x, y)
LINE_ABS_3 (x, y, z)
LINE_REL_2 (dx, dy)
LINE_REL_3 (dx, dy, dz)

Un segment de droite est tracé entre la position actuelle et celle spécifiée dans l'appel. Le point d'arrivée devient le point courant.

POLYLINE_ABS_2 (tabx, taby, n)
POLYLINE_ABS_3 (tabs, taby, tabz, n)
POLYLINE_REL_2 (tabdx, tabdy, n)
POLYLINE_REL_3 (tabdx, tabdy, tabdz, n)

Tracé d'une ligne brisée, le premier segment partant du point courant, d'extrémité finale *tabx(1)*, *taby(1)*, *tabz(1)*. Les coordonnées de l'extrémité du dernier segment tracé définissent la position finale du point courant.

- primitives de tracé de textes

TEXT (chaîne-de-caractères)

La suite de caractères spécifiée dans l'appel est affichée à partir du point courant. Chaque caractère dessiné modifie la position du point courant. Cette modification dépend des attributs du dessin de caractères : hauteur, largeur, taille, orientation. Il est proposé de distinguer quatre qualités de texte

("low-quality", "medium-quality", "high-quality", "special-output"). Les résultats du tracé (et donc la position finale du point courant dépendent de chacune de ces catégories. Par exemple, la rotation d'une chaîne de caractères ne donnera pas les mêmes résultats suivant la qualité de texte.

- marqueurs

La notion de point n'existe pas en elle-même. Elle est remplacée par la notion de marqueur ("marker"), symbole destiné à repérer un emplacement de la surface de visualisation. Les primitives de tracé de marqueur sont :

MARKER_ABS_2 (x, y, n)
MARKER_ABS_3 (x, y, z, n)
MARKER_REL_2 (dx, dy, n)
MARKER_REL_3 (dx, dy, dz, n)

Plusieurs symboles peuvent être utilisés, qui sont sélectionnés par le numéro n. La position finale du point courant correspond à la position donnée dans l'appel de la primitive et est donc indépendante de la forme du marqueur.

A2.112 - Les primitives de définition du graphisme

Les graphiques associés à chaque primitive de tracé sont statiques, et ne peuvent être modifiés une fois l'attribution réalisée.

- attribution de graphismes

SET_CURRENT_COLOR (color)
SET_CURRENT_INTENSITY (intensity)
SET_CURRENT_LINEWIDTH (linewidth)
SET_CURRENT_LINESTYLE (linestyle)
SET_CURRENT_FONT (font)
SET_CURRENT_CHARSIZE (charwidth, charheight)
SET_CURRENT_CHARSPACE (dx, dy, dz)
SET_CURRENT_CHARPLANE (dx, dy, dz)
SET_CURRENT_CHARQUALITY (charquality)
SET_CURRENT_PICK_ID (id)

Des primitives d'interrogation sont fournies, qui permettent de connaître la valeur courante de l'un des attributs. Pour en obtenir la liste, il suffit de remplacer, pour toutes les primitives d'attribution, le mot *SET* par le mot *INQUIRE*.

- valeurs par défaut

COLOR	couleur habituelle de travail du dispositif utilisé
INTENSITY	intensité moyenne
LINEWIDTH	épaisseur moyenne
LINESTYLE	trait continu
FONT	police habituelle de la console
CHARSIZE	la plus petite taille fournie par le générateur de caractères
CHARSPACE	espacement normal sur l'axe des x, nul sur les autres
CHARQUALITY	qualité ordinaire
PICK_ID	0

A2.12 Les primitives de structuration

Elles sont explicites, différenciant la modification de la dénomination. Un seul niveau de modification est prévu, la figure ("segment"), représentant un ensemble de primitives de tracé. La dénomination ("naming") peut par contre s'adresser à une primitive (ou plusieurs) d'une figure.

A2.121 - Les primitives de définition de figures

CREATE_SEGMENT (segment_name)

La création d'une figure répond aux règles suivantes :

- il n'y a pas de figure en cours de création (pas d'emboîtement de niveaux),
- il n'existe pas déjà de figure portant le nom passé en paramètre,
- tous les paramètres de visualisation sont figés et ne peuvent être modifiés au cours de la création de la figure,
- il existe deux types de figures :
  - . les figures permanentes ("retained"), dont on conserve la liste des primitives qui les composent,
  - . les figures temporaires ("non retained segments"), qui sont affichées immédiatement et ne sont pas conservées.

CLOSE_SEGMENT ()

La figure en cours de création est entièrement définie. Une autre figure peut être créée.

- destruction de figure

DELETE_SEGMENT (segment_name)
DELETE_ALL_SEGMENTS ()

La première primitive permet d'effacer une figure de type permanent. Tous les renseignements la concernant sont détruits. La deuxième primitive permet de détruire toutes les figures créées, quel que soit leur type.

- modification de nom d'un segment

RENAME_SEGMENT (segment_name, new_name)

Cette primitive permet l'utilisation de techniques de mémoire tampon lors de la mise à jour de dessins.

A2.122 - Les primitives de gestion des attributs de figure

- attribution de valeurs statiques

SET_CURRENT_SEGMENT_TYPE (segment_type)

- attribution de valeurs dynamiques

SET_VISIBILITY (segment_name, visibility)
SET_HIGHLIGHTING (sement_name, highlighting)
SET_DETECTABILITY (segment_name, detectability)
SET_IMAGE_TRANSLATE_2 (segment_name, dx, dy)
SET_IMAGE_TRANSLATE_3 (segment_name, dx, dy, dz)
SET_IMAGE_ROTATE_2 (segment_name, a)
SET_IMAGE_ROTATE_3 (segment_name, ax, ay, az)
SET_IMAGE_SCALE_2 (segment_name, sx, sy)
SET_IMAGE_SCALE_3 (segment_name, sx, sy, sz)

- contrôle des valeurs

Pour connaître la valeur actuelle de l'un des attributs, il suffit d'utiliser une primitive d'interrogation. La liste est obtenue en remplaçant dans toutes les primitives d'attribution le mot *SET* par le mot *INQUIRE*. On dispose de plus de la primitive *INQUIRE_NAMED_SEGMENT_TYPE (segment_name, segment_type)*.

- valeurs par défaut

SEGMENT_TYPE temporaire
VISIBILITY affichage immédiat
DETECTABILITY non sensible à la désignation
HIGHLIGHTING pas de mise en valeur
IMAGE_TRANSFORMATION transformation identique

A2.13 Les primitives de préparation à la visualisation

A2.131 - Visualisation de scènes à trois dimensions

- définition du point de vue

VIEW_REFERENCE_POINT (x, y, z)

- définition du plan de projection

VIEW_PLANE_NORMAL (dx, dy, dz)
VIEW_PLANE_DISTANCE (view_distance)

Le plan de projection est déterminé par la donnée d'une normale et de la distance par rapport au point de vue.

- projections

PARALLEL (dx, dy, dz)
PERSPECTIVE (dx, dy, dz)

- définition du cône de vision

VIEW_UP_3 (dx_up, dy_up, dz_up)
WINDOW (umin, umax, vmin, vmax)

Ces deux primitives permettent de définir, dans le plan de projection, une portion d'espace rectangulaire (dont les bords sont parallèles aux axes, ou dont l'inclinaison par rapport aux axes est donnée par un vecteur ("view up vector")).

VIEW_DEPTH (front_distance, back_distance)

Cette primitive permet de spécifier un cône de vision tronqué, délimité par deux plans parallèles au plan de projection.

- divers

NDC_SPACE_2 (width, height)

permet d'ajuster les transformations vers le système de coordonnées réel, au cas où l'on souhaite avoir des transformations à l'échelle exacte

VIEWPORT_2 (xmin, xmax, ymin, ymax)

définition d'une clôture

A2.132 Visualisation de scènes à deux dimensions

Ces primitives ne sont introduites que pour simplifier la tâche des programmeurs n'ayant à dessiner que dans le plan. Une vue de la scène est alors complètement définie par l'utilisation des quatre primitives suivantes :

VIEW_UP_2 (dx_up, dy_up)
WINDOW (xmin, xmax, ymin, ymax)
VIEWPORT_2 (xmin, xmax, ymin, ymax)
NDC_SPACE_2 (width, height)

A2.133 - Mise en page

Ces possibilités ne sont utilisables que lorsque la scène est prête à être visualisée. Elles correspondent à la mise en place d'éléments, les transformations géométriques telles que rotation, homothétie, translation pouvant être appliquées à toute figure de type permanent. Ces transformations sont spécifiées à l'aide des attributs de figure (voir A2.122).

A2.142 - La gestion des dispositifs logiques

- autorisation ou interdiction

ENABLE_DEVICE (device_class, device_num)
DISABLE_DEVICE (device_class, device_num)
DISABLE_CLASS (device_class)
DISABLE_ALL ()

- traitement des dispositifs à relevé périodique

READ_LOCATOR (locator_num, x, y)
READ_VALUATOR (valuator_num, value)

- traitement des dispositifs à interruption

AWAIT_EVENT (time, event_class, event_num)
FLUSH_DEVICE_EVENTS (event_class, event_num)
FLUSH_CLASS_EVENTS (event_class)
FLUSH_ALL_EVENTS ()

ASSOCIATE (event_class, event_num, sampled_class, sampled_num)
DISASSOCIATE (event_class, event_num, sampled_class, sampled_num)
DISASSOCIATE_DEVICE (device_class, device_num)
DISASSOCIATE_CLASS (device_class)
DISASSOCIATE_ALL ()

- lecture des réponses

GET_PICK_DATA (segment_name, pick_id)
GET_KEYBOARD_DATA (input_string, num_input)
GET_LOCATOR_DATA (locator_num, x, y)
GET_VALUATOR_DATA (valuator_num, value)

- définition de caractéristiques des dispositifs
  . réponse éventuelle à l'action de l'opérateur

SET_ECHO (device_class, device_num, on_off)
SET_ECHO_CLASS (device_class, on_off)
SET_ECHO_POSITION (device_class, device_num, echo_x, echo_y)
SET_ECHO_SURFACE (device_class, device_num, surface_num)
SET_LOCATOR (locator_num, loc_x, loc_y)
SET_VALUATOR (valuator_num, initial_value, low_value, high_value)
SET_KEYBOARD (keyboard_num, buffer-size)
SET_BUTTON (button_num, prompt_switch)
SET_BUTTON_ALL (prompt_switch)

A2.15 Les primitives de contrôle

Il s'agit d'un ensemble de primitives qui permettent de contrôler le fonctionnement du logiciel et son adaptation au matériel utilisé.

- initialisation et terminaison d'une session

INITIALIZE_CORE (level)
TERMINATE_CORE ()

- sélection de la console

INITIALIZE_VIEW_SURFACE (surface_name)
SELECT_VIEW_SURFACE (surface_name)
DESELECT_SURFACE_VIEW (surface_name)
TERMINATE_VIEW_SURFACE (surface_name)

- attribution de valeurs par défaut

SET_DEFAULT_ATTRIBUTES (attribute_array)

- contrôle de valeurs

INQUIRE_CURRENT_POSITION_2 (x, y)
INQUIRE_CURRENT_POSITION_3 (x, y, z)
INQUIRE_VIEWING_PARAMETERS (viewing_parameters_array)
INQUIRE_VIEWING_TRANSFORMATION (transformation)
INQUIRE_DEFAULT_ATTRIBUES (attribute_array)
INQUIRE_OUTPUT_STATUS_PARAMETERS (output_status_array)
INQUIRE_INPUT_STATUS_PARAMETERS (input_status_array)

INQUIRE_SEGMENT_SURFACES (segment_name, number_of_surfaces,view_surfaces_array,
INQUIRE_NUMBER_OF_SEGMENTS (number_of_segments)
INQUIRE_SEGMENT_NAMES (segment_name_list)
INQUIRE_OUTPUT_CAPABILITIES (surface_name, output_array)
INQUIRE_INPUT_CAPABILITIES (input_array)

- contrôle d'erreurs

REPORT_MOST_RECENT_ERROR (error_report)

- gestion de l'écran

NEW_FRAME ()
BEGIN_BATCH_OF_UPDATES ()
END_BATCH_OF_UPDATES ()

- contrôle du cône de vision

CLIP_WINDOW (on_off)
CLIP_PLANES (on_off)
COORDINATE_SYSTEM_TYPE

A2.16 Liens spéciaux avec le programme d'application

- transformations liées au système de description

MODELLING_TRANSFORMATION (transformation)
INQUIRE_MODELLING_TRANSFORMATION (transformation)

ces deux primitives permettent un gain d'efficacité, en combinant les matrices de préparation à la visualisation et les matrices du logiciel de description. Les transformations géométriques ne sont donc calculées qu'une fois.

- utilisation de possibilités non standard

ESCAPE (function_name, parameter_count, parameter_list)

Permettent d'utiliser des fonctions spécifiques à un dispositif donné.

Deux primitives permettent de spécifier des transformations dans l'espace, en faisant passer d'un cône dans l'espace utilisateur à un cône dans un espace normalisé :

NDC_SPACE_3 (width, height, depth)
VIEWPORT_3 (xmin, xmax, ymin, ymax, zmin, zmax)

A2.14 Les primitives de dialogue

A2.141 - Les dispositifs logiques

Il en existe cinq types :

- les dispositifs d'identification *(PICK)*,
- les dispositifs de lecture de chaînes de caractères *(KEYBOARD)*,
- les dispositifs de sélection d'action *(BUTTON)*
- les dispositifs de collecte de coordonnées *(LOCATOR)*
- les dispositifs de lecture de valeurs numériques *(VALUATOR)*

Chaque dispositif logique est identifié par un nom formé de deux parties :

- son type ("device class"),
- son numéro dans sa classe ("device number"),

Deux modes de fonctionnement sont prévus :

- certains dispositifs ne sont lus qu'une seule fois, avant retour au programme d'application (*PICK*, *KEYBOARD*, *BUTTON*). Ils ont un fonctionnement sur interruption.
- les autres dispositifs sont relevés régulièrement, à des intervalles de temps donnés (*LOCATOR*, *VALUATOR*).

A2.2 LES PRIMITIVES DU LOGICIEL AW2250

- définition de l'espace utilisateur

FIGURE (numfig, type, gx, gy, dx, dy)	permet de spécifier le type des coordonnées (absolu ou relatif) et la fenêtre utilisée par le programmeur. L'affichage se fait sur tout l'écran.

- tracé : l'appel d'une primitive provoque l'affichage immédiat sur l'écran

. *POINT (numfig, n, tabx, taby)*	affichage de n points
. *VECTEUR (numfig, mode, n, tabx, taby)*	mode permet de définir ligne brisée ou segments disjoints
. *TEXTE (numfig, n, chaîne, x, y)*	affichage de n caractères à partir de x, y
. *EFFACE (numfig)*	effacement d'une figure.

- dialogue

. entrée de données

CLAVIER (numfig, groupe, format, x, y)	affichage d'un message, avec des zone laissées en blanc, pour remplissage avec les primitives suivantes :
GREEL (numfig, groupe, numpar)	lecture d'un réel dans la zone *numpar* du message *groupe*
GCHAINE (numfig, groupe, numpar)	lecture d'une chaîne de caractères
. *ATTENTE (codinter, listinter)*	utilisation du clavier de fonctions. Les touches sont spécifiées par la liste de leurs numéros.
. *POINTEUR (numéro, x, y)*	collecte d'un couple de coordonnées grâce au photostyle

A2.3 LES PRIMITIVES DU LOGICIEL GRIGRI (PREMIERE VERSION)

- mise en page

. *FIGURE (numfig, type, gx, gy, dx, dy)*	définition d'une fenêtre
. *CLOTURE (numfig, gx, gy, dx, dy)*	définition d'un espace de travail sur l'écran

- affichage

. *AFFICHER (numident, mode, n, tabx, taby)*	affichage guidé par le *mode*, qui permet de différencier des points, des lignes brisées et des segments disjoints. *Numident* est une combinaison de la corrélation et du numéro de figure
. *EDENTIER (numfig, numligne, numzone, entier)*	affichage d'un entier. *Numligne* référence un message (voir dialogue) .
EDREEL (numfig, numligne, numzone, format, réel)	le *format* spécifie le format d'entrée du réel
. *EFFACER (numfig)*	si 0, tout l'écran

- dialogue

. *EDITER (numident, numligne, description, x, y)*	édition d'un message avec des zones de travail, remplies en sortie avec *edreel* et *edentie* en entrée en utilisant :
. *RVAL (numfig, numligne, numzone)*	lecture d'un réel
NVAL (numfig, numligne, numzone)	lecture d'un entier
CHAINE (numfig, numligne, numzone)	lecture d'une chaîne de caractères
. *IDENTIFIER (numfig)*	
. *POSITION (numfig, n, tabx, taby)*	lecture de *n* couples de coordonnées au maximum. Si n = 0, nombre a priori quelconque. Au retour, *n* contient le nombre de couples lus.
. *MENU (liste de fonctions)*	attente de la sélection d'une action affichage automatique des valeurs étiquetées.

A2.4 LES PRIMITIVES DU LOGICIEL GRIGRI (DEUXIEME VERSION)

- mise en page

FENETRE (nfen, igx, igy, sdx, sdy)
CLOTURE (nclot, igx, igy, sdx, sdy)

- tracé

POINTS (nfig, tablong, tabx, taby, tablegende)
TEXTES (nfig, tablong, tabtextes, tabx, taby)
MODE (nfig, ncorrel, nsection, descripteur)

- gestion de l'écran

AFFICHER (nfig, ncorrel, nfen, nclot)
EFFACER (nfig, ncorrel)

- messages

MESSAGE (nmess, descripteur, nclot, x, y)
EDENTIER (nmess, nzone, entier)
EDREEL (nmess, nzone, réel)
EDCHAINE (nmess, nzone, chaîne)
EFMESS (nmess)

- dialogue

RVAL (nb, nmess, tréel)
NVAL (nb, nmess, tentier)
CVAL (nb, nmess, tchaîne)
POSITION (nb, nfig, nfen, nclot)
IDENTIFIER (nb, nfig, tfig, tcorrel)
MENU (nb, liste, tretour)

BIBLIOGRAPHIE

BIBLIOGRAPHIE

[Alp 73] S.R. ALPERT
Graphics Software for BASIC
Computer Graphics, SIGGRAPH-ACM, vol 7, n°4, 1973

[Ani 77] Les systèmes informatiques d'aide à la production de films
Compte-rendu de la journée d'étude du 21 avril 1977, Paris

[AnL 74] J.P. ANDRE, A. LEDUC-LEBALLEUR
Système interactif pour la visualisation tridimensionnelle de fonctions de deux variables
Rapport de projet de 3ème année ENSIMAG, juin 1974

[App 67] A. APPEL
The Notion of Quantitative Invisibility and the Machine Rendering of Solids
Proceedings ACM National Conference, 1967

[Arc 72] M. ARCHELUTA
Hidden Surface Line Drawing Algorithm
Rapport UTEC-CSc- 72-121, Univ. Utah, juin 1972

[Are 74] J. ARESTEIN
Ronds, traits, points. Le dessin pour les enfants.
F. Nathan, Paris, 1974

[Arn 70] R. ARNHEIM
Visual Thinking
Faber and Faber Limited, Londres, 1970

[Art 77] Dossiers Arts Plastique n°1
L'ordinateur et les arts visuels
Centre de Recherche des Arts Plastiques, Paris I, Musée National d'art Moderne, 1977

[Aze 69] J. AZEMA
Grammaires de graphes - algorithmes d'analyse - Applications
Thèse de 3ème Cycle, Grenoble, mars 1975

[Bae 69] R.M. BAECKER
Interactive Computer-Mediated Animation
Ph D Thesis, MAC-TR-61, MIT, juin 1969

[BaR 74] R.E. BARNHILL, R.F. RIESENFELD
Computer Aided Geometric Design
Academic Press, Londres, 1974

[Ber 73] J. BERTIN
Sémiologie graphique
Les diagrammes - les réseaux - les cartes
GAUTHIERS-VILLARS, 1973

[Ber 75] M. BERTHOD
Une méthode syntaxique de reconnaissance de caractères manuscrits en temps réel avec apprentissage continu
Thèse de 3ème Cycle, Paris VI, mars 1975

[BeB 75] BERTOGLI, BUISSONET
Etude de diverses méthodes employées pour la réalisation de reconnaisseurs de caractères en temps réel
Projet de 3ème année ENSIMAG, Grenoble, juin 1975

[Bez 77] P. BEZIER
Essai de définition numérique des courbes et des surfaces expérimentales. Contribution à l'étude des propriétés des courbes et des surfaces paramétriques polynomiales à coefficients vectoriels
Thèse de Doctorat d'Etat, Paris VI, février 1977

[Bli 77] J.F. BLINN
Models of Light-Reflection for Computer Synthesized Pictures
Computer Graphics, SIGGRAPH ACM, vol 11 n°2, Summer 1977

[BlN 76] J.F. BLINN, M.E. NEWELL
Texture and Reflexion in Computer Generated Images
CACM vol 19 n°10, octobre 1976

[BØ 77] K. BØ
IGP - Intermediate Graphic Package
Interactive Graphic Package
Integrated Graphic Package
Exploratory Meeting on Computer Graphics, Londres, février 1977

[Bou 70] W.J. BOUKNIGHT
A Procedure for Generation of Three-Dimensional Half-tone Computer Graphics Representation
CACM, vol 13, n°9, septembre 1970

[Boy 75] A.R. BOYLE
The Present Status of Automated Cartography
Computer Graphics SIGGRAPH ACM, vol 9, n°1, Spring 1975

[Bra 74] I.C. BRAID
Designing with Volumes
Cantab Press, Cambridge, 1974

[Bra 75] I.C. BRAID
The Synthesis of Solids Bounded by Many Faces
CACM, vol 18, n°4, avril 1975

[Bre 65] J.E. BRESENHAM
Algorithm for Computer Control of a Digital Plotter
IBM System Journal, vol 4, N°1, 1965

[Bre 77] J. BRESENHAM
A Linear Algorithm for Incremental Digital Display of Circular Arc
CACM, vol 20, n°1, janvier 1977

[Bro 76] D.K. BROTZ
Intersecting Polyhedra with successive Planes
Computer and Graphics, vol 2, n°1, 1976

[Bru 76] J.M. BRUN
Le concept de filtrage progressif et l'élimination des parties cachées dans EUCLID
Rapport LIMSI, 1976

[Bur 77] W. BURTON
Représentation of Many-Sided Polygons and Polygonal Lines for Rapid Processing
CACM, vol 20, n°3, mars 1977

[BuW 76] N. BURTNICK, M. WEIN
Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame animation
CACM, vol 19, n°10, octobre 1976

[Cat 74] E.E. CATMULL
A Subdivision Algorithm for Computer Display of Curved Surfaces
Ph. D. Thesis, University of Utah, décembre 1974

[Car 76] F.D. CARLSON
Graphics Terminal Requirements for the 1970's
Computer, vol 9, n°8, août 1976

[Cha 74] J. CHAUCHE
Transducteurs et arborescences
Etude et réalisation de systèmes appliqués aux grammaires transformationnelles
Thèse de Doctorat d'Etat, Grenoble, 1974

[Cla 76] J.H. CLARK
Hierarchical Geometric Models for Visible Surface Algorithms
CACM vol 19, n°10, octobre 1976

[Cla 77] R. CLARK
Graphics Algorithms
Computer Graphics SIGGRAPH ACM, vol 11, n°1, Spring 1977

[Cle 69] E. CLEEMANN
Un macro-langage de programmation graphique
Thèse 3ème Cycle, Grenoble, mars 1969

[Coh 71] D. COHEN
On Linear Difference Curves
in "*Advanced Computer Graphics*", Plenum Press, 1971

[Col 73] A. COLMERAUER, H. KANOUI, P. ROUSSEL, R. PASERO
Un système de communication homme-machine en français
Groupe de Recherche en IA, UER de Luminy, Université d'Aix-Marseille, juin 1973

[Coo 67] S.A. COONS
Surfaces for Computer Aided Design of Space-forms
MIT MAC TR-41, juin 1967

[Cre 73] J.P. CRESTIN
L'utilisation des terminaux graphiques pour la mise au point des programmes de commande numérique
Compte-rendu du groupe de travail ADEPA
Journées graphiques AFCET-IRIA, Paris, décembre 1973

[CrL 76] J.P. CRESTIN, M. LUCAS
What Might Be Computer Graphics ?
Position Paper, Workshop IFIP "Towards a Methodology in Computer Graphics", Seillac, mai 1976

[Cro 77] F.C. CROW
Shadow Algorithms for Computer Graphics
Computer Graphics SIGGRAPH ACM, vol 11, n°2, Summer 1977

[Dal 77] P.L. DAHAN, P. LETUAN
Approche théorique d'une technique : perspectives et ombres calculées
Thèse de Docteur-Ingénieur, ENST, juin 1977

[Dav 74] B. DAVID
Aspect graphique d'un système conversationnel pour la conception architecturale assistée par ordinateur
Automatisme, n°4, avril 1974

[Dav 75] B. DAVID
Pour une généralisation des systèmes de CAO. Approche et Applications
Thèse de 3ème Cycle, Grenoble, octobre 1975

[Das 69] S. DAVIS
Computer Data Displays
Prentice Hall, 1969

[Dre 74] P.E. DRENICK
Computer Animated Algorithms
IEEE Transactions on Education, E. 17, mai 1974

[Drn 64] A. DRESDEN
Solid Analytical Geometry and Determinants
Dover Publications, New York, 1964

[Duc 74] A. DUCROT
Principes d'organisation et de réalisation du système graphique interactif GIPSY
Thèse 3ème Cycle, Lille, mai 1974

[Ear 77] R.A. EARNSHAW
Line-Tracking for Incremental and Raster Devices
Computer Graphics, SIGGRAPH ACM, vol 11, n°2, Summer 1977

[Eas 75] J.F. EASTMAN
An Efficient Scan Conversion and Hidden Surface Removal Algorithm
Computer and Graphics, vol 1, n°2-3, 1975

[Eck 77] R. ECKERT
Functional Aspects and Specification of Graphic Systems (part II)
Rapport, Technische Hochschule Darmstadt, mai 1977

[Enc 75] J. ENCARNACAO
Computer-Graphics : Programmierung und Anwendung von graphischen Systemen
R. OLDENBURG Verlag, Munich, 1975

[Eng 73] M.E. ENGELI
A Language for 3-D Graphics Applications
International Computing Symposium, Davos, 1973

[EPS 77] R. ECKERT, F.F. PRESTER, E.G. SCHLECHTENDAHL, P. WISSKIRCHEN
Functional Description of the Graphical Core System GKS as a Step Towards Standardisation
Report from FNI Working Group 5.9, first draft, juillet 1977

[Fol 76] J.D. FOLEY
Picture Naming and Modification : an Overview
SIGGRAPH-SIGPLAN-FIU Graphics Languages Symposium, Miami, avril 1976

[FoW 74] J.F. FOLEY, V.L. WALLACE
The Art of Natural Graphic Man-Machine Conversation
Proceedings of the IEEE, vol 62, n°4, avril 1974

[Fre 74] H. FREEMAN
Computer Processing of Line-Drawings Images
ACM Computing Surveys, vol 6, n°1, mars 1974

[Gai 75] G. GAILLAT
Une procédure statistique de décision avec apprentissage et son application à la reconnaissance des caractères manuscrits
Thèse de 3ème Cycle, Paris VI, mars 1975

[GaM 69] R. GALIMBERTI, U. MONTANARI
An Algorithm for Hidden Line Elimination
CACM, vol 12, n°4, avril 1969

[Gil 75] W. GILOI
On High-Level Programming Systems for Structured Display Programming
Computer Graphics SIGGRAPH ACM, vol 9, n°1, Spring 1975

[Gip 75] J. GIPS
Shape Grammars and their Uses : Artificial Perception, Shape Generation and Computer Aesthetics
Birkhaüser Verlag, Interdisciplinary Systems Research, Bâle, 1975

[Glo 70] M.B. GLOWES
Picture Syntax
in *Picture Language Machines*, S. KANEFF ed, 1970

[GoR 74] W.J. GORDON, R.F. RIESENFELD
BERNSTEIN-BEZIER Methods for the Computer Aided Design of Free-Forms Curves and Surfaces
JACM vol 2, n°2, avril 1974

[Gou 71] H. GOURAUD
Computer Display of Curved Surfaces
University of Utah, UTEC CSc 71 113, juin 1971

[Gra 76] M. GRAVE
Manipulation de surface en conception assistée par ordinateur
Rapport de Recherche IRIA LABORIA n°210, décembre 1976

[Gri 75] J.G. GRIFFITHS
A Data-Structure for the Elimination of Hidden Surfaces by Patch Subdivision
Computer Aided Design vol 7, n°3, juillet 1975

[GSP 77a] GSPC State-of-the-Art Subgroup
State-of-the-Art of Graphics Software Packages
Computer Graphics, SIGGRAPH ACM, vol 11, n°3, Fall 1977

[GSP 77b] GSPC core definition subgroup
First Report, draft 0, octobre 1976
First Report, draft 1, mars 1977
Second Report, juillet 1977

[GSP 77c] GSPC Interface subgroup
Interface Guidelines for Graphics Standards Development
Report, septembre 1976

[Gue 76] R.A. GUEDJ
Report on the IFIP W.G. 5.2. Workshop on "Methodology in Computer Graphics"
avant-projet, juillet 1976

[HaG 77] G. HAMLIN, C.W. GEAR
Raster-scan Hidden Surface Algorithm Techniques
Computer Graphics SIGGRAPH ACM, vol 11, n°2, Summer 1977

[HCY 67] A. HURWIRTZ, J.P. CITRON, J.B. YEATON
GRAF : graphical extensions to FORTRAN
Spring Joint Computer Conference, 1967

[Her 77] G.C. HERTLEIN
Computer Art for Computer People - A Syllabus
Computer Graphics SIGGRAPH ACM, vol 11, n°2, Summer 1977

[Hop 76] F.R.A. HOPGOOD
Is a Graphic Standard Possible ?
Position paper, Workshop "Methodology in Computer Graphics", Seillac, mai 1976

[HHK 77] T. HAGEN, P.J.W. TENHAGEN, P. KINT, H. NOOT
ILP Intermediate Language for Pictures
Preliminary Report, IW 68/77, Mathematical Centrum, Amsterdam, octobre 1977

[HuE 74] W.H. HUGGINS, D.R. ENTWISLE
Iconic Communication : an Annotaded Bibliography
The John Hopkins University Press, Londres, 1974

[IBM -] IBM System 360 Operating System
Graphic Subroutine Package (GSP) for FORTRAN IV, COBOL and PL/1
Form C27-6932

[IBM 76] N° spécial "Art et Ordinateur"
IBM Informatique n°13, 1976

[IRI 76] Groupe Graphique de l'IRIA
GIPSY, un système général pour la programmation graphique interactive
Congrès AFCET-TTI, Paris, novembre 1976

[ISO 77] Report of Exploratory Meeting on Computer Graphics
ISO/TC 97/SC5/Graphics n°11, Londres, février 1977

[JJN 76] J.F. JARVIS, C.N. JUDICE, W.H. NINKE
A Survey of Techniques for the Display of Continuous Tone Pictures on Bilevel Displays
Computer Graphics and Image Processing, n°5, 1976

[Jud 75] C.N. JUDICE
Display of Two-Dimensional Functions Using Gray-Scale Simulated on a Bilevel Display
Computer Graphics SIGGRAPH ACM, vol 9, n°1, Spring 1975

[Kul 68] H.E. KULSRUD
A General Purpose Graphic Language
CACM, vol 11, n°4, avril 1968

[Laf 76] B. de LA FAYOLLE
Analyse du français comme langage de commande dans un système de construction graphique
Thèse de 3ème Cycle, Grenoble, novembre 1976

[Lal -] Y. LALOUM
Titre non encore précisé

Thèse de 3ème Cycle, Grenoble, *à paraître*

[LaM 74] C. LAUGIER, F. MARTINEZ
Utilisation du 2250 à partir d'Algol W
Note technique n°T4, équipe graphique, mai 1974

[Lan 75] C.A. LANG
Achievements in Computer-Aided Design
CACM, vol 18, n°4, avril 1975

[Lau 76a] C. LAUGIER
Un système d'interprétation graphique de données - Application à l'illustration dynamique de programmes
Thèse de 3ème Cycle, Grenoble, octobre 1976

[Lau 76b] C. LAUGIER
Illustration dynamique de programmes à l'aide d'une console de visualisation
Congrès AFCET (TTI), Gif-sur-Yvette, novembre 1976

[Lec 70] O. LECARME
Contribution à l'étude des problèmes d'utilisation des terminaux graphiques. Un système de programmation graphique conversationnelle
Thèse de Doctorat d'Etat, Grenoble, septembre 1970

[Led 76] A. LEDUC-LEBALLEUR
Conception et réalisation d'un logiciel graphique sur le concept de console virtuelle
Rapport interne, équipe graphique, Grenoble, janvier 1976

[Led 77] A. LEDUC-LEBALLEUR
Conception et réalisation d'un logiciel graphique de base, indépendant de son contexte. Application au logiciel GRIGRI
Thèse de Docteur-Ingénieur, Grenoble, juin 1977

[LeK 73] C. LEDU, R. KONISKI
Algorithme d'effacement des lignes cachées (Galimberti - Montanari)
Projet de 3ème année ENSIMAG, Grenoble, juin 1973

[LEL -] A. LEDUC-LEBALLEUR, M. LUCAS
Etude de quelques problèmes de conception et de réalisation de logiciels graphiques indépendants de leur contexte d'exploitation
Rapport de Recherche, IMAG, *à paraître*

[Lem 76] M. LEMOINE
La standardisation : mythe ou réalité - Une expérience probante de standardisation d'un logiciel graphique
Congrès AFCET (TTI), Gif-sur-Yvette, novembre 1976

[LeP -] Ch. LEPROVOST, A. PONCET
Finite Element method for spectral modelling of tides
A paraître dans : International Journal for Numerical Methods in Engineering

[Ler 74] A. LERAY
Le logiciel de visualisation graphique GRAFOR
Note de travail, groupe graphique AFCET, mai 1974

[Lev 76] J. LEVIN
A Parametric Algorithm for Drawing Pictures of Solid Objects Composed of Quadric Surfaces
CACM, vol 19, n°10, octobre 1976

[LiZ 75] B. LISKOV, S.N. ZILLES
Specification Techniques for Data Abstraction
IEEE Transactions on Software Engineering, SE 1, n°1, mars 1975

[LLL 75] C. LAUGIER, A. LEDUC-LEBALLEUR, F. MARTINEZ
GRIGRI : logiciel de base pour l'utilisation des consoles de visualisation du CICG
Equipe graphique, note technique n°29, novembre 1975

[LLM 76] A. LEDUC-LEBALLEUR, M. LUCAS, F. MARTINEZ
GRIGRI : logiciel de base pour l'utilisation des consoles de visualisation du CICG
Equipe graphique, note technique n°49, décembre 1976

[LLM 77] A. LEDUC-LEBALLEUR, M. LUCAS, F. MARTINEZ
Conception et réalisation d'un logiciel graphique interactif indépendant du contexte d'utilisation. Le logiciel de base GRIGRI
Séminaire d'informatique, IMAG, juin 1977
A paraître dans la revue Bleue Informatique de l'AFCET

[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

[Lue 74] A.W. LUEHRMAN
An Elaboration of Some Thoughts on a Graphical Syntax in BASIC
Conference on Graphics in BASIC, octobre 1974

[Luc 72a] M. LUCAS
Computer Driven Animation on a Graphic Display
8ème Congrès DECUS-EUROPE, Strasbourg, septembre 1972

[Luc 72b] M. LUCAS
Production de dessins animés à l'aide d'un terminal graphique
Congrès AFCET, Grenoble, 6-9 novembre 1972

[Luc 73] M. LUCAS
L'exploitation et l'organisation des systèmes graphiques
Compte-rendu d'activité du groupe "GRAPHIQUE" de l'AFCET, journées AFCET-IRIA, décembre 1973

[Luc 74a] M. LUCAS
Technologie, programmation et utilisation des consoles de visualisation - Etat des recherches actuelles
Rapport DRME, IMAG, mai 1974

[Luc 74b] M. LUCAS
La programmation des consoles de visualisation : les techniques de base
Polycopié, ENSIMAG, décembre 1974

[Luc 75a] M. LUCAS
Principaux composants d'un système de production d'images animées
Rapport DRME n°1, juillet 1975

[Luc 75b] M. LUCAS
La conception interactive des dessins animés
38ème Journées de l'ASSPA, Genève, 25-26 septembre 1975

[Luc 76a] M. LUCAS
Les systèmes de visualisation graphique : état des recherches actuelles
Séminaire de Programmation, IMAG, mars 1975
Paru également dans Bulletin de l'IRIA, n°25, avril 1976

[Luc 76b] M. LUCAS
Evolution des matériels graphiques interactifs
Journée d'étude AFCET-IRIA, Paris, février 1976

[Luc 77a] M. LUCAS
Technologie des consoles de visualisation : présent et perspectives
Journée d'étude SEE, Gif-sur-Yvette, 10 février 1977
Paru dans Onde Electrique

[Luc 77b] M. LUCAS
La normalisation des logiciels graphiques interactifs
Le Courrier de la Normalisation, n°257, octobre 1977

[LuM 76] M. LUCAS, F. MARTINEZ
Elements pour un système de production d'images animées
Séminaire de Programmation, IMAG, mai 1976

[Mah 70] R. MAHL
Visible Surface Algorithms for Quadric Patches
University of Utah UTEC CSc 70 111, 1970

[Mar 73] J. MARTIN
Design of Man-Computer Dialogues
Prentice Hall, New Jersey, 1973

[Mar 75] A. MARROCO
Tracé automatique de courbes planes
Rapport de Recherche IRIA n°110, mars 1975

[Maz 75] F. MARTINEZ
Eléments d'un système d'animation d'objets en deux dimensions satisfaisant certaines contraintes mécaniques
Rapport DRME n°1, juillet 1975

[Maz 77a] F. MARTINEZ
Techniques de passage d'un dessin à un autre par déformations successives. Application à un système d'animation
Rapport de Recherche IMAG n°65, Grenoble, janvier 1977

[Maz 77b] F. MARTINEZ
Etude des problèmes de conception et de réalisation d'animation : le système SAFRAN
Thèse de 3ème Cycle, Grenoble, mai 1977

[Mea 73] J.A. MEADS
Basic Graphics Software
Computer Graphics, SIGGRAPH-ACM, vol 7, n°3, 1973

[MEC 74] Méthodes graphiques en mécanique
Compte-rendu des journées GAMI-ADEPA
Sous le patronnage AFCET/ICF, Paris, juin 1974

[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, ENSIMAG, juin 1977

[Mie 71] J.M. MIERMONT
Etude algébrique et programmation de la discrétisation de figures planes
Thèse de 3ème Cycle, Grenoble, juin 1971

[Mil 56] G.A. MILLER
The Magical Number Seven, Plus or Minus Two : Some Limits on our Capability for Processing Information
Psychol. Rev., vol 63, pp 81-97, mars 1956

[Moh 71] R. MOHR
Formalisation d'un outil pour la description d'images
RIRO, R-2, 1971

[Moh 73] R. MOHR
Modèle algébrique pour l'analyse syntaxique de figures
Thèse de 3ème Cycle, Nancy, 1973

[Mol 71] A. MOLES
Art et Ordinateur
CASTERMAN, Paris 1971

[MoL 76] P. MORVAN, M. LUCAS
Images et Ordinateur - Introduction à l'infographie interactive
LAROUSSE, Collection Sciences Humaines et Sociales, Série Informatique, septembre 1976

[Nar 69] R. NARASHIMAN
On the Description, Generation and Recognition of Classes of Pictures
in "Automatic Interpretation and Classification of Images", GRASSELLI ed, 1969

[NeS 73] W. NEWMAN, R.F. SPROULL
Principles of Interactive Computer Graphics
Mc. GRAW-HILL, New York, mai 1973

[Nes 74] W.M. NEWMAN, R.F. SPROULL
An Approach to Graphics System Design
Proceedings of the IEEE, vol 62, n°4, avril 1974

[New 73] C. NEWTON
Graphics in Medicine and Biology
AFIPS Conference Proceedings, vol 43, juin 1973

[NNS 72] M.E. NEWELL, R.G. NEWELL, T.L. SANCHA
A New Approach to the Shaded Picture Problem
Procedings ACM National Conference, 1972

[Not 73] R. NOTESTINE
Graphics Computer Aided Design in Aerospace
AFIPS Conference Proceedings, vol 43, juin 1973

[OBB 75] C.D. O'BRIEN, H.G. BOWN
IMAGE, a language for the interactive Manipulation of a Graphics Environment
Computer Graphics SIGGRAPH-ACM, vol 9, n°1, Spring 1975

[OdA 76] M. ODIER, C. ABOULKER, M. IAS VERGNAS, D. BOURDIN
Importance des structures hélicoïdales en biologie. Calcul et construction de quasi-hélices par itérations de deltaèdres
Revue de Biomathématique, 53, 1, 1976

[Odi 75] M. ODIER
Isoëdres quasi hélicoïdaux et symétries d'ordre 5
C.R. Acad. Sciences, Patis, tome 280, série D, février 1975

[Par 72] D.L. PARNAS
A Technique for Software Modules Specification
CACM, vol 15, n°5, mai 1972

[Pfa 70] J.L. PFALTZ
Web Grammars and Picture Description
Tech. Rep 70-138, Computer Science Center, Univ. Maryland, 1970

[Pfi 74] G. PFISTER
The Computer Control of Changing Pictures
Ph. D. Thesis, MAC TR-135, MIT, septembre 1974

[Pho 73] B.T. PHONG
Illumination for Computer Generated Images
University of Utah, Ph. D. Thesis, août 1973

[Pho 75] B.T. PHONG
Improved Rendition of Polygonal Models of Curved Surfaces
Proceedings of Second USA-JAPAN Computer Conference, 1975

[Pol 72] B.W. POLIACK
A Bibliography on Computer Graphics
Stanford University STAN-CS-72-306, août 1972

[Pon 74] A. PONCET
Ecriture d'un code d'éléments finis
Journées d'Etudes AFCET-GAMI - Ecole Centrale, Chatenay Malabris, décembre 1974

[Poo 76] U.W. POOCH
Computer Graphics : Interactive Technique, and Image-Proceesing 1970-1975 : a bibliography
COMPUTER, vol 9, n°8, août 1976

[Ram 76] G. RAM
Analysis of Images Specified by Graphlike Descriptions
Computer Graphics and Image Processing, n°5, 1976

[Ray 76] J. RAYMOND
LG : a Language for Analytic Geometry
CACM, vol 19, n°4, avril 1976

[Rib -] H. RIBEROL
Dessin industriel
Collection "Technique et Normalisation", DELAGRAVE

[Rie 73] R. RIESENFELD
Applications of B-spline Approximation to Geometric Problems of CAD
University of Utah, UTEC CSc 73, 126, mars 1973

[Riv 77] V. RIVERO
Une contribution à la conception architecturale assistée par ordinateur : le système SIGMA-ARCHI
Thèse de 3ème Cycle, Grenoble, juin 1977

[RoA 76] D.F. ROGERS, J.A. ADAMS
Mathematical Elements for Computer Graphics
Mc GRAW HILL, New York, 1976

[Rob 63] L.G. ROBERTS
Machine Perception of Three-Dimensional Solids
TR 315, MIT Lincoln Laboratory, mai 1963

[Rom 70] G.W. ROMNEY
Computer Assisted Assembly and Rendering of Solids
TR-4-20, Univ. of Utah, 1970

[Sab 76a] M.A. SABIN
Software Interfaces for Graphics
A Digestion of Some of the Litterature
Position Paper, Workshop "Towards a Methodology for Computer Graphics", Seillac, mai 1976

[Sab 76b] M.A. SABIN
A Method for Displaying the Intersection Curve of Two Quadric Surfaces
The Computer Journal, vol 19, n°4, novembre 1976

[Sim 71] J.P. SIMERAY
Les graphiques au service de l'entreprise
Editions Hommes et Techniques, 1971

[SJL 76] B. SELE, P. JALBERT, B. VAN CUTSEM, M. LUCAS, C. MOURIQUAND
Chromosomes Proximities on the Metaphase Plate
5th International Congress of Human Genetics, Mexico, 10-15 octobre 1976

[SJL -] B. SELE, P. JALBERT, B. VAN CUTSEM, M. LUCAS, C. MOURIQUAND
R. BOUCHEZ
Distribution of Human Chromosomes on the Metaphase Plate Using Banding Techniques
Human Genetics, n°39, septembre 1977

[Sha 67] A.C. SHAW
A Proposal Language for the Formal Description of Pictures
Tech. Report, G.S.G 28, Stanford Linear Accelerator Center, 1967

[Sha 70] A.C. SHAW
Parsing of Graph Representable Pictures
J. ACM, vol 17, n°3, juillet 1970

[Smi 71] D.N. SMITH
GPL/1- A PL/1 Extension for Computer Graphics
Spring Joint Computer Conference, 1971

L.B. SMITH
An Example of a Pragmatic Approach to Portable Interactive Graphics
Computer and Graphics, vol 1, n°1,

[SpT 74] R.F. SPROULL, E.L. THOMAS
A Network Graphics Protocol
Computer Graphics SIGGRAPH-ACM, vol 8, n°3, Fall 1974

[StC 75] M.B. STEPHENSON, H.N. CHRISTIANSEN
A Polyhedron Clipping and Capping Algorithm and a Display System for Three Dimensional Finite Element Models
Computer Graphics SIGGRAPH-ACM, vol 9, n°3, Fall 1975

[Str -] CL. STROBINO
Cours de perspective avec et sans emploi de la règle à calcul
Notes de cours, Atelier Ecole de Lausanne, ?

[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

[SuH 74] I.E. SUTHERLAND, G.W. HODGMAN
Reentrant Polygon Clipping
CACM, vol 17, n°1, janvier 1974

[Sut 76] D.C. SUTCLIFFE
An Algorithm for Drawing the Curve f(x,y) = o
The Computer Journal, vol 19, n°3, août 1976

[The 72] F. THERON
Sur le programme EUCLID : création, manipulation et visualisation de formes tridimensionnelles dans un langage géométrique à génération dynamique
Thèse de 3ème Cycle, Paris, 1972

[Tra 75] U. TRAMBACZ
Towards Device Independent Graphics Systems
Computer Graphics SIGGRAPH-ACM vol 9, n°1, Spring 1975

[VCV 77] J. VAN DEN BOS, L.C. CARUTHERS, A. VAN DAM
GPGS : A Device Independent General Purpose Graphic System for Stand-alone and Satellite Graphics
Computer Graphics SIGGRAPH-ACM, vol 11, n°2, Summer 1977

[War 69] J.E. WARNOCK
A Hidden Surface Algorithm for Computer Generated Half-Tone Pictures
University of Utah, TRA.15 Cs Department 1969

[Wat 70] G.S. WATKINS
A Real Time Visible Surface Algorithm
University of Utah, UTEC, CSs 70101, juin 1970

[WeA 77] K. WEILER, P. ATHERTON
Hidden Surface Removal Using Polygon Area Sorting
Computer Graphics, SIGGRAPH-ACM, vol 11, n°2, Summer 1977

[Wil 72] H. WILLIAMSON
Algorithm 420, Hidden Line Plotting Program
CACM, vol 15, n°2, février 1972

[Woo 73] P.A. WOODSFORD
The Design and Implementation of the GINO 3-D Graphics Software Package
Software, Practice and Experience, vol 1, n°4, octobre 1973

[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

ANNEXES

CLASSIFICATION PAR THEMES

Ouvrages généraux [Ber 73], [Drn 64], [Enc 75], [Mar 73], [MoL 76], [NeS 73], [RoA 76], [Sim 71].

Bibliographies [Ant 77], [Fre 74], [HuE 74], [Luc 74a], [Luc 76a], [Pol 72], [Poo 76], [SSS 74]

chapitre I

Eléments de base de la communication graphique [Arc 74], [Arn 70], [Ber 73], [Fre 74], [Mil 56]

Art et ordinateur [Ant 77], [Her 77], [IBM 76], [Mol 71]

Technologie [Car 76], [Das 69], [Luc 76b], [Luc 77a]

Généralités sur les logiciels graphiques :

Logiciels de description
- . Codages topologiques [Bur 77], [Fre 74],
- . Codages syntaxiques [Aze 75], [Gip 75], [Glo 70], [Mie 71], [Moh 71] [Moh 73], [Nar 69], [Pfa 70], [Sha 67], [Sha 7
- . Codages géométriques
  - deux dimensions [Ray 76] .
  - trois dimensions [Bra 74], [Bra 75], [Cla 76], [Duc 74] [Eng 73], [The 72]
  - surfaces analytiques [BaR 74], [Bez 77], [Coo 67], [GoR 7 [Gra 76], [Mah 70], [Rie 73]

Logiciels de visualisation
- . Algorithmes deux dimensions [Bre 65], [Bre 77], [Coh 71], [Ear 77], [JJN 76], [Jud 75], [Luc 74b], [Maz 77a], [Sut 76], [SuH 74]
- . Algorithmes trois dimensions [Bro 76], [Lev 76], [Mar 75], [Sab 7

Logiciels de base [GSP 77b], [Led 77], [LLM 76], [LLM 77]

Reconnaisseurs de symboles [Ber 75], [BeB 75], [Gai 75]

chapitre II

Propositions générales [Gil 75], [GSP 77b], [Gue 76], [Mea 73], [NeS 74]

Dialogue [Fol 76], [FoW 74], [Luc 74b], [Tra 75]

Extensions syntaxiques [Alp 73], [HCY 67], [Luc 74], [Smi 71]

Logiciels généraux [Cle 69], [IBM -], [IRI 76], [Kul 68], [LEC 70], [Ler 74], [OBB 75], [Sab 76a], [VCV 77], [Woo 73]

GRIGRI [Led 76], [Led 77], [LeL --], [LLL 75], [LLM76], [LLM 77]

Normalisation [BØ 77], [Cla 77], [CrL 76], [Eck 77], [EPS 77], [GSP 77a], [GSP 77b], [GSP 77c], [Gue 76], [Hop 76], [HTK 77], [ISO 77], [Luc 77b]

Réseaux [SpT 74]

chapitre III

Généralités [SSS 74], [Cla 76]

Techniques de calcul [Dxn 64], [RoA 76]

Algorithme de GALIMBERTI - MONTANARI [App 67], [Bru 76], [Eng 73], [GaM 69], [LeK 73], [Lou 70], [Rob 63], [The 72],

Algorithme de WARNOCK [War 69], [WeA 77]

Algorithme de WATKINS [Arc 72], [Bou 70], [Bro 76], [Eas 75], [Rom 70], [Wat 70]

Algorithme de NEWELL SANCHA [HaG 77], [MeM 77], [NNS 72], [WeA 77]

Coloriage des faces [Bli 77], [BlN 76], [Gou 71], [NNS 72], [Pho 73], [Pho 75

Ombres Portées [Gro 77], [DaL 77]

Surfaces analytiques [Cat 74], [Gri 75], [Mah 70], [Lev 76]

chapitre IV

Généralités [Boy 75], [Cre 73], [Lau 75], [MEC 74], [New 73], [Not 73]

Animation

- Illustration de programmes [Dre 74], [Lau 76a], [Lau 76b], [Smi 71]
- Dessin animé [ANI 77], [Bae 69], [BuW 76], [Luc 72a], [Luc 72b], [Luc 75a], [Luc 75b], [LuM 76], [Maz 75], [Maz 77a], [Maz 77b], [Pfi 74]

Fonctions de 2 variables [AnL 74], [Wil 72], [Wri 73]