Vers une approche systématique de la synthèse d'image : aspects logiciel et matériel

Francis Martinez

THESE

présentée à

l'Institut National Polytechnique de Grenoble

pour obtenir le grade de
DOCTEUR D'ETAT ES SCIENCES

par

Francis MARTINEZ

VERS UNE APPROCHE SYSTEMATIQUE DE LA SYNTHESE D'IMAGE

ASPECTS LOGICIEL ET MATERIEL

Thèse soutenue le 3 novembre 1982 devant la commission d'examen

G. VEILLON	Président
G. ALLAIN V. CORDONNIER J.C. LATOMBE M. LUCAS J. MERMET	Examinateurs

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1979-1980

Président : M. Philippe TRAYNARD

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

PROFESSEURS DES UNIVERSITES

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

.../...

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

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

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

Personnalités habilitées à diriger des travaux de recherche (décision du Conseil Scientifique) E.N.S.E.E.G.

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

.../...

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

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

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

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

MM. BOREL Joseph
CHEHIKIAN Alain
VIKTOROVITCH Pierre

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

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

E.N.S.H.G.

M. DELHAYE Jean-Marc

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

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

REMERCIEMENTS

Je tiens à exprimer à Monsieur G. VEILLON, professeur à l'Institut National Polytechnique de Grenoble, l'honneur qu'il me fait en acceptant de présider ce jury. Qu'il me soit permis de profiter de cette occasion pour le remercier également de son intérêt et de sa présence lors des étapes importantes de mes études, notamment à mes débuts dans le domaine des techniques graphiques et lors de ma thèse de troisième cycle.

Monsieur J. MERMET, maître de recherche au CNRS, trouvera ici l'expression de ma gratitude pour avoir dirigé cette thèse qui n'était pas directement liée à son domaine principal d'activité. J'ai bien conscience que nos résultats lui doivent beaucoup et qu'une grande partie de nos réalisations n'auraient pu voir le jour sans son aide et sa participation efficace.

Que Monsieur M. LUCAS, professeur à l'Université de Nantes, me permette de faire état de la grande dette que j'ai contracté envers lui. S'il est clair que, lors de sa présence à Grenoble, son influence a été prédominante dans l'orientation de mes recherches, je me dois de préciser que son départ n'a, en rien, entamé cette influence et que son expérience et sa clairvoyance sont toujours une source d'enseignements pour moi. Je n'omettrai pas de le remercier également pour la célérité avec laquelle il a "épluché" les diverses versions du manuscrit, et pour ses critiques constructives qui ont largement contribué à améliorer le texte initial. Son intérêt pour mes travaux est pour moi un encouragement constant.

Monsieur V. CORDONNIER, professeur à l'Université de Lille, a bien voulu lire et critiquer cette thèse. Qu'il soit remercié pour avoir assumé cette tâche ingrate avec compétence et efficacité, et qu'il veuille bien me pardonner pour lui avoir infligé cette punition pendant la période des vacances.

Je tiens à remercier Monsieur G. ALLAIN, de la société Thomson LMT, d'avoir accepté, pour la deuxième fois, de participer au jury d'une thèse de l'équipe. L'intérêt amical qu'il témoigne ainsi à nos travaux ne fait que conforter les espoirs que je fonde sur une collaboration plus étroite entre nos deux équipes.

Monsieur J.C. LATOMBE, professeur à l'Institut National Polytechnique de Grenoble, est sensé jouer dans ce jury le rôle du "candide". Je tiens donc à le remercier pour avoir spontanément accepté cette étiquette, mais je suis intimement convaincu que ses connaissances et sa compétence le rendent tout à fait apte à juger ce travail. Je tiens également à lui exprimer ma reconnaissance, ainsi qu'à son équipe, pour l'intérêt porté à nos réalisations et pour le climat de confiance qui a toujours entouré nos relations.

Ces remerciements ne seraient pas complets si je ne tentais pas de louer, à leur juste valeur, les mérites des membres de l'équipe de recherche sur la Communication Graphique. Il m'est donné, ici, une occasion très formelle de remercier Ph. BOULLE, F.NUNES FERREIRA, M.T. SARRASIN, qui tous ont apporté les pierres indispensables à la construction et à la solidité de l'ensemble. Les conditions de travail, souvent précaires et difficiles, n'ont jamais entamé leur enthousiasme et leur dynamisme et j'ai plaisir à dire que cette thèse est aussi la leur.

J'associerai dans ces remerciements l'ensemble du personnel du Service Electronique et notamment R. BOUTTAZ pour l'aide substantielle et les conseils qu'il nous a prodigués tout au long de la phase de conception du prototype HELIOS. Ma gratitude ira plus particulièrement à J.P. SERPAGGI pour le sérieux, la compétence et la bonne volonté qu'il a su mettre dans le cablage des cartes et leur montage final.

J'adresserai une part non négligeable de ma reconnaissance à mes condisciples, contaminés comme moi par la synthèse d'image, notamment Messieurs M. GRAVE et P. JANCENE de l'INRIA, M. GANGNET de l'Ecole des Mines de Saint-Etienne, M. MERIAUX de l'Université de Lille, pour leur sympathie et les échanges fructueux qui ont toujours résulté de nos réunions de travail.

Pour être sûr de n'oublier personne, je voudrais remercier l'ensemble du personnel du Laboratoire dans lequel j'ai toujours trouvé l'aide et la compréhension dont j'avais besoin. Je mentionnerai, plus particulièrement, les membres de l'Equipe Méthodologie de la C.A.O. pour avoir patiemment subi mes humeurs pendants la période de gestation de cette thèse, et j'adresserai un grand merci à E. TOURNIER qui, pendant cette même période, a spontanément pris à sa charge une grande partie de notre travail commun d'enseignement.

Je voudrais enfin que le lecteur apprécie le soin apporté dans la réalisation matérielle de ce document par le service de reprographie de l'IMAG et notamment la qualité des photographies, en couleur et en noir et blanc, qui l'accompagnent.

Si une thèse est en général le fruit de toute une équipe, je suis particulièrement fier de dire que celle-ci est également une affaire de famille. Comment ne pas faire une place dans ces pages à ma femme qui a assumé, à elle seule, la frappe des 521.912 caractères qui composent ce livre. Je ne serai pas complet si je ne parlais pas des heures de découragement où j'ai toujours trouvé chez elle le réconfort et l'aide nécessaire. Je crains également que ma fille ne me pardonne pas de passer sous silence sa participation à la mise en place des figures et au classement des références bibliographiques. Je m'acquitte d'autant plus volontiers de cette formalité que son offre était sincère et désintéressée.

TABLE DES MATIERES

PREMIERE PARTIE
VERS UNE APPROCHE SYSTEMATIQUE DE LA SYNTHESE D'IMAGE

CHAPITRE II
Aspect technique
Le point de vue du concepteur

CHAPITRE III
Etat de l'art et exemples de réalisations

DEUXIEME PARTIE
UN EXEMPLE DE REALISATION

CHAPITRE IV
Présentation générale du système

CHAPITRE V
Aspect matériel
Le synthétiseur cablé HELIOS

CHAPITRE VI
Aspect logiciel
Les synthétiseurs programmables : CLOVIS

CHAPITRE VII
Aspect fonctionnel
Les opérateurs de description et de visualisation

TABLE DES PHOTOGRAPHIES COULEUR

PREMIERE PARTIE

VERS UNE APPROCHE SYTEMATIQUE

DE LA SYNTHESE D'IMAGE

INTRODUCTION

La puissance d'évocation de l'image et les différents niveaux de perception dont elle peut faire l'objet en font incontestablement le moyen de communication le plus naturel et le plus efficace. Chacun peut l'analyser à sa guise, pour en dégager une impression d'ensemble, ou au contraire s'attarder sur des détails particuliers afin d'en extraire des informations précises.

Alors que l'analyse d'un texte demande au lecteur une connaissance approfondie du vocabulaire et des règles grammaticales utilisées, la compréhension d'une image ne requiert aucune connaissance "consciente", aucun apprentissage préalable. Ainsi, la finalité de toute image, quelle qu'elle soit, est d'apporter à l'observateur un ensemble de références visuelles, susceptibles de stimuler la compréhension de la scène représentée. Ces références visuelles obéissent aux lois régissant les phénomènes de la perception visuelle dans le monde réel, lois auxquelles les mécanismes d'analyse du cerveau humain sont assujettis.

L'avènement de l'image dans l'univers de l'informatique et l'intérêt toujours croissant qui lui est témoigné, découlent précisément de la grande quantité d'informations qu'elle transmet sous une forme synthétique ou qu'elle suggère, sans nécessiter de la part de l'observateur aucune connaissance ni "culture" particulière.

Les études menées au cours des vingt ans d'age de la synthèse d'image à l'aide d'un ordinateur ont eu pour objectif constant l'accroissement de la quantité d'information traitée. Les grandes étapes de cette progression ont concerné successivement: le dessin au trait d'objets bi-dimensionnels puis tri-dimensionnels pour lesquels les techniques d'élimination des parties cachées ont contribué à lever bien des ambiguités de perception. Plus

récemment le domaine des informations "visuelles" relatives à la couleur et à la "texture" des objets ainsi qu'aux phénomènes d'éclairage, a été abordé pour apporter à l'observateur de nouvelles références à travers une présentation plus "réaliste".

Les progrès réalisés parallèlement dans le domaine du matériel, dûs notamment à la grande densité d'intégration atteinte et à l'avènement des micro-processeurs, ont permis que les rêves d'hier deviennent les réalités d'aujourd'hui.

Cette mutation des techniques a eu, des répercussions considérables dans le domaine de la synthèse d'image, compte-tenu de la complexité des calculs nécessaires et de la masse généralement très importante des éléments concernés.

L'optimisme relatif de ce bref tableau, ne doit cependant pas abuser le lecteur, car loin de croire que tous les problèmes sont résolus, nous pensons pour notre part que la synthèse d'image par ordinateur se trouve à un tournant capital de son développement. En effet, si les images spectaculaires publiées par la littérature prouvent que la plupart des problèmes techniques ont trouvé une solution, une étude plus approfondie montre qu'il s'agit dans la plupart des cas d'une solution ponctuelle adaptée au cas particulier de l'image à présenter, et qu'aucun concept généralisable ne s'en dégage. Nous n'en voulons pour preuve que la grande diversité dont témoigne la littérature actuelle et les difficultés évidentes des tentatives de normalisation.

A l'heure où l'on ne sait plus distinguer ce qui est logiciel de ce qui est matériel, ce qui est "graphique" de ce qui est synthèse d'image, nous pensons que la période actuelle sera celle de la mise en valeur des concepts généraux qui sont à la base de toute production visuelle à l'aide de processeurs cablés ou programmés.

Cette thèse rapporte la contribution dans ce domaine de l'Equipe de Communication Graphique du laboratoire d'Informatique et de Mathématiques

Appliquées de Grenoble. Les différentes expériences réalisées au cours des quatre dernières années tant du point de vue logiciel que matériel, sont exposées, ainsi que les réflexions qu'elles nous ont suscité. Ces travaux ont été menés dans le cadre de deux thèses de Docteur-Ingénieur (Bou80), (Fer81); d'un mémoire d'ingénieur du Conservatoire National des Arts et Métiers (Sar82), ainsi que de nombreux projets d'élèves ingénieurs et d'étudiants du D.E.A (Led79), (AQT80), (BaG81), (BaH81), (Tar81), (Pay82), (DeT82).

Cet ouvrage, dans sa première partie, présente l'approche systématique qui a guidé nos travaux jusqu'aux réalisations effectives qui font, quant à elles, l'objet de la deuxième partie. Le chapitre I est consacré à la mise en valeur des concepts et des notions sur lesquels s'appuie notre approche de la synthèse d'image. Après avoir présenté un bref historique permettant de motiver notre étude, nous proposons de distinguer les entités de base manipulées, ainsi que les grandes étapes nécessaires. Cette présentation expose le point de vue de l'utilisateur.

Le chapitre II s'attache plus au point de vue du concepteur d'un système général de synthèse d'image. Les problèmes relatifs à l'organisation des "synthétiseurs" et au choix des techniques de réalisation sont évoqués, et des solutions sont proposées. Nous terminons en étudiant l'incidence des objectifs visés sur les critères de choix entre les différentes approches possibles.

Le chapitre III présente, à la lumière des concepts de base et des différentes solutions techniques évoqués aux chapitres précédents, l'état de l'art dans le domaine de la communication graphique, et de la synthèse d'image. Cette revue permet de situer nos travaux par rapport aux réalisations étudiées et permet également de caractériser bon nombre de systèmes matériels et logiciels, du point de vue des processus utilisés et des architectures adoptées.

Le chapitre IV constitue la présentation générale de notre expérience personnelle qui fait l'objet de la deuxième partie de cette thèse. Après

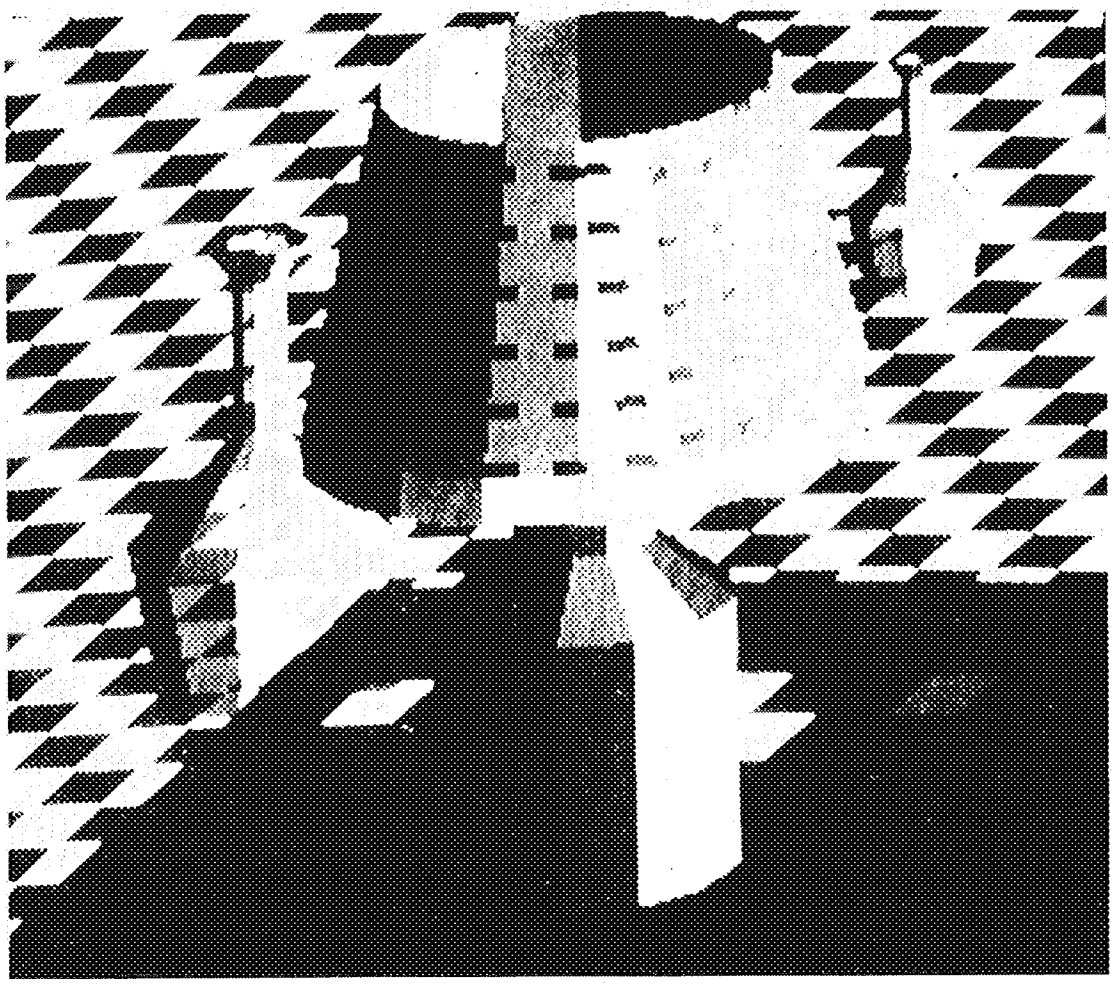
avoir justifié les choix qui nous ont conduits vers cette réalisation, l'architecture générale du système est évoquée ainsi que les éléments de base proposés aux utilisateurs. La présentation du matériel et des divers postes de travail utilisés fait également l'objet de ce chapitre.

Le chapitre V fait état de notre expérience dans le domaine de la conception de matériel de visualisation, à travers la construction du prototype HELIOS, adapté à la synthèse d'images réalistes. Après avoir exposé les originalités de ce synthétiseur cablé ainsi que son architecture générale, les différents processeurs de visualisation intégrés sont examinés tour à tour. L'étude des principaux problèmes de construction rencontrés, et des solutions adoptées termine le point de vue matériel.

Le chapitre VI s'attache, en revanche, au point de vue logiciel et plus particulièrement aux divers composants développés dans le cadre du projet d'étude CLOVIS. Outre l'intérêt apporté par l'utilisation d'une structure arborescente au niveau du logiciel principal, ce chapitre présente une réalisation permettant la prise en charge totale du processus de visualisation par le système. Les interfaces entre les différents niveaux de logiciel et l'application sont évoqués ainsi que la gestion des structures de données graphiques.

Le chapitre VII présente, à travers quelques exemples de processus de visualisation et de description, les performances et les résultats de l'ensemble du système. Une nouvelle approche du problème d'élimination des parties cachées est également proposée ainsi qu'une réalisation du remplissage des taches polygonales utilisant trois processeurs en parallèle. En ce qui concerne la description, l'accent est mis plus particulièrement sur le réalisme des images produites et sur les possibilités de traitement en temps réel.

Il va de soi que cet ouvrage n'est qu'une contribution à l'étude des problèmes liés à la synthèse d'image, et nous présenterons en guise de conclusion les nouvelles voies de recherche ouvertes autour des thèmes abordés.



Chapitre I

Aspect conceptuel

Point de vue de l'utilisateur

Ce chapitre est consacré à la mise en valeur des concepts qui nous semblent être à la base de la synthèse d'image à l'aide d'un ordinateur. Afin de dégager cette étude de l'influence d'un acquis issu de l'évolution progressive des techniques, nous considérons un système de synthèse comme une entité globale, sans contraintes syntaxiques, sans séparation préalable entre logiciel et matériel, sans préjugé aucun des techniques passées ou à venir. Seul le point de vue de l'utilisateur, à qui ces concepts sont destinés, est pris en compte.

Nous débuterons cette présentation par un historique très schématisé de l'évolution des techniques de base de la synthèse d'image. Cette revue n'est pas destinée à faire la liste des travaux effectués, mais plutôt de faire apparaître les tendances et les raisons qui nous ont conduits à adopter l'approche présentée dans cette thèse.

1. Historique et état de l'art.

Si l'on schématise quelque peu les vingt ans d'âge de la synthèse d'image à l'aide d'un ordinateur, on peut considérer trois périodes successives : la découverte, l'apprentissage et actuellement le développement.

1.1. La découverte (1963, 1970)

Si l'année 1952 (Eve52) vit le premier tube à rayons cathodiques connecté à un calculateur, il fallut attendre 1963 pour voir apparaître le premier logiciel graphique digne de ce nom : SKETCHPAD développé par I.E Sutherland (Sut63). Cette même année, le premier algorithme d'élimination des parties cachées (Rob63) fut également publié. La découverte des notions

de base du dessin au trait allait se poursuivre, puisque en 1965 Brésenham (Bre65) publie un recueil d'algorithmes permettant de dessiner sur un traceur digital. Mais il fallut attendre 1967 pour qu'un réel "démarrage" puisse être observé. Cette année là vit en effet les premières présentations de surfaces gauches (Coo67), la prise en compte des notions d'éclairage et d'ombres portées (App67), des mécanismes de description et de modélisation (Sha67) . Les points forts qui ont marqué la fin de cette période sont liés aux algorithmes d'élimination de parties cachées, Galimberti et Montanari (GaM69) pour le dessin au trait, Warnock (War69), Watkins (Wat70) pour les images. On vit en effet apparaître dans le même temps les premières consoles couleur à balayage de trame associées à une mémoire d'image.

1.2. L'apprentissage (1971,1978)

Cette période d'apprentissage s'est caractérisée par deux tendances principales, l'une vers une amélioration et une extension des techniques de base, l'autre vers une réflexion en vue de faire émerger les concepts de base. Dans la première tendance bon nombre de travaux ont porté sur l'amélioration de la qualité visuelle des images. On peut citer dans cette catégorie la technique de "lissage" des surfaces proposée par Gouraud (Gou71), la modélisation des phénomènes d'éclairage (Pho75), la génération de textures (Bli76), de reliefs (Bli78), ainsi que les techniques d'anti-aliasing (Cro77). Au rayon des nouveautés on trouve des algorithmes d'élimination de parties cachées, adaptés aux nouvelles possibilités du matériel Newell, Newell et Sancha (NNS72), Atherton et Weiler (AtW77), des logiciels de description tels que EUCLID (The72) ou celui développé par Braid (Bra74) opérant par composition de volumes élémentaires de base. On peut citer également de nombreuses techniques d'approximation et de présentation de surfaces gauches (Mah72), (Cat74), (Lev76), et un grand nombre de logiciels graphiques 2D et 3D (Woo73), (VCV77).

Cette explosion et cette diversité des techniques suscita dans le même temps des travaux de réflexion visant à mettre en valeur les concepts de base des différentes opérations, bien souvent masqués par les artifices techniques. L'année 1974 vit deux essais dans ce sens à travers la

classification des algorithnmes d'élimination de parties cachées proposée par Sutherland, Sproull et Schumacker (SSS74), et l'approche de Newman et Sproull vers la définition d'un logiciel graphique de base indépendant du matériel (NeS74). Cette idée d'indépendance fut d'ailleurs l'un des principaux leitmotifs de cette fin de période puisqu'elle suscita successivement : la conception du logiciel GRIGRI (LLM76), (Luc77), LLM78), la proposition de norme du Graphic Standards Planning Committee de l'ACM (GSP77), et les premiers pas de GKS (EGK78). En ce qui concerne le matériel, les progrès réalisés parallèlement au niveau de l'intégration des circuits LSI et des performances des micro-processeurs, ont influé directement sur le développement des consoles couleur à balayage de trame. Cette influence s'est manifestée dans trois directions:

- augmentation des capacités de mémorisation et des vitesses de traitement,

- utilisation de processeurs dévolus à des tâches spécifiques telles que le remplissage de contour (Eng75) ou encore des objets particuliers tels que les sphères (Sta78),

- développement de contrôleurs vidéo complets intégrés dans un boitier unique (Mat78) assurant la gestion de la mémoire d'image, la génération de vecteurs, de caractères et la production des signaux de synchronisation nécessaires pour piloter le moniteur TV.

1.3. Le développement (depuis 1979)

En dépit du nombre toujours croissant de publications concernant la synthèse d' images, fort peu de "nouveaux concepts" voient le jour à l'heure actuelle. La tendance la plus marquée est la recherche de techniques logicielles ou matérielles permettant d'améliorer les performances (vitesse, qualité et complexité d'image) et ce pour des applications de plus en plus spécifiques.Sans donner ici des références bibliographiques que l'on trouvera dans les chapitres suivants, on peut citer quelques uns des pôles d'intérêts actuels.

* Etude d'architectures parallèles cablées ou à base de VLSI,
* Techniques d'anti-aliasing cablées ou micro-programmées,

* Techniques de transformation d'images (rotation, perpective, etc.)
* Techniques de synthèse adaptées à des objets spécifiques tels que polyèdres convexes, équation à deux variables, sphères, etc.,
* Amélioration des modèles de réflexion en vue d'augmenter le réalisme,
* Adaptation des techniques à des applications particulières.

D'un autre coté, quelques études théoriques sont également proposées sur des points particuliers tels que:

* Les transformations et la perception des espaces de couleur,
* La classification et la comparaison d'algorithmes d'élimination de parties cachées,
* La classification et la comparaison d'algorithmes élémentaires pour la synthèse d'image.

1.4. Conclusion

Ce bref historique, bien que très schématisé, permet de tirer quelques conclusions.

Tout d'abord du point de vue des techniques, il est clair que l'on se dirige vers une spécialisation de plus en plus marquée vers un type d'application ou un type de matériel. Cette tendance se justifie par les limites assez contraignantes des méthodes générales issues des deux premières périodes, et les nouvelles possibilités offertes par les progrès de la micro-électronique. Cette évolution est tout à fait prometteuse et illustre bien le caractère dynamique de ce domaine de l'informatique.

D'un autre côté, celui de la connaissance et de la "normalisation" des concepts de base des logiciels, il semble au contraire que fort peu de chemin ait été fait. Si de gros efforts ont été réalisés vers une unification des concepts au cours des années 1974 à 1979, les résultats obtenus sont aujourd'hui démentis par la diversité observée dans les travaux actuels. Afin d'illuster les raisons qui, a notre avis, ont provoqué cet état de fait, nous prendrons l'exemple caractéristique de la proposition de norme du groupe GSPC de l'ACM (GSP77) (voir étude détaillée en (Luc77)). Ce

travail, remarquable sous bien des aspects nous semble néanmoins résulter de quelques erreurs "stratégiques".

- Tout d'abord on peut constater qu'il s'agit d'une normalisation des primitives (aspect syntaxique et sémantique) alors même que les concepts de base ne sont pas définis.

- Les propositions ont été indiscutablement guidées par les possibilités du matériel disponible à l'époque. Elle sont aujourd'hui totalement inadaptées pour des terminaux très évolués possédant de nombreuses fonctions locales, ou manipulant des images réalistes par exemple.

- Les primitives concernaient initialement le dessin au trait, ce qui limitait considérablement leur intérêt . Cet "oubli" fut réparé en 1979 par un additif à la proposition de norme (GSP79) qui apparaît plus comme une remise en cause de la version initiale que comme une simple prise en compte de nouveaux éléments. Ceci montre clairement qu'il a suffi d'introduire quelques notions non prévues initialement pour que l'ensemble des concepts sous-jacents soit radicalement remis en cause. Ainsi le tracé dynamique à l'aide de primitives gérant le déplacement du faisceau (MOVE, DRAW) n'a plus aucune signification si l'on considère les taches.

Si l'on considère les développements récents de GKS (EEK80), un progrès indiscutable apparaît notamment grâce à la définition statique du tracé (disparition de MOVE et DRAW) et au concept de poste de travail (workstation) permettant une plus grande indépendance par rapport au matériel.

Si, à présent on considère la synthèse d'images réalistes nécessitant la manipulation de textures, le calcul d'éclairage, la prise en compte des ombres portées, les primitives proposées sont à nouveau insuffisantes, et bien des notions proposées deviennent caduques. En outre, les spécificités des terminaux actuels qui intégrent de plus en plus de fonctions spécifiques, de génération, de stockage, etc., sont inexploitées.

Ces constatations sont à l'origine des travaux présentés dans cette thèse. Il nous semble, en effet, plus cohérent d'aborder le problème par le haut, c'est à dire par la mise en valeur des concepts régissant la synthèse d'images au sens large, allant du dessin au trait aux images réalistes. Afin de ne pas être le reflet d'une époque, cette étude préalable sera indépendante de toute séparation entre logiciel et matériel, et de toute technique. Dans un second temps les performances requises et les contraintes réelles viendront forcer le choix vers l'une ou l'autre approche, et vers l'un ou l'autre type de matériel, sans pour cela remettre en cause les notions de base.

Les paragraphes qui suivent, observent cette démarche pour la présentation des concepts de base. Les implications techniques et la réalisation effective font l'objet des chapitres suivants.

2. Les concepts de base de la synthèse interactive d'image

2.1. Les différents types d'images

Par essence même, une image est destinée à apporter à l'observateur un ensemble de références visuelles permettant de stimuler les mécanismes de compréhension et d'analyse d'une scène donnée. Ces références visuelles doivent bien entendu obéir aux lois régissant les phénomènes de la perception visuelle dans le "monde réel", lois auxquelles les mécanismes de reconnaissance du cerveau humain sont assujettis.

Nous garderons le vocable scène pour désigner l'idée d'ensemble voulue ou perçue par le cerveau humain. Cette scène virtuelle voulue par le concepteur passera par des étapes de représentation intermédiaire avant de parvenir à l'état d'image puis à nouveau à l'état de scène dans l'esprit de l'observateur, comme l'illustre la figure 1.

Le terme "image synthétique" quant à lui désignera toute représentation visuelle obtenue à l'aide d'un ordinateur à partir des seules directives du concepteur, sans référence à des images réelles. Plusieurs

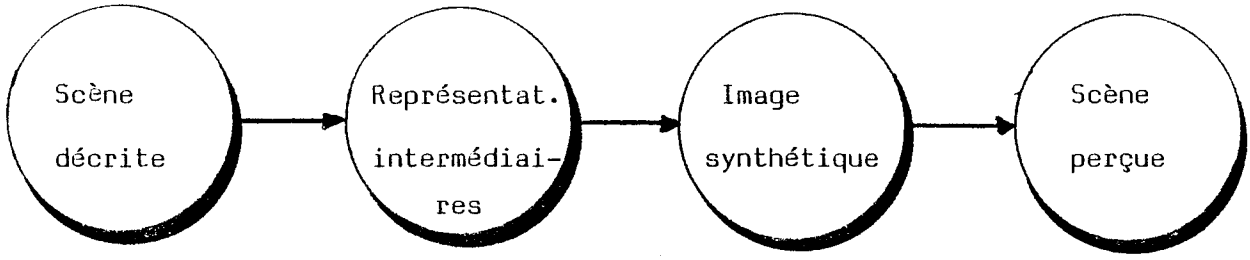


Figure I.1: Représentations intermédiaires des informations

types d'images peuvent être distingués, relatifs aux techniques picturales employées, c'est à dire aux éléments de base dont elles sont constituées: traits, points, taches, etc.

Une deuxième classification des images peut être opérée en considérant la nature des informations véhiculées:

- les images abstraites: il s'agit là d'images qui ne véhiculent aucune information tangible.
- les images symboliques telles que diagrammes, graphiques, schémas synoptiques qui expriment des informations quantitatives, topologiques, structurelles, etc.
- les images figuratives qui représentent à l'aide de dessins ou de croquis la forme et l'attitude des objets du monde réel, sous un aspect simplifié,
- les images réalistes qui s'attachent également à rendre l'aspect réel des objets.

La frontière entre ces diverses catégories est souvent subjective, en particulier pour les images figuratives et réalistes entre lesquelles divers degrés de réalisme peuvent être introduits.

2.2. Les interlocuteurs, leur rôle et leurs besoins

La synthèse interactive d'images met en jeu deux interlocuteurs principaux : le programme d'application (vocable regroupant l'application elle-même et l'ensemble des logiciels qu'elle utilise) et l'opérateur humain. Ce dialogue "visuel" se situe dans une boucle d'interaction un peu particulière compte tenu de la complexité des informations manipulées (figure 2). Chacun des deux interlocuteurs joue un rôle bien défini que l'on peut tenter d'approfondir quelque peu.

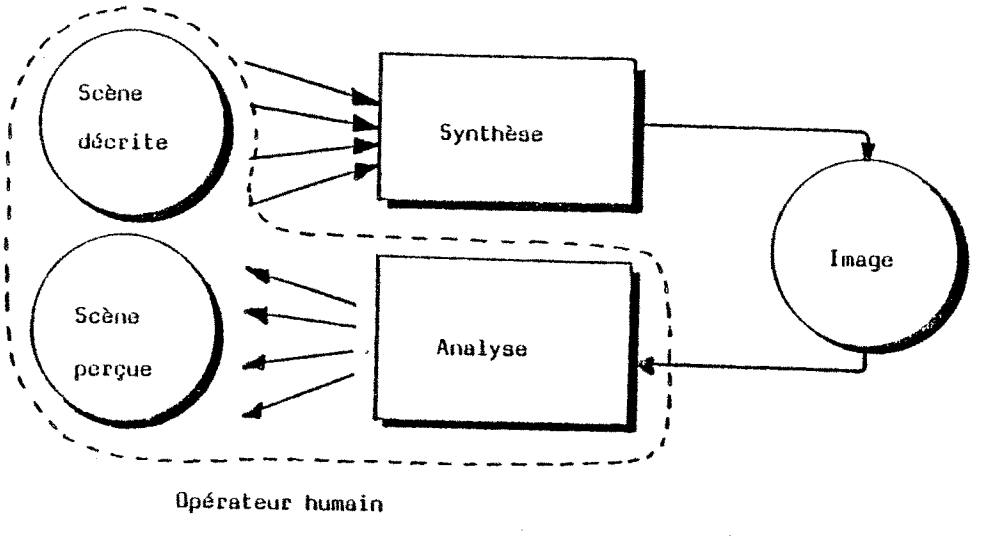


Figure I.2: Boucle d'interaction

Le programme d'application: Il est en général le maître du dialogue par la limitation du nombre d'actions proposées à l'opérateur dans un contexte donné. Son rôle principal est de synthétiser l'ensemble des informations dont il dispose afin de donner au cerveau de l'observateur, les éléments de référence suffisants pour stimuler la compréhension. Cette opération nécessite une connaissance des mécanismes du cerveau humain, ainsi que des phénomènes physiques de la vie réelle dans lesquels il puise ses références. Les principaux problèmes rencontrés au niveau des programmes de synthèse d'image découlent principalement de la dificulté à modéliser de manière approximative les phénomènes physiques régissant la perception visuelle.

L'opérateur: Son premier rôle est d'analyser les images présentées afin d'en extraire les informations qui y sont synthétisées. Les données graphiques fournies en retour par l'opérateur en vue de modification, adjonction, suppression sont en général décrites sous forme analytique à l'aide des dispositifs de communication classiques disponibles sur le site (tablette à numériser, photostyle, réticule, etc.). Il est clair que la principale difficulté est la "quantification" des divers paramètres à communiquer au programme d'application. Deux exemples illustrent parfaitement cette difficulté: la description de couleurs ou de rotations dans l'espace tri-dimensionnel.

Il incombe ainsi au programme d'application ou aux logiciels qui l'accompagnent la vocation supplémentaire de fournir à l'opérateur des outils de description "auto-quantifieurs" procédant par exemple par référence visuelle. Ainsi pour la description d'une couleur, le procédé le plus aisé consiste à présenter une palette dans laquelle l'opérateur désignera la nuance désirée (cf. VII.2.1.2).

Il peut, en outre, s'avérer extrêmement intéressant de décrire certaines informations sous forme synthétique, par exemple à l'aide d'images (réelles ou synthétiques) saisies au moyen de caméras numériques. Nous rejoignons ici l'ensemble des techniques propres à l'analyse d'image, qui peuvent prendre place dans la synthèse en tant qu'outils évolués de description .

2.3. Définition des entités de base

2.3.1. La notion de classe d'information

Dans l'approche ci-dessus, il apparaît que l'image elle même est bien moins significative que l'impression qu'elle suscite, et qu'elle ne représente qu'un support d'information privilégié permettant une transmission performante entre un émetteur (le moniteur T.V.) et un récepteur (l'oeil humain) : figure 3.

Néanmoins comme dans toute transmission, il faut tenir compte d'une certaine perte d'information aux niveaux de l'émission et de la réception. Une image synthétique doit, par conséquent, rassembler au moins les informations destinées à être extraites par le cerveau après analyse, mais en général un grand nombre d'informations présentes dans une image ne sont pas réellement perçues. Des travaux précédents (Mar77),(Mar80), nous ont conduit à partitionner en six classes indépendantes les informations véhiculées par une image:

I Identité ,

M Morphologie,

A Aspect ,

G Géométrie ,

E Eclairage ,

S Structure .

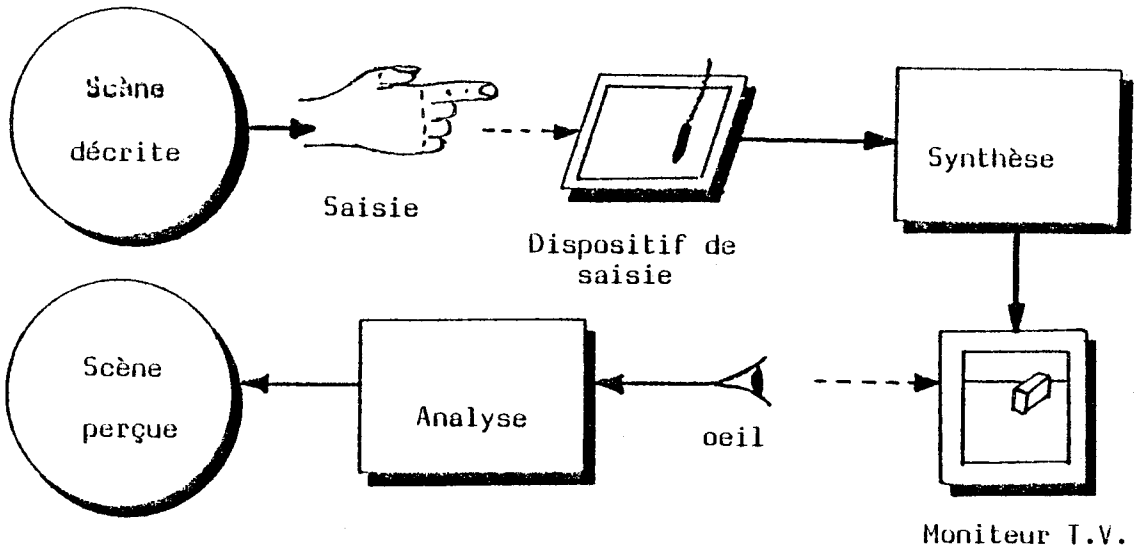


Figure I.3: Le cheminement des informations

L'identité (I) est la rubrique regroupant les informations relatives à la nomination des objets ou des ensembles d'objets représentés par une image.

La morphologie (M) exprime la forme intrinsèque de chaque objet, indépendamment de son attitude, de sa taille ou de son point de vue. Il s'agit d'une information prépondérante au niveau des mécanismes de reconnaissance des objets.

L'aspect (A), de la même manière, exprime l'apparence intrinsèque de chaque objet indépendamment des conditions d'éclairage. Il s'agit en fait d'une expression des propriétés de reflexion de la lumière, ou en d'autres termes les informations relatives au matériau dont l'objet est constitué (cf. III). On y trouvera principalement des notions telles que la couleur, la texture, la brillance, la transparence, etc.

La géométrie (G), rubrique complémentaire de la morphologie, regroupe les données permettant de situer les objets les uns par rapport aux autres, ainsi que les paramètres régissant la "prise de vue", les domaines de visibilité, etc. Ces informations expriment principalement des translations, des rotations, des projections, des découpages etc.

L'éclairage (E), indique pour la scène, la nature, le nombre et la position des sources lumineuses, ainsi que les conditions de visibilité; brume, fumées, effets atmosphériques,etc.

La structure (S) permet d'exprimer les relations liant les divers objets entre-eux. Ces relations peuvent être de nature:
Logique: appartenance, inclusion, etc.
Topologique: proximité, contact, etc.
Fonctionnelle au libre choix de l'utilisateur.

On peut noter que ces six classes d'informations peuvent être réparties en trois catégories distinctes, ayant au niveau de l'image finale des répercussions de nature fondamentalement différente.

Les informations topologiques dans lesquelles prennent place la morphologie et la géométrie, ont pour finalité la définition des domaines du plan de l'image correspondant aux divers objets visibles. Elles ne sont perçues par l'observateur qu'à travers une analyse qui peut être délicate ou ambigue.

Les informations visuelles (l'aspect et l'éclairage) ont au contraire une incidence en chacun des points de l'image pour lesquels elles doivent être obligatoirement définies. Leur nom est justifié par le fait que ce sont les seules informations directement perçues par l'observateur.

Pour leur part, les informations nominatives (Identité et structure) n'apparaissent pas au niveau de l'image. Elles sont déduites de celle-ci par l'observateur humain à travers ses facultés de reconnaissance et ses connaissances au sujet de la scène visualisée ou de son contexte. On peut d'ores et déjà noter la perte de ces informations au cours des opérations de synthèse, perte qui aura des répercussions importantes en particulier sur le degré d'interaction (cf II).

2.3.2. La notion de type d'information

Dans chacune des six classes d'informations de base il est indispensable de pouvoir considérer plusieurs types distincts. La notion de type correspond à la nécessité de distinguer des informations en fonction de critères:

- sémantiques, c'est à dire relatifs à la nature de l'information concernée. Dans le cas de la morphologie, on peut citer les types: cercle, vecteur, tache, caractère, polyèdre, etc.

- syntaxiques, c'est à dire relatifs à la modélisation des informations (Par exemple coordonnées cartésiennes ou polaires, entières ou réelles, compressées ou développées, etc.).

- structurels, permettant le regroupement d'entités de même nature: suite

de points, ensemble de taches, etc.

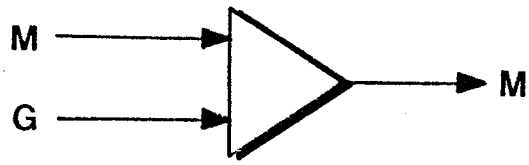
La notion de type déborde cependant des différentes classes d'informations, puisque au cours du processus ces informations vont être synthétisées pour former des classes "mixtes", dans lesquelles la distinction de type est également nécessaire. On peut ainsi considérer que le type d'une information implique la classe à laquelle elle appartient.

Les opérateurs élémentaires de synthèse, nécessaires pour passer des six types des informations initiales à celui de l'information finale comportant la couleur en chaque point, modifient les types d'informations qu'ils concernent.

On peut énoncer quelques régles caractéristiques correspondant aux cas des opérateurs les plus fréquents.

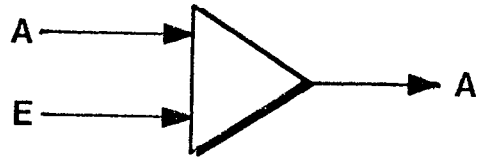
- Synthèse (M.G)

  Il s'agit généralement d'une simple transformation géométrique effectuée par une multiplication matricielle. Le résultat appartient à la classe (M).



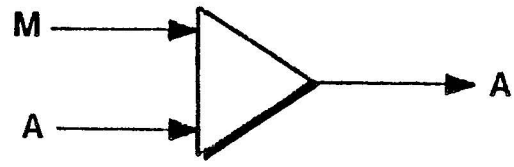
- Synthèse (A.E)

  Cet opérateur module les couleurs intrinsèques des objets, en fonction des conditions d'éclairage. Le résultat représente un aspect (A).



- Synthèse (M.A)

Il s'agit ici d'un opérateur permettant d'affecter l'aspect (A) à tous les points du domaine défini par (M). On trouve par exemple, dans ces opérateurs les opérations de remplissage de taches, la génération de diverses texture de traits, etc. Le résultat est l'aspect (A) des points de l'image. La morphologie est reconstituée par le cerveau de l'observateur.



2.3.3. La notion d'élément

La scène à synthétiser est composée d'objets. Nous garderons également cette appellation pour désigner les objets virtuels, tels qu'ils sont voulus ou perçus par le cerveau humain. Le terme élément désignera les différentes représentations internes de ces objets dans la machine. On peut, par extension, définir le type d'un élément en fonction:

* du type de ses informations d'identité I
* du type de ses informations morphologiques M
* du type de ses informations d'aspect A
* du type de ses informations géométriques G

La structure (S) et l'éclairage (E) concernent l'ensemble de la scène et non pas les objets en particulier.

Toutes ces rubriques d'information constituent les attributs associés aux éléments ou à la scène. Il se peut en outre que certains éléments soient définis à partir d'attributs préalablement synthétisés. Ainsi un élément pourrait être défini par:

* le type de l'attribut (I)
* le type de l'attribut (M.G.A) : (informations synthétisées).

En d'autres termes, ceci revient à dire qu'un opérateur de synthèse appliqué aux attributs d'un élément de type X, produit un nouvel élément de type Y, généralement différent du premier, puisque les attributs sont modifiés.

2.4. Les grandes étapes de la synthèse d'image

La synthèse d'une image à partir d'une scène fictive (ou réelle, mais innaccessible) nécessite, comme dans la réalité, trois étapes fondamentales:

- la description d'une maquette de la scène à synthétiser,
- la construction de cette maquette,
- la visualisation proprement dite.

2.4.1. La description de la maquette

Cette première étape est celle à travers laquelle le concepteur décrit les éléments constitutifs de la maquette, et les relations qui les unissent. Il s'agit en quelque sorte de la saisie des plans de la maquette. Chaque élément est décrit complètement c'est à dire que tous ses attributs (I,M,A,G) sont communiqués séparément à la machine. La structure et les paramètres d'éclairage sont décrits également (E,S), mais s'appliquent à l'ensemble de la maquette.

Les outils proposés à l'opérateur pour effectuer cette description doivent être indépendants, autant que possible, des problèmes relatifs à la construction et à la visualisation. Les caractéristiques des objets eux mêmes, doivent au contraire être exploitées afin de permettre une description aisée et adaptée.

2.4.2. La construction de la maquette

Cette construction, consiste à réaliser effectivement des modèles d'objets à partir des attributs fournis par la description. Pour fixer les idées nous prendrons comme exemple le cas d'une sphère décrite par la seule donnée de son rayon et de son centre, mais dont la construction effective peut nécessiter l'approximation de sa surface à l'aide de faces polygonales, de quadriques, etc.

De même que pour la réalisation de maquettes "réelles" cette opération est directement liée aux conditions dans lesquelles la maquette sera observée, si celles ci sont connues d'avance. Ainsi une maquette destinée à être observée à partir d'une direction donnée ne comporte généralement pas les "faces arrières" des objets présentés. En outre, la grande diversité des objets réels impose l'utilisation de techniques de construction variées. Dans le cas de la synthèse d'image à l'aide d'un ordinateur, cette diversité se traduit par l'utilisation d'éléments de types différents dans une même maquette. Les problèmes soulevés par cette étape de description seront évoqués au chapitre II.

2.4.3. La visualisation

Pour mettre en valeur et étudier les opérations de base de la visualisation, on peut assimiler celle-ci à un processus visuel de la vie quotidienne: le processus photographique. Deux opérations principales sont à considérer:

* la prise de vue elle*même,
* l'affichage de la vue obtenue.

Lors d'une photographie réelle, les informations visuelles issues de la maquette sont mémorisées sur la pellicule sous forme chimique, puis passent par des étapes intermédiaires de développement avant d'aboutir à l'image finale. Le nombre et la nature de ces étapes intermédiaires (les épreuves) dépendent du procédé utilisé (tirage papier, diapositives, polaroid, etc.). Il en est exactement de même pour la synthèse d'image, où

les épeuves successives contiennent de façon analogue, des images latentes modélisées à l'aide d'attributs incomplètement synthétisés.

La deuxième étape est simplement la présentation de la vue (ou sa projection) sur laquelle il est encore possible d'intervenir, par exemple pour la situer par rapport à d'autres vues présentées simultanément sur l'image finale.

2.4.4. Les étapes intermédiaires

La figure 4 schématise le processus général présenté ci-dessus.

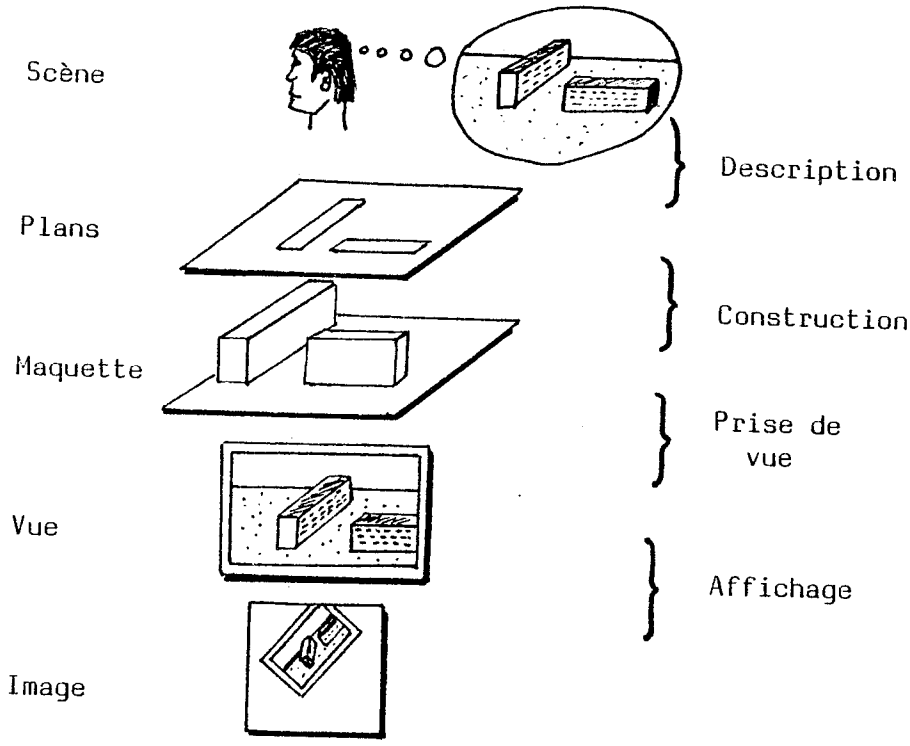


Figure I.4: Processus général de synthèse

Il est clair qu'un grand nombre d'étapes intermédiaires sont nécessaires pour passer de la scène voulue, à la scène perçue. Ces étapes conduisent à des représentations intermédiaires de la scène initiale, aux

divers stades de transformation.
Nous nommerons:

- maquettes, toutes les structures de données situées avant la prise de vue.

- épreuves, toutes les structures de données situées après la prise de vue.

Une vue est un type particulier d'épreuve dans laquelle toutes les informations (sauf celles relatives à l'affichage) sont synthétisées.

Une image est une épreuve dans laquelle toutes les informations ont été synthétisées.

3. Les éléments de base d'un système général de synthèse d'image

3.1. Les opérations

Sans vouloir faire la liste exhaustive des fonctions exécutées lors d'une synthèse d'image, nous évoquerons ci-après, pour chacune des quatre étapes mises en valeur au paragraphe précédent, les opérations "indispensables" ainsi que les attributs qu'elles concernent ou qu'elles affectent.

3.1.1. Description de la maquette

Cette première étape nécessite bien évidemment la description, puis la modélisation de toutes les classes d'attributs (I,M,A,G,E,S) afin d'obtenir la maquette initiale correspondant à la scène décrite. Outre les outils adaptés à la description morphologique (M) de telle ou telle catégorie d'objets, il faudra donc prévoir des fonctions permettant de situer les objets les uns par rapport aux autres (G), de leur affecter un aspect (A), d'indiquer les conditions d'éclairage de la scène (E) et bien sur de les identifier (I), et de les structurer (S). Cette dernière opération inclue les possibilités de création et de suppression.

3.1.2. La construction de la maquette

Dans cette seconde étape, on trouve essentiellement des opérations de génération, permettant à partir de paramètres de contrôle d'obtenir une information complète,

* à l'aide d'algorithmes,
* en faisant référence à des banques d'informations,
* à l'aide d'opérateurs de composition d'informations élémentaires,
* par approximation, interpolation, etc.

Tous les attributs quels qu'ils soient, peuvent ainsi faire l'objet d'une description "synthétique", à partir de quelques paramêtres, ce qui minimise la place accupée, et facilite la description . En revanche le temps passé dans l'étape de construction est plus important.

3.1.3. Prise de vue

Dans cette troisième étape, par analogie avec la prise de vue photographique, nous trouvons:

- La description des paramètres régissant cette prise de vue:
  * le point de vue,
  * la direction de visée,
  * les caractéristiques de l'appareil de prise de vue, en particulier: le type d'objectif utilisé, sa distance focale ainsi que le format de la vue. Ces paramètres permettent de définir le type de projection utilisée, ainsi que les frontières du domaine visible.
- la prise de vue elle-même.

Il est à noter que tous les attributs ci-dessus peuvent se ranger dans la classe des attributs géométriques (G), mais sont sensiblement différents de ceux de l'étape de description par deux points essentiels:

- ils s'appliquent de façon globale et identique à tous les éléments de la maquette,

- ils permettent la transformation des coordonnées exprimées dans le repère de la maquette, en coordonnées exprimées dans le repère de la vue. Dans le cas d'une maquette tri-dimensionnelle, cette prise de vue effectue par conséquent la projection dans le plan de la vue.

Pour les différencier nous noterons (Gv) ces divers attributs.

3.1.4. Affichage

La dernière étape de la synthèse nécessite également la description de quelques attributs géométriques notés (Ga) et permettant de décrire la façon dont la vue est "cadrée" pour obtenir l'image finale. Il s'agit par conséquent d'une transformation des coordonnées exprimées dans le repère de la vue, en coordonnées exprimées dans celui de l'image, tous deux bi-dimensionnels.

3.1.5. Récapitulatif

La figure 5 illustre les attributs mis en jeu au cours des quatre étapes fondamentales de la synthèse.

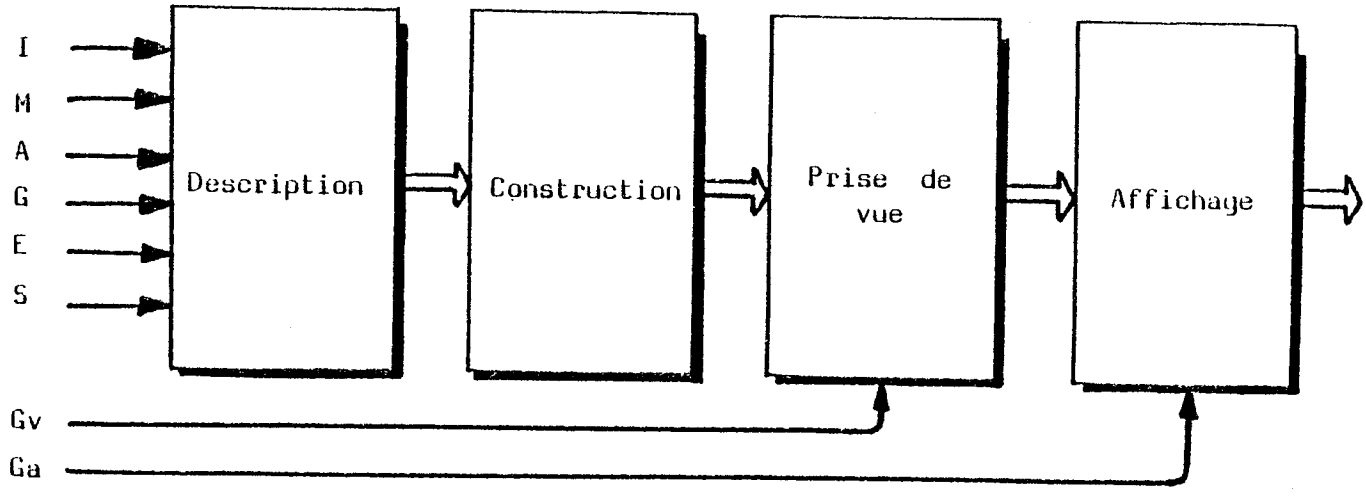
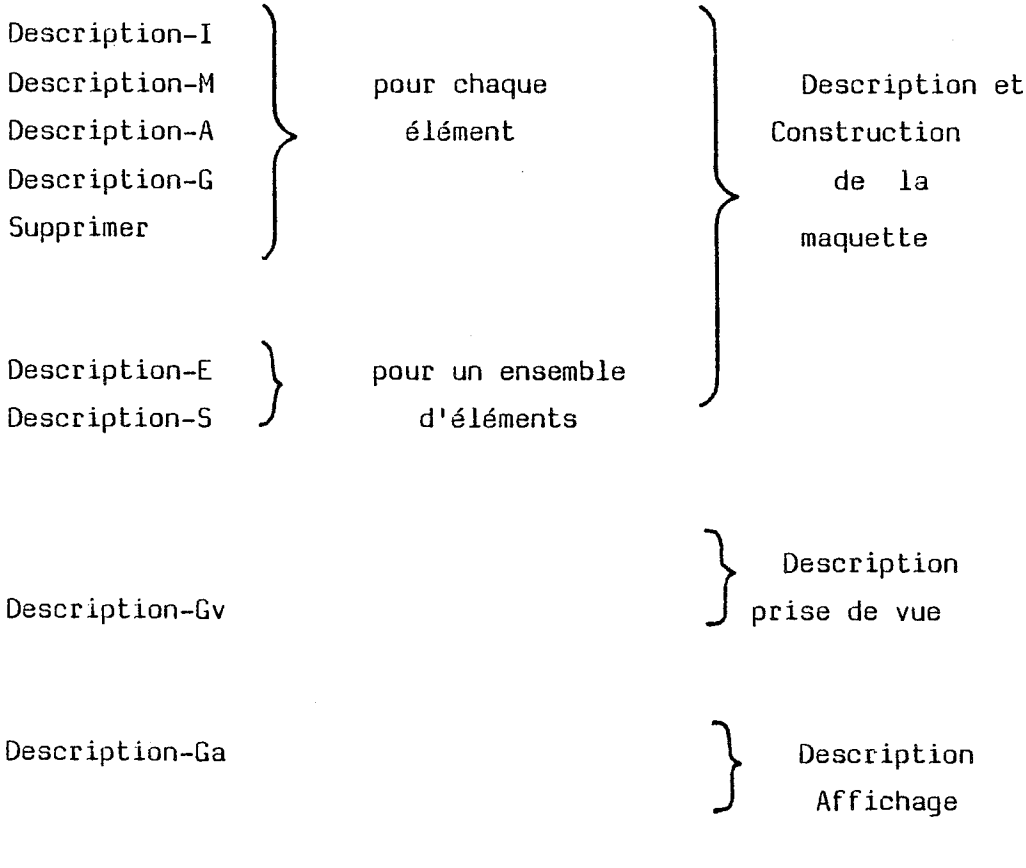


Figure I.5: La prise en compte des attributs

On peut donner également une première liste d'opérations pouvant entrer dans la définition d'un système de synthèse d'image.



L'opération de suppression se rapporte à un élément, et provoque également la suppression des attributs qui lui sont associés. L'opération de visualisation concerne la maquette toute entière.

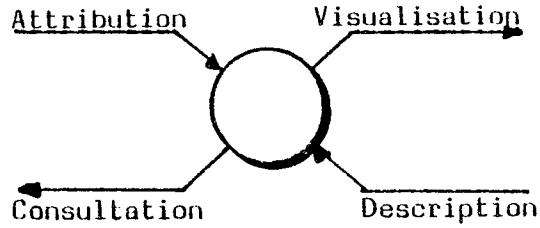
3.2. Les différents processus

Un système de synthèse d'image joue le rôle d'intermédiaire entre les deux interlocuteurs principaux qui sont :

* le programme d'application,
* l'opérateur.

Il doit par conséquent être capable d'assumer les échanges dans les deux sens avec ces deux participants, à travers quatre processus:

- programme --> système : attribution
- système --> programme : consultation
- système --> opérateur : visualisation
- opérateur --> système : description



Ces quatre processus sont, bien entendu, sous le contrôle de l'application qui peut les invoquer par le biais de primitives. Nous passerons en revue, ci-après ces processus et donnerons des "primitives possibles" pour leur invocation. Il ne faut y voir en aucun cas, la définition d'un langage, mais simplement une formalisation de l'interface entre l'application et le système.

3.2.1. L'attribution

Il est relativement rare que tous les attributs proviennent d'une saisie directe effectuée par l'opérateur. Bon nombre d'entre eux proviennent de calculs effectués dans le programme d'application et font l'objet d'une affectation dans les structures de données du système. Ce processus d'affectation pourrait être invoqué par le programme d'application, à l'aide d'une primitive telle que celle présentée ci-dessous.

```
ATTRIBUER (identité-éléments, type-attribut, info)
```

identité-élément représente le, ou les éléments auxquels l'information 'info' de type 'type-attribut' doit être affectée.

Si l'attribut appartient à l'une des classes (S,E,Gv,Ga), il concerne tous les éléments (ou un sous-ensemble représenté par 'identité-éléments').

Il s'agit donc d'une description effectuée par le programme à l'aide de variables, tableaux, etc., communiqués par "valeur" ou par "adresse".

Dans le premier cas, les données sont réellement transmises au système, alors que dans le second, seules les adresses sont communiquées. Le processus d'attribution peut être décomposé en deux opérateurs élémentaires, appliqués successivement.

- Consultation des structures de données du système à la recherche des éléments concernés.

- Affectation proprement dite.

3.2.2. La consultation

Le programme doit pouvoir récupérer des attributs en provenance du système. Ce besoin se fait sentir principalement lorsqu'une description a été réalisée, afin de communiquer à l'application les attributs décrits.

La primitive duale de celle de l'attribution est de la forme:

```
CONSULTER (identité-éléments, type-attribut, info)
```

dans laquelle 'info' est cette fois ci le résultat du processus. Dans le cas où seules les adresses sont présentes au niveau du système, cette consultation peut être effectuée directement à l'intérieur du programme d'application.

3.2.3. Le processus de visualisation

La visualisation d'un ou plusieurs éléments d'une maquette nécessite l'application d'un processus susceptible d'assurer la synthèse des attributs descriptifs des éléments en une information finale exprimant la couleur en chaque point de l'image.

Le processus global présenté en 2, et mettant en jeu successivement, la description, la construction, la prise de vue et l'affichage, n'est cependant pas universel.

En premier lieu, il n'est pas toujours possible ou souhaitable, dans

le cas d'images synthétisées par ordinateur, de séparer aussi nettement la construction de la prise de vue elle-même. Cette étape de construction peut ainsi être répartie au long du processus de visualisation, au fur et à mesure des diverses synthèses effectuées à partir des attributs initiaux, pour parvenir à l'information finale de couleur.

D'autre part des considérations relatives à l'efficacité de certaines opérations peuvent influer considérablement sur ce processus de visualisation. Le chapitre II de cette thèse est principalement consacré à ce problème, mais on peut d'ores et déjà entrevoir que cette influence portera principalement sur l'ordonnancement et la nature des différents opérateurs élémentaires utilisés. Nous décomposerons tout processus de visualisation selon quatre types d'opérateurs élémentaires:

- Les opérateurs de synthèse, chargés de la synthèse de deux attributs de base.

- Les opérateurs de mémorisation, chargés de stocker les étapes intermédiaires du processus.

- Les opérateurs de construction (ou modélisation), chargés de la transformation d'un attribut d'un type donné dans un autre, par exemple la génération des points constituant le contour d'un cercle.

- Les opérateurs de composition, permettant de composer les attributs (de même type) de plusieurs éléments de la maquette. Le résultat de ces opérateurs peut être le tri des éléments, ou simplement la sélection de ceux qui vérifient une propriété donnée, ou encore une réelle composition (union, intersection, etc.). Ces opérateurs sont utilisés principalement dans les algorithmes d'élimination des parties cachées, ils sont les seuls à concerner globalement plusieurs éléments.

Nous illustrerons ces propos à l'aide de l'exemple représenté sur la figure 6 . Le processus représenté pourrait être appliqué à la synthèse d'un élément tel que la "face 3D polygonale plane uniforme". Les différents opérateurs sont les suivants:

1) Synthèse M.G, entre les attributs morphologiques définissant le contour

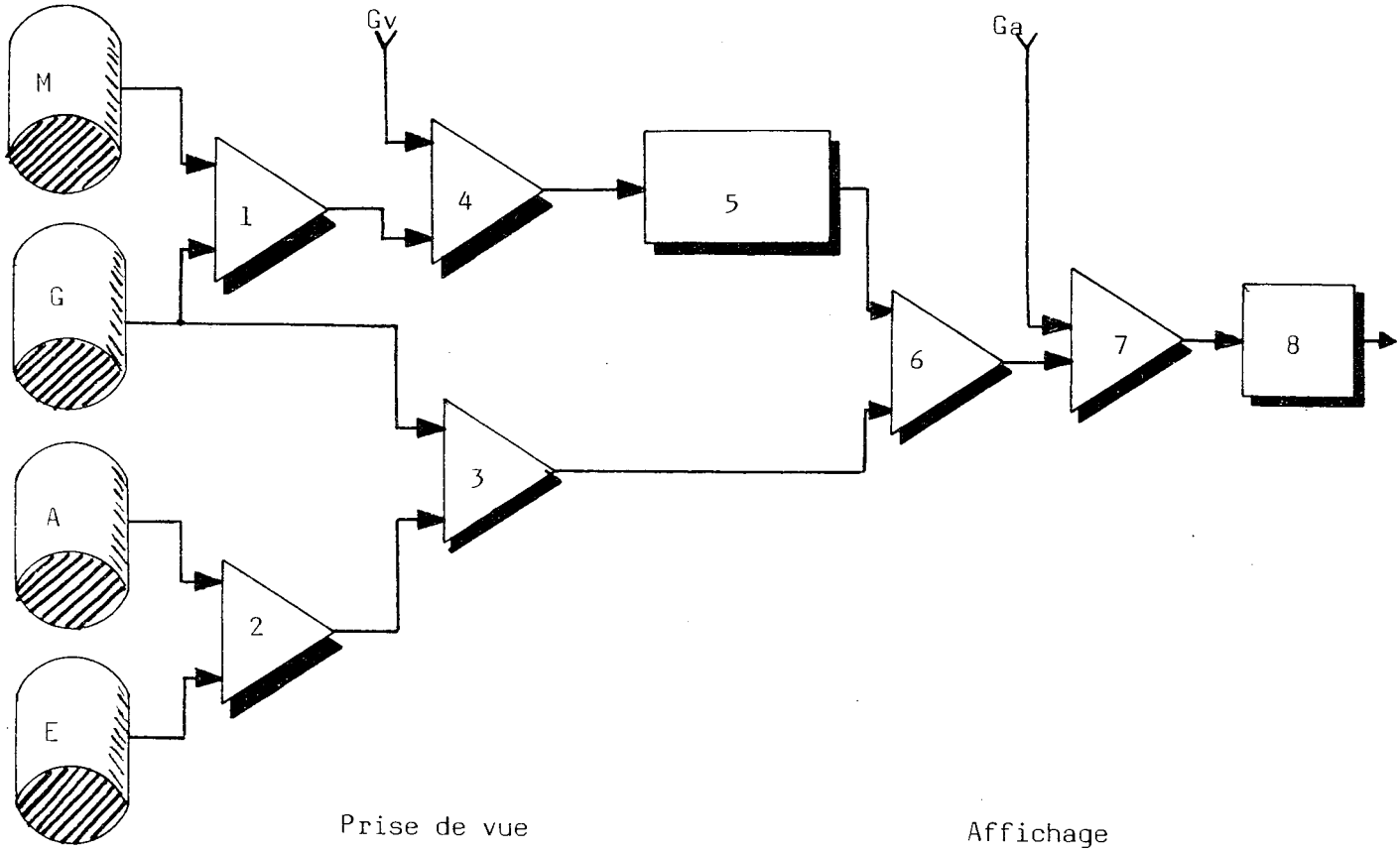


Figure I.6: Exemple de processus de synthèse

et les attributs géométriques permettant de situer le polygone dans l'espace.

2) Synthèse A.E, entre la couleur (A) et l'intensité d'éclairage (E).

3) Synthèse G.(A.E): la couleur (A.E) est modulée en fonction de la direction de la source lumineuse (E) et de la normale au plan de la face (G).

4) Synthèse (M.G).Gv: correspond aux opérations de projection de la face dans le plan.

5) Détermination et construction des points intérieurs de la face ("remplissage").

6) Synthèse (M.G.Gv).(G.A.E): attribue à chaque point intérieur la couleur modulée par l'éclairage .

7) Synthèse (M.G.A.E.Gv).Ga: permet de cadrer la face sur l'écran (zoom, translation, rotation, coupage).

8) Transformation de l'information finale (M.G.A.E.Gv.Ga): effectue une éventuelle transposition des couleurs, permettant par exemple la mise en valeur de certains phénomènes.

On peut également remarquer sur cet exemple qu'il n'existe aucun opérateur de mémorisation intermédiaire et que par conséquent, toute modification de l'un quelconque des attributs présents, implique l'exécution du processus complet. Etant donné qu'il s'agit d'un processus défini pour un élément unique, aucun opérateur de composition n'est nécessaire. De nombreux exemples de processus seront donnés dans les chapitres qui suivent.

De manière analogue aux processus d'attribution et de consultation, le programme d'application peut invoquer la visualisation à l'aide d'une primitive unique:

VISUALISER (identité-éléments, processus)

L'indication du processus permet au système de déterminer l'ordre et la nature des opérateurs nécessaires. Cette approche permet de dégager l'utilisateur de toute gestion du processus de visualisation, qui est intégralement à la charge du système. Nous verrons que ce choix, soulève des problèmes nouveaux, et que de nombreux axes de recherche, sont ouverts pour la détermination des processus les mieux adaptés aux possibilités du matériel, au type des informations manipulées, et aux performances requises.

Il faut également faire la distinction entre les processus de visualisation différée pour lesquels la primitive "visualiser" doit être invoquée pour demander explicitement la mise à jour de l'image, et les processus de visualisation immédiate pour lesquels le processus est déclenché implicitement après chaque attribution ou description.

3.2.4. Processus de description

Si les processus d'attribution et de consultation se limitent à de simples opérations élémentaires, le processus de description peut quant à lui être extrêmement complexe. Il s'agit en effet de décrire les informations de base (I.M.A.G.E.S.Gv.Ga) de la maquette à synthétiser à l'aide des informations fournies par l'utilisateur. La principale

difficulté provient du fait que les dispositifs de saisie associés aux consoles de visualisation ne fournissent que des couples de coordonnées (informations morphologiques) à partir desquels tous les autres types d'attributs doivent être décrits.

La description d'un type d'attribut nécessite donc un processus spécifique permettant de transformer les informations initiales, fournies par les dispositifs de saisie, en attributs du type considéré. Les informations initiales de base sont:

- des coordonnées exprimées dans le repère du dispositif de saisie, qui définissent des informations synthétiques (M.G.Gv.Ga),

- des paramètres numériques ou alphanumériques permettant d'exprimer certaines caractéristiques des attributs à engendrer. En outre, afin d'éviter une perte de temps au moment de la visualisation, certaines opérations de construction peuvent être effectuées au moment même de la description. Il apparaît donc que la séparation entre description et visualisation n'est pas évidente, puisque toutes deux renferment des opérations relatives à la construction. Comme il convient de s'entendre sur la signification des termes, nous parlerons de "description d'un élément", pour désigner les parties du processus global permettant de définir séparément les attributs associés à l'élément concerné.

Si cet élément subit plusieurs transformations au cours du processus, on obtient le schéma ci-dessous (figure 7) dans lequel le processus complet peut être assimilé à la description de l'image.

La figure 8 illustre un processus de description, permettant la définition dans l'espace, des points du périmètre d'un cercle, à partir des coordonnées du centre et d'un point de la circonférence, désignés sur l'écran. Le nombre de points est communiqué en paramètre:
1) Modélisation des coordonnées sous forme d'équations ou sous forme centre-rayon, exprimée dans le repère de l'image.
2) En fonction des attributs géométriques qui ont été utilisés pour l'affichage (Ga), expression du modèle dans le repère de la vue.

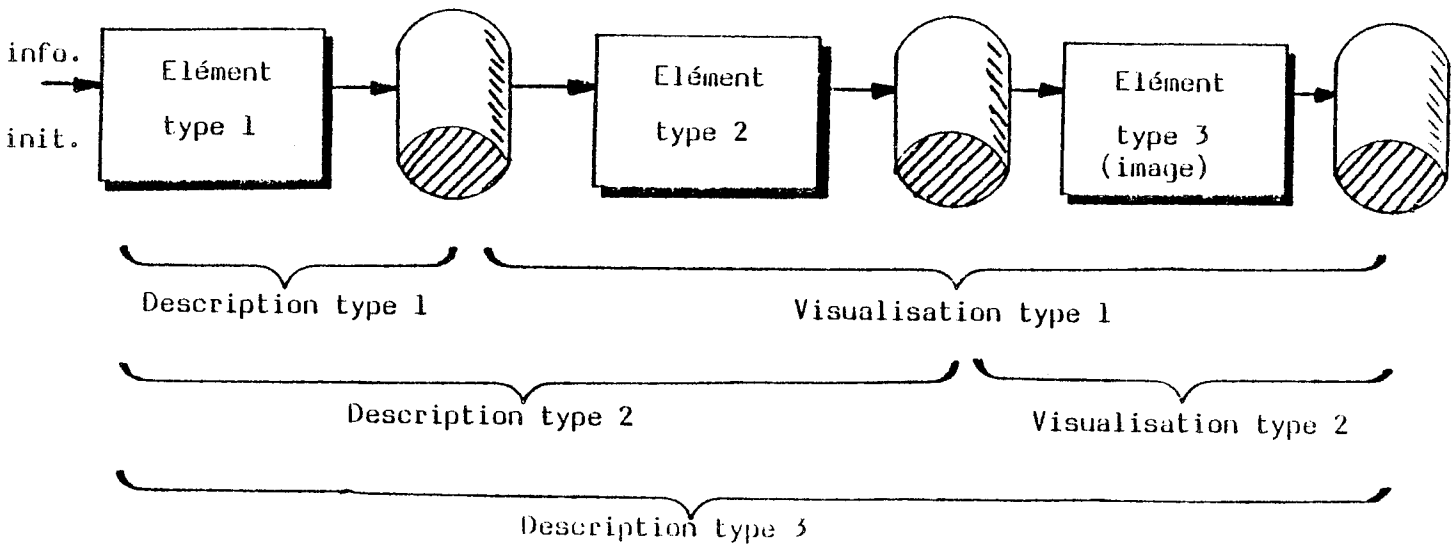


Figure I.7: Les limites entre la description et la visualisation

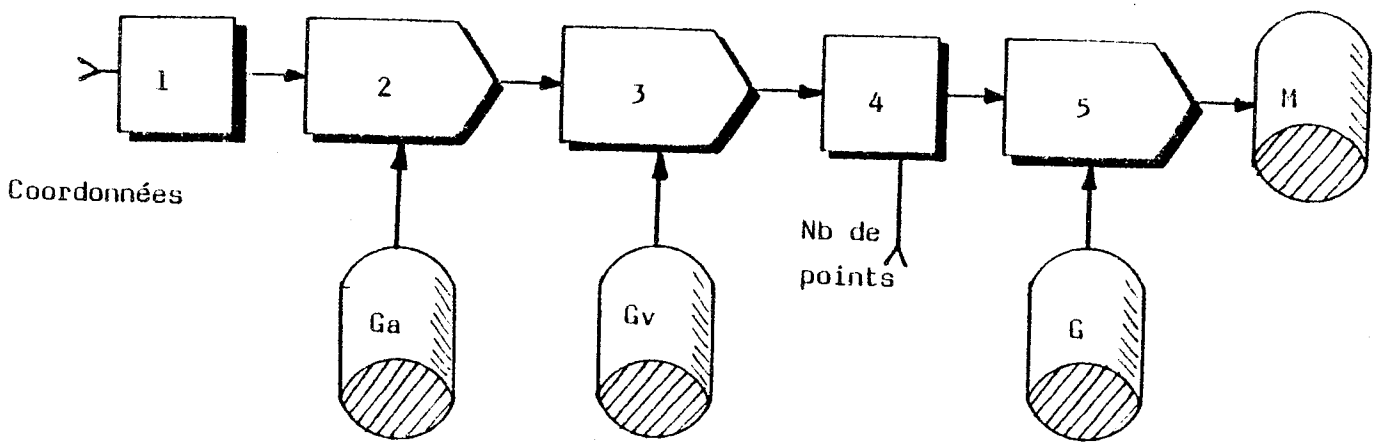


Figure I.8: Processus de description d'un cercle

3) Etant donné qu'il n'est pas possible de reconstituer la troisième dimension sans informations supplémentaires, cet opérateur exprime le modèle par exemple dans le plan z = 0 du repère absolu de la maquette.

4) Génération des points du contour dans ce plan, le nombre de points est fourni en paramètre.

5) Reconstitution de la troisième dimension en fonction de l'équation du plan de l'espace devant servir de support (attribut (G) de l'élément).

L'identification

Un cas particulièrement intéressant est celui où le processus de description s'effectue par référence à des éléments précédemment définis et mémorisés. Ce type de traitement est basé sur un opérateur permettant de décrire un sous-ensemble de la maquette (attributs (I) et (S)) à partir de désignations effectuées directement sur l'écran . Ce processus particulier d'identification par désignation peut se révéler relativement complexe, si des opérateurs de mémorisation ne figurent pas dans le processus de visualisation.

Il s'agit en effet

1) de collecter les coordonnées du point désigné sur l'écran,

2) de "remonter" le processus de visualisation jusqu'à la première mémorisation contenant un attribut d'identité. Cependant, les opérations inverses des opérateurs de synthèse ne sont généralement pas définis, et l'identification nécessite alors une pseudo-visualisation des éléments "identifiables", afin de vérifier si le point désigné les concerne. Cette pseudo-visualisation est généralement plus simple que l'originale, pour trois raisons principales.

- Elle débute à partir de la dernière mémorisation du processus.
- Elle ne considère pas les attributs A et E.
- Il n'est pas nécessaire de déterminer tous les points d'un élément mais simplement de vérifier si le point désigné appartient ou non à cet élément.

De la même manière que pour la visualisation, les processus de description peuvent être invoqués par une primitive unique:

DECRIRE (identité-éléments, processus, type-attribut)

L'attribut de type 'type-attribut' est décrit selon le processus indiqué et attribué aux éléments référencés par 'identité-éléments'.

3.3. Caractérisation des systèmes de synthèse

3.3.1. Cas général

On peut tenter de définir globalement un système de synthèse d'image comme une interface entre un univers initial I et un univers terminal T. Chaque univers, à son tour, est défini comme un couple d'ensembles:

$$\begin{aligned} I &= \{P(I),E(I)\} \\ T &= \{P(T),E(T)\} \end{aligned}$$

où

P(j) représente un ensemble de processus de visualisation, d'attribution, de consultation ou de description.

E(j) représente l'ensemble des éléments sur lesquels ces processus portent. Il est à noter que chaque processus p(k,j) de P(j) n'est pas nécessairement défini sur E(j) tout entier, mais sur

$$\begin{aligned} D(k,j) &\subset E(j) \\ E(j) &= \bigcup_k D(k,j) \end{aligned}$$

Dans le cas de la synthèse d'image, l'univers terminal T (qui est l'univers initial du tube à rayons cathodiques lui-même) est défini comme suit:

- l'élément unique appartenant à E(T) est le pixel,
- le processus unique appartenant à P(T) est l'attribution d'une couleur à un pixel donné.

Pour ce qui est de l'univers initial I, si l'ensemble E(I) de tous les éléments envisageables est illimité, compte tenu de la diversité des situations, l'ensemble P(I) de processus, en revanche, peut être déduit des considérations précédentes. Un système maximal comprendrait ainsi le jeu de processus suivant:

Pour chaque type d'élément:

	Description	Attribution	Consultation
Identité	Décrire-I	Attribuer-I	Consulter-I
Morphologie	Décrire-M	Attribuer-M	Consulter-M
Aspect	Décrire-A	Attribuer-A	Consulter-A
Géométrie	Décrire-G	Attribuer-G	Consulter-G

Pour la maquette

E	Décrire-E	Attribuer-E	Consulter-E
S	Décrire-S	Attribuer-S	Consulter-S
Gv	Décrire-Gv	Attribuer-Gv	Consulter-Gv
Ga	Décrire-Ga	Attribuer-Ga	Consulter-Ga

A ceci il faut ajouter l'ensemble des processus de visualisation concernant tous les éléments, en fonction des possibilités du matériel, du type de représentation choisi, et des performances requises.

3.3.2. Les cas particuliers et leurs propriétés

En vue de réduire le nombre de processus nécessaires les concepteurs de systèmes de synthèse ont utilisé plusieurs schémas de simplification.

- Diminuer le nombre d'éléments traités, ce qui limite la portée du système.

- Utiliser les mêmes types d'attributs pour tous les éléments (en particulier I,G,A), ce qui permet d'utiliser les mêmes processus de description pour tous ces éléments, et parfois les mêmes opérateurs de base dans les processus de visualisation.

- Proposer des processus combinés permettant par exemple de décrire simultanément plusieurs attributs. Cette combinaison offre beaucoup moins de souplesse mais procure souvent une plus grande facilité d'utilisation.

- Laisser les données au niveau du programme d'application afin d'éviter les besoins de consultation. Ce choix impose alors une connaissance de la modélisation employée par le système pour les divers éléments.

- Sacrifier certains processus, par exemple favoriser l'attribution (ensemble de primitives) ou au contraire favoriser la description (logiciel intégralement interactif).

La grande diversité des solutions suggère de mettre en valeur un certain nombre de propriétés permettant de caractériser quelques cas particuliers. Ainsi nous dirons qu'un système est:

- cohérent, si les processus disponibles sont définis pour tous les éléments traités, et applicables au(x) même(s) niveau(x) de la structure;

- symétrique, si tous les éléments de l'univers peuvent être intégralement décrits de façon interactive;

- ouvert, si tous les attributs décrits peuvent être consultés par le programme d'application.

Dans la majorité des cas ces propriétés sont rarement aussi "tranchées" et doivent être modulées à des degrés différents.

4. Conclusion

Si l'on examine les systèmes de synthèse d'image (fort rares) et les systèmes "graphiques" existants à l'heure actuelle, on peut faire les remarques suivantes:

En premier lieu, on peut constater le nombre très important de primitives proposées (I.G.L (Hvi79) en propose 175), en regard des 4 "primitives-processus" que nous avons présentées. Cette différence s'explique de plusieurs façons:

- Bon nombre de ces primitives n'ont aucun rapport avec la synthèse

d'image, qu'il s'agisse d'actions permettant de configurer le système, le terminal, ou d'opérations permettant d'utiliser les dispositifs non graphiques associés au poste de travail: clavier alpha-numérique, potentiomètres divers, etc.

- Ces primitives sont en fait les opérations élémentaires et non les processus. On y trouve ainsi des primitives permettant d'engendrer un cercle, d'effectuer une rotation-2D, d'approximer une courbe, etc.

- Le nombre des primitives est également dû aux contraintes de modélisation syntaxique des divers attributs, ainsi une information bi-dimensionnelle ne nécessite que deux paramètres alors qu'un attribut tri-dimensionnel en nécessite trois.

Nous pensons pour notre part qu'il est tout à fait souhaitable de distinguer les concepts et les primitives proposés à l'utilisateur. Le choix de celles-ci résulte généralement des schémas de simplification utilisés, et de la prise en charge ou non des processus par le système. Cette gestion des processus est le plus souvent effectuée partiellement par le système, ce qui ne facilite ni la compréhension ni l'utilisation. Nous verrons au chapitre III, les principales approches adoptées par les systèmes de synthèse d'image. Le point fort des concepts présentés dans ce chapitre, concerne la possibilité de laisser au système le choix et l'ordonnancement des divers processus. Il va de soi que cette approche libère considérablement l'application qui n'a plus rien à connaître de l'organisation interne du système ni des possibilités du matériel utilisé.

Le report de cette gestion au niveau du système soulève cependant des problèmes délicats et ouvre de nouveaux axes de recherche, relatifs à la conception et à la réalisation du système. Nous apporterons, au cours du chapitre suivant quelques éléments de réponse à ces questions.

Chapitre II

Aspect technique

Point de vue du concepteur

Nous nous intéresserons dans ce chapitre à la mise en oeuvre d'un système général de synthèse d'image. A partir des concepts mis en valeur au chapitre précédent, nous serons amenés à considérer les différents facteurs influant sur le choix des processus, des primitives et des niveaux de matériel. Nous retrouverons ainsi la plupart des approches proposées dans la littérature, constituant des cas particuliers intéressants, et nous en indiquerons, les avantages et les limitations. Nous proposerons également de nouvelles solutions découlant directement des concepts de base ci-dessus, approches qui, bien entendu, sont à la base de notre propre expérience.

Pour étudier la conception et la réalisation des processus nécessaires à la synthèse (description, visualisation) nous ferons l'analogie avec une "chaîne de fabrication", dont la mise en oeuvre passe généralement par trois phases.

- l'ordonnancement des opérations,
  * de description de ces éléments (description)
  * de transformation (construction),
  * d'assemblage (synthèse),
  * de stockage (mémorisation) à divers stades de la fabrication.

- la répartition des travaux parmi les différents postes de travail (processeurs).

- la détermination et l'adaptation des processeurs nécessaires (logiciel ou matériel).

Cette étude est compliquée par le fait qu'il ne s'agit pas simplement de fabriquer un produit particulier sur une chaîne particulière, mais plutôt de donner les outils permettant de fabriquer toutes sortes de produits sur

toutes sortes de postes de travail. En d'autres termes, il s'agit dans notre cas de proposer un système "général" capable:

- d'assurer l'indépendance vis-à-vis du matériel (consoles de visualisation et dispositifs de description), c'est-à-dire d'assurer la visualisation et la description de tous les éléments de base, quel que soit le poste de travail utilisé,

- d'accepter une grande diversité d'éléments afin de satisfaire le plus grand nombre d'applications (C.A.O., cartographie, mathématiques, etc.)

- de proposer une grande diversité de performances afin de satisfaire au mieux le plus grand nombre de situations (applications interactives sur matériel lent , applications complexes, ou rapides, etc.)

Ces propriétés découlent directement des choix effectués lors de l'organisation générale du système, de l'ordonnancement des processus, et de l'organisation interne de chaque "poste de travail". Les paragraphes qui suivent traitent de ces trois questions, et des critères influant sur les choix.

1. Organisation générale du système

1.1. La notion de synthétiseur

Nous appelerons synthétiseur d'images (ou tout simplement "synthétiseur") une unité de traitement indivisible possédant les caractéristiques suivantes:

- Il existe au moins un processeur, de technologie quelconque (cablé, micro-programmé, programmé),

- Il existe au moins une ressource locale de mémoire, quel qu'en soit le support (RAM, disque souple, écran, etc).

- Il existe au moins un processus de visualisation et un processus d'attribution dans la mémoire locale.

- Il peut exister des processus de description et de consultation.

Un synthétiseur peut ainsi être caractérisé par son univers initial, qui indique les éléments et les processus acceptés. Le synthétiseur le plus simple est constitué par le tube cathodique lui-même pour lequel la mémoire est la surface de l'écran, le processus d'attribution est assuré par le canon à électrons et le dispositif de déviation, le processus de visualisation implicite est assuré par la luminescence de l'écran. L'univers comporte un élément unique, le pixel défini par ses coordonnées et sa couleur, le processus unique est l'attribution de la couleur au point considéré.

1.2. Les couches de synthétiseurs

Comme dans le cas d'une entreprise, l'organisation du système de synthèse d'image dépend principalement de la localisation des unités de traitement, c'est-à-dire des synthétiseurs.

On peut ainsi définir un système de synthèse d'image comme une suite de synthétiseurs dont le dernier est le tube cathodique, et dont l'univers du premier est l'univers du système entier. Nous parlerons alors de couches de synthétiseurs. Les connexions entre ces divers synthétiseurs, (en parallèle ou en série), constituent autant de coupures physiques que les informations doivent franchir sous une forme déterminée par l'univers initial du récepteur. Le processus de synthèse d'un élément quelconque doit être, par la force des choses, scindé en autant de sous-processus successifs qu'il existe de couches de synthétiseurs. Cette scission dépend, par conséquent:

* du nombre de synthétiseurs,
* de l'univers de chacun d'eux.

Nous examinerons dans le paragraphe suivant les cas les plus caractéristiques de configuration matérielles, et leur influence sur l'organisation générale du système.

1.3. Influence de la configuration matérielle

Le concepteur d'un système de synthèse d'image peut se trouver confronté à l'une des situations suivantes:

- 1- Possibilité de déterminer le nombre de couches et de réaliser lui-même tous les synthétiseurs adaptés à ses besoins, y compris la console de visualisation.

- 2- Nécessité de choisir la console de visualisation parmi celles disponibles sur le marché.

- 3- Nécessité d'utiliser toute la configuration existante (logiciels et matériels).

Si la première situation laisse au concepteur une entière liberté à tous les niveaux, la seconde limite son rôle à la réalisation de logiciels adaptés, implantés dans les synthétiseurs "amont". Quant à la troisième, il est peu probable que le découpage imposé et les possibilités de la console de visualisation, correspondent précisément à ses besoins.

Quoi qu'il en soit, l'influence mutuelle entre la configuration matérielle et les objectifs visés est considérable. Entre des installations comportant une console bas de gamme connectée directement à un gros calculateur et des configurations utilisant une console "évoluée" connectée via un calculateur satellite, le fossé est grand et révèle une multitude de problèmes que nous évoquerons ci-après. Nous nous proposons de différencier les configurations selon les critères suivants:

- nombre de synthétiseurs indépendants et type (programmé, micro-programmé, cablé),

- capacité et type des mémoires locales (RAM, support magnétique) associées à chaque synthétiseur,

- univers de chaque synthétiseur (éléments et processus acceptés).

Le nombre de configurations possibles étant quasi-illimité, nous étayerons cette étude par trois situations caractéristiques. Le chapitre III nous permettra d'étudier plus en détail les configurations proposées dans les systèmes existant à l'heure actuelle.

1.3.1. Console bas de gamme

Une configuration est dite minimale dans le cas où elle se compose de deux synthétiseurs (figure 1).

- le premier est le calculteur hôte avec son logiciel et toutes ses ressources de mémorisation,
- le second est une console de visualisation "bas de gamme" (nous ferons abstraction du tube cathodique lui-même).

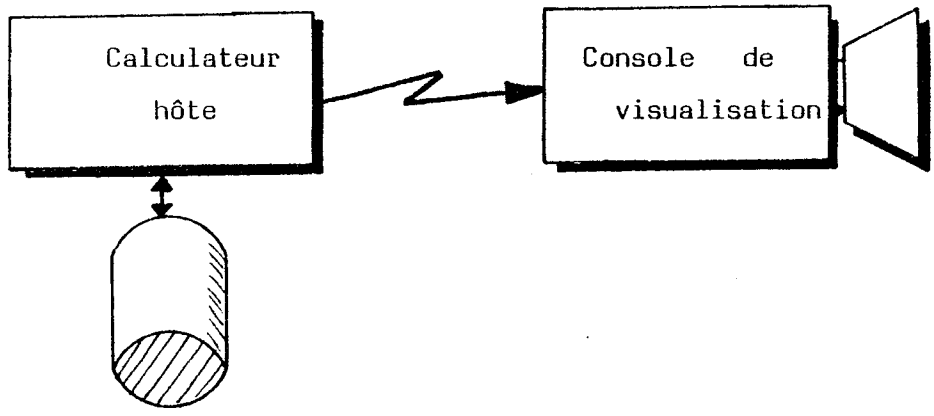


Figure II.1: Configuration minimale

La liaison est généralement du type série asynchrone à vitesse moyenne (1200 à 9600 bauds). Dans cette situation, l'univers du terminal est trés réduit:

- au niveau des éléments proposés (point, vecteur, caractère),
- et au niveau des processus disponibles souvent limités aux attributions et à la visualisation implicite ou explicite de ces éléments. La majeure partie des générations est donc assurée par le logiciel implanté

entièrement dans le calculateur principal. En contre-partie cette configuration est la plus facile à mettre en oeuvre, car elle soulève peu de problèmes relatifs au découpage des processus qui s'exécutent presque intégralement au niveau logiciel.

1.3.2. Console évoluée

Cette catégorie regroupe les installations dans lesquelles une console de visualisation évoluée est connectée directement au calculateur principal.

Nous dirons qu'une console est "évoluée" si elle offre des fonctions locales de description, c'est-à-dire permettant des modifications de l'image sans intervention du calculateur principal.

Ce type de console résulte généralement de l'adjonction d'un synthétiseur intermédiaire (avec son logiciel), associé à une mémoire locale (RAM et/ou disques souples). La liaison est également du type série asynchrone (figure 2).

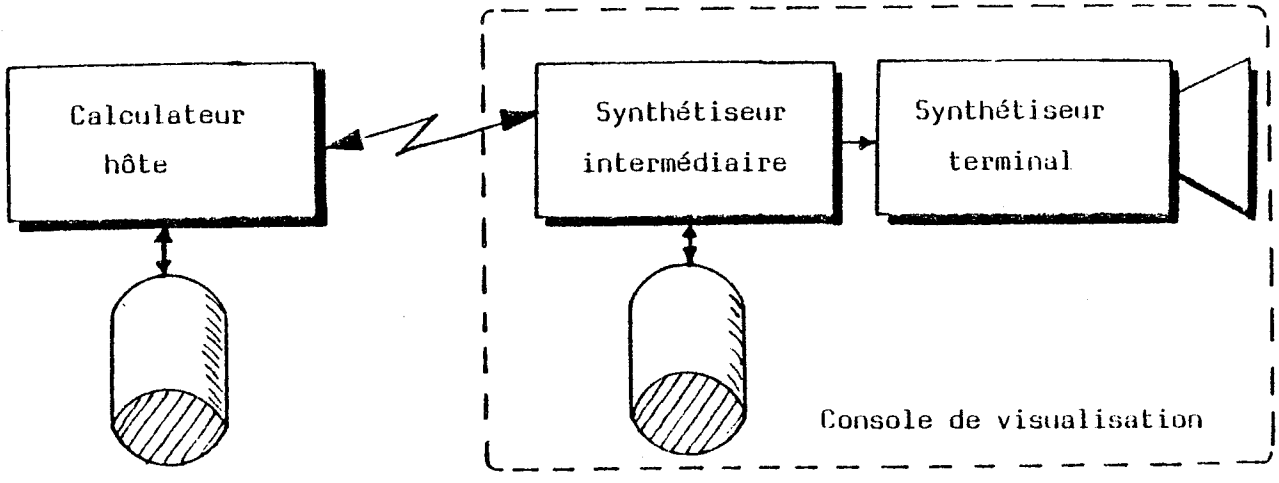


Figure II.2: Console évoluée

Cette situation, qui tend à se généraliser actuellement grâce à l'utilisation de plus en plus intensive de micro-processeurs, place le

concepteur d'un système général devant des problèmes délicats, principalement dûs à l'incohérence et à la dissymétrie de la couche micro-programmée du matériel.

L'incohérence

Quelques fonctions ne sont applicables qu'à certains types d'objets (par exemple rotation impossible sur les caractères). Le logiciel doit alors détecter ces lacunes du matériel puis les combler en proposant des fonctions de remplacement. Un deuxième exemple encore plus frappant est celui des terminaux permettant des transformations géométriques 2D et même 3D sur chaque élément : ces fonctions, qui font partie de la construction, sont dans la plupart des cas totalement inutilisables puisqu'elles doivent être effectuées avant l'élimination des parties cachées, qui elle n'est pas disponible au niveau du matériel. Le logiciel doit également "détecter" cette incohérence et décider de l'utilisation ou non des fonctions spécifiques du terminal, ce qui complique considérablement sa mise en oeuvre.

La dissymétrie

Il s'agit là comme nous l'avons vu (cf I.3.3), de l'impossibilité de décrire et de consulter les mêmes types d'attributs que ceux acceptés en entrée. Cette dissymétrie affecte la plupart des terminaux à cause d'une synthèse prématurée des attributs mémorisés, ce qui impliquerait l'application d'opérations d'analyse pour restituer les attributs initiaux . C'est le cas, par exemple, du remplissage de taches dans le terminal Tektronix 4027, après lequel aucune information n'est plus disponible sur le contour polygonal ni sur la texture utilisée. Une telle situation conduit irrémédiablement l'utilisateur (ou le logiciel) à conserver lui-même ces attributs, d'où une duplication importante. Une autre conséquence fâcheuse de la dissymétrie est l'amoindrissement de l'intérêt des fonctions d'interactions locales proposées par le matériel. Leur utilisation sans possibilité de consultation pour l'application peut conduire à des incohérences entre les structures conservées par le logiciel et l'image

présentée par le matériel.

1.3.3. Connexion à un calculateur satellite

Dans ce dernier cas, la console de visualisation n'est plus connectée au calculateur hôte directement, mais à travers un calculateur satellite possédant sa mémoire propre comme l'illustre la figure 3. La configuration comporte alors un synthétiseur de plus que dans les précédentes. Les liaisons sont généralement du type:

- serie asynchrone à vitesse moyenne entre les deux calculateurs,

- parallèle à grande vitesse entre le calculateur satellite et le terminal.

Cette architecture a suscité quelques travaux au cours des dernières années (Fol73), (Fol75). Ces études sont généralement des cas particuliers basés sur l'utilisation d'un nombre restreint d'éléments pour lesquels les processus de synthèse sont semblables.

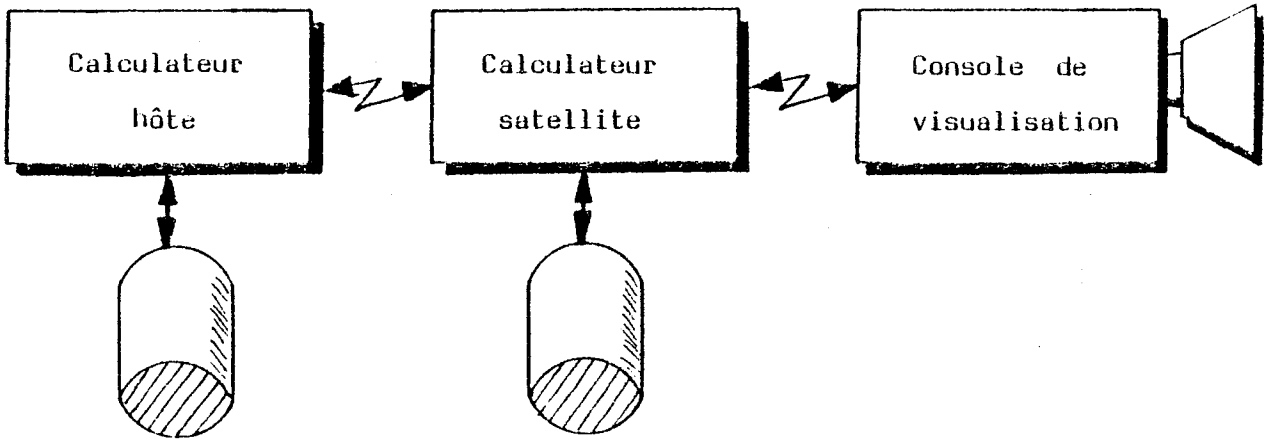


Figure II.3: Connexion à un calculateur satellite

A première vue cette configuration est assez semblable à la précédente mis à part le fait que le synthétiseur intermédiaire est situé à l'extérieur du terminal, mais en réalité une différence fondamentale les distingue.

Le calculateur n'est plus asservi au terminal, mais au contraire permet d'asservir celui-ci aux exigences de l'application, à travers une couche cohérente et symétrique, située au niveau du satellite.

1.3.4. Configuration multiple

Il est fréquent sur un site donné d'être simultanément confronté aux trois situations ci-dessus. L'organisation générale peut revêtir l'un des deux aspects suivants:

- le calculateur satellite est dévolu à un terminal particulier,
- le calculateur satellite est connecté en "frontal" de l'ensemble du parc des terminaux.

Cette dernière solution présente l'avantage de maximiser l'indépendance du logiciel situé sur le calculateur hôte. Celui-ci ne s'adresse en fait qu'à un synthétiseur unique et cohérent: celui simulé par la couche du calculateur satellite.

1.4. La réalisation d'un synthétiseur

Lorsque le système a été découpé selon le nombre et les possibilités des synthétiseurs disponibles ou désirés, il reste à organiser et à réaliser effectivement ces derniers.

La réalisation d'un synthétiseur peut s'effectuer de plusieurs façons différentes

- réalisation d'un logiciel sur un calculateur standard,
- utilisation d'un micro-processeur standard associé à un logiciel intégré en PROM ou en EPROM,
- utilisation de micro-processeurs spécialement adaptés intégrés dans des boitiers VLSI,

- utilisation de micro-processeurs en tranches,

- utilisation de logique cablée.

En ce qui concerne l'organisation interne d'un synthétiseur d'image, les problèmes sont beaucoup plus complexes et dépendent essentiellement du (ou des) processus à réaliser.

Les paragraphes qui suivent traitent de cette question, tout d'abord dans le cas de synthétiseurs dévolus à un processus de visualisation unique, puis dans celui de synthétiseurs multi-processus.

2. Organisation d'un synthétiseur mono-processus

Le cas le plus simple de synthétiseur correspond à la situation suivante:

- l'univers initial ne comporte qu'un seul type d'élément à synthétiser,

- l'univers terminal est unique, c'est à dire que l'on utilise les possibilités d'un seul et unique synthétiseur situé en "aval".

Ce type de synthétiseur ne comporte alors qu'un seul type de processus:

* de visualisation,
* de description,
* d'attribution,
* de consultation.

Il est clair que le processus le plus délicat à réaliser est celui de visualisation dont l'organisation influe directement sur les performances et les possibilités du synthétiseur. En effet seul le processus de visualisation est susceptible de traiter simultanément tous les attributs de tous les éléments de la maquette à synthétiser, alors que la description, l'attribution et la consultation ne considèrent qu'un type donné d'attribut.

Nous nous attacherons donc, dans les paragraphes qui suivent, à montrer l'influence du processus de visualisation sur l'organisation du synthétiseur associé.

2.1. L'ordonnancement du processus

2.1.1. L'influence des attributs

Si l'on reprend l'exemple du processus adapté à la visualisation des faces polygonales planes, présenté au chapitre précédent, on peut remarquer que la modification d'un attribut d'un élément quelconque, nécessite l'exécution complète du processus pour cet élément. Cette exécution selon le processus, est effectuée implicitement après chaque modification (visualisation immédiate), ou est demandée explicitement par une primitive (visualisation différée).

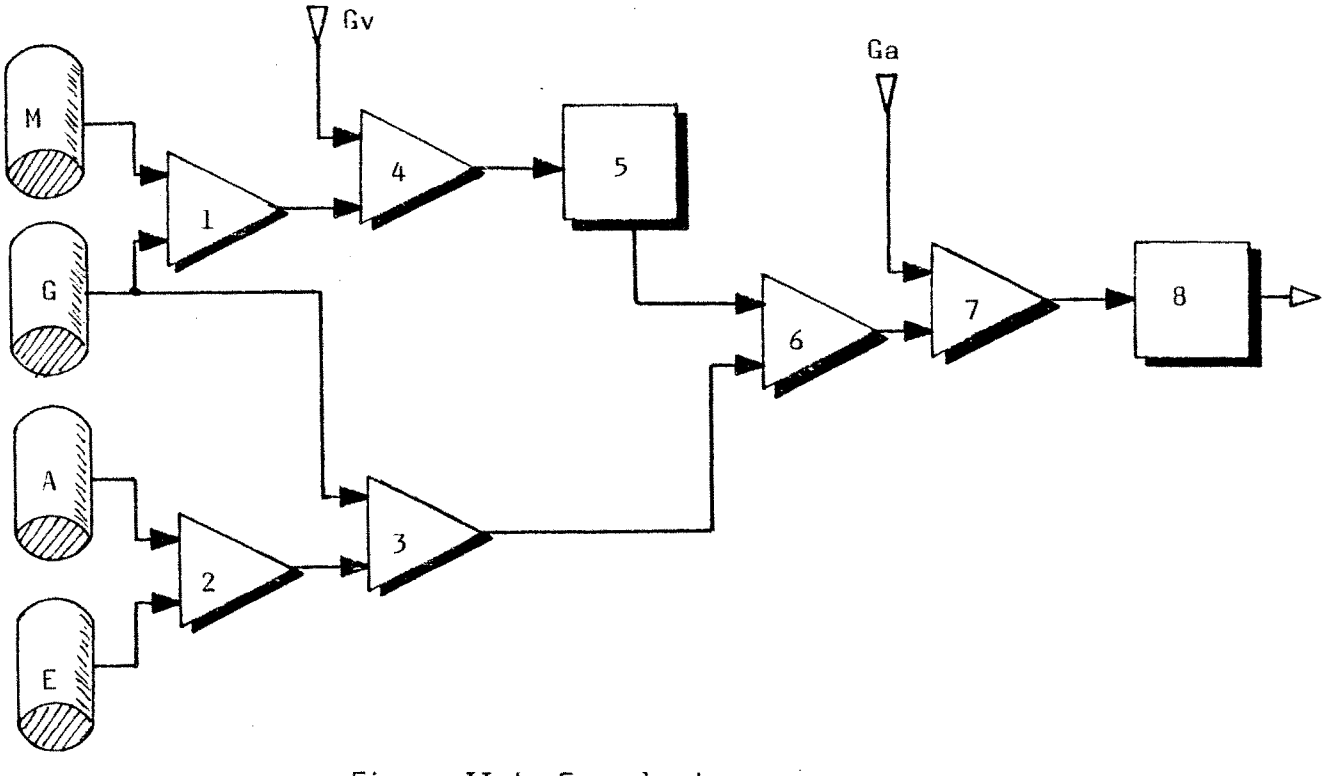


Figure II.4: Exemple de processus

Nous nommerons influence d'un attribut le nombre d'opérations élémentaires qu'il est nécessaire d'effectuer pour visualiser l'élément

auquel il appartient. Dans l'exemple de la figure 4, l'influence de chaque attribut est égale à 8. Cette notion ne reflète pas, bien sûr, la complexité des différents opérateurs, ni leurs performances, mais elle permet de quantifier de façon relative, les divers ordonnancements possibles d'un processus.

Dans la plupart des applications, il existe des opérations cruciales, dont les performances conditionnent celles de l'ensemble. Ainsi dans les applications de simulation de conduite, il est indispensable de privilégier les opérations de prise de vue et d'affichage si l'on veut prétendre à une animation en temps réel. Il importe alors de réduire au maximum l'influence des attributs (Gv) et (Ga) qui conditionnent directement ces opérations. De manière similaire, si les modifications d'aspect doivent être favorisées, il convient de réduire l'influence des attributs d'aspect.

En vue de réduire l'influence des attributs privilégiés, deux techniques sont utilisables, séparément ou conjointement:

* La mémorisation des attributs synthétisés,
* La pénétration des attributs non synthétisés.

2.1.2. Mémorisation des attributs synthétisés

Cette approche résulte généralement de la vonlonté de privilégier les opérations ayant trait à la prise de vue ou à l'affichage. Ainsi sur la figure 4, il est clair que l'introduction d'une opération de mémorisation après l'opérateur 6, réduit le processus nécessaire pour l'affichage aux opérateurs 7 et 8, ce qui porte à 2 l'influence de l'attribut (Ga), et diminue considérablement le temps de réponse (cf.figure 5).

Dans le cas fréquent où l'on désire déplacer le point de vue autour de la maquette, il est souhaitable de favoriser le processus de prise de vue, en réduisant l'influence des attributs (Gv) à 5, qui est la valeur minimale que l'on peut obtenir avec ce processus. Cette approche conduit à introduire une mémorisation après les opérateurs 1 et 3, comme le montre la figure 6.

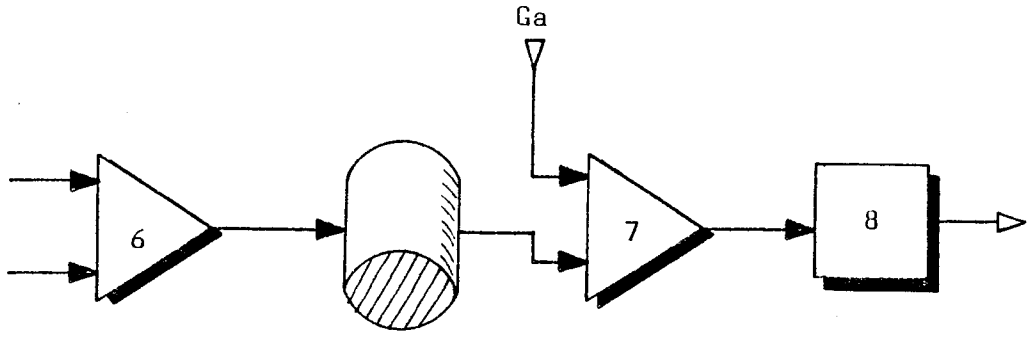


Figure II.5: Mémorisation intermédiaire

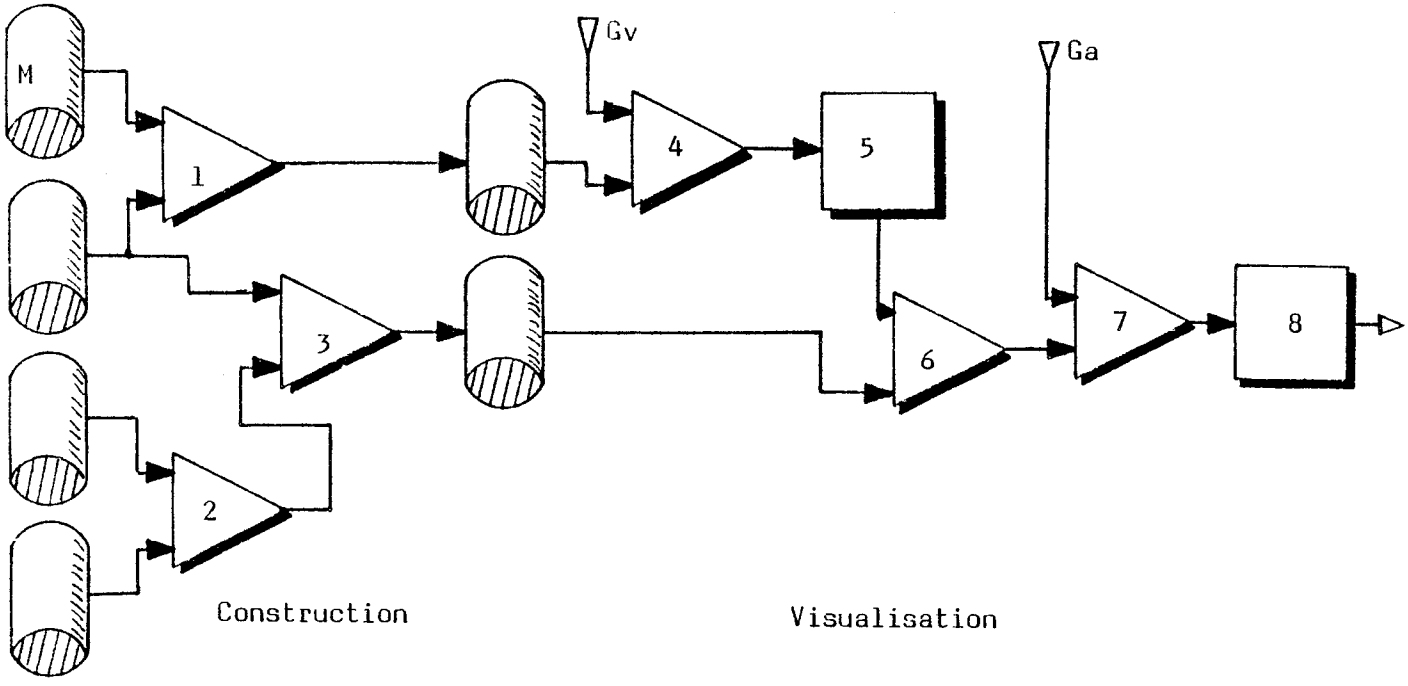


Figure II.6: Autre exemple de mémorisation intermédiaire

La mémorisation intermédiaire induit également une nette accélération du processus d'identification par désignation, puisque celui-ci requiert une pseudo-visualisation qui se trouve elle-même accélérée. Il est nécessaire dans ce cas de mémoriser également les informations d'identité (voire de structure) afin de permettre l'identification. Si ces informations ne figurent pas à ce niveau, il faut effectuer la régénération à partir d'une structure amont, ce qui bien entendu, est plus coûteux.

2.1.3. Pénétration des attributs non synthétisés

L'exemple précédent illustre parfaitement les limites de la technique de mémorisation intermédiaire. L'influence de chaque attribut est en fait limitée par le processus lui-même, ce qui est préjudiciable notamment aux attributs M,A,G,E. Dans le cas de la figure 6 l'influence de A est fixée à 7 et ne peut être réduite.

La solution consiste alors à modifier le processus, de telle façon que l'attribut à privilégier (A) "pénètre" le plus profondément possible vers l'intérieur. En conjuguant cette approche avec la mémorisation des "autres attributs" synthétisés on peut parvenir à réduire l'influence dans des proportions apréciables. La figure 7 illustre un exemple dans lequel l'influence des attributs d'aspect est réduite à 2.

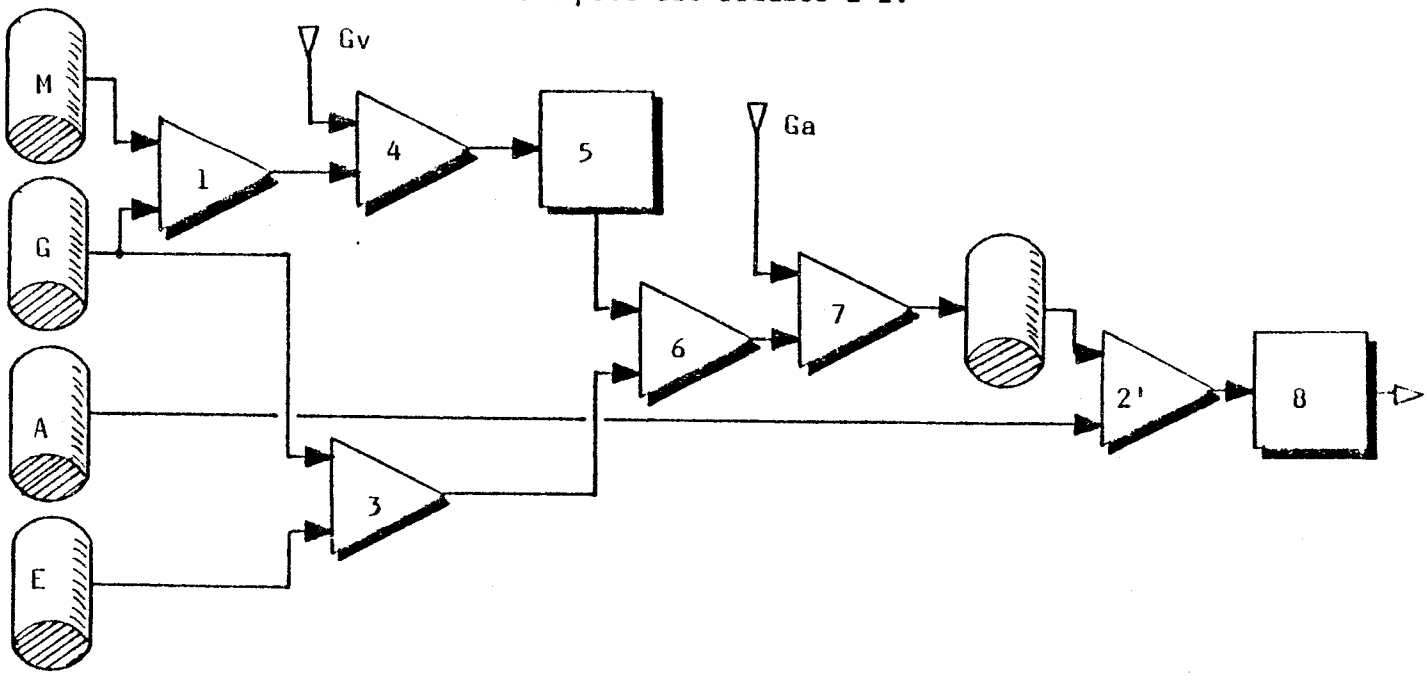


Figure II.7: Pénétration des attributs

La pénétration des attributs soulève cependant un certain nombre de problèmes.

- En premier lieu, la transformation du processus n'est posssible que si l'attribut considéré est relativement indépendant des autres. Dans

l'exemple précédent la couleur de la face est effectivement indépendante de sa morphologie et du point de vue, ce qui n'est pas le cas si l'on considère des textures. Cette transformation a donc pour conséquence de limiter la "qualité" des informations concernées, et par suite celle des éléments traités par le processus initial.

- En second lieu, la plupart des algorithmes "classiques" de la synthèse d'image sont remis en cause par la synthèse "tardive" des informations privilégiées. C'est ainsi qu'il devient très difficile, dans le processus de la figure 7, d'utiliser les techniques de lissage de l'éclairement (Gou71), ou "d'anti-aliasing" (Cro77) (le mot français correspondant reste encore à trouver; il semble que le néologisme "anti-aliassage" se dégage de plus en plus). On peut noter qu'il s'agit là d'une nouvelle réduction de la qualité des informations concernées (A). En ce qui concerne le processus d'identification par désignation, cette pénétration est sans effet si elle concerne les attributs (A) et (E). Pour les autres, elle est tout à fait souhaitable puisque ce processus nécessite généralement une pseudo-visualisation des attributs M,G,Gv,Ga.

2.1.4. Interactions inter-éléments

Les paragraphes précédents ont montré le grand nombre de solutions possibles pour la visualisation d'un élément unique. Lorsqu'un grand nombre d'éléments identiques doivent être synthétisés simultanément, cette situation se trouve à nouveau compliquée dans des proportions importantes.

les opérateurs de composition doivent, en effet, être appliqués globalement à un ensemble d'éléments et non pas indépendamment sur chacun d'eux et provoquent, par conséquent, des interactions mutuelles entre ces éléments. Les principaux opérateurs de composition mettant en jeu une interaction mutuelle des éléments sont:

- l'élimination des parties cachées (interactions morphologiques et géométriques),

- la production d'ombres portées (opération quasiment identique à la

précédente),

- la restitution de la transparence (interactions d'aspect),

- l'atténuation des effets "d'aliasing": marches d'escaliers, disparition d'objets reduits, etc. (interactions au niveau des informations finales synthétisées).

La prise en compte de ces opérations de composition nécessite par conséquent une synchronisation entre les processus des divers éléments concernés. Nous examinerons ci-après les principales conséquences de cette synchronisation sur les processus de visualisation.

On peut considérer plusieurs types d'opérateurs de composition (cf. III.4)

- 1- Les opérateurs qui ne modifient pas le nombre d'éléments:
  Il s'agit généralement de simples tris, dont les résultats sont utilisés ultérieurement au cours d'un autre opérateur de composition.

- 2- Les opérateurs qui modifient le nombre d'éléments:
  * en plus, tel le découpage des polygones dans l'algorithme de Newell, Newell et Sancha (NNS72),
  * en moins, telle la suppression des segments cachés dans l'algorithme de Watkins (Wat70).

- 3- Les opérateurs convergents qui produisent en sortie un élément unique, telle la détermination de l'élément visible en un point donné (exploitation d'un tri précédent), ou encore la synthèse de plusieurs éléments dans les traitements d'anti-aliassage ou de transparence.

Ce dernier type d'opération constitue alors un véritable processus de synthèse "transversal" aux processus de visualisation des éléments. Il faut de plus ajouter à ceci que ces opérations de composition conditionnent généralement les processus individuels des éléments. Ainsi certains algorithmes d'élimination des parties cachées travaillant dans le repère de l'image nécessitent l'application préalable des opérateurs de projection

(Gv), d'autres encore nécessitent une transformation préalable des éléments initiaux en éléments plus simples afin de faciliter l'opération de tri.

Nous examinerons en détail ces problèmes au chapitre III, pour les principaux algorithmes publiés, mais il est d'ores et déjà évident que le choix de l'algorithme sera prépondérant pour l'ordonnancement du processus de visualisation. La figure 8 représente le cas de l'algorithme du "Z-buffer" (Cat74) appliqué au processus de la figure 4, dans lequel l'opérateur de composition est convergent et constitue un processus transversal.

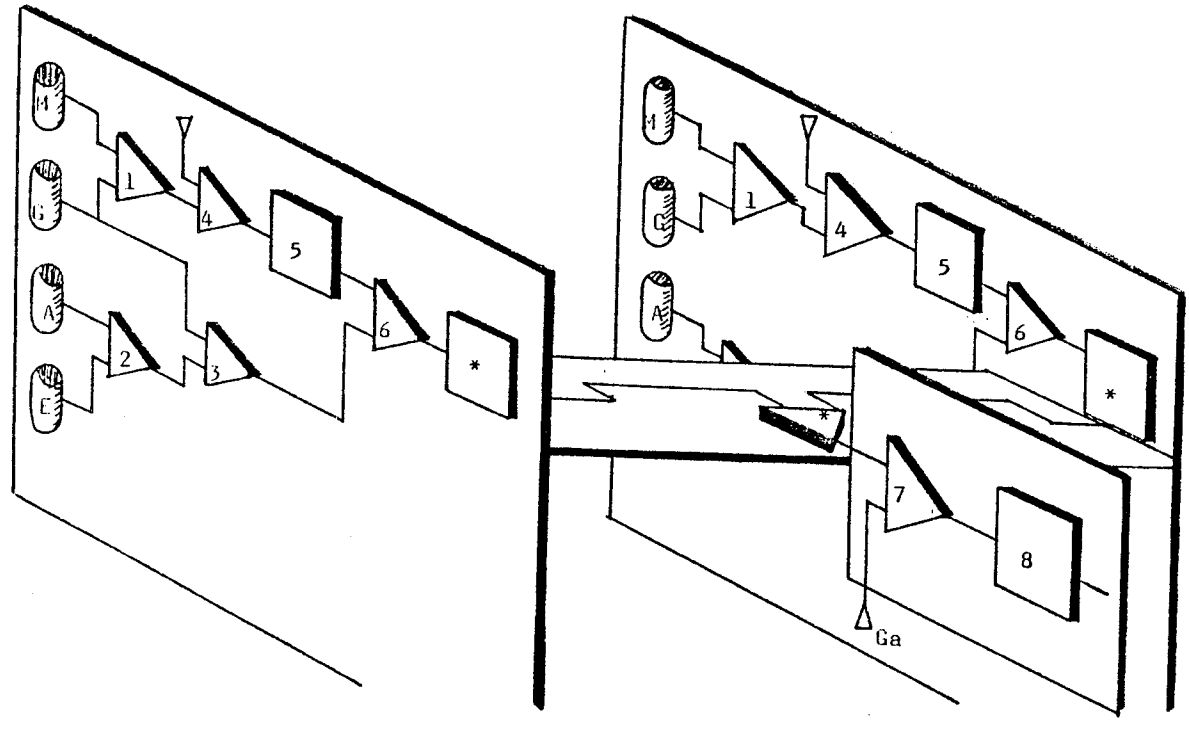


Figure II.8: Cas de l'algorithme du Z-buffer

Si n est le nombre d'éléments traités, l'influence de chaque attribut dans ce cas est: 5n + 2. On peut réduire à nouveau cette influence en utilisant les techniques proposées dans les paragraphes précédents. Néanmoins la, ou les, synchronisations nécessaires limitent la marge de manoeuvre. La pénétration des attributs nécessite quant à elle la

transformation du processus global, transformation qui peut être incompatible avec le processus interne de l'algorithme utilisé. Ainsi, si l'algorithme d'élimination des parties cachées permet de traiter l'anti-aliasing (Cat78), les attributs (A) et (E) doivent impérativement être synthétisés avant l'étape de composition.

Le processus de désignation pose à son tour des problèmes complexes. L'identification de l'élément visible au point désigné nécessite en effet:

- soit la "remontée" du processus complet, ce qui est généralement impossible compte-tenu de la perte d'informations provoquée par les opérateurs de composition,

- soit la pseudo-visualisation de tous les éléments afin d'effectuer une comparaison point par point,

- soit une combinaison des deux techniques précédentes
  * 1) remontée en appliquant les opérateurs inverses (si possible) jusqu'à l'opérateur de composition,
  * 2) pseudo-visualisation des éléments jusqu'à cet opérateur également,
  * 3) étude de coincidence.

La deuxième étape peut en outre être complètement supprimée par la mémorisation des seuls éléments visibles dans la partie commune du processus. Cette mémorisation doit bien entendu comporter l'identité de ces éléments (cf figure 9).

2.2. Architectures des synthétiseurs

La représentation schématique des processus de synthèse, utilisée dans les paragraphes précédents, met nettement en évidence les possibilités de parallélisme intra ou inter processus. Cependant l'exploitation effective de ce parallélisme potentiel dépend du nombre et de l'architecture des processeurs utilisés, et également de la répartition des tâches entre ceux-ci.

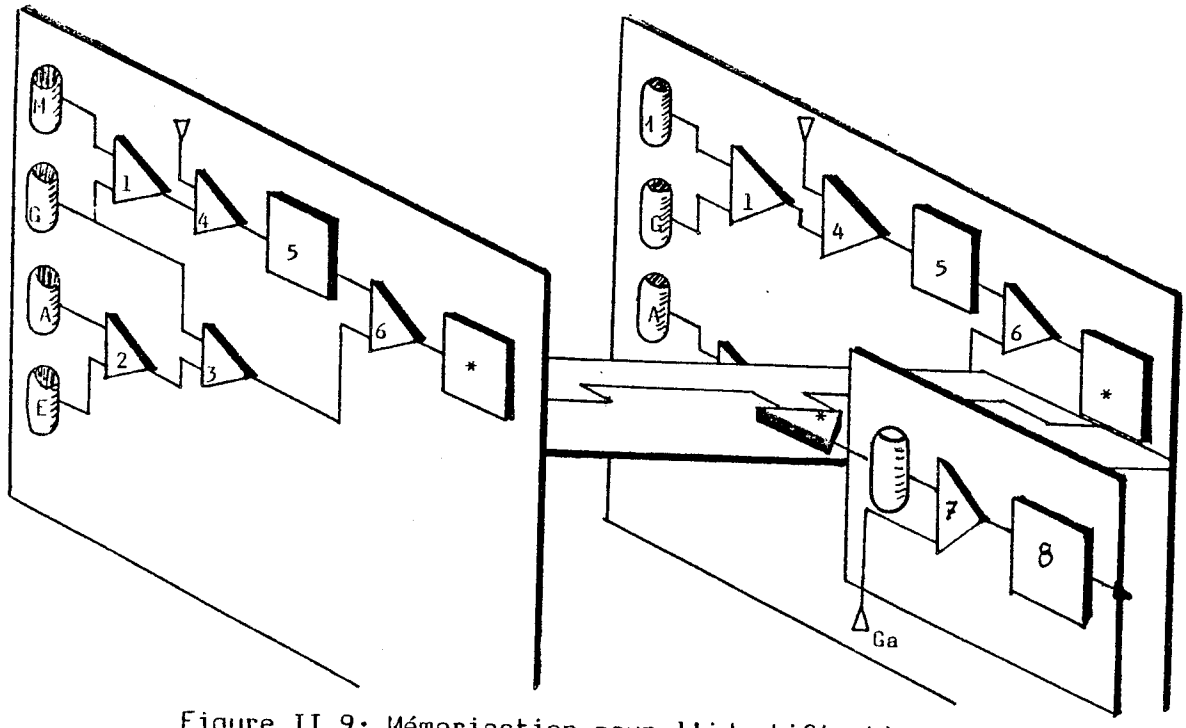


Figure II.9: Mémorisation pour l'identification

Entre le cas minimal dans lequel un processeur unique est chargé de toutes les opérations du processus et le cas maximal où un processeur est affecté à chaque opérateur, un grand nombre de compromis existent. Nous évoquerons ici les grandes tendances, et nous donnerons au chapitre suivant des exemples de réalisations actuelles utilisant ces divers compromis.

2.2.1. Séquentiel intégral

Pour simplifier l'exposé nous nous appuierons sur l'exemple d'un processus unique P appliqué à n éléments identiques. Ce processus comporte k opérations (p1,p2,..,pk) ordonnées de telle manière que:

$\forall\ l,j \in [1,k],\ l<j \Longrightarrow$(pl est antérieur à pj dans le processus)

On notera pj(i) l'application de la jème opération au ième élément.

L'organisation séquentielle intégrale est caractérisée par le fait qu'à un instant t donné, une seule opération (la jème) est en cours sur un élément unique le (ième). On peut représenter cette séquence par une itération.

```
pour i:=1 jusqu'à n
|    pour j:=1 jusqu'a k
|    |    pj (i)
|    fin
fin
```

2.2.2. Pipe-line

Il s'agit du cas le plus simple de parallélisme entre plusieurs processeurs dévolus à des phases successives du processus. Prenons le cas du partitionnement du processus précédent en deux sous-processus séquentiels successifs P1=(p1,..,ps) et P2=(ps+1,..,pk), attribués à deux processeurs. On peut représenter cette situation par les deux algorithmes ci-dessous.

```
pour i:=1 jusqu'à n
|    pour j:=1 jusqu'à s
|    |  pj(i)
|    fin
fin
```

Premier processeur

```
pour i:=1 jusqu'à n
|    pour j:=s+1 jusqu'à k
|    |  pj(i)
|    fin
fin
```

Second processeur

La bonne exécution simultanée de deux ou plusieurs processus nécessite une synchronisation qui peut être assurée de trois manières distinctes.

- 1- Si les processus Pj sont de périodes strictement égales, leur exécution peut s'effectuer en parfait synchronisme, avec toutefois une période de retard pour chaque nouvelle étape. La figure 10 illustre une période de pipe-line synchrone dans laquelle la j-ème partie des processus de l'élément i s'éxécute dans Pj alors que l'étape précédente de l'élément i+1 s'exécute dans Pj-1.

Ce type de pipe-line est particulièrement adapté aux implémentations cablées, pour lesquelles la période est souvent fixée par une horloge commune. La période du processus total est ainsi divisée par le nombre d'étages de pipe-line.

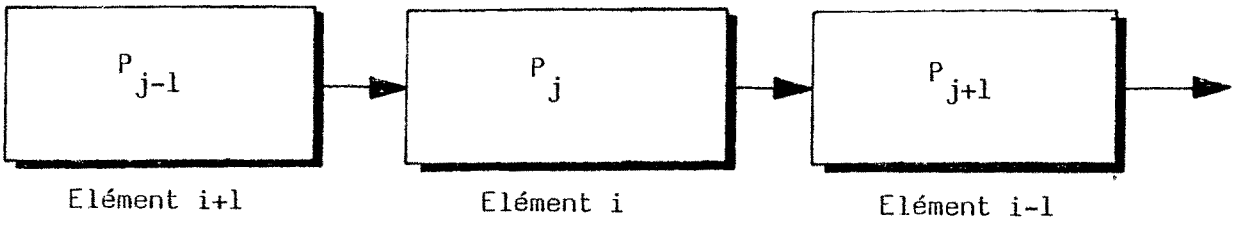


Figure II.10: Pipe-line synchrone

- 2- si un processus PJ nécessite une période supérieure à celle de son successeur PJ+1, il doit comporter un mécanisme d'attente. La période de l'ensemble est alors égale à la plus longue période.

- 3- Si l'on ne peut assurer que

$$\forall J, \text{période } (Pj) >= \text{période } (Pj+1),$$

il faut prévoir un mécanisme d'attente au niveau Pj, ou insérer entre Pj et Pj+1 une mémoire tampon susceptible d'absorber l'ensemble du décalage.

Cette dernière solution est la plus courante dans le cas des processus de synthèse d'image comportant plusieurs étapes "programmées", pour lesquelles il est impossible de calculer une période. La figure 11 schématise cette architecture de pipe-line asynchrone dans laquelle chaque processeur est quasiment indépendant des autres.

A l'inverse, si un processus de synthèse comporte une opération de mémorisation, en vue de privilégier certaines opérations ou d'assurer l'interface entre des éléments distincts (cf.2.2.1), cette structure de données peut assurer le rôle de tampon entre deux étapes du processus. La figure 6 représentait cette possibilité de scinder le traitement en deux processus asynchrones: construction, et prise de vue/affichage.

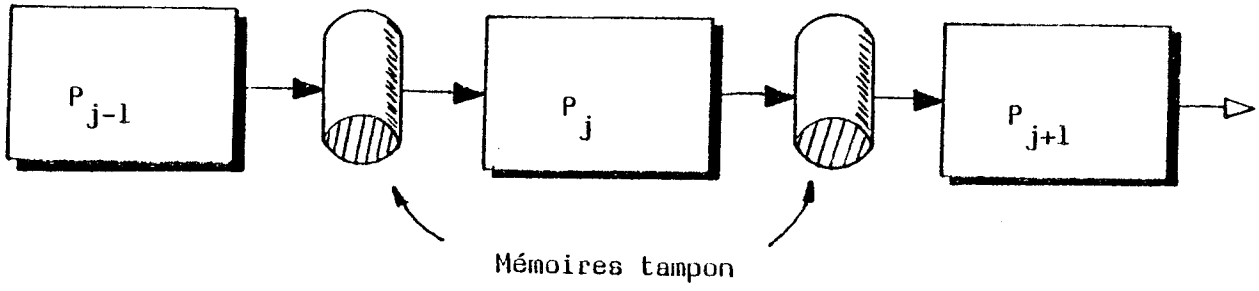


Figure II.11: Pipe-line asynchrone

A noter la différence de niveau existant entre le synthétiseur lui-même tel que nous l'avons défini, qui est maître de tout le processus concernant l'élément traité, et les différents processeurs internes attachés à des opérateurs élémentaires particuliers afin d'accroître les performances d'ensemble.

2.2.3. Parallélisme intra-processus

La figure 12 qui reprend l'exemple du processus présenté en 2.1, fait clairement apparaître des possibilités de parallélisme au niveau des opérateurs.

L'exécution en parallèle des opérations 1 et 2 puis 3 et 4 permet ainsi de réduire à 6 l'influence des attributs M, G, A et E, au lieu de 8 dans l'approche séquentielle. Il est clair en outre, que la pénétration des attributs favorise le parallélisme alors que la mémorisation intermédiaire est plus adaptée à une architecture pipe-line. Nous verrons au cours des chapitres qui suivent des exemples de réalisations fondées sur une architecture pipe-line, comportant un parallélisme intra-processus.

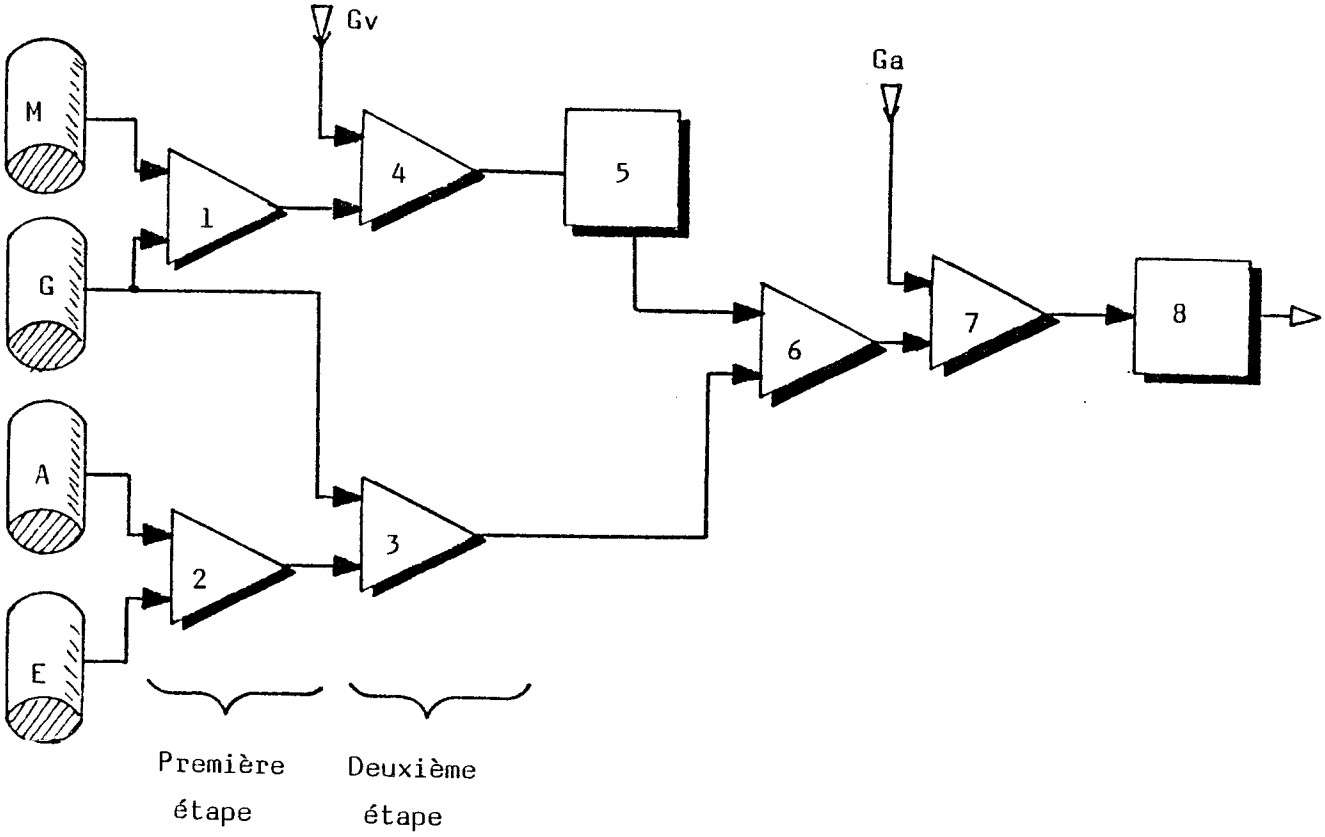


Figure II.12: Possibilités de parallélisme

2.2.4. Parallélisme inter-processus

Il s'agit là du cas le plus intéressant du parallélisme puisqu'il permet de traiter simultanément tous les éléments de la maquette, mais également le plus coûteux et aussi le plus difficile à mettre en oeuvre compte-tenu du nombre de processeurs nécessaires. Cette difficulté provient essentiellement de la nécessité de synchroniser tous les processus au moment d'une opération de composition. Celle-ci constitue un véritable "processus transversal" qui peut, à ce titre, présenter à son tour une architecture, séquentielle, pipe-line ou parallèle. La figure 13 représente cette organisation bi-dimensionnelle.

Il est possible de combiner les deux types de parallélisme. Cependant, si le parallélisme intra-processus pénètre plus profondément que l'opérateur de composition, plusieurs processus transversaux doivent être envisagés. Cette possibilité conduit à des architectures assez complexes, dont la figure 14 est un exemple représentant une solution possible pour

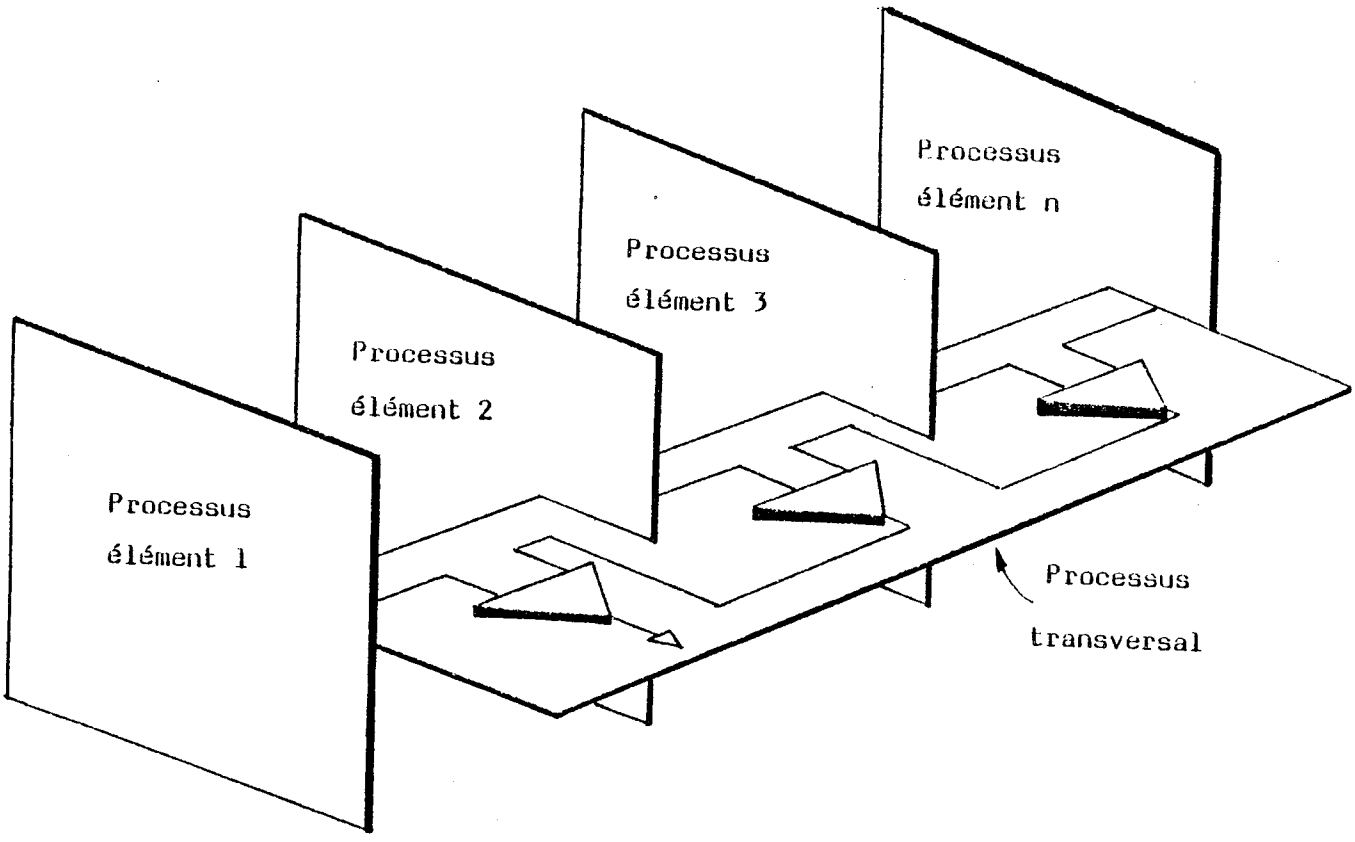


Figure II.13: Parallélisme inter-processus

l'utilisation du parallélisme dans un algorithme tel que celui proposé par Newell, Newell et Sancha (NNS72) (voir chapitre suivant).

3. Organisation d'un synthétiseur multi-processus

3.1. Les problèmes inhérents aux processus multiples

Deux problèmes résultent de la co-existence de processus différents

- la difficulté de la synchronisation nécessaire aux opérateurs de composition,

- la nécessité de réduire le nombre d'opérateurs élémentaires nécessaires.

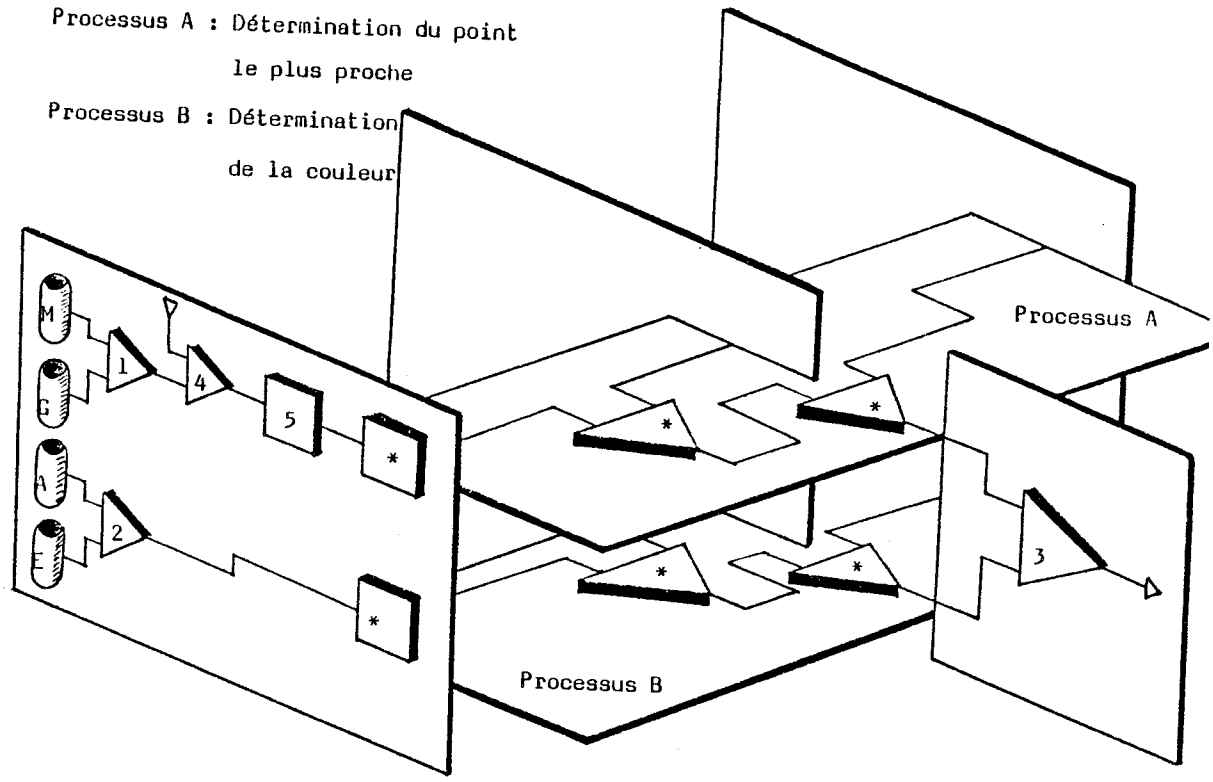


Figure II.14: Parallélisme intra et inter-processus combinés

3.1.1. La synchronisation de processus différents

Si les processus attachés aux divers éléments sont différents, les problèmes de synchronisation se posent en des termes encore plus aigus. Etant donné que la différence de processus est fréquemment liée à une différence d'éléments, la difficulté est encore accentuée par l'impossibilité de trouver des opérations de composition susceptibles de traiter des éléments quelconques. Ce problème se fait particulièrement sentir à propos des algorithmes d'élimination des parties cachées, qui ne traitent généralement qu'un type unique d'élément, par exemple les faces polygonales planes. La question qui se pose alors est de déterminer le point de convergence et de synchronisation des divers processus en présence.

Nous avons apporté dans une étude précédente (Mar79), quelques

éléments de réponse à ces problèmes, dans le cas particulièrement complexe de la visualisation de paysages, où les objets rencontrés sont très différents d'aspect et de morphologie (constructions, terrains, arbres, nuages, etc.).

Trois types de solutions sont évoqués:

- 1- Suppression des disparités.
  Il s'agit ni plus ni moins, de transformer au niveau de la maquette les éléments initiaux en un élément standard de façon à se ramener au cas des processus identiques. L'élément commun pourrait être dans ce cas la face polygonale plane, mais on imagine aisément le manque de réalisme d'un nuage modélisé ainsi.

- 2- Convergence au niveau du pixel.
  Il est clair que le pixel est l'élément terminal de tout processus de synthèse. On peut ainsi envisager des processus totalement différents, convergeant au niveau du pixel final, vers un algorithme du style "Z-buffer". Chaque processus doit, dans ce cas, être capable de calculer la profondeur pour chaque pixel de l'élément considéré, et de propager cette information à travers les opérateurs de synthèse.

- 3- Convergence intermédiaire.
  Cette troisième solution, plus difficile à mettre en oeuvre, est un compromis entre les deux précédentes. Elle est conditionnée par la possibilité de partitionner chacun des processus de telle manière que tous les sous-processus terminaux soient identiques. L'élimination des parties cachées est effectuée à ce niveau de coupure, ce qui permet d'appliquer le sous-processus terminal aux seuls éléments visibles. L'existence de cette solution dépend principalement des types d'éléments concernés (Mar79).

3.1.2. La réduction du nombre d'opérateurs élémentaires

Si l'on envisage de réaliser un synthétiseur d'image adapté à plusieurs situations différentes, il faut considérer:

- l'application ou la classe d'applications concernée, c'est à dire les synthétiseurs situés en amont,
- le matériel disponible, c'est à dire les synthétiseurs situés en aval.

L'application permet de déterminer :

* les p types d'éléments à considérer,
* les m performances (rapidité, qualité, complexité) requises pour chaque opération.

Le matériel permet à son tour de fixer:

* les types d'éléments disponibles,
* les opérateurs élémentaires disponibles.

Un synthétiseur spécifique à une classe d'application et à un matériel donné peut être défini comme une bibliothèque de n = p.m processus, adaptés aux p types d'éléments et aux m performances requises. Si chaque processus comporte en moyenne k opérateurs, le nombre total d'opérateurs nécessaires est

$$T = n.k$$

Si l'on désire, au contraire, réaliser un synthétiseur général susceptible de répondre au maximum de situations, il faut également considérer:

* les classes d'applications à satisfaire,
* les types de matériels disponibles.

On peut alors organiser l'ensemble en synthétiseurs indépendants adaptés à chaque situation. On obtient ainsi le schéma de la figure 15, dans lequel C représente l'univers nécessité par une classe d'application, M

représente l'univers proposé par un matériel donné, U(C,M) représente l'unité du synthétiseur permettant à la classe C d'utiliser le matériel M.

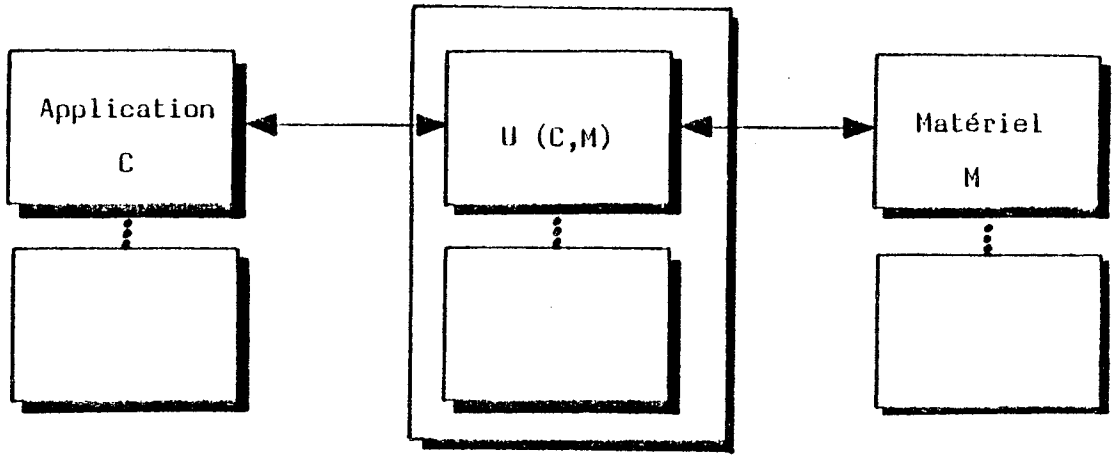


Figure II.15: Synthétiseurs adaptés à chaque situation

Si NC représente le nombre de classes, NM le nombre de matériels, n le nombre moyen de processus par classe, et k le nombre moyen d'opérateurs par processus; le nombre total d'opérateurs est :

$$T = NC * NM * n * k \qquad (1)$$

Cette approche combinatoire conduit bien entendu à un nombre prohibitif d'opérateurs, et nous examinerons ci-après deux méthodes d'organisation permettant de réduire de façon appréciable, le nombre d'opérateurs élémentaires.

3.2. L'organisation hiérarchique

3.2.1. Présentation

Si l'on examine un grand nombre de processus de synthèse correspondant à des situations classiques, on peut faire deux remarques intéressantes:

- 1- bien que dans un ordre très différent, une quantité de processus font appel à des opérateurs identiques tels que:

* mémorisation dans la base de données,
* multiplication par une matrice de transformation (synthèse M.G),
* remplissage d'une tache polygonale par une couleur uniforme (synthèse M.A),
* etc.

- 2- certains éléments "évolués" se décomposent en éléménts plus élémentaires ayant déjà fait l'objet d'un processus de synthèse. C'est le cas par exemple des polyèdres qui peuvent se décomposer en faces polygonales.

Ces deux remarques peuvent ainsi servir de point de départ à une nouvelle technique de répartition basée sur la séparation entre l'ordonnancement du processus et les opérateurs utilisés. La figure 16 représente cette approche dans laquelle une unité de contrôle dispose, pour chaque situation, d'un schéma de comportement exprimant l'enchaînement des opérations, alors que les deux autres unités sont en fait des "banques d'opérateurs" contenant les opérations elles mêmes. La ventilition des opérations en trois unités, si elle n'est pas indispensable, procure néanmoins au système une grande modularité, et de nombreux avantages qui seront évoqués dans les lignes qui suivent.

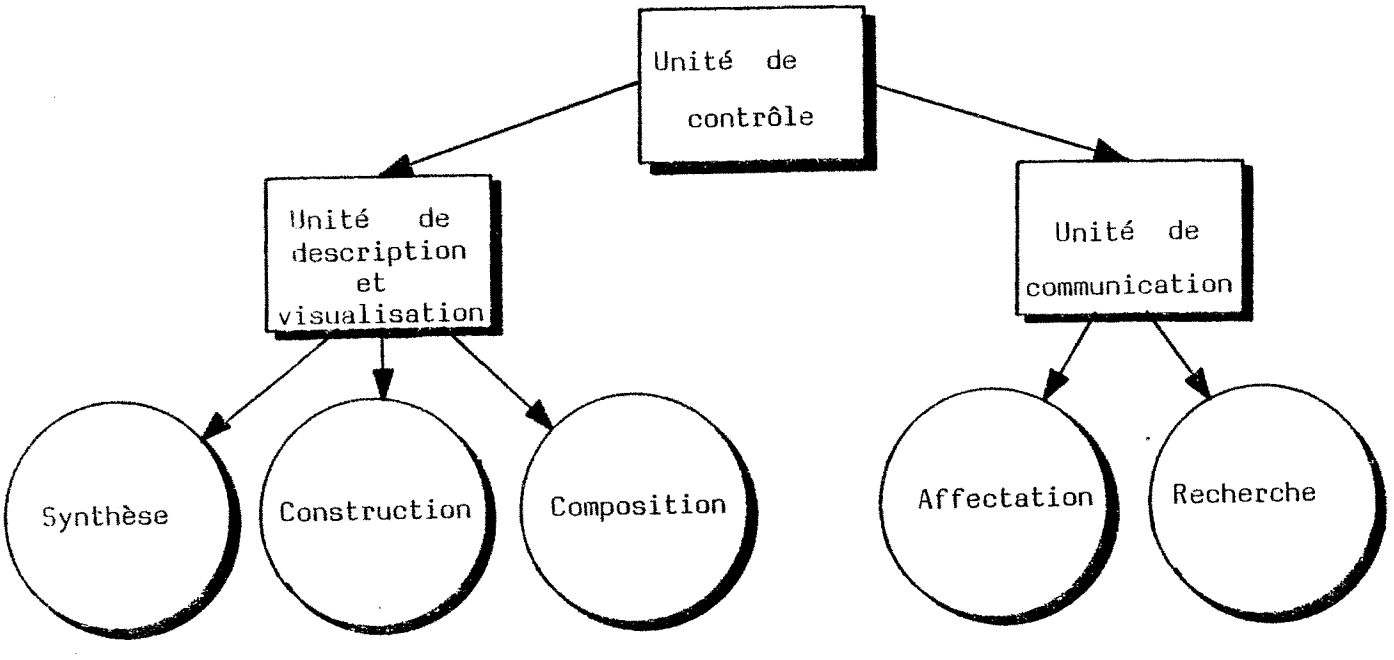


Figure II.16: L'organisation hiérarchique

3.2.2. Unité de description et de visualisation

L'unité de description et de visualisation, assure l'indépendance, vis à vis des dispositifs de saisie et d'affichage. Elle comprend ainsi des opérations dépendant du matériel et des opérations indépendantes. Elle constitue une banque regroupant trois types d'opérateurs élémentaires:

* les opérateurs de synthèse,
* les opérateurs de construction ou de modélisation,
* les opérateurs de composition ou de décomposition.

De nombreux arguments militent en faveur d'un regroupement description-visualisation.

- Les processus de description aussi bien que ceux de visualisation utilisent fréquemment les mêmes opérateurs élémentaires, en particulier les opérateurs de construction.

- Les processus de description nécessitent fréquemment la visualisation d'objets sur l'écran, servant de référence aux informations collectées, par exemple pour décrire des points clés d'un objet (centre de rotation, points d'attache, etc.) ou encore lorsque la morphologie de l'objet à décrire, dépend de son environnement.

- Les processus d'identification par désignation, impliquent généralement la visualisation ou la pseudo-visualisation des objets "identifiables".

3.2.3. Unité de communication

Compte tenu du fait que plusieurs mémorisations intermédiaires peuvent intervenir au sein des processus de visualisation et de description, un grand nombre de structures de données peuvent être nécessaires. Dans un schéma classique on trouve généralement:

- la maquette (2D ou 3D),

- l'épreuve constituant la vue (2D),

- la liste de visualisation souvent nécessaire au niveau du logiciel pour pallier les carences du matériel.

La multiplicité des structures de données mises en jeu dans un synthétiseur, même si elles sont parfois dégénérées, soulève quelques difficultés.

- Les attributs de structures et d'identité se voient répartis ou dupliqués, ce qui nuit à leur efficacité et complique leur gestion.

- La plupart des attributs sont mémorisés à plusieurs niveaux, ce qui surcharge la mémoire.

- Les mécanismes de gestion des diverses structures de données sont nombreux et différents, ce qui surcharge et complique la mise en oeuvre du synthétiseur.

La solution que nous préconisons pour pallier ces inconvénients est l'utilisation d'une base de données unique, banalisée de manière à permettre la structuration, l'affectation et la recherche d'informations graphiques quelconques, quelle que soit leur provenance ou leur destination. La gestion de cette base de données est confiée à une unité spécialisée que nous nommerons unité de communication. De manière similaire aux deux unités précédentes, cette dernière assure l'indépendance vis-à-vis des contraintes physiques telles que la représentation interne et le support des informations.

Outre ce rôle d'interface, l'unité de communication est généralement chargée du traitement des informations d'identité et de structure. Elle doit à cet effet comporter des mécanismes permettant,

* la création et la nomination de structures ou sous*structures,
* la suppression de sous-structures,
* la recherche des éléments appartenant à une sous*structure nommée.

Pour reprendre l'analogie avec une entreprise il s'agit là du "magasin

général", service chargé de stocker, de classer et d'étiqueter les produits qui lui sont communiqués. Les opérateurs élémentaires présents dans cette unité concernent essentiellement, la structuration, l'affectation, la recherche et la destruction des attributs des éléments ou de la maquette. Au fil des processus, de nouveaux attributs sont construits ou obtenus par synthèse, et peuvent être attribués à l'élément ou à la maquette concernés. Tous les attributs mémorisés sont directement accessibles à partir de l'identité de l'élément auquel ils appartiennent. L'allocation et la récupération de l'espace mémoire sont entièrement prises en charge par les opérateurs élémentaires.

3.2.4. Unité de controle

Outre le rôle d'ordonnancement des processus qui lui est attribué, cette unité prend en charge les tâches suivantes:

- Elle assure l'indépendance syntaxique et sémantique de l'application vis-à-vis des processus et des opérations élémentaires, en proposant un jeu de primitives standard, tel que celui présenté au premier chapitre.

- Elle décharge l'application de la responsabilité du transfert d'informations entre les unités de description/visualisation et de communication.

- Elle effectue les tests permettant de contrôler le déroulement des divers processus, et peut effectuer des opérations de "recouvrement d'erreur".

- Elle assure la liaison avec le synthétiseur "aval" en invoquant ses primitives d'attribution, de consultation, de visualisation et de description.

On peut considérer, à son tour, cette unité comme une banque de processus mis à la disposition des synthétiseurs "amont". Chaque processus d'un synthétiseur est ainsi assimilé à un opérateur élémentaire, disponible pour les synthétiseurs supérieurs. Un exemple d'imbrication hiérarchique des processus est représentée sur la figure 17 ci-dessous.

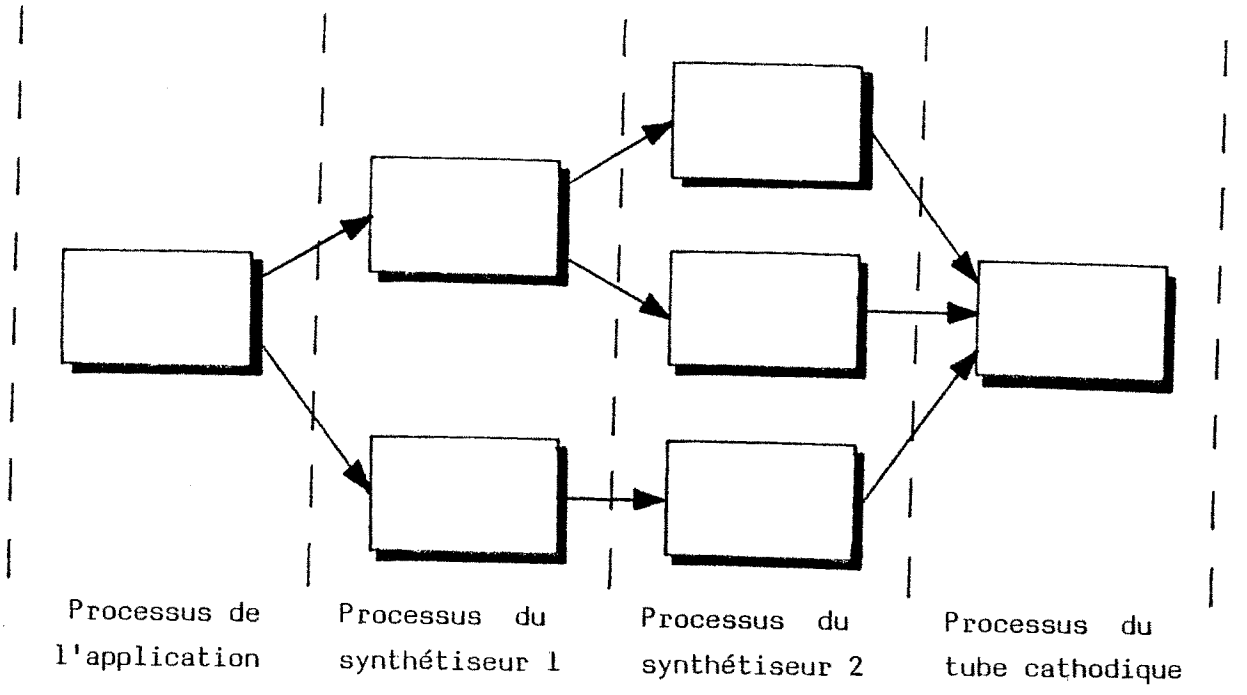


Figure II.17: Imbrication hiérarchique des processus

3.2.5. Les retombées de l'organisation hiérarchique

La figure 18 illustre un exemple d'organisation hiérarchique appliquée aux différents synthétiseurs (logiciel et matériel) d'un système de synthèse d'image.

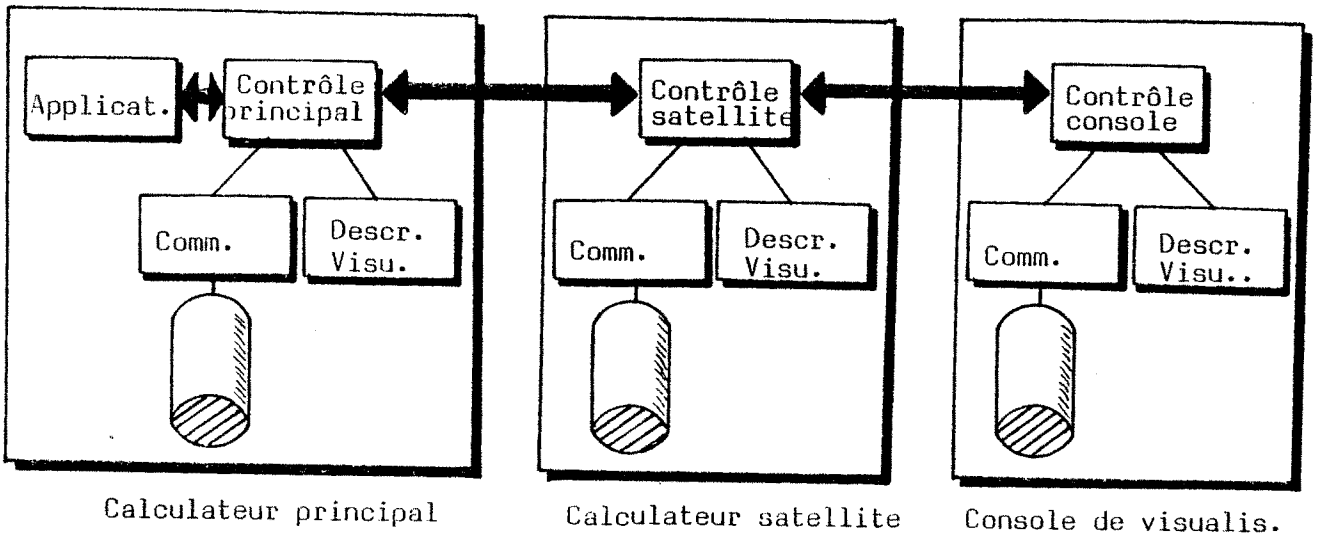


Figure II.18: Exemple d'organisation hiérarchique

Seule la liaison entre l'application et l'unité de contrôle du premier synthétiseur peut s'effectuer à l'aide des adresses des informations, alors que les transmissions entre les unités de contrôle nécessitent le transfert effectif des valeurs de ces informations.

En ce qui concerne l'indépendance vis-à-vis du matériel (c'est-à-dire en fait du synthétiseur aval), elle est assurée au niveau même de l'unité de contrôle qui invoque, en dernier ressort, un opérateur dépendant du matériel utilisé. Il est clair qu'un changement de matériel peut conduire à invoquer simplement un opérateur terminal différent, mais il peut conduire également à une refonte totale du processus afin d'utiliser au mieux les possibilités offertes par le nouveau poste de travail(voir l'exemple du paragraphe 3.3.3).

De la même manière la prise en compte d'un nouvel élément peut être obtenue à l'aide d'un opérateur unique permettant la conversion vers des éléments existants, ou encore faire l'objet d'un nouveau processus.

La mise en oeuvre de cette organisation soulève cependant un certain nombre de problèmes:

Le premier est relatif aux différentes structures de données utilisées dans chaque synthétiseur, pour lesquelles on ne peut aucunement préjuger des types (et par conséquent de la modélisation) des éléments qui seront ajoutés au fil des extensions logicielles ou matérielles. Cette situation confirme l'hypothèse d'une structure de données "banalisée" telle que nous l'avons proposé au paragraphe précédent.

Le second problème concerne l'unité de contrôle qui doit être conçue de manière à réduire au maximun la place occupée par les différents ordonnancements de processus. Il est à noter que fort peu de systèmes prennent en charge l'ensemble des processus. Dans ce cas l'application doit déterminer elle même l'enchaînement des opérations, et le synthétiseur se présente simplement comme une bibliothèque d'opérateurs. Dans le cas contraire, le système doit être capable de déterminer l'incidence de chaque

modification d'attribut, et d'exécuter le processus correspondant. Ainsi par exemple, après la modification de la couleur d'un élément dans la maquette, faut-il effacer l'image puis la reconstruire entièrement ? ; faut-il simplement effacer cet élément et le reconstruire ? ; ou encore suffit-il de transmettre la modificpation au matériel si celui-ci est capable de l'effectuer ? . Nous examinerons au chapitre V quelques solutions permettant de mettre en oeuvre ces mécanismes.

Un nouvel axe de recherche réside par ailleurs dans la détermination automatique de processus en fonction des types d'attributs et des opérateurs disponibles.

3.3. Le partitionnement en couches

3.3.1. La notion de couche

Bien que cette organisation soit sous-jacente à la plupart des systèmes graphiques actuels, la notion de couche reste toujours assez imprécise. Pour étayer la discussion nous nous appuierons sur la définition suivante.

Nous dirons qu'un synthétiseur est organisé en couches (de logiciel ou de matériel) s'il comporte un ensemble d'unités de traitement telles que l'univers terminal de la ième unité soit l'univers initial de la (i+1)ème unité, et telles que chaque unité simule un synthétiseur complet pour les éléments de son univers.

Chaque unité, qui forme une couche, comporte donc sa propre structure de données, et propose des processus de visualisation, d'attribution, de consultation et de description des éléments concernés. Cette notion de couche ne doit par conséquent pas être assimilée à une simple partition de processus, puisqu'on trouve dans chaque couche, l'ensemble des processus complets assurant la conversion des éléments initiaux vers les éléments acceptés par la couche suivante. Il faut noter que les couches possèdent des processus de visualisation totalement indépendants, invoqués par des

primitives distinctes. Il en est d'ailleurs de même pour les processus d'attribution, de consultation, et de description. Il s'agit ni plus ni moins d'une partition (fictive) du synthétiseur initial en sous-synthétiseurs successifs comme l'illustre la figure 19. Ces pseudo-synthétiseurs partagent tous le même processeur, ils sont simplement simulés par logiciel.

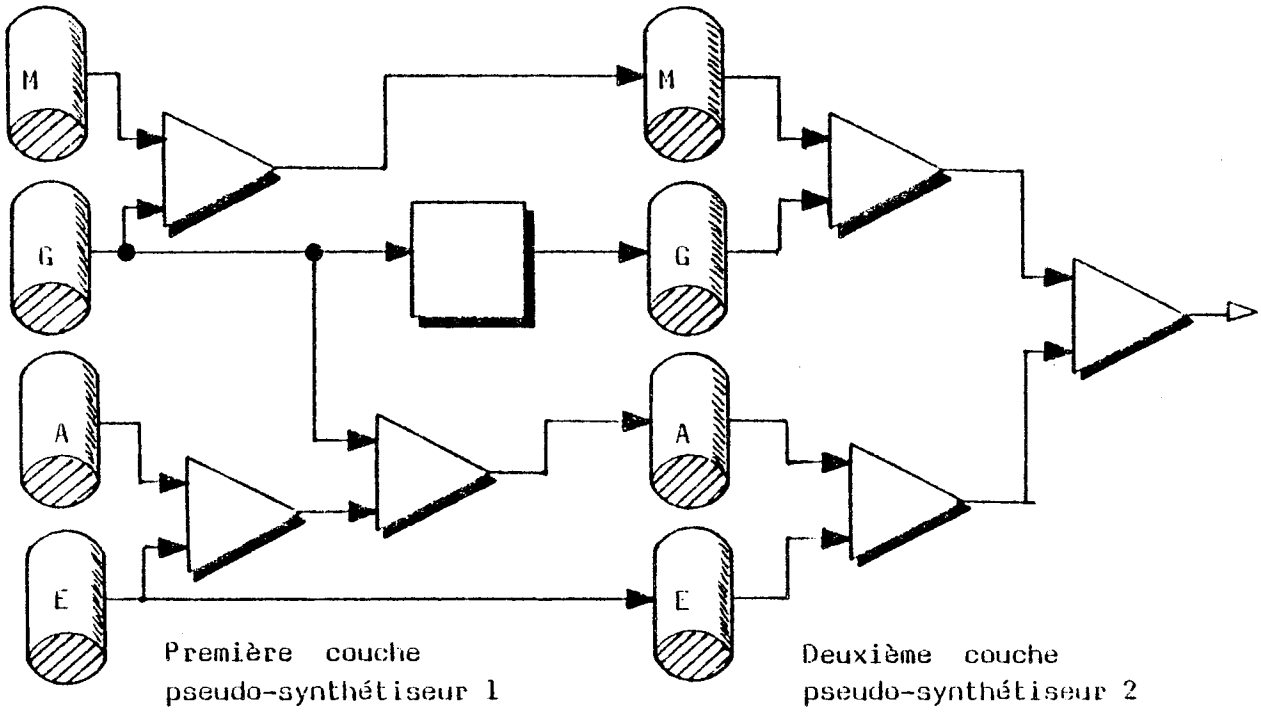


Figure II.19: Partitionnement du processus

3.3.2. Le partitionnement

Dans le cas le plus simple on cherche à partitionner le synthétiseur en deux couches relativement indépendantes. Ce découpage s'effectue en choisissant un univers intermédiaire S standard susceptible de "convenir" au plus grand nombre de situations (applications et matériels).

Chaque unité de contrôle U(C,M) est alors découpée en deux sous-unités:

U(C,S) et U(S,M)

Le nombre d'opérateurs nécessaires se réduit alors à

$$T=(n * NC * k1) + (n2 * NM * k2)$$

où k1 représente le nombre moyen d'opérateurs de la première couche,
k2 représente le nombre moyen d'opérateurs de la deuxième couche,
n2 représente le nombre moyen de types d'éléments acceptés par la deuxième couche.

Dans le cas particulier où

$$k1 = k2 = k \text{ et } n2 = n$$

on obtient

$$T=n * k * (NC + NM) \qquad (2)$$

ce qui, comparé à l'expression (1), devient intéressant à partir de 2 classes et 2 terminaux.

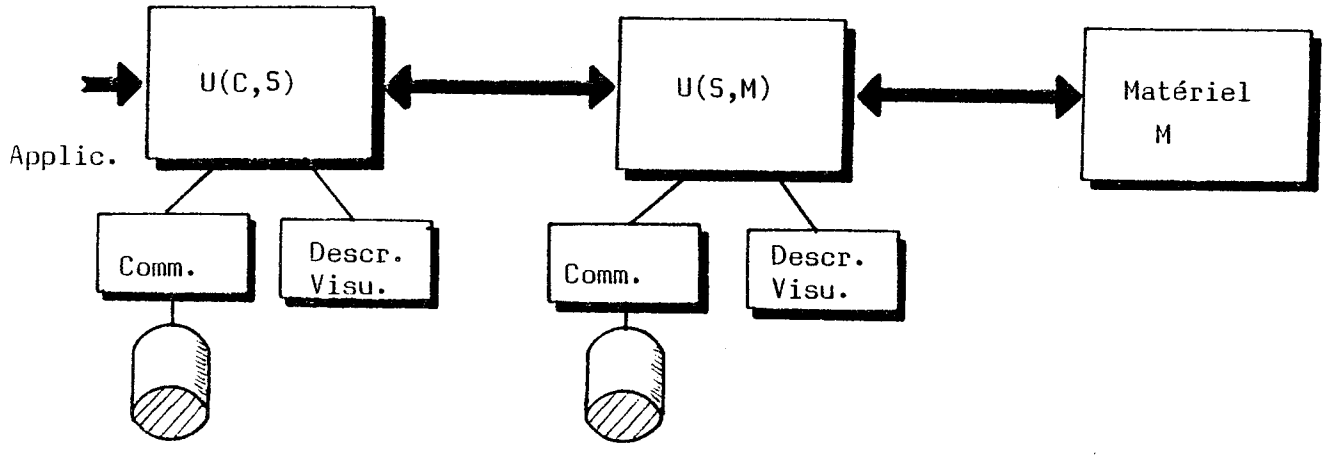


Figure II.20: Partition en deux couches

La figure 20 représente le schéma ainsi obtenu, dans lequel la couche "amont" U(C,S) est seule dépendante de l'application, et la couche "aval" U(S,M) est seule dépendante du matériel. Si ce type de répartition est étendu à plus de deux couches, les couches intermédiaires deviennent totalement indépendantes.

L'organisation qui résulte de cette approche présente de nombreux avantages:

- 1- Chaque couche, si elle est suffisamment symétrique et cohérente, permet de définir des pseudo-synthétiseurs hiérarchisés, spécifiques à un "niveau" d'éléments. Par exemple, Michel Lucas (Luc77) propose de considérer:
  * un logiciel élémentaire composé lui même de deux couches: l'interpréteur (couche aval dépendante du matériel) fondé sur des éléments de trés bas niveau: points, vecteurs, caractères; le module de communication (couche intermédiaire indépendante) fondé sur des éléments 2D un peu plus évolués et structurés: sections de points, textes, messages.
  * des logiciels de description et de préparation à la visualisation (couches "amont" dépendantes de l'application) adaptés aux diverses classes d'applications à considérer.

- 2- L'indépendance vis-à-vis de l'application est assurée par la première couche en fixant définitivement son univers terminal S. La prise en compte d'une nouvelle classe d'application C ne nécessite en moyenne que l'écriture de n*k nouveaux opérateurs.

- 3- L'indépendance vis à vis du terminal est assurée de façon analogue par la dernière couche en fixant son univers initial S. La prise en compte d'un nouveau terminal demande alors ns*ks nouveaux opérateurs.

L'ensemble du système se présente alors comme une suite de couches, dont certaines sont dues à une séparation physique des synthétiseurs, et d'autres à une séparation simulée par logiciel sur un processeur unique. La figure 21 ci-dessous représente une situation de ce type.

Il existe cependant une différence fondamentale dans les modes de transmission entre ces différentes couches. Alors qu'une information peut être communiquée à l'aide de son adresse entre C1,C2 et C4,C5, elle doit impérativement être transmise "par valeur" entre C2,C3 et C3,C4. La tendance

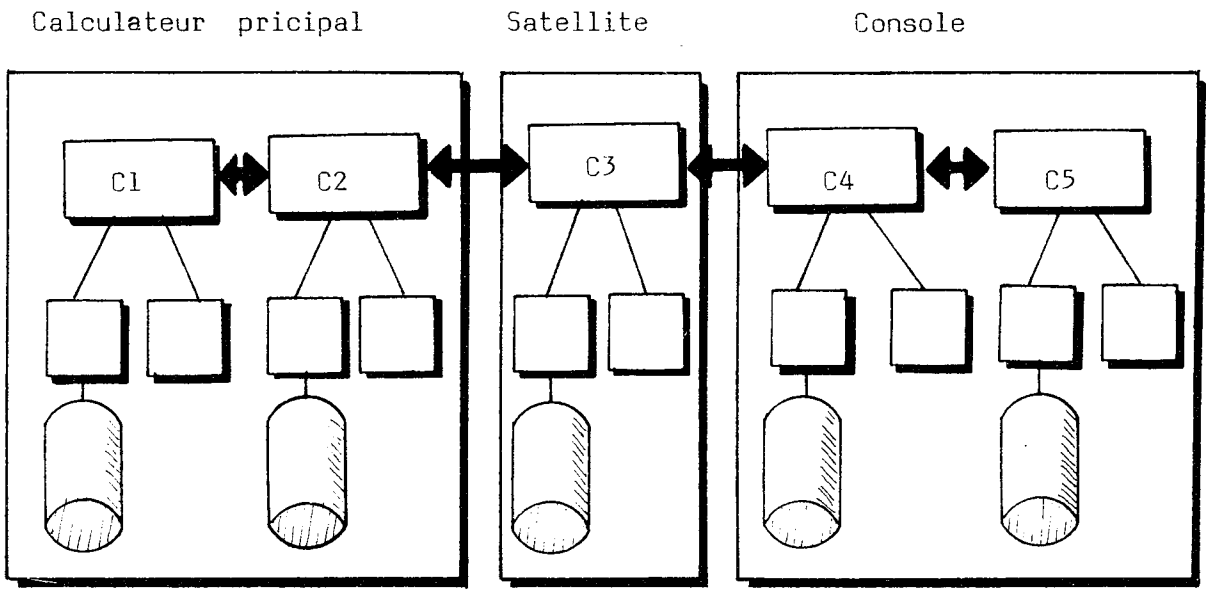


Figure II.21: Organisation d'un système en couches

à minimiser ces transmissions pénalisantes, revient à inclure un plus grand nombre d'opérateurs dans les couches inférieures, c'est-à-dire à opter pour une pénétration des attributs non synthétisés.

3.3.3. Les limitations du partitionnement en couches

La notion de couche est apparue pour satisfaire deux besoins essentiels: la réduction du nombre total d'opérateurs et l'indépendance vis-à-vis des matériels et des applications. Cependant si l'on passe du dessin au trait aux images, le prix de ces avantages est relativement conséquent.

En premier lieu on peut remarquer les transformations successives subies par les éléments à travers les différentes couches. Cette multiplicité est préjudiciable en temps et en place mémoire puisque les étapes intermédiaires sont généralement mémorisées pour les besoins de l'interaction.

Les informations de structure et d'identité se voient réparties entre différentes couches, ce qui complique considérablement leur gestion et limite leur utilisation et leur transmission jusqu'aux niveaux les plus bas.

L'utilisation des possibilités offertes par le matériel est limitée par le goulot d'étranglement constitué par la fixation, a priori, des univers intermédiaires. D'une manière plus générale, on peut dire que les propriétés du synthétiseur et même du système entier sont conditionnées par celles de la couche (logicielle ou matérielle) la moins "performante". A l'heure où les progrès considérables de la micro électronique confèrent aux terminaux à balayage de trame des possibilités de plus en plus grandes, cette limitation nous semble très pénalisante dans un système de synthèse d'image.

- La hiérarchie induite par les couches sur les univers est purement fictive et ne correspond plus nécessairement à des "niveaux" de complexité ni même à un regroupement par dimension 2D ou 3D. Certains terminaux proposent par exemple des fonctions de manipulation dans l'espace qu'une couche "2D" ne permettrait pas d'utiliser.

- Les processus de visualisation permettant de passer de la description des éléments à leur image sur l'écran ne peuvent être découpés a priori en étapes identiques. Celles-ci dépendent la plupart du temps : du matériel utilisé, du type des attributs composant l'élément, et des performances souhaitées.

Nous illustrerons ces remarques par un exemple caractéristique. Soient quatre synthétiseurs terminaux T1,T2, T3, T4 tels que:

* T1 soit limité à l'affichage d'un point avec une couleur donnée,
* T2 comporte également cette possibilité plus l'affichage de carrés de couleur uniforme,
* T3 comporte la possibilité de remplir les points intérieurs d'un polygone avec une couleur uniforme,
* T4 comporte la possibilité d'afficher une tache circulaire uniforme exprimée par son centre et son rayon.

Si l'on désire réaliser un synthétiseur logiciel capable de visualiser sur ces terminaux, un disque de couleur uniforme décrit à partir de son centre et de son rayon, on peut utiliser les 4 processus de visualisation

ci-dessous (figure 22), dont seules les grandes étapes de construction sont représentées. L'impossibilité de trouver, dans ces quatre processus, une étape commune pouvant justifier la présence de deux couches, apparaît clairement.

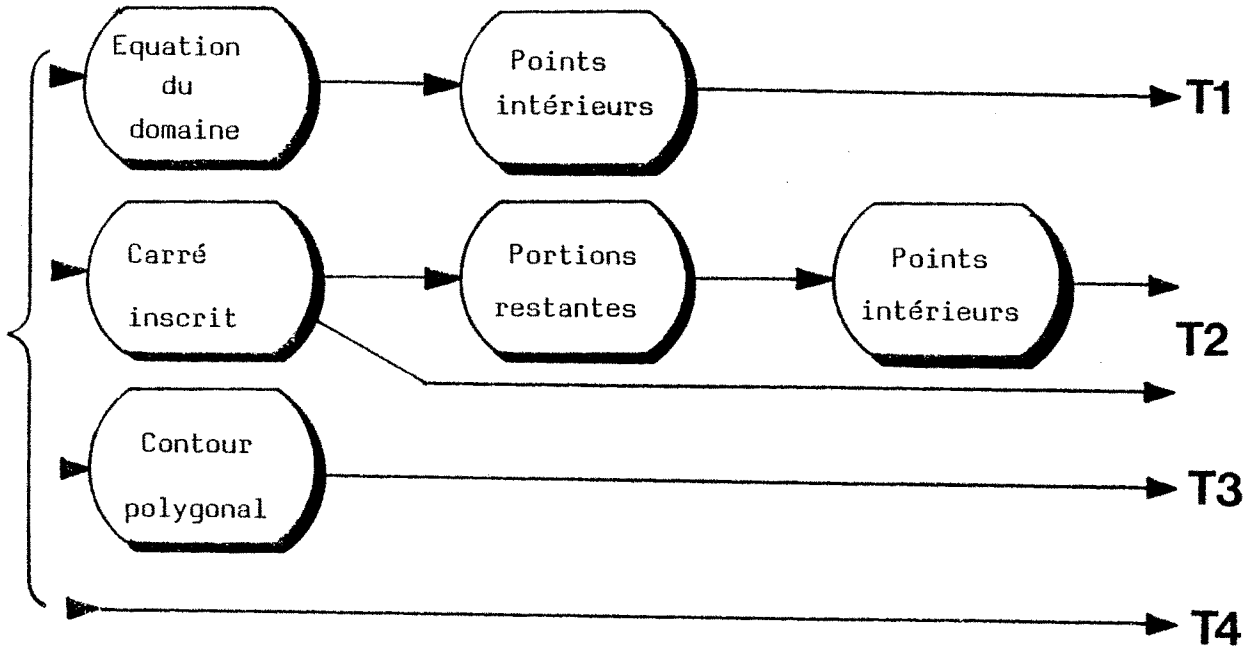


Figure II.22: Visualisation d'un disque uniforme

Si l'on désire à présent, visualiser ce même disque recouvert d'une texture à damier, le problème se trouve à nouveau complètement changé. On constate sur ce deuxième exemple que le processus est également dépendant de l'aspect de l'élément, en particulier pour le terminal T4 pour lequel on revient à une génération point par point (figure 23).

Afin d'illustrer les effets produits par une répartition en couches sur les exemples précédents nous choisirons comme élément standard intermèdiaire, la tache 2D polygonale uniforme. Ce choix correspond au désir de partitionner en deux le synthétiseur initial, de telle façon que le sous-synthétiseur amont soit unique (cf figure 24).

Si les avantages de cette approche sont évidents, les inconvénients ne

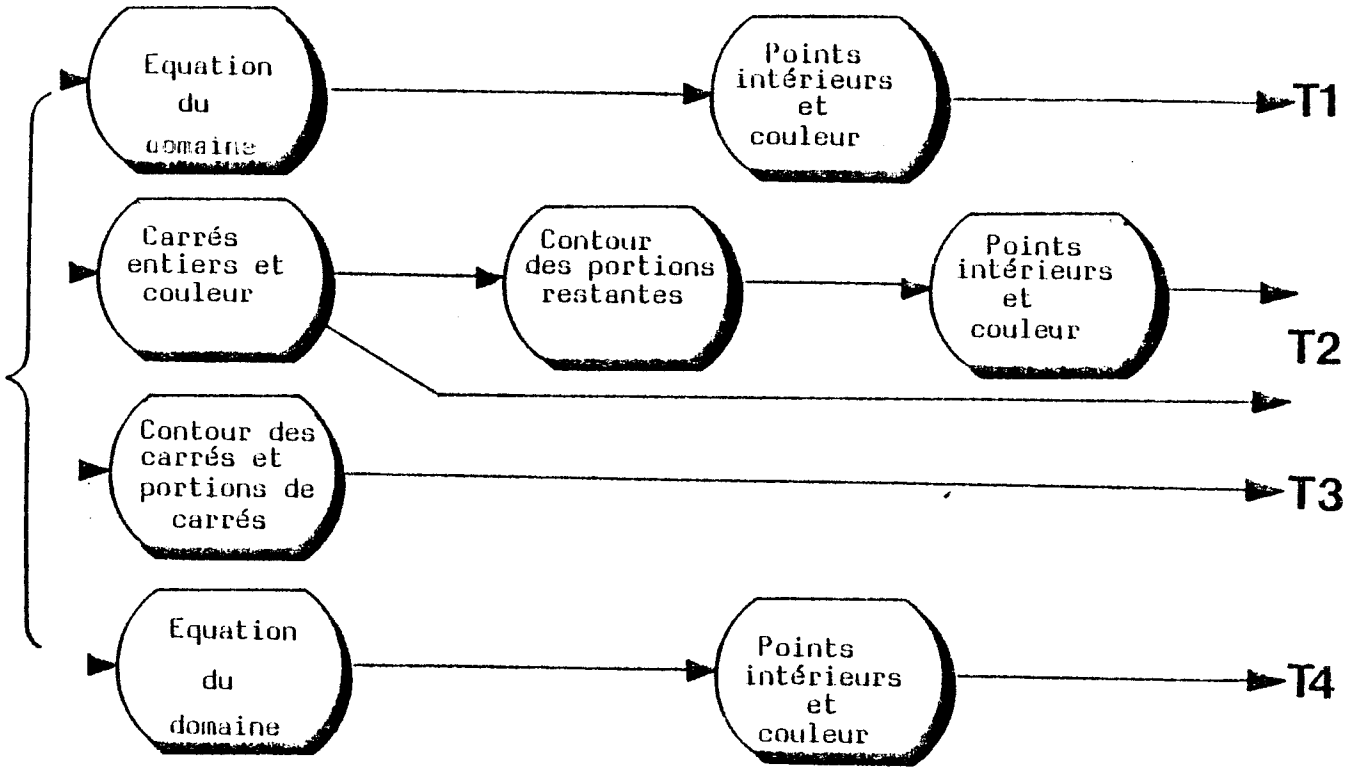


Figure II.23: Visualisation d'un disque à damier

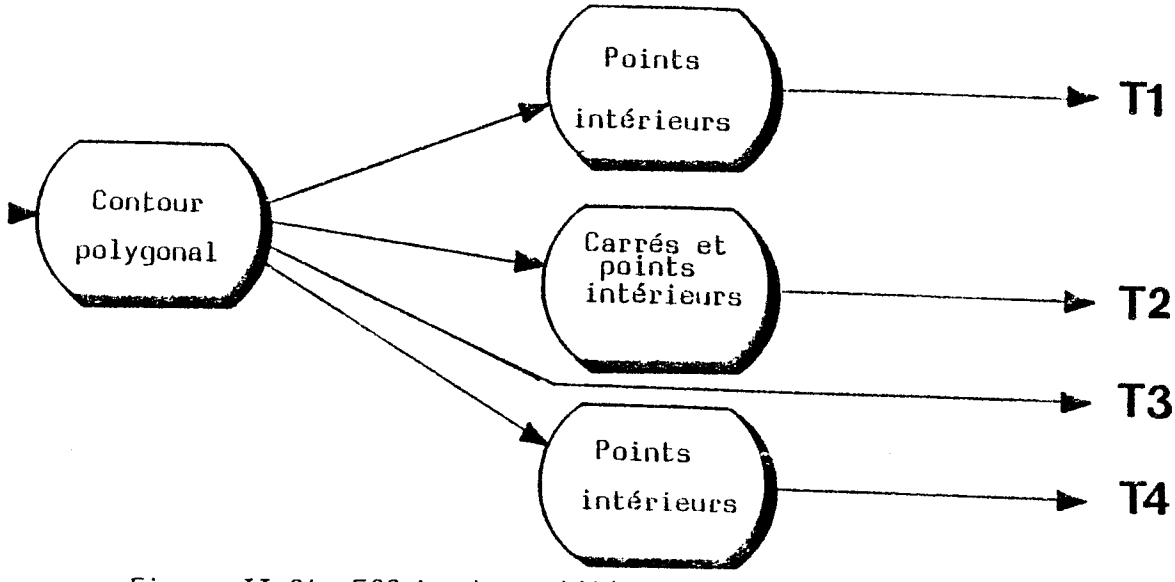


Figure II.24: Effets du partitionnement en couches

le sont pas moins.

- Dans le cas de T1 et T2, elle entraîne: l'approximation plus ou moins bonne du contour par un polygone et le "remplissage" de ce dernier qui aurait pu être effectué plus rapidement sur le cercle même, grâce à

l'utilisation d'algorithmes appropriés (Dor79), (SKK79), (MaB79).

- Dans le cas de T4, elle supprime la possibilité d'utiliser directement le matériel.

En résumé, le partitionnement d'un synthétiseur en couches revient à fixer a priori, pour toutes les situations quelles qu'elles soient:

- les mêmes niveaux de mémorisation des attributs,

- la même pénétration des attributs non synthétisés,

- les mêmes étapes imposées de pipe-line, fermant ainsi la porte à des réalisations plus parallèles.

4. Les différentes approches face aux objectifs visés

Les paragraphes précédents nous ont permis de mettre en valeur le nombre et la diversité des approches possibles et leur principales incidences. La démarche suivie ici est inverse puisqu'il s'agit de prendre en compte les objectifs visés lors de la conception et la réalisation d'un système de synthèse d'images, et d'en déduire les approches les mieux adaptées. Les objectifs que nous étudierons sont les suivants:

* les performances requises,
* la puissance d'interaction souhaitée,
* l'indépendance et l'adaptativité nécessaires,
* le coût de développement.

Ces différentes contraintes jouent des rôles antagonistes et le choix final passe donc nécessairement par l'établissement de priorités dépendant de la classe d'applications que l'on désire privilégier ainsi que des moyens disponibles. Nous ne donnerons, dans les paragraphes qui suivent, que les grandes directions impliquées, étant bien entendu que chaque cas particulier peut donner lieu à des modulations plus ou moins importantes. Un tableau récapitulatif regroupe l'ensemble des incidences en fin de paragraphe, et quelques systèmes seront examinés sous cet aspect au chapitre III.

4.1. Les performances

Les performances requises concernent en général trois domaines concurrents:

- le temps de réponse,
- la qualité des images,
- la complexité des scènes.

La conception d'un système de synthèse d'image passe donc généralement par le choix d'un compronis, illustré sur la figure 25.a.

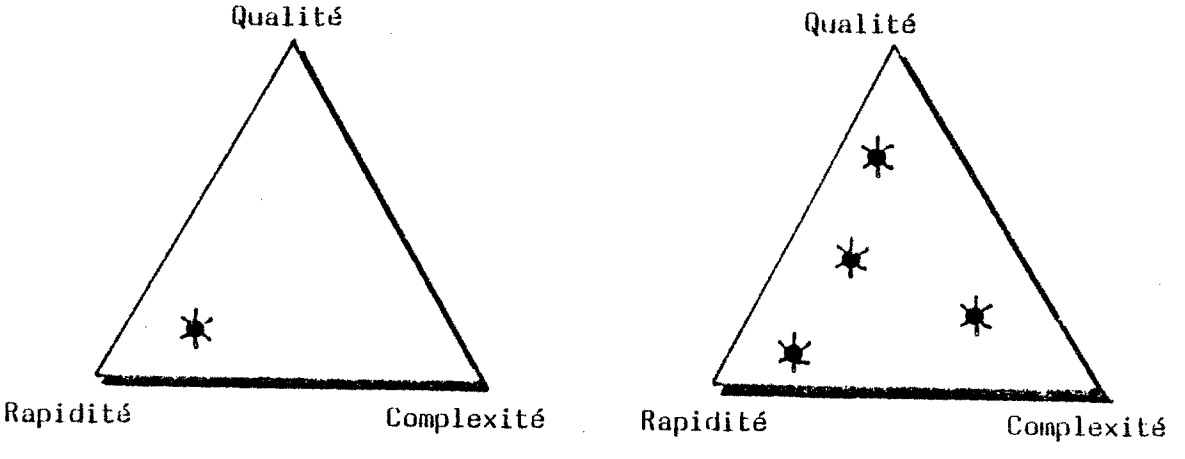


Figure II.25: Compromis des performances

Si ce choix favorise la qualité, (généralement au prix de traitements additionnels), il est clair qu'il va du même coup pénaliser les applications pour lesquelles le temps de réponse et la complexité de la scène sont prépondérants. En d'autres termes, la question soulevée ici est la suivante: peut-on concevoir un système adaptatif susceptible de répondre au plus grand nombre d'exigences ?

Si chaque élément est concerné par un processus qui lui est propre, on peut concevoir et offrir à l'utilisateur des processus possédant certaines propriétés fonctionnelles. En particulier des processus:

* "rapides" mais limités en qualité et en complexité,
* complexes mais limités en rapidité et en qualité,
* de qualité mais limités en rapidités et en complexité.

On obtient ainsi le schéma de la figure 25.b illustrant les choix offerts à l'utilisateur, dans un tel système.

Si l'on utilise une répartition en couches, la scission "a priori" des processus, limite la marge de manoeuvre à l'intérieur de chacune des parties, et impose globalement pour tous les processus, des performances semblables.

Quoi qu'il en soit, le même compromis rapidité-qualité-complexité se présente à nouveau au niveau de chaque processus et nous examinerons ci-dessous les incidences directes des performances requises sur le choix d'une architecture de système.

4.1.1. La rapidité

La grande quantité d'informations manipulées, ainsi que les calculs complexes nécessaires, notamment dans le cas d'images réalistes, font de cet objectif l'un des plus prioritaires. Deux précisions supplémentaires sont cependant nécessaires pour le choix d'une organisation:

- l'ordre de grandeur de la vitesse désirée, qui peut couvrir une plage allant de plusieurs minutes au temps réel (40 ms), pour la modification intégrale d'une image,

- les opérations pour lesquelles cette rapidité est réellement cruciale, par exemple les modifications géométriques ou les modifications d'aspect ou encore la prise de vue.

L'ordre de grandeur de la vitesse va conditionner directement le choix entre architecture séquentielle ou parallèle ce qui revient dans la plupart des cas à choisir:

- 1) un niveau dans l'échelle ci-dessous.
  * séquentiel
  * pipe-line asynchrone
  * pipe-line synchrone
  * parallélisme intra-processus
  * parallélisme inter-processus

- 2) la technologie des différents processeurs employés:
  * cablés,
  * micro-programmés,
  * programmés.

Les opérations privilégiées, quant à elles, vont dicter l'insertion de structures de données intermédiaires et la profondeur de pénétration des attributs concernés. Nous avons vu, par exemple que la modification de l'aspect d'un objet sera d'autant plus rapide que la synthèse de (A) avec les autres attributs sera tardive.

4.1.2. La qualité d'image

La recherche d'une grande qualité d'image plaide au contraire en faveur d'une faible pénétration. La perte énorme d'information provoquée généralement par l'opération de prise de vue (perte d'une dimension, découpage), implique que certaines fonctions relatives à l'esthétique de l'image soient effectuées préalablement. C'est le cas, notamment, des opérations "d'anti-aliassage"; de la détermination des ombres portées et des anamorphisme de textures, pour lesquelles de nombreux algorithmes ont été développés (cf. chapitre III). Cette opposition confirme d'ailleurs l'antagonisme existant entre rapidité et qualité.

En résumé le besoin de qualité oriente vers:

- une faible pénétration des informations,

- une réalisation programmée pour prendre en compte les phénomènes réels de la manière la plus exacte possible.

4.1.3. La complexité

Si l'opposition entre la complexité de la maquette et la rapidité du processus de visualisation est évidente, l'incidence sur la qualité de l'image mérite quelques remarques. La complexité de la scène (notion délicate à définir), engendre généralement une image complexe, composée d'un grand nombre d'éléments. Afin d'analyser correctement l'ensemble de la scène perçue, le cerveau humain doit diposer d'une image d'autant plus nette et réaliste qu'elle est "chargée". Ce besoin accru de qualité contribue à accentuer la perte de temps.

D'autre part le grand nombre d'éléments à considérer rend peu envisageable les réalisations cablées et à plus forte raison parallèles.

4.2. La puissance d'interaction

Nous prendrons ici le terme interaction dans son sens strict c'est-à-dire la partie de la description qui est effectuée par référence à l'image visualisée. Ce processus de description s'appuie sur deux opérations fondamentales:

- l'identification par désignation (description de I et S)

- la collecte d'informations (consultation de la structure de données)

4.2.1. L'identification par désignation

Cette fonction permet d'obtenir l'identité et la structure d'un ou plusieurs éléments désignés directement sur l'écran, à l'aide de dispositifs interactifs tels que réticule, photostyle, boule roulante, etc.

Cet objectif milite incontestablement en faveur d'une approche fondée sur une forte pénétration des attributs, accompagnée d'une mémorisation intermédiaire. Nous avons vu que cette approche permet de réduire les opérations nécessaires à la pseudo-visualisation, mais elle permet également

de conserver et de transmettre les attributs d'identité et de structure jusqu'aux niveaux les plus bas, voire même jusqu'au sein du matériel. L'identification peut s'effectuer alors directement au niveau du matériel ou des synthétiseurs logiciels les plus "proches" de l'image ce qui confère une grande efficacité à l'interaction.

4.2.2. La collecte d'informations

A travers la désignation d'un élément, le programme d'application peut exprimer:

* soit une identification pure (attributs I et S),
* soit la consultation d'un attribut associé à cet élément (M,A,G,E).

Cette faculté permet d'exprimer des actions s'appliquant, par exemple, à tous les éléments ayant le même aspect que celui désigné. Puisque la portée de telles actions peut atteindre la maquette entière et non plus l'image, l'utilisation d'une structure de données unique permet d'effectuer une correspondance directe entre les éléments désignés sur l'image et ceux de la maquette. La puissance et l'efficacité de l'interaction se trouvent par conséquent défavorisées par une répartition en couches dans laquelle la structure de données est répartie.

4.3. L'indépendance et l'adaptativité

Nous examinerons dans ce paragraphe les critères permettant de différencier les systèmes,

- généraux, c'est à dire susceptibles de servir une vaste gamme d'applications sur divers matériels,

- ou spécifiques, c'est à dire dévolus à une application ou une classe d'applications particulière et/ou un matériel donné.

Ces critères vont influer sur l'opportunité d'un partitionnement en couches.

4.3.1. L'indépendance

L'indépendance qu'il faut considérer ici est celle qu'un système de synthèse d'images confère à l'application vis à vis de lui-même, c'est à dire vis à vis:

- du matériel de visualisation et de saisie,
- du support et de l'organisation de la structure de données,
- de la modélisation interne des éléments.

Nous verrons au cours du chapitre VI qu'une indépendance syntaxique peut être obtenue, quelle que soit l'organisation choisie, à travers l'utilisation de structures de données "banalisées".

En ce qui concerne l'indépendance sémantique, liée à la nature des éléments manipulés, les problèmes sont beaucoup plus complexes, et voient leur solution dans la prise en charge et la détermination automatique des processus par le système. Le programme d'application ne dépend plus, dans ce cas, du type des éléments dont la visualisation peut être effectuée à l'aide d'une primitive unique.

4.3.2. L'adaptativité

Il s'agit ici de la faculté du logiciel à s'adapter aisément à des contextes d'utilisation variés, résultant du développement de nouveaux postes de travail. Cette faculté qui rend compte de l'utilisation optimale du matériel pour l'application considérée, joue en faveur:

- d'une réalisation programmée,
- d'un partitionnement en couches qui minimise les développements nécessaires pour passer d'un contexte à un autre.

4.4. Coût du développement

Le coût du développement, bien souvent prépondérant dans le choix d'un système, vient malheureusement sanctionner la plupart des qualités exposées ci-dessus.

Deux politiques peuvent être observées pour limiter le coût du développement, qu'il s'agisse de matériel ou de logiciel.

- 1- Limiter l'investissement total en fixant a priori les limites des extensions futures, à travers un partitionnement en couches. Cette organisation conduit en effet à utiliser des éléments de bas niveaux demandant par conséquent peu de matériel et peu de développements logiciels.

- 2- Limiter l'investissement initial, en réalisant un noyau minimal laissant la porte ouverte à toutes les extensions. L'organisation hiérarchique permet alors de repousser les problèmes liés au coût du développement, au niveau de chaque situation. Il est clair cependant que ces objectifs ambitieux conduisent à des développement importants et des difficultés de mise en oeuvre, et donc à un coût plus élevé.

4.5. Récapitulatif

Le tableau ci-après résume brièvement les grandes incidences des objectifs examinés ci-dessus. Les cases non remplies correspondent aux cas où l'incidence est insignifiante ou difficile à évaluer.

	Mémorisation intermédiaire	Pénétration des attributs	Partitionnement en couches	Architecture parallèle	Réalisation cablée	Prise en charge du processus
Rapidité d'ensemble	N	/	N	0	0	0
Rapidité d'opérations privilégiées	0	0	N	0	0	0
Qualité d'image	/	N	/	N	/	/
Complexité de la scène	/	/	/	N	N	/
Puissance d'interaction	0	0	N	0	/	0
Facilité d'utilisation	/	/	/	/	/	0
Facilité d'adaptation	/	/	0	N	N	N
Limitation du coût	/	/	0	N	N	N

CHAPITRE III

Etat de l'art et exemples de réalisations

1. Présentation

Afin de mieux dégager les originalités et les limites de notre expérience, nous consacrerons ce chapitre à une revue critique des diverses approches utilisées dans les principaux systèmes actuels. Nous prendrons ici le mot système au sens large et nous considèrerons aussi bien les composants matériels que logiciels. Cette revue, qui ne se veut aucunement exhaustive, sera effectuée à la lumière des concepts de base énoncés au premier chapitre, et des différentes approches caractérisées au chapitre précédent. Ainsi le choix et la présentation des systèmes est plus destinée à faire ressortir ces notions qu'à présenter les solutions techniques utilisées. Cependant, pour les cas où l'aspect technique est à la base du choix, et également pour fixer les idées et la terminologie, nous présenterons brièvement les principales techniques de réalisation des opérateurs élémentaires du logiciel et du matériel, avant d'aborder les différentes organisations, et les processus utilisés. Nous serons ainsi amenés à considérer successivement:

- les processus élémentaires du matériel,
- l'architecture du matériel,
- les processus élémentaires du logiciel,
- l'organisation du logiciel.

2. Les processus élémentaires du matériel

La synthèse d'une image sur un écran nécessite au moins deux synthétiseurs. L'un est le dispositif d'affichage lui-même (l'écran), qui exécute le processus d'affectation.

- L'autre est un dispositif d'entretien, destiné à rafraîchir constamment l'image, qui comporte au moins un processus d'affectation et un processus de visualisation.

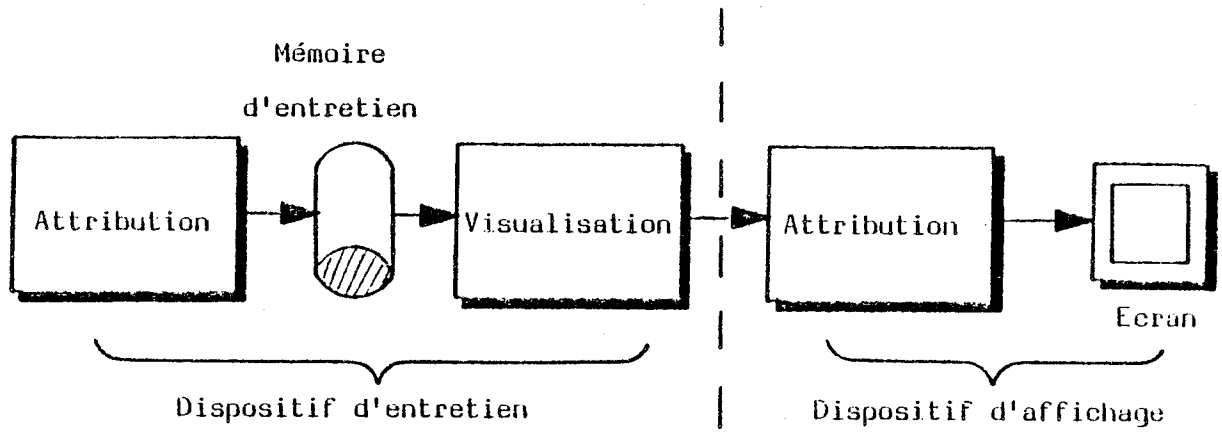


Figure III.1: Les dispositifs du matériel

Pour assurer la description et l'interaction, les consoles de visualisation intègrent également un processus de description permettant d'utiliser le ou les dispositifs de saisie disponibles, et un processus de consultation permettant de récupérer l'information saisie.

La technologie utilisée pour réaliser ces divers dispositifs conditionne directement le type des éléments manipulés, et nous lui consacrerons le premier volet de cette étude. L'évolution des techniques d'affichage, d'entretien et de saisie ayant fait l'objet de nombreux articles (Das79), (Luc77a), nous n'y reviendrons ici que pour énumérer brièvement les principaux dispositifs couramment utilisés à l'heure actuelle.

2.1. Les dispositifs d'affichage

Nous éluderons dans cette revue, les technologies "marginales" telles que: (les panneaux à plasma, les écrans lasers, les cristaux liquides, etc.) pour ne considérer que le cas du tube à rayons cathodiques qui constitue encore 99è des dispositifs d'affichage.

Deux technologies concurrentes, fondées sur deux techniques de balayage, se partagent actuellement le marché:

- le balayage cavalier dans lequel le faisceau d'électrons suit le contour du dessin à afficher,

- le balayage de trame dans lequel le faisceau d'électrons parcourt l'écran tout entier ligne par ligne, quelle que soit l'image affichée (figure 2).

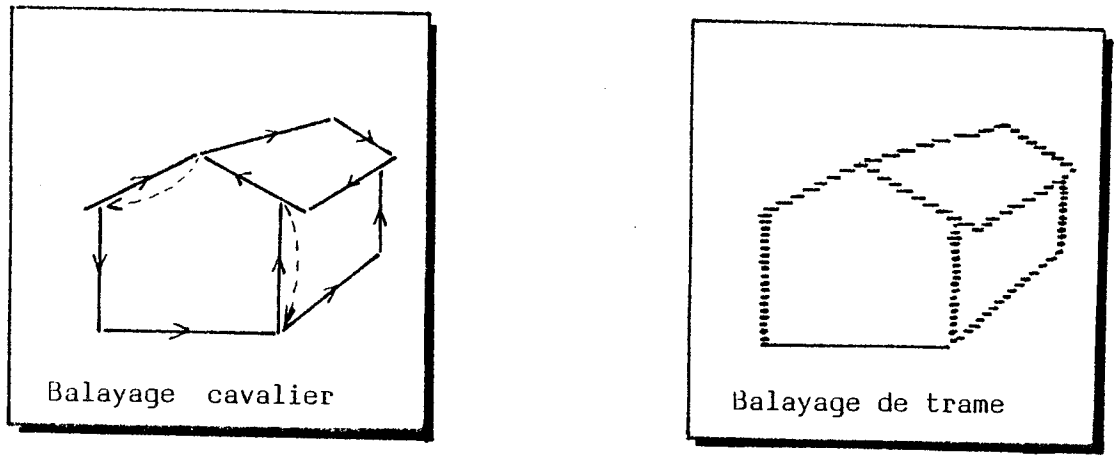


Figure III.2: Les deux types de balayage

Il apparaît que le balayage cavalier est beaucoup mieux adapté au dessin au trait que le balayage de trame, alors que ce dernier est plus approprié à la présentation d'images. Cette tendance est confortée par les possibilités de couleurs offertes par les deux technologies.

- Dans le cas du balayage cavalier, des tubes couleurs actuellement disponibles proposent quatre couleurs (rouge, orange, jaune, vert) obtenues en dosant la force de pénétration du faisceau d'électrons à

travers deux couches fluorescentes (une rouge, une verte). superposées sur l'écran (figure 3).

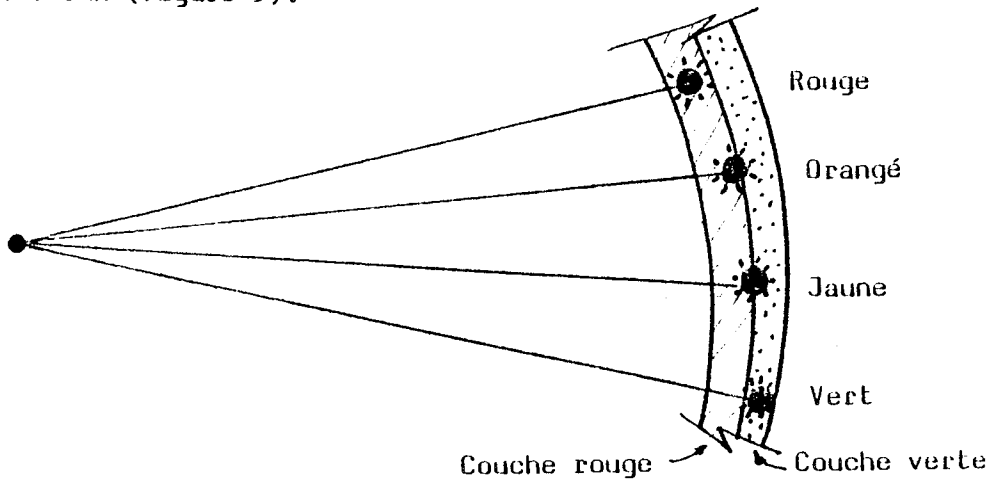


Figure III.3: Principe du tube à pénétration

- Dans le cas des tubes à balayage de trame, l'obtention d'une vaste gamme de nuances est obtenue grâce à l'utilisation de trois canons à électrons, intégrés dans un même tube. Ces tubes en tous points identiques à ceux utilisés en télévision, bénéficient ainsi d'une large diffusion, et par conséquent d'un coût considérablement plus faible que les précédents. La technologie du tube trinitron (ou chromatron) vient peu à peu remplacer celle du tube à masque, beaucoup plus complexe et plus consommatrice d'énergie. La figure 4 schématise le principe d'un tube trinitron. Le phénomène de synthèse additive des couleurs à partir de trois composantes primaires sera évoqué en VI.4.

2.2. Les dispositifs d'entretien (ou rafraîchissement)

La rémanence des luminophores apposés sur l'écran est relativement faible, en particulier pour les tubes "télévision" destinés à présenter des images dynamiques. Il est donc nécessaire de rafraîchir périodiquement l'image afin qu'elle persiste.

Liées aux technologies d'affichages, trois techniques de rafraîchissement sont utilisées:

* Le tube mémoire,

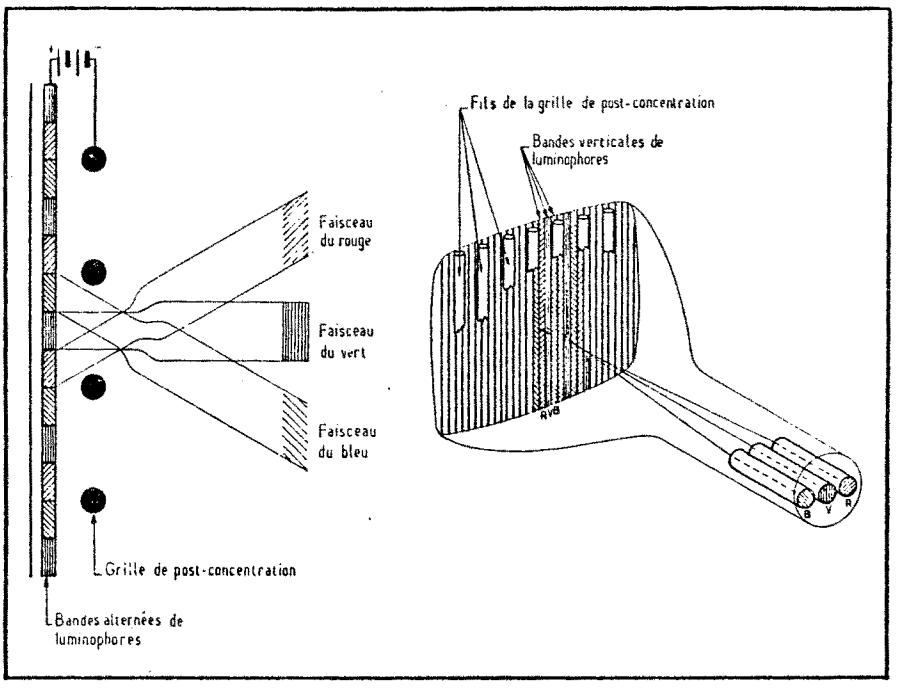


Figure III.4: Principe du tube trinitron

* L'interprétation continue d'une liste de visualisation,
* L'interprétation continue d'une mémoire de points.

2.2.1. Le tube mémoire

Utilisé à l'heure actuelle par un seul constructeur (Tektronix), il n'en est pas moins le plus répandu compte tenu du faible coût qu'il engendre. Dans ce procédé (réservé au balayage cavalier), le dispositif d'entretien est intégré au tube lui-même et se compose:

- d'une grille mémorisante située près de l'écran,
- d'un faisceau d'électrons d'entretien couvrant tout l'écran en permanence.

Le but de cette présentation n'étant pas de détailler l'aspect technique, le lecteur intéressé pourra trouver une description complète du processus dans (MoL76).

Les avantages et les faiblesses de cette technique sont principalement:

- son faible coût,
- l'impossibilité d'effectuer un effacement sélectif, seul l'effacement complet du tube étant autorisé, sauf sur certains matériels et pour une faible partie de l'image,
- l'impossibilité d'effectuer une identification directe d'un élément de l'image sans recours à un processus logiciel,
- l'absence de couleurs,
- la dégradation de l'image au cours du temps.

2.2.2. Liste de visualisation

Appliquée également au cas du balayage cavalier, cette technique nécessite (figure 5):

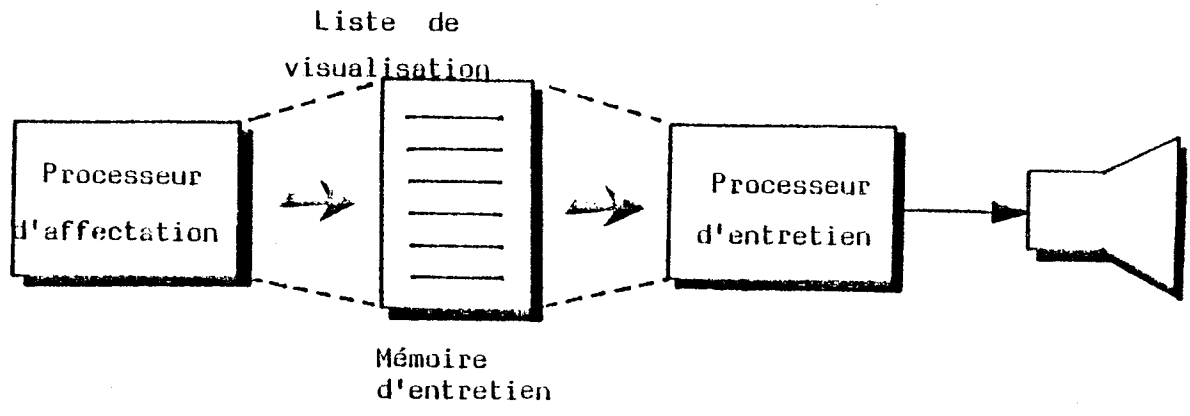


Figure III.5: Terminal avec liste de visualisation

- une mémoire d'entretien RAM (dimension variant entre 1 et 64 k),
- un processeur d'entretien, en partie cablé, chargé du processus de

visualisation s'effectuant par interprétation périodique des ordres graphiques enregistrés dans la mémoire. Ces ordres graphiques constituent un véritable programme qui est habituellement nommé liste de visualisation.

- un processeur chargé du processus d'affectation dans la mémoire d'entretien.

Les possibilités de ce type de console sont étroitement liées au jeu d'ordres graphiques acceptés par le processeur d'entretien.

Les principaux avantages de cette technique résident dans les possibilités de structuration et de modification de la liste de visualisation. Les principales retombées sont:

- la possibilité d'effacement sélectif par adjonction d'un "branchement" dans la liste de visualisation,

- la possibilité d'utiliser des sous-dessins, à l'aide de "sous-programmes" graphiques,

- la possibilité d'animation, en modifiant directement les attributs géométriques des éléments de la liste de visualisation.

Ces avantages sont néanmoins contre-balancés par:

* un coût plus élevé,
* une limitation du nombre d'éléments affichés, due principalement au temps nécessaire pour interpréter la mémoire et également à la taille mémoire Au delà de 40 ms ce procedé provoque en effet un clignotement désagréable.

2.2.3. La mémoire de trame

Associée aux tubes balayés par trames, ce dispositif comprend (figure 6):

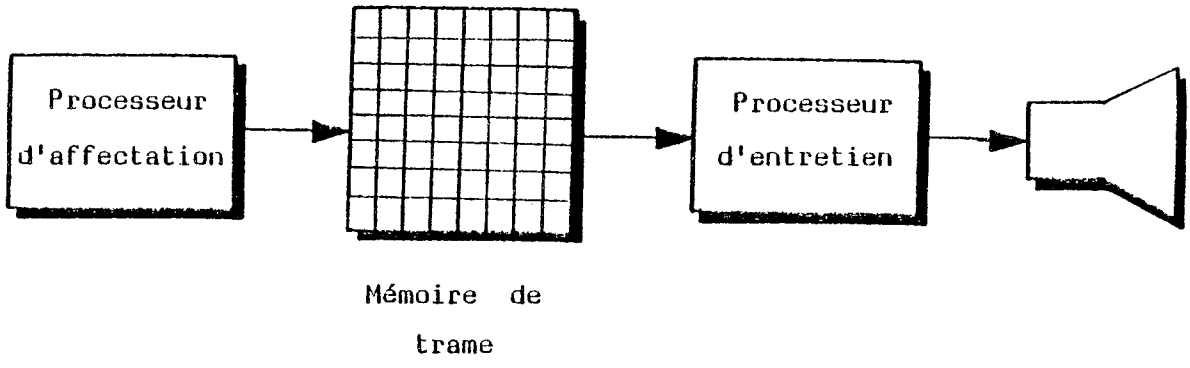


Figure III.6: Terminal avec mémoire de trame

- une mémoire de points, RAM contenant autant de points que l'image à présenter (en général 512*512),

- un processeur d'entretien, chargé de balayer cette mémoire par trames, d'interpréter et d'afficher l'information contenue dans chaque point.

- un processeur d'affectation chargé du remplissage de la mémoire.

Dans le cas le plus simple, la mémoire de trame exprime la luminosité ou la couleur de chaque point de l'image (attribut (A)). Le processeur d'entretien est alors un simple convertisseur numérique analogique, chargé de communiquer cette information au dispositif d'affichage, accompagnée des signaux de synchronisation du balayage (attribut (M)).

Les qualités et les faiblesses de cette technique sont principalement:

- le coût peu élevé, dû à la grande diffusion des techniques de télévision (on peut utiliser directement des récepteurs T.V),

- la stabilité de l'image indépendante de la complexité et de la durée d'affichage,

- la faible résolution de l'image en comparaison des techniques précédentes. A noter que des tubes comportant 1024x1024 points sont

disponibles à l'heure actuelle.

2.3. Les dispositifs de saisie

Au même titre que les précédents, les dispositifs de saisie ont peu évolué, quant à leur principe, au cours des dernières années. Ils concernent exclusivement la saisie d'informations morphologiques (coordonnées) exprimées dans le plan, et pour quelques uns dans l'espace. Nous considérerons deux catégories:

* les dispositifs liés au processus de visualisation,
* les dispositifs indépendants.

2.3.1. Les dispositifs liés au processus de visualisation

On trouve essentiellement dans cette catégorie le photostyle, muni d'une cellule photo-électrique qui engendre une interruption lors de la détection d'une intensité lumineuse. Lié principalement aux consoles avec mémoire d'entretien, il permet grâce à l'adresse dans cette mémoire de l'ordre graphique ayant provoqué l'interruption, la consultation des attributs relatifs à l'élément concerné.

Il est à noter que des photostyles sont également proposés depuis peu, sur des consoles à balayage de trame. L'interruption permet dans ce cas là de retrouver les coordonnées du point désigné, et à travers une lecture de la mémoire de trame, l'attribut qui lui est associé.

Dans un cas comme dans l'autre, le photostyle ne permet "d'identifier directement" un élément que si son identité figure dans la mémoire d'entretien (cas général) ou dans la mémoire de trame (cas du prototype HELIOS).

2.3.2. Les dispositifs indépendants

Nous rangerons dans cette catégorie, les dispositifs tels que: réticule, boule roulante, manche à balai, qui font généralement partie intégrante d'une console de visualisation, mais utilisant un processus de description totalement indépendant du processus de visualisation. L'information délivrée est un couple de coordonnées (x,y) exprimées dans le repère attaché à l'écran. Afin de faciliter la saisie, un "écho" est affiché sur l'écran sous forme d'un symbole quelconque (croix, caractère, etc.).

La tablette à numériser constitue cependant le moyen de saisie le plus puissant et le mieux adapté, à l'heure actuelle. Ces avantages par rapport aux outils précédents sont nombreux:

- position horizontale ou inclinée,
- grandes dimensions possibles (1m * 1m),
- grande précision (jusqu'à 0.025 mm),
- possibilité de relever directement des plans, des schémas, etc.,
- possibilité de projeter des diapositives en vue de relever des contours ou des distances entre objets.

Les tablettes actuelles sont de véritables périphériques indépendants, pourvus de leurs circuits d'interface permettant de les connecter directement via une ligne V 24 série, à un calculateur ou à un terminal évolué.

La saisie d'informations tridimensionnelles peut être également effectuée,

- à partir d'une tablette ordinaire en pointant successivement deux vues du même objet,
- à partir d'une tablette spéciale comportant deux "crayons" et se chargeant de reconstituer la troisième dimension (Sut74), à l'aide d'un

processus intégré,

- à partir d'un numériseur 3D.

3. Les architectures du matériel

3.1. Méthodologie de l'étude

La console de visualisation regroupe les synthétiseurs terminaux de tout système de synthèse d'images. Il s'agit généralement de synthétiseurs mono-processus, et à ce titre les questions d'organisation ne se posent pas. En revanche l'ordonnancement du processus, l'exploitation éventuelle du parallélisme , ainsi que la nature des processeurs utilisés vont conditionner directement l'architecture du matériel.

3.1.1. L'architecture minimale

Compte-tenu de la nécessité de rafraîchir l'écran en permanence, toute console de visualisation comporte, en plus du dispositif d'affichage, au moins:

- une mémoire d'entretien,
- un processeur d'entretien,
- un processeur d'échange, chargé des processus d'affectation et de consultation.

Cette architecture minimale représentée par la figure 7, peut être complètée par quelques remarques générales.

- Compte tenu du rythme nécessaire au rafraîchissement du tube, le processeur d'entretien doit être cablé, ou intégré au tube (cas du tube mémoire).
- Les conflits d'accès à la mémoire d'entretien, partagée entre deux processeurs, sont résolus de manière interme.

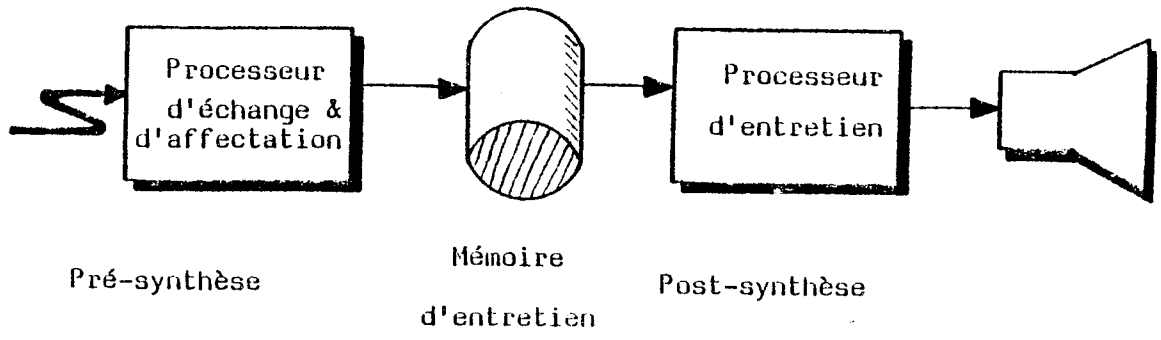


Figure III.7: Architecture minimale

- La mémoire d'entretien peut également servir de mémorisation intermédiaire.

- Les opérateurs élémentaires de synthèse peuvent être répartis aussi bien au niveau du processeur d'échange (nous parlerons de pré-synthèse) qu'au niveau du processeur d'entretien (post-synthèse).

Nous verrons cependant que la tendance actuelle est à l'augmentation du nombre de processeurs, qu'il s'agisse d'insérer des couches supplémentaires de pipe-line asynchrone entre les processeurs d'échange et d'entretien, ou encore d'exploiter le parallélisme du processus de visualisation.

Outre le nombre de processeurs, nous examinerons pour chacun d'eux:

* la technologie utilisée (cablé ou programmé)
* la nature et la structure des mémoires associées.

3.1.2. La pénétration des attributs au sein du matériel

Afin d'évaluer globalement les performances des différents matériels étudiés, nous considèrerons l'ordre et la place des opérateurs de synthèse, c'est-à-dire la profondeur de "pénétration" des différentes classes d'attribut. Les points intéressants sont principalement:

- la classe d'attribut la plus "pénétrante", qui est aussi la plus privilégiée du point de vue de la rapidité de modification,
- la faculté de pénétration à l'intérieur des couches du matériel.

3.2. Les systèmes minimaux

Nous rassemblerons dans cette catégorie, les sytèmes matériels bi-processeurs à faible pénétration. Ces deux propriétés induisant les caractèristiques suivantes:

- Le système est organisé autour d'une mémoire unique (mémoire d'entretien),
- Aucune synthèse n'est effectuée au niveau du processeur d'entretien,
- Le processeur d'échange effectue une synthèse limitée.

3.2.1. Contrôleurs vidéo intégrés

La tendance indiscutable vers le balayage de trame trouve son origine dans la faible quantité de matériel nécessaire pour la construction d'un système minimal. Cette construction qui peut être réalisée à partir de produits "grand public" (RAM dynamique, convertisseur N/A, moniteur T.V. standard), est encore simplifiée par l'avènement de contrôleurs vidéo complets intégrés dans un boitier 40 broches. La société EFCIS propose depuis peu un tel circuit (EF9365), issu d'une étude effectuée par P. Matherat (Mat78), qui prend en charge les processus de visualisation, de description, d'affectation et de consultation, pour des éléments simples (points, vecteurs):

* génération des signaux de synchronisation destinés au moniteur,
* gestion des conflits d'accès (écriture, lecture) et du rafraîchissement de la mémoire de trame (RAM dynamique),
* gestion d'un réticule,
* génération de vecteurs,
* génération de caractères,
* zoom.

L'architecture d'un système réalisé avec ce circuit est représenté figure 8.

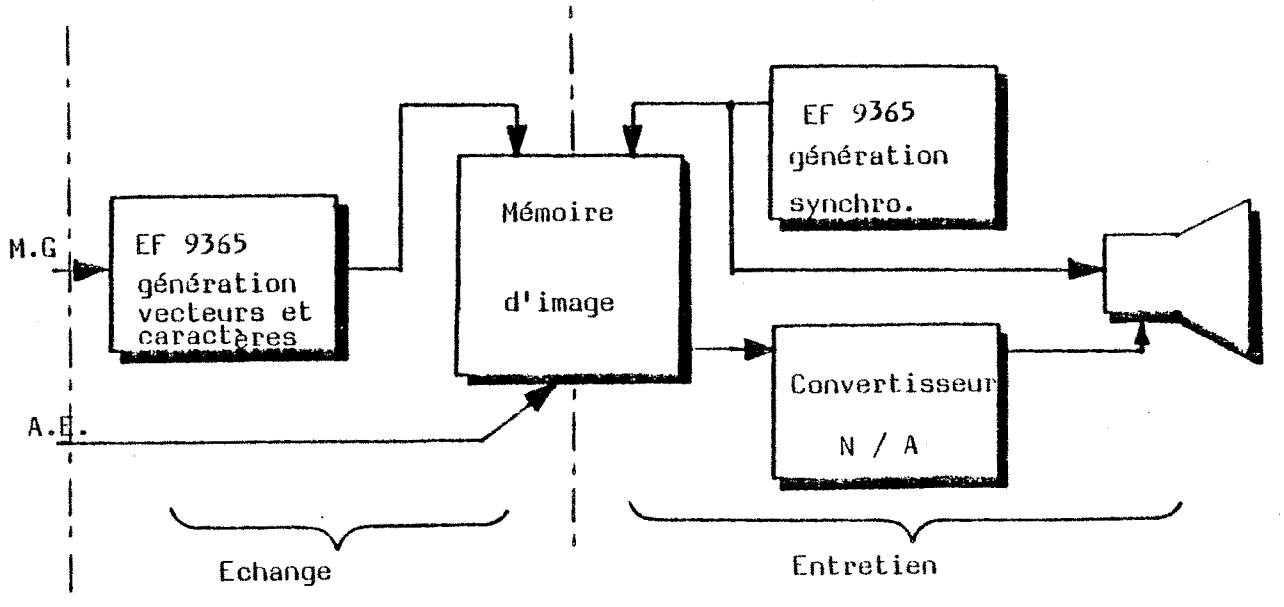


Figure III.8: Système utilisant un contrôleur intégré

Il faut noter, toutefois que ce circuit se contente d'engendrer les accès mémoire nécessaires pour l'affichage de vecteurs et de caractères (conversions morphologiques). Le contenu des informations communiquées à la mémoire est laissé à la responsabilité du logiciel externe.

3.2.2. Systèmes à tube mémoire

Etant donné que le dispositif d'entretien est intégré au tube, l'architecture de ce type de matériel est sensiblement plus simple, comme en témoigne la figure 9 représentant un système du type Tektronix 4010.

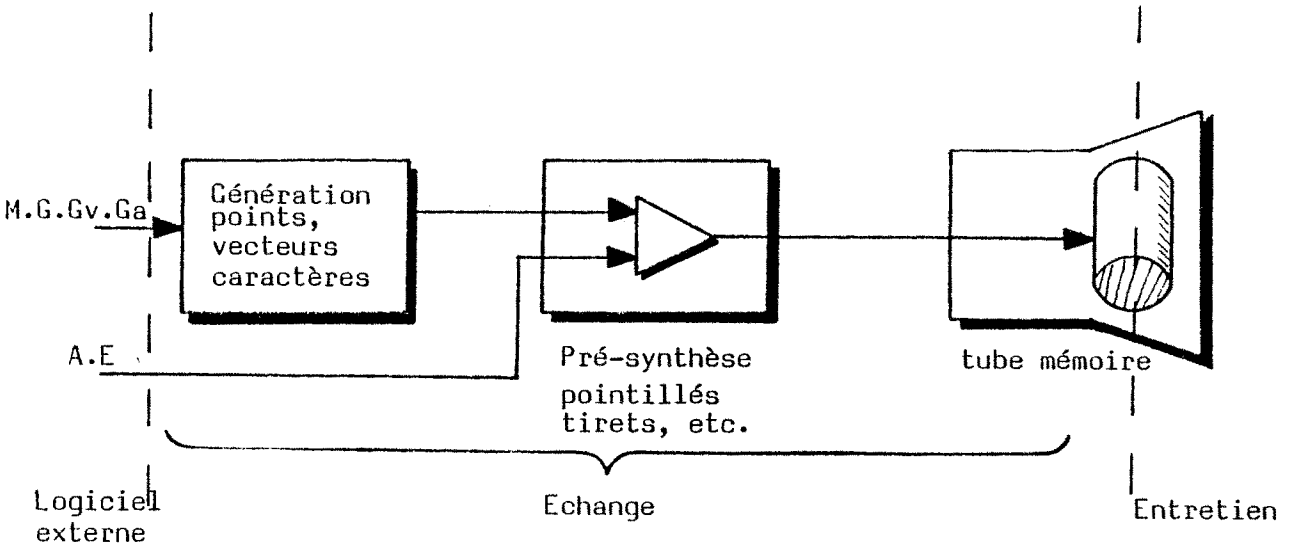


Figure III.9: Système à tube mémoire

Comme dans le cas précédent, la couche du processeur d'échange se charge de l'inscription des éléments de base: vecteurs, points, caractères qui peuvent être regroupés en:

* lignes brisées ou segments disjoints pour les vecteurs ou
* chaînes pour les caractères.

Le terminal effectue également une synthèse d'aspect limitée aux tirets et pointillés, pour les vecteurs uniquement.

3.2.3. Systèmes à balayage de trame

La plupart de ces systèmes utilisent (ou pourraient utiliser) le contrôleur présenté en 3.2.1 . Ils constituent en effet pour la plupart le regroupement sous un même "carter":

* d'un ou deux processeurs réalisant les fonctions du 9365.
* d'une mémoire de trame,

* d'un convertisseur numérique/analogique,
* d'un moniteur TV,
* et de circuits et accessoires annexes tels que claviers, ports d'entrée/sortie, etc.

Ces systèmes bas de gamme sont généralement orientés vers le dessin au trait, et présentent pour certains une compatibilité totale avec le Tektronix 4010, tels que le VT640 Retrographics de DEC ou les terminaux de Secapa.

3.2.4. Systèmes à liste de visualisation

Contrairement aux systèmes précédents, on se trouve ici dans le cas où la synthèse est effectuée par le processeur d'entretien. Quelques attributs d'aspect (texture du trait, couleur) et quelques attributs géométriques (translations 2D et parfois homothétie) sont directement enregistrés dans la mémoire d'entretien par le processeur d'échange, en vue de leur synthèse au cours des cycles d'entretien.

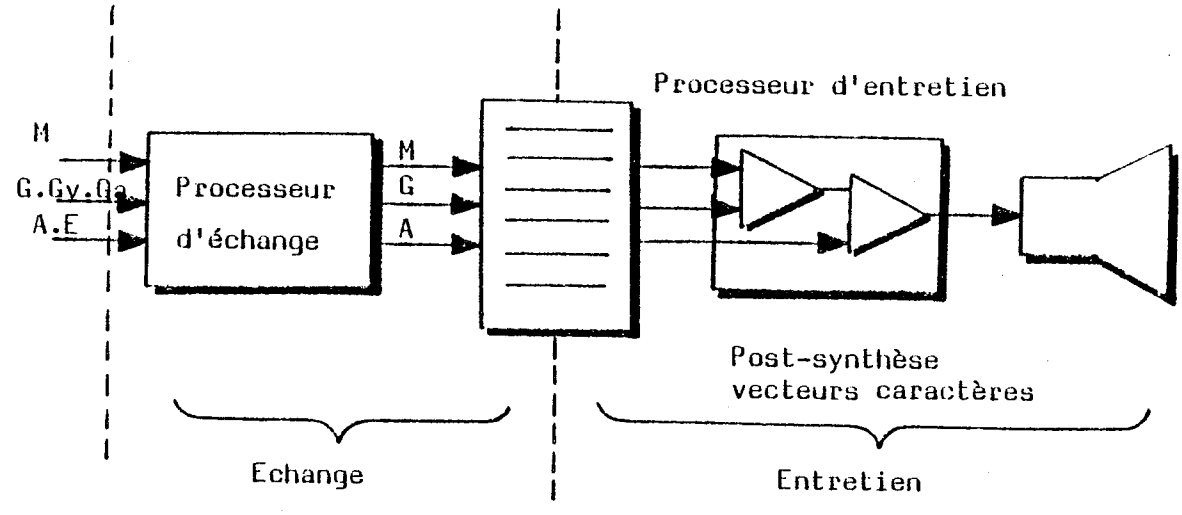


Figure III.10: Système à liste de visualisation

Le temps de réponse à une modification de l'une des informations contenues dans la mémoire d'entretien, est considérablement diminué grâce à une pénétration assez forte des attributs (A) et (G) au niveau de la logique cablée.

Quelques exemples de systèmes: IBM 2250 et 3250, AFIGRAF 6000 et 8000, CONCEPT 2160, etc.

3.3. Systèmes généraux

Afin d'aborder un maximum d'applications, les constructeurs ont cherché à élargir l'univers d'entrée du matériel, en proposant des processus et des éléments plus "évolués". Cette tendance se traduit généralement par une extension du niveau logiciel:

* soit à travers l'adjonction d'un micro-processeur et d'un logiciel spécialisé au niveau du processeur d'échange,
* soit à travers l'incorporation d'une couche supplémentaire intégrée souvent constituée d'un micro-ordinateur complet comportant ses propres ressources de mémoire (RAM et/ou disques). Il s'agit donc là d'un synthétiseur complet.

3.3.1. Utilisation de micro-processeurs

La première extension proposée par les constructeurs fut d'incorporer au niveau du matériel un micro-processeur et son logiciel, permettant d'accroître le nombre d'opérateurs de synthèse et de construction pris en charge par le terminal. Le Tektronix 4027, dont l'architecture est présentée sur la figure 11, en est un exemple.

Les opérateurs de synthèse sont effectués au niveau du processeur d'échange et permettent notamment:

- la génération de points, vecteurs, caractères sous diverses textures (pointillé, trait mixte, tiretés, etc.).

- le remplissage de taches à l'aide de "motifs", sortes de textures simplifiées comportant 14*8 points, de 2 couleurs choisies parmi 64.

Nous rangerons également dans cette catégorie le prototype STYX, développé par M. Mériaux (Mer79) de l'université de Lille, qui comprend un

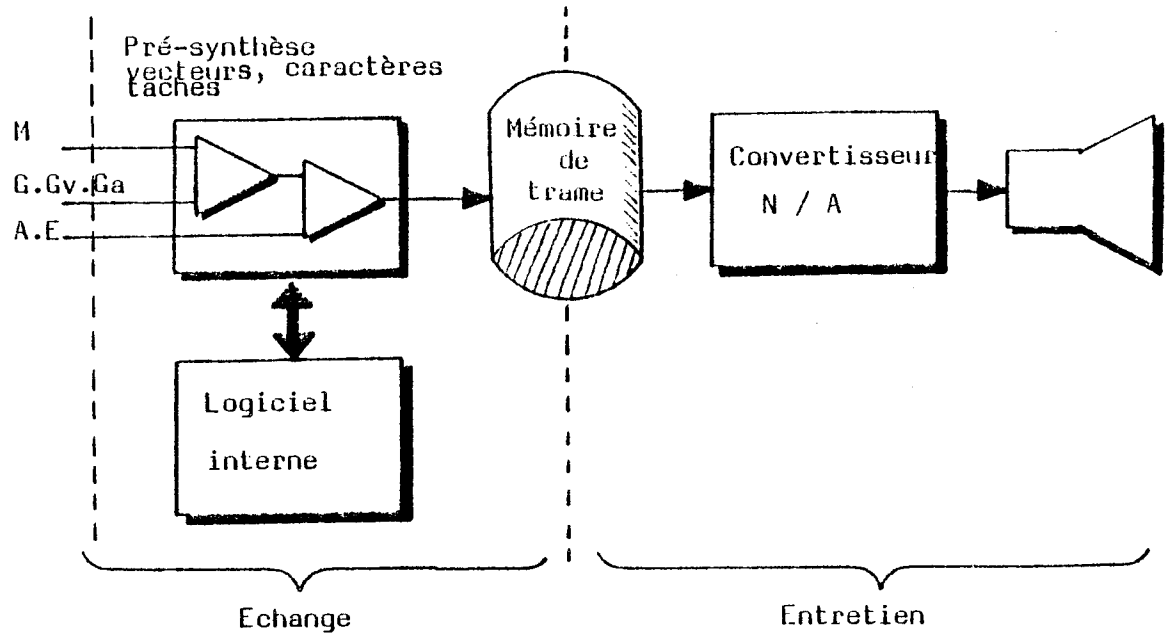


Figure III.11: Architecture du Tektronix 4027

micro-processeur Z80 chargé du remplissage des taches.

3.3.2. Utilisation de micro-ordinateurs

Il s'agit là de la tendance actuelle des constructeurs, désireux de fournir le maximum de fonctions locales intégrées. Nous avons vu en II.1.3.2 les problèmes soulevés par l'utilisation de ce type de matériel qui s'apparente plus à un "calculateur à vocation graphique" qu'à un terminal. Toutes les technologies sont touchées par cette "évolution":

- les systèmes à base de liste de visualisation telle la console 4 couleurs Concept 2160 fabriquée par CIT Alcatel, comportant trois processeurs et deux mémoires RAM.

- les systèmes à tube mémoire tel le 4114 de Tektronix pouvant gérer deux unités intégrées de disques souples.

- les systèmes à balayage de trame tel le Ramtek 9400 dont l'architecture caractéristique est schématisée par la figure 12.

L'originalité de ce système est de concilier les avantages du balayage

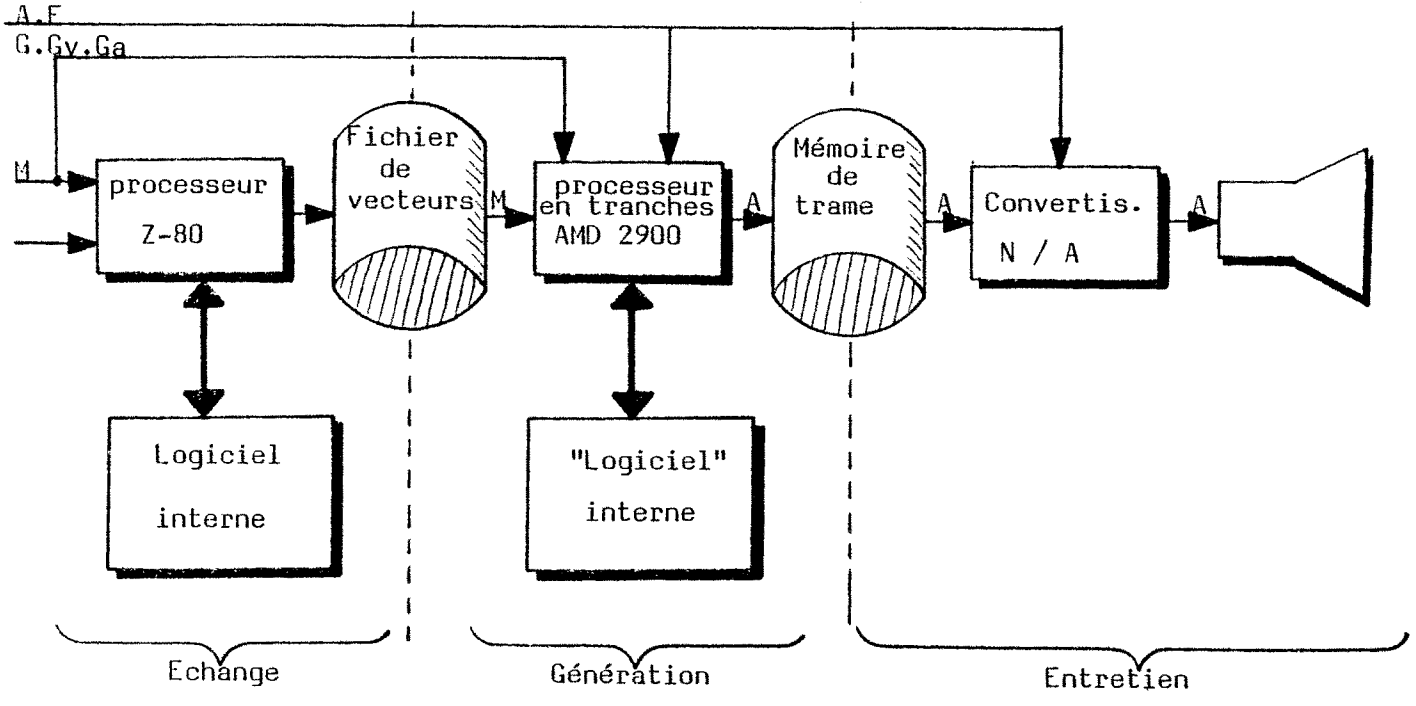


Figure III.12: Architecture du Ramtek 9400

cavalier (en particulier la définition d'images à partir de points et de vecteurs), et ceux du balayage de trame notamment le remplissage de taches avec une gamme étendue de nuances. L'accès au fichier de vecteurs contenant une épreuve intermédiaire est géré par le logiciel d'un micro-ordinateur utilisant un micro-processeur Z80. Le traitement des attributs géométriques (zoom, translation, rotation, coupage) est effectué par un micro-processeur bipolaire en tranches AMD 2900 dont le cycle est de l'ordre de 280 ns. La génération des éléments (vecteurs, points, taches) et le remplissage de la mémoire de trame sont également à la charge de ce processeur. Un zoom cablé est possible au niveau du convertisseur ainsi qu'une transposition des couleurs.

3.4. Systèmes orientés vers un processus spécifique

Nous quittons ici le domaine du matériel "grand public" pour nous tourner vers des systèmes plus spécifiques produits généralement en petites séries, ou encore à l'état de prototype. Ils préfigurent néanmoins le marché de demain, et à ce titre nous leur consacrerons une part importante.

La plupart des recherches en matière de matériel de visualisation, sont issues des deux constatations suivantes:

- la technologie actuelle ne permet pas de concevoir un système doué des meilleures performances, et susceptible de satisfaire toutes les situations.

- L'adaptation du matériel à un processus particulier permet d'accroître les performances, en exploitant au maximum les propriétés de ce processus. Cette adaptation en revanche, limite la portée de ce matériel à un domaine restreint d'applications. L'opération à privilégier détermine le choix du processus intégré.

3.4.1. Animation d'images en temps réel

On trouve généralement dans cette classe, des systèmes ne comportant qu'un nombre très restreint d'opérateurs élémentaires. Le principal objectif est l'animation en temps réel d'images synthétiques offrant un réalisme "modeste". La méthode utilisée est la suivante

* précalcul en différé de toutes les images de la séquence,
* enregistrement sur disque de ces images sous une forme condensée,
* décompression et visualisation en temps réel.

L'architecture de la figure 13 est celle utilisée dans deux systèmes, l'un commercialisé (CVD/2 de Rivers Computer Corporation), l'autre issu de travaux de recherches (StT79).

Il est à noter l'absence de mémoire de trame au sens propre du terme. L'asynchronisme existant entre la lecture du disque et le balayage de l'écran, est géré grace à un processeur de transfert qui se charge d'alimenter la mémoire tampon au fur et à mesure qu'elle est visualisée. Le compactage utilisé (nombre de points consécutifs de la même couleur + couleur) réduit considérablement les performances requises au niveau du disque, mais limite du même coup le réalisme des images.

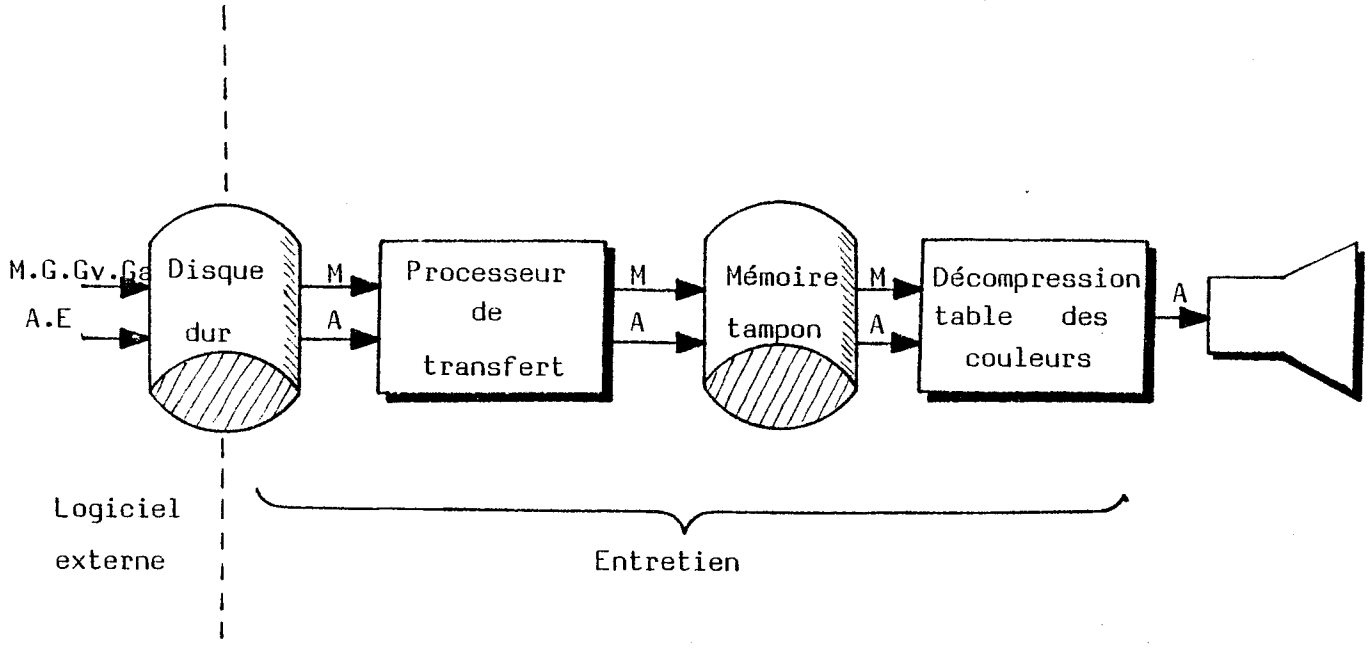


Figure III.13: Système permettant l'animation temps réel

3.4.2. Visualisation d'objets particuliers (sphères)

Nous illustrerons ce type d'adaptation en considérant un système développé par J. Staudhammer (Sta78), destiné à la visualisation de molécules en temps réel. Chaque atome est symbolisé par une sphère, et le matériel est capable de présenter 2000 atomes simultanément. Bien que l'auteur espère beaucoup des techniques de tri multi-dimensionnelles utilisant des processeurs parallèles (Boi79), l'élimination des parties cachées est à l'heure actuelle effectuée par le logiciel et les images précalculées sont, comme précédemment, stockées sur disque. Le matériel quant à lui détermine l'éclairement en chaque point de chaque sphère.

3.4.3. Adaptation à une opération particulière (prise de vue)

Il s'agit plus en fait d'une adaptation mutuelle entre processus logique et matériel, une sorte de compromis entre ce que l'on voudrait faire, et ce que l'on peut faire. Le but recherché dans ce type de système étant généralement la rapidité (voire le temps réel), les processus de synthèse concernés sont du type à forte pénétration, et le matériel, pour ne pas pénaliser l'ensemble, doit lui même accepter cette pénétration et

prendre à sa charge bon nombre d'opérations de synthèse. Nous prendrons comme exemple le système GSI 1 réalisé au CELAR (Centre d'études de l'armée de l'air) (Ler80), qui utilise un processus privilégiant la prise de vue.

Le processus utilisé, directement inspiré de celui proposé par R.A. Schumacker et ses collaborateurs (SBG69), repose sur le principe suivant:

- précalcul des priorités statiques sur les faces polygonales composant la maquette (classement des faces indépendamment du point de vue),
- détermination en temps réel au niveau du matériel, des priorités dynamiques (modification des priorités statiques en fonction du point du vue).

Le système du CELAR intègre également le calcul de l'éclairage des faces dépendant de la position d'une source lumineuse, "lissé" à l'aide de la méthode de Gouraud (Gou71). Toutes ces informations au même titre que les priorités dynamiques, et les transformations géométriques sont traitées par un microprocesseur en tranches AMD 2900, et stockées en vue du remplissage de la mémoire de trame effectué par un module cablé (cf. figure 14).

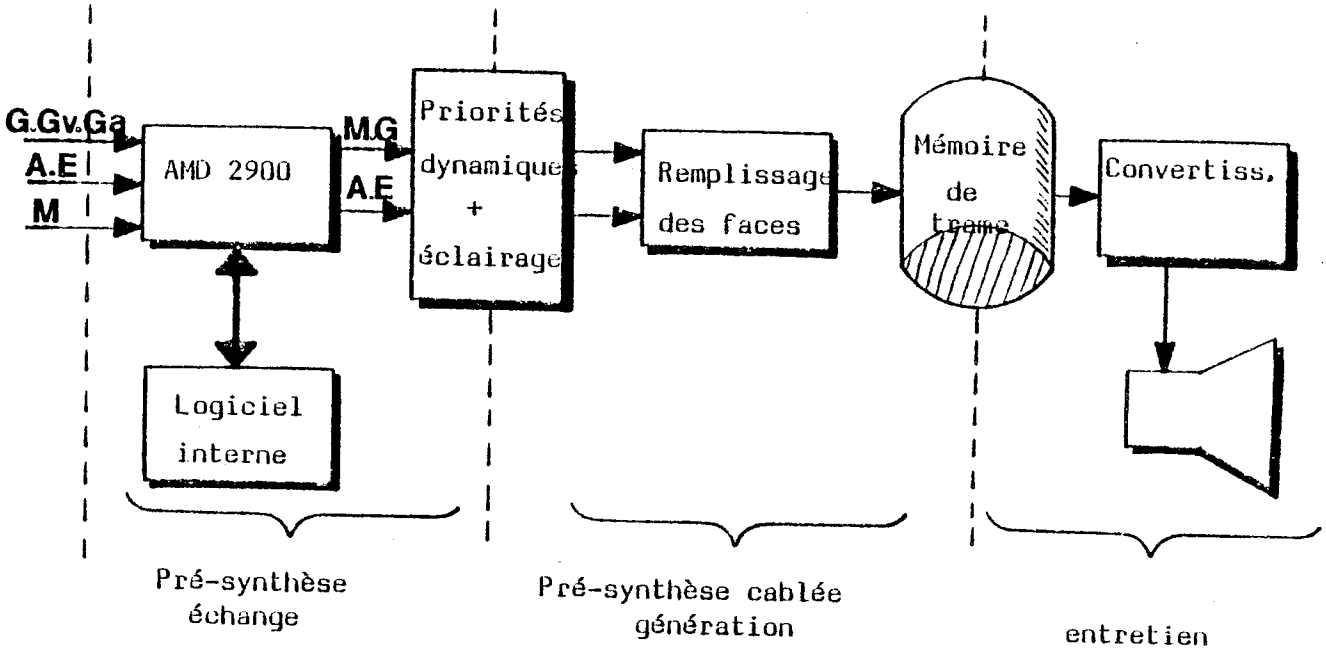


Figure III.14: Architecture du GSI 1

Ce processus ne fait pas exception à la règle générale et "paye" le privilège accordé aux attributs géométriques au prix de quelques limitations:

* le réalisme est assez pauvre (pas de couleurs ni texture ni reflets),
* les éléments sont composés uniquement de faces convexes,
* les mouvements relatifs des objets présentés sont possibles en temps réel à condition qu'il existe toujours un plan séparateur entre chacun d'eux.

3.5. Les architectures parallèles

3.5.1. Parallélisme intra-processus

Fort peu de systèmes ont exploité, même modestement, ce type de parallélisme. Les raisons de ce peu d'enthousiasme découlent sans doute des difficultés suivantes:

- difficulté de trouver un processus suffisamment parallèle,
- difficulté de synchroniser les divers processeurs,
- difficulté des opérations de synthèses effectuées sur des informations "pénétrantes".

Nous aurons l'occasion au cours du chapitre suivant de présenter notre expérience dans ce domaine.

3.5.2. Parallélisme inter-processus

Compte tenu du fait que le nombre d'éléments dans l'image est généralement très supérieur au nombre d'opérateurs dans le processus de visualisation, le gain de temps apporté par un parallélisme inter-processus est beaucoup plus intéressant que celui de l'approche précédente. En revanche l'investissement matériel est considérable puisque un processeur est nécessaire pour chaque élément. De nombreux travaux récents ont dirigé leurs efforts vers cette technique (Mer81), (Wei81), (GSS81), devenue

envisageable grâce à la densité d'intégration atteinte dans les circuits VLSI (Very Large Scale Intégration). L'idée de base de ce type d'approche consiste à affecter à chaque élément de la maquette un processeur spécialisé chargé du calcul en parallèle de la profondeur et de la couleur associées à chaque point (x,y) de l'image. La détermination de l'objet visible en ce point est confiée à un pipe-line synchrone de comparateurs permettant de sélecter le point le plus proche de l'observateur et même d'effectuer un opérateur d'anti-aliasing (figure 15).

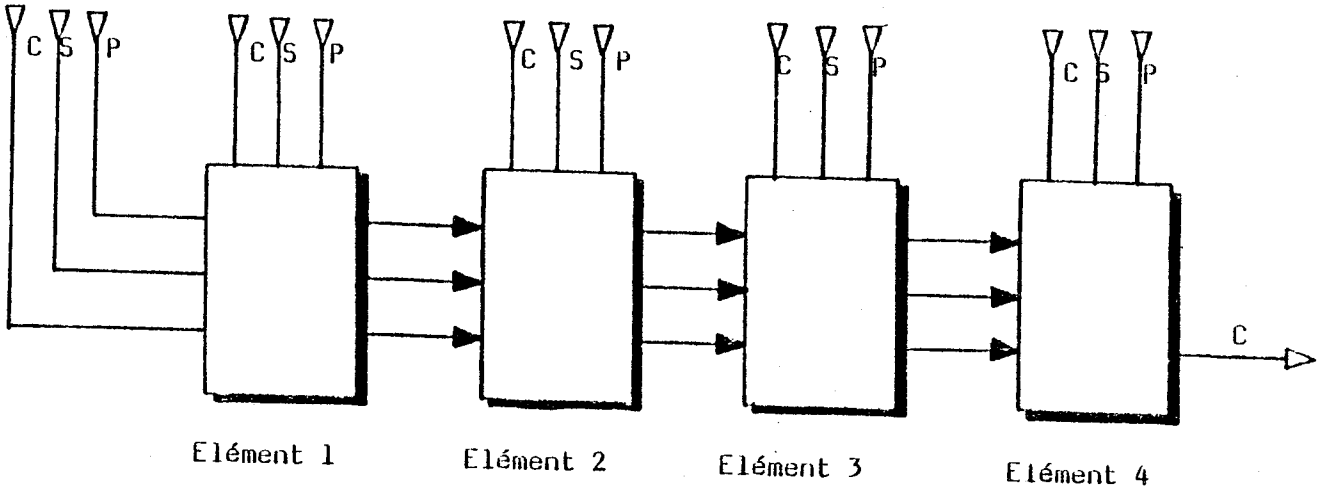


Figure III.15: Exemple de parallélisme inter-processus

Quelques remarques peuvent être faites sur ce type de processus:

- l'état d'avancement de la technologie liée au coût de la méthode n'a permis jusqu'àlors que des implantations simulées àl'aide de programmes.

- les objets sont limités à l'heure actuelle à des polygones convexes du plan de l'épreuve, la projection dans ce plan étant effectuée de manière séquentielle, c'est à dire successivement pour chaque objet, ce qui retire à cette technique un peu de son intérêt: la rapidité du temps de réponse.

Ce type de solution pourrait néanmoins conduire, dans un avenir dont la proximité est liée à l'accroissement des densités d'intégration des VLSI,

à des réalisations ne nécessitant plus aucune mémoire de trame et permettant l'animation en temps réel d'objets tridimensionnels.

4. Les processus élémentaires du logiciel

Nous passerons en revue dans ce paragraphe quelques processus élémentaires classiques du logiciel. Il est clair que quelques uns de ces processus tendent de plus en plus vers des réalisations mixtes comportant des parties cablées et des parties programmées, mais tous nécessitent encore une contribution logicielle. Nous avons choisi de limiter cette étude aux opérateurs fondamentaux concernant les attributs d'aspect, d'éclairage et de morphologie. Ce choix se justifie principalement par les progrès constants réalisés dans la mise en oeuvre de ces opérateurs et par l'incidence qu'ils ont sur les performances des systèmes de synthèse. Nous examinerons ci-après:

- la modélisation de l'aspect: opérateurs de construction sur les attributs de la classe 'A',
- l'application des textures : opérateurs de synthèse entre les attributs 'M,G' d'une part, et 'A' d'autre part,
- le remplissage des taches : opérateurs de construction appliqué aux attributs 'M',
- la modélisation de l'éclairage : opérateurs de construction des attributs 'E' et de synthése 'E.G', 'E.A'.

4.1. Modélisation de l'aspect

4.1.1. Les principes de la trichromie

Le phénomène d'intégration, dans l'espace et dans le temps, lié au pouvoir séparateur limité de l'oeil (angle minimum = 1 minute), conduit à chercher à représenter chaque couleur, subjective ou non, à l'aide de quelques couleurs primaires seulement. Tous les procédés de reproduction des couleurs s'appuient sur les principes de la trichromie, énoncés par Young,

Maxwell et Helmholtz, d'après lesquels on peut recréer pratiquement toutes les sensations colorées, par mélange de trois couleurs primaires judicieusement choisies. On a constaté que le "trio" rouge, vert, bleu permettait la reproduction du plus grand nombre de couleurs.

On peut ainsi représenter une couleur quelconque (C) à l'aide de trois coefficients représentant l'intensité des trois primaires Rouge, Vert et Bleu (r, v, b).

$$m\,(C) = r\,(R) + v\,(V) + b\,(B)$$

m représentant l'intensité de la couleur C.

Cette représentation obéit aux lois de l'addition et de la multiplication dans l'espace vectoriel R .

$$k.m\,(C) = k.r\,(R) + k.v\,(V) + (k.b)\,(B)$$

$$m1(C1) + m2\,(C2) = (r1 + r2)\,(R) + (v1 + v2)\,(V) + (b1 + b2)\,(B)$$

Le choix des primaires adopté en télévision couleur est dicté par les caractéristiques de radiation des substances luminescentes utilisées pour les écrans des récepteurs. Dans le diagramme de la C.I.E. (Commision Internationale de l'Eclairage) (figure 16), ces trois primaires déterminent un triangle à l'intérieur duquel se trouvent toutes les couleurs reproductibles en télévision. Le nombre de couleurs différentiables par l'oeil est de l'ordre d'une douzaine de mille.

Les inconvénients de la trichromie

La trichromie, plus ou moins imposée par les consoles de visualisation actuelles, présente certains inconvénients au niveau du traitement numérique des couleurs, et de leur manipulation par un utilisateur non averti (GrJ78).

La majorité des utilisateurs est peu familiarisée avec la synthèse additive des couleurs, utilisée en trichromie, il est en effet fort peu "naturel" de produire un jaune en mélangeant du rouge et du vert. La

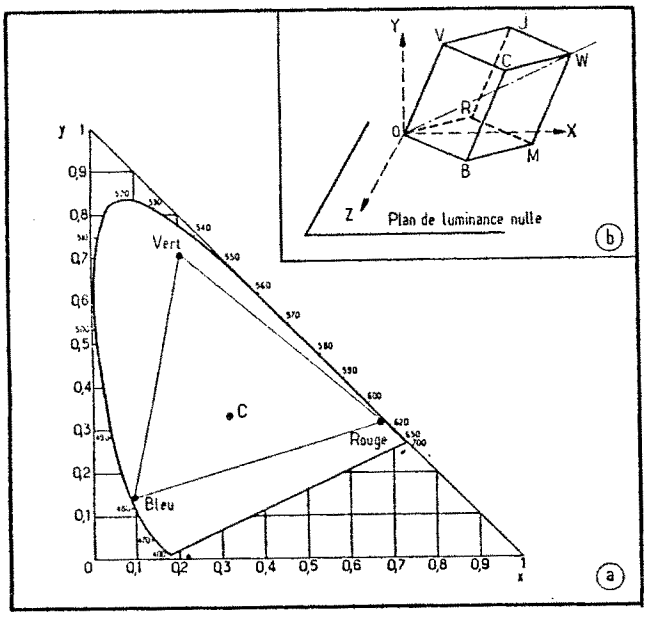


Figure III.16: Diagramme de la C.I.E.

synthèse de couleur utilisée en peinture ou en imprimerie, est au contraire une synthèse soustractive, agissant par une superposition de filtres colorés, absorbant une partie du spectre visible.

On peut ainsi fournir un espace de couleurs plus adapté à l'utilisateur en considérant l'espace inverse de (R, V, B) : (C, M, J) (cyan, magenta, jaune).

La transformation des coefficients d'intensité normalisés est immédiate :

$$(r, v, b) = (1, 1, 1) - (c, m, j)$$

4.1.2. Luminance, teinte et saturation

Les deux représentations ci-dessus comportent de grosses difficultés de manipulation, dues à l'abstraction de trois paramètres prépondérants de la vision: la luminance, la teinte et la saturation.

- La luminance correspond à l'énergie lumineuse perçue, différente de l'énergie lumineuse émise en raison de la courbe de sensibilité relative de l'oeil. Ainsi, certaines couleurs apparaissent plus "lumineuses" que d'autres à éclairement égal (figure 17). L'oeil peut distinguer près de mille niveaux de luminance.

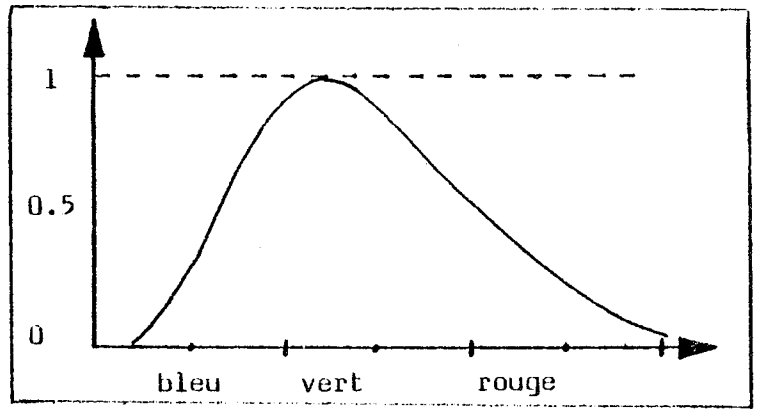


Figure III.17: Luminance relative des couleurs

- La teinte est en relation avec les longueurs d'onde prédominantes du rayonnement perçu par l'oeil.

- La saturation (ou pureté) permet de distinguer les couleurs "vives" des couleurs "pastels" ou "délavées". Dans le spectre et dans les pourpres, les radiations sont toutes de pureté égale à 1 (saturées à 100 pour 100). En ajoutant du blanc à des couleurs pures, on obtient toute une gamme de couleurs jusqu'à 0 pour 100.

On peut représenter l'espace des couleurs en fonction de ces trois paramètres, comme le physicien Munsell en eut l'idée dès 1905 (Mun46) (figure 18).

Outre l'intérêt de manipulation aisée de l'espace (luminance, teinte,

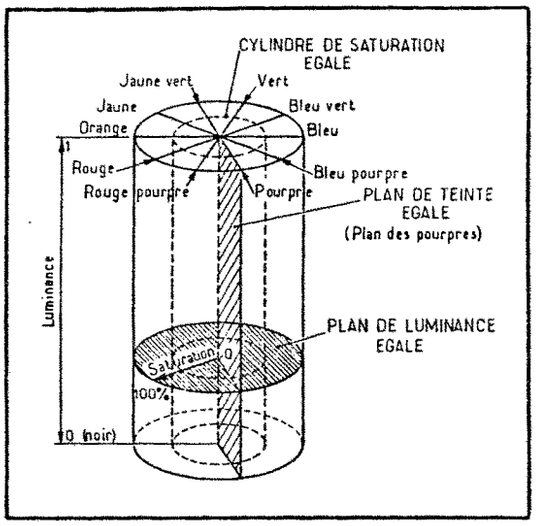


Figure III.18: Représentation cylindrique de l'espace des couleurs

saturation) pour un utilisateur non initié, cette représentation présente également l'avantage de faciliter le traitement numérique des informations colorées.

En particulier, l'effet d'ombrage ou de vision crépusculaire d'un objet coloré se traduit par une simple interpolation linéaire sur le paramètre de luminance (entre la couleur initiale et le noir). Les effets de brumes, fumées ou dispersions atmosphériques lors de visions lointaines peuvent être approchés par simple interpolation linéaire sur le paramètre de saturation (entre la couleur initiale et le gris). La même interpolation dans l'espace (R, V, B) produirait des teintes intermédiaires abérrantes. Les propriétés des différents espaces de couleurs ainsi que les opérations permettant de passer d'une représentation à l'autre, ont suscité un grand nombre de travaux (Ken77), (Smi78), (Job78), (MeG80), (Mas81), (Tru81).

4.1.3. Modélisation d'une texture colorée

Dans la majeure partie des applications, la présentation d'images constituées de taches uniformes est insuffisante. L'aspect peu réaliste de ces images limite en effet considérablement le champ d'application possible. La présentation de paysages en particulier, nécessite une approche

différente, qui consiste à donner aux objets présentés un aspect moins "synthétique" en leur associant une texture colorée, sorte de "papier-peint" apposé sur ces objets.

La totalité des études présentées ici, comporte comme première phase l'obtention d'un modèle de texture représenté par une matrice (carrée dans la plupart des cas) de points . Chaque point comporte des informations de couleur ou de luminance selon le cas. Le modèle ainsi obtenu peut être assimilé à une image numérisée représentant les détails d'une portion limitée de la texture. La synthèse effective de la texture est obtenue en "pavant" la zone considérée à l'aide du modèle défini ci-dessus.

Les méthodes présentées ci-après diffèrent essentiellement par la provenance du modèle.

- 1- Modèles issus d'images numérisées. La matrice modèle est obtenue directement à partir d'une image rèelle saisie par une caméra numérique, sous laquelle il suffit de placer la photographie ou le dessin du modèle de texture désiré (Bln76), (DSS78), (Lie78), (Bli78). Cette technique présente de gros avantages quant à l'aspect très réaliste du modèle mais produit en revanche, des discontinuités gênantes lors du pavage des objets.

- 2- Modèles calculés. L'application d'une texture résulte généralement du "pavage" de la zone à remplir à l'aide d'un modèle de texture donné. La juxtaposition de modèles quelconques risquant de provoquer des discontinuités nuisibles à l'aspect de la texture, il convient lors du calcul de celle-ci, de prendre quelques précautions.

Deux techniques sont utilisées:

- génération de modèles "cohérents", c'est à dire dont les frontières supérieure et inférieure, gauche et droite, coincident parfaitement.

Cette condition est facilement remplie pour un grand nombre de textures régulières: briques, grilles, etc.

- utilisation de méthodes de pavage syntaxiques, ou structurelles, (Zuc75), (FuL78), (Lir70).

Un modèle de texture est considéré comme étant composé d'un ensemble de sous-modèles de taille fixe, et ainsi de suite. Ces modèles ou sous-modèles forment un arbre, dont les feuilles sont les points de l'image.

La texture finale est définie par l'application de règles de grammaire d'un haut niveau, décrivant l'arrangement des différents éléments. L'emploi de règles stochastiques permet d'obtenir des textures présentant des répartitions aléatoires (figure 19).

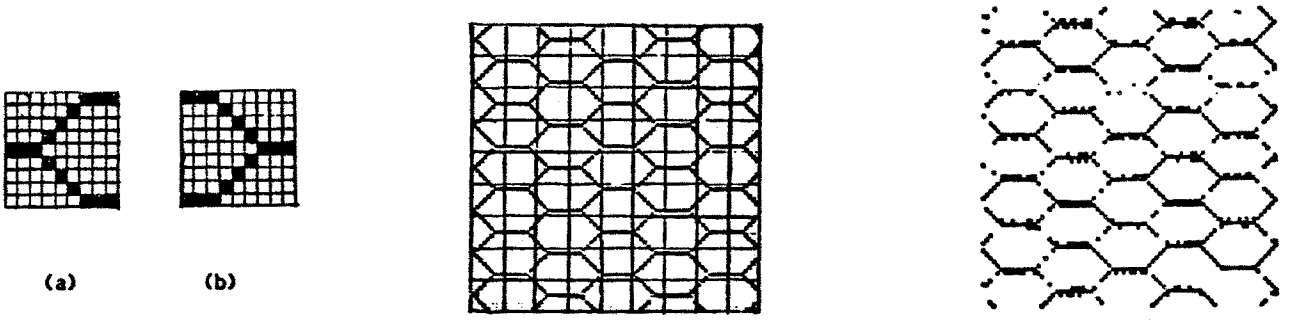


Figure III.19: Obtention d'une texture brouillée

Perturbation et transformation des textures

Cette technique utilise, dans la plupart des cas, des méthodes issues du traitement d'images: les transformées de Fourier, les perturbations aléatoires, ainsi que toutes les méthodes agissant par 'filtrage' ou sommation de régions plus ou moins grandes de l'image initiale.

L'ensemble de ces techniques trouve sa principale application dans la modification d'une texture initiale en fonction de certains paramètres tels que distance du point de vue, conditions climatiques (neige, brouillard, etc.), ou encore pour atténuer l'aspect trop "synthétique" d'une texture

calculée.

4.1.4. La transparence

Les problèmes relatifs au rendu de la transparence sont étroitement liés aux caractéristiques des algorithmes d'élimination de parties cachées.

En effet, les épreuves produites par ces algorithmes ne contiennent généralement que les seules parties visibles, alors que la prise en compte de la transparence peut nécessiter toutes les parties situées dans une direction donnée.

L'algorithme de Newell, Newell et Sancha (NNS72) est sans doute le mieux adapté à la présentation d'objets transparents.

Les faces polygonales sont traitées de l'arrière vers l'avant, l'intensité des faces opaques venant alors simplement remplacer la couleur précédemment calculée pour un point donné. Si la face rencontrée est transparente ou semi-transparente, il n'y a plus remplacement pur et simple, mais combinaison des couleurs. Les lois de composition des couleurs sont linéaires dans l'espace RVB.

Une autre technique proposée par Myers (Mye75) utilise également une variante de l'algorithme de Watkins, exposée ci-après.

Lors de l'examen d'une ligne de balayage, les segments opaques sont traités point par point (étude point par point en fonction de la profondeur) alors que les segments transparents sont simplement stockés, pour être étudiés après traitement des segments opaques.

Chaque point d'un segment transparent est alors éliminé s'il est derrière le point opaque le plus proche, sinon la nouvelle intensité est calculée d'une manière analogue à celle utilisée par Newell, Newell et Sancha.

4.2. Pavage et projection des textures

Après l'obtention d'un modèle de texture, le problème qui se pose est celui de son intégration dans la scène à visualiser. Deux aspects distincts apparaissent, découlant d'une part du pavage lui-même, et d'autre part des transformations subies par le modèle en vue de satisfaire à la perspective ou à la forme de l'objet.

4.2.1. Problèmes issus du pavage

Si dans le cas des modèles calculés le problème de la continuité du pavage peut être résolu lors de la génération du modèle, il n'en est pas de même lorsque l'on utilise une image numérisée qui n'a aucune raison, a priori, de respecter les conditions requises. La non-concordance des frontières fait alors apparaître une macro-texture indésirable, sauf dans quelques cas particuliers. Cet effet peut être considérablement atténué en appliquant à l'image finale les techniques de filtrage "anti-aliasing" (Cro77), (BlN76), (DSS78), (FlC80). Il s'agit en fait de filtres passe-bas agissant par pondération de chaque point en fonction des points environnants.

4.2.2. Les transformations géométriques

L'apposition d'une texture sur une surface quelconque soulève de nombreux problèmes relatifs à la visualisation de celle-ci.

En effet, l'application d'une simple projection perpective fait apparaître de nombreuses abérrations dûes à un échantillonnage insuffisant de la texture (aliasing). Cet effet se traduit par la disparition des détails fins, ou au contraire leur apparition dans des conditions ou ils devraient disparaître (vision lointaine). La rotation, le zoom, et la projection perspective d'une texture ont fait l'objet de publications concernant principalement l'optimisation du traitement (Cas80), et la qualité visuelle des résultats (Wei80), (FLC80), (Beg80), (PGC82).

Blinn et Newell (BlN76) ont proposé une méthode permettant de projeter une texture sur une surface gauche.

Pour chaque pixel on considère un carré centré en ce point et de côté égal à deux éléments. Les coordonnées X, Y des sommets du carré sont alors transformées en coordonnées U,V exprimées à l'aide de paramètres régissant la surface, les éléments du modèle de texture (défini dans ces mêmes coordonnées) situés à l'intérieur de la zone ainsi définie font alors l'objet d'une sommation pondérée (figure 20).

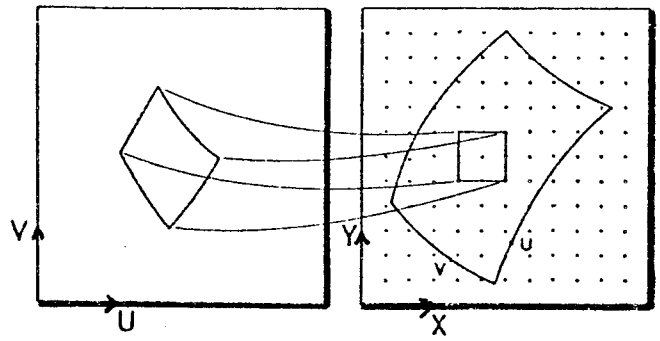


Figure III.20: Anamorphisme des textures

4.3. Le remplissage d'une tache

Cet opérateur effectue une transformation morphologique en déterminant l'ensemble des points intérieurs d'une tache définie à partir de son contour.

On trouve dans la littérature deux grandes classes d'algorithmes selon qu'ils travaillent sur le contour numérisé ou sur la définition formelle du contour.

4.3.1. Contour numérisé

Les algorithmes de cette catégorie travaillent en deux temps

- numérisation et inscription du contour dans une mémoire de trame,
- remplissage effectif.

- 1- Codification du contour M. Lucas (Luc77) et T. Pavlidis (Pav78) proposent de coder le contour lors de sa numérisation en fonction des singularités qu'il présente (segments horizontaux, sommets communs à plusieurs arêtes, recouvrement des arêtes). Le remplissage s'effectue ensuite en balayant les codes des points de contour qui s'y trouvent. Chaque code indique si l'on change d'état (intérieur/extérieur) ou non.

- 2- Structuration du contour. Les algorithmes les plus récents utilisent une structuration du contour (ou des régions internes) sous forme de graphes exprimant la connectivité. La détermination du graphe s'effectue à partir d'un point intérieur supposé connu, comme le montre la figure 21 ci-dessous.

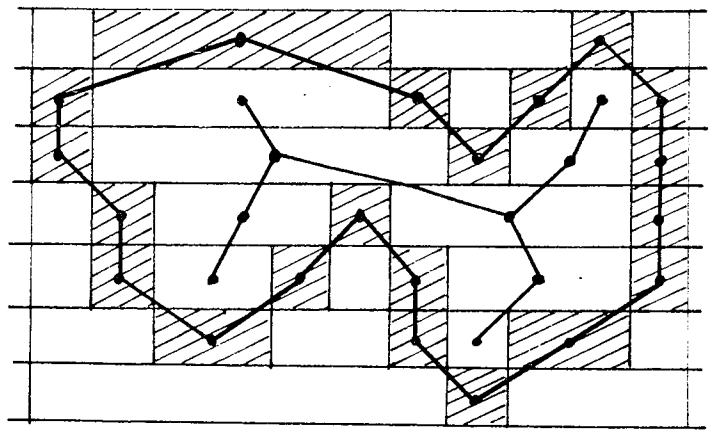


Figure III.21: Graphes des régions d'un contour

Le remplissage effectif consiste alors à parcourir le graphe ainsi obtenu. De nombreuses études ont été publiées sur ce sujet, différant essentiellement par la nature du graphe utilisé et la technique de parcours

employée (Lie78), (Smi79), (Sha80), (Pav81).

4.3.2. Contour formel

Les algorithmes de cette classe travaillent directement sur la définition formelle du contour, dont la numérisation est intégrée au remplissage. A part quelques techniques consacrées à la génération et au remplissage de contours particuliers, tels que les cercles et les ellipses (Coh76), (Por78), (Dor79), MaB79), la grande majorité des algorithmes concerne les taches définies à l'aide de polygones, et met en jeu des techniques nécessitant la numérisation de segments de droite (Bre77), (SKK79), (Bre82), (PiG82).

Dans cette approche les arêtes formant le contour sont étudiées pour mettre en valeur des régions cohérentes composées de triangles ou de trapèzes (voir figure 22 ci-dessous).

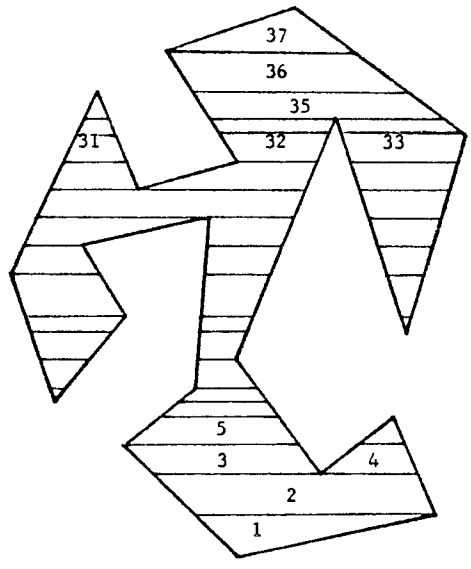


Figure III.22: Régions cohérentes d'un contour polygonal

Les méthodes proposées diffèrent essentiellement par l'ordre dans lequel ces régions élémentaires sont remplies (glissement sur les contours, tris des arêtes selon Y) (Nes79), (Brf79), (Met79) et par la réduction

éventuelle du nombre de régions (Lee81) illustrée sur la figure 23. Ce problème sera étudié en détail au cours du chapitre VII.

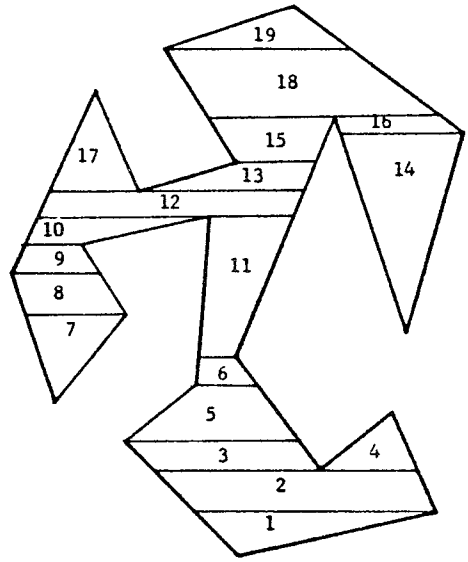


Figure III.23: Réduction du nombre de régions

4.4. Modélisation de l'éclairage

L'étude d'un modèle d'éclairage pour un objet, comporte deux parties distinctes :

- le calcul de la luminance en un point quelconque de l'objet,
- les méthodes permettant d'accélérer, d'améliorer ou de modifier ce calcul pour l'objet tout entier.

4.4.1. Les modèles de calcul de la luminance

Le nombre de facteurs entrant en jeu dans le phénomène d'éclairage d'un point est considérable, et sa modélisation exacte est loin d'être résolue. Néanmoins, la plupart des applications se contentent d'approximations plus ou moins bonnes obtenues à partir de modèles simplifiés. La littérature propose depuis une dizaine d'années un certain nombre de modèles plus ou moins "raffinés" que nous passerons rapidement en

revue en dégageant pour chacun d'eux les paramètres "oubliés".

Afin de faciliter cette comparaison, nous nous appuierons sur une notation commune (figure 24).

θi = angle d'incidence : angle entre $\vec{L}$ et $\vec{N}$

θv = angle de vue : angle entre $\vec{N}$ et $\vec{V}$

α = angle médian : demi angle entre $\vec{L}$ et $\vec{V}$

ds = distance de la source au point considéré

d = coefficient relatif au facteur de réflexion spéculaire du matériau

r1 = coefficient relatif au facteur de réflexion diffuse du matériau

r2 = coefficient relatif à la dispersion de la réflexion spéculaire

Lamb = luminance due à la lumière ambiante

Lmax = luminance maximum autorisée par le matériel.

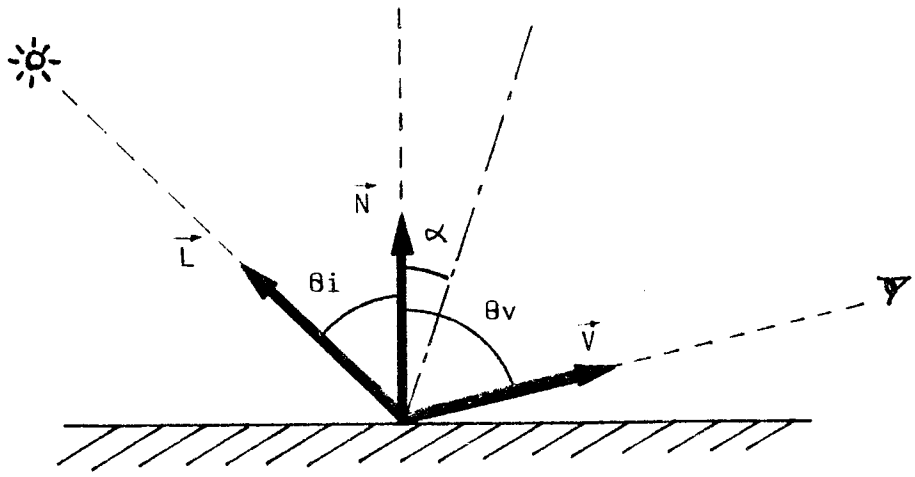


Figure III.24: Géométrie de l'éclairage

Les modèles marqués (*) dans le tableau ci-après, considèrent uniquement le cas où la source lumineuse est située au point de vue (sur l'axe oz).

Le modèle le plus couramment utilisé à l'heure actuelle est celui proposé par B.T. Phong, qui a été repris et amélioré de diverses façons (Bli77), (Whi80). Une étude de R.L. Cook et K.E. Torrance (CoT81), montre qu'il ne s'agit que d'un cas particulier, inadapté pour bon nombre de matériaux, et présente un modèle beaucoup plus proche de la réalité. Cette

Auteurs et références	Modèle	Influence des paramètres					
		facteur de diffusion	facteur de réflexion	facteur de dispersion (reflets)	distance de la source	lumière ambiante	point de vue
APPEL [APP 68]	$L = d.\cos\theta i.L_{max} + r_1(\cos\theta i - \cos\theta v).\cos\alpha$ $d = 0.8$ (en général) $r_1 = 0$ si $\alpha > \frac{\pi}{2}$	oui	oui				oui
BOUKNIGHT-KELLEY	$L = d \lvert\cos\theta i\rvert .L_{max} + L_{amb}$	oui				oui	
GOURAUD [GOU 71] WATKINS [WAT 70]	$L = \frac{\cos^2 \theta i}{z_s}$ | z_s = composante selon oz de la distance d_s				faible		(*)
NEWELL NEWELL.NEWELL.SANCHA [NNS 72]	$L = \cos^{r2}\theta i \,.\, L_{max} + L_{amb}$ (si $\theta i < \frac{\pi}{2}$) $L = L_{amb}$ (si $\theta i \geq \frac{\pi}{2}$)			oui		oui	
NEWMAN-SPROULL [NES 73]	$L = \frac{L_{max} \,.\, \cos\theta i}{d_s} + \frac{r1.(\cos\theta i)^{r2} + L_{amb}}{d_s}$	oui	oui	oui	faible	oui	(*)
PHONG [PHO75]	$L = L_{amb} + d \lvert\cos\theta i\rvert + r1\,(\cos\alpha)^{r2}$	oui	oui	oui		oui	oui
ROMNEY	$L = \frac{\cos^2 \theta i}{d_s^4} \,.\, L_{max}$				forte		(*)
WARNOCK [WAR 69]	$L = \left\lvert\frac{\cos\theta i}{d_s}\right\rvert + \frac{(\cos\theta i)^{r2}}{d_s}$			oui	faible		(*)

amélioration se traduit néanmoins par des formules beaucoup plus complexes et un temps de calcul considérable.

4.4.2. Les modèles discrets

Il est clair que le calcul de la luminance en chaque point de chaque objet est très coûteux, aussi a-t-on cherché à n'effectuer ce calcul qu'en quelques points, pour en déduire la luminance en chaque point.

4.4.2.1. Interpolation linéaire de la luminance

Dans le cas des surfaces gauches approchées par des treillis de faces polygonales, l'attribution de la luminance fait apparaître chacune des faces d'une manière très nette. Gouraud (Gou71) propose dans ce cas de ne calculer la luminance que pour les sommets du treillis, la luminance des autres points étant obtenue par une interpolation linéaire bi-dimensionnelle. L'emploi de l'algorithme de Watkins pour l'élimination des parties cachées, permet en outre d'effectuer cette interpolation linéaire sur une ligne de balayage, éventuellement par un procédé cablé.

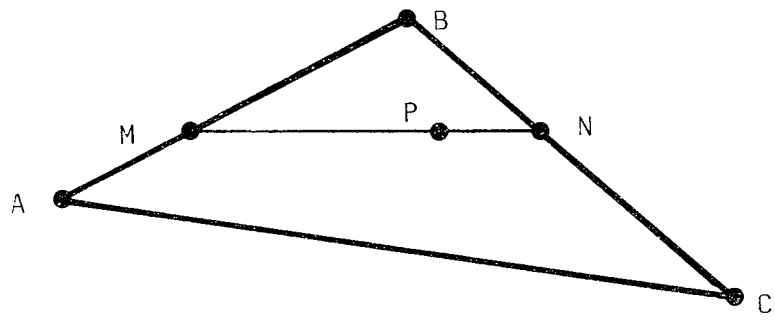


Figure III.25: Interpolation linéaire de la luminance

Soient La, Lb, Lc les luminances calculées aux points A, B, C de la figure 25.

$$\begin{aligned} Ln &= Lb\ (1-a) + Lc\ .\ a \\ Lm &= Lb\ (1-b) + La\ .\ b \\ Lp &= Lm\ (1-x) + Ln\ .\ x \end{aligned}$$

où a, b, et x varient de 0 à 1, respectivement entre B et C, B et A, M et N.

Cette méthode donne à la surface représentée un aspect "lisse" en supprimant les discontinuités lumineuses. Il semble, cependant, que la représentation de reflets soit problématique avec cette méthode a cause de l'atténuation due à l'interpolation linéaire.

4.4.2.2. Interpolation linéaire de la normale

En 1975, Phong (Pho75), (CrP75) propose une amélioration de la méthode de Gouraud. L'interpolation a lieu non plus directement sur la luminance mais sur la normale à la surface, c'est à dire sur les trois composantes de celle-ci. Le temps de calcul est donc supérieur et encore augmenté par le fait que la luminance doit être calculée pour chaque point de la surface. L'avantage de la méthode proposée par Phong est de permettre la présentation exacte des reflets (figure 26).

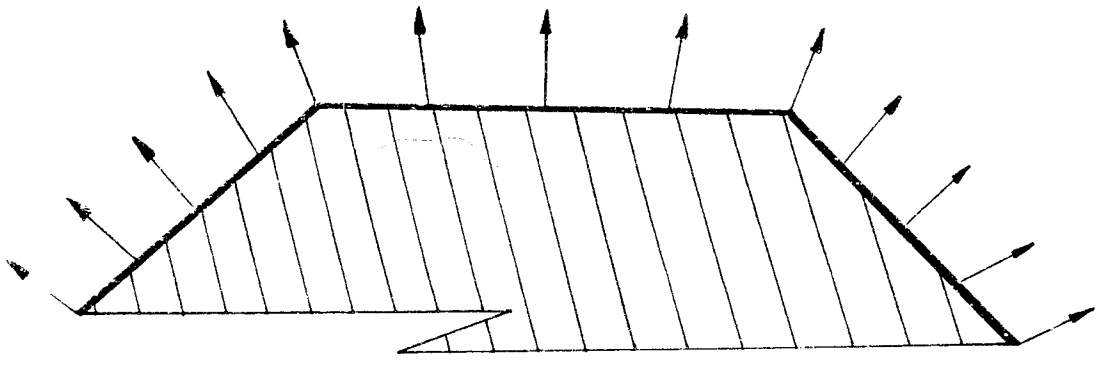


Figure III.26: Interpolation linéaire de la normale

4.4.3. Les textures de relief

De la même manière que pour les textures colorées, la notion de texture de relief augmente considérablement l'aspect réaliste des objets.

La perception du relief s'effectue principalement grâce aux variations de luminance qu'il provoque en perturbant notamment la direction de la normale à la surface. La présentation d'objets dont les surfaces sont irrégulières nécessite, par conséquent, une modulation de la luminance donnant cette illusion de relief.

Cette technique est particulièrement appliquée aux objets présentant des surfaces tourmentées d'une manière aléatoire (peau d'orange, écorce d'arbre, champ labouré, etc.) ou au contraire d'une manière régulière (pavages, tôles ondulées, motifs géométriques, etc.).

4.4.4. Modélisation du relief

Une méthode exposée par Blinn (Bli78) s'inspire de celles évoquées pour les texture colorées.

Le modèle est, ici encore, représenté à l'aide d'une matrice de points dont l'obtention peut être calculée, issue d'une image numérisée, etc.

Le modèle proposé par Blinn représente une fonction F(u,v) exprimant "l'altitude" du relief à ajouter (ou à retrancher) à la surface régulière initiale (figure 27).

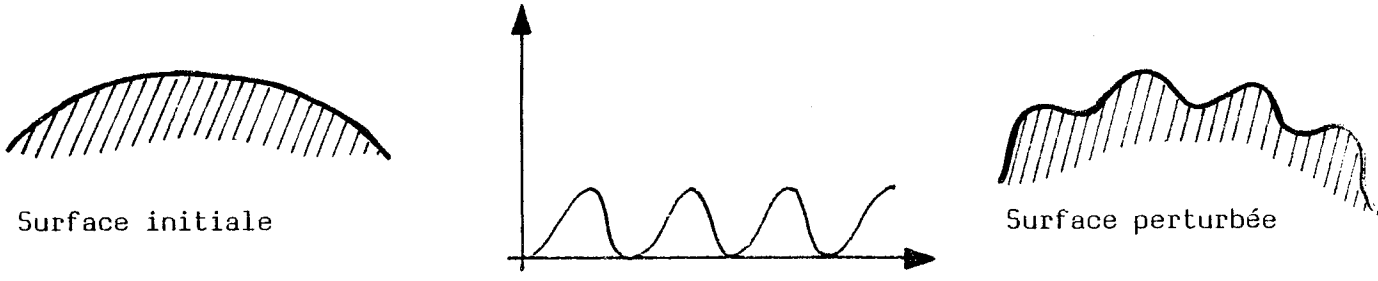


Figure III.27: Fonction d'altitude du relief

Le calcul du nouveau vecteur normal N' en chaque point de la nouvelle surface, est le suivant:

$$\vec{N'} = \vec{N} + \vec{D}$$

avec $\vec{N}$ vecteur normal initial

$$\vec{D} = \frac{F'u\,(\vec{N} * \vec{P}'v) - F'v\,(\vec{N} * \vec{P}'u)}{\| \vec{N} \|}$$

Cette méthode permet d'obtenir des effets d'un réalisme saisissant mais présente néanmoins quelques inconvénients:

- La perturbation engendrée est indépendante de la taille de l'objet. En effet, un grossissement de celui-ci ne provoque aucune augmentation de l'altitude du relief. Blinn propose une technique permettant de supprimer cet inconvénient en considérant la fonction F, non plus comme l'altitude du relief, mais comme l'expression d'un angle de rotation appliqué au vecteur normal.

- Le temps de calcul nécessité par cet algorithme est considérable : environ 4 fois le temps passé pour la présentation du même objet sans relief.

5. L'organisation du logiciel

Nous examinerons dans ce paragraphe les processus les plus caractéristiques de la synthèse d'image actuelle. Aucun système, à notre connaissance, ne prend en charge complètement les processus de visualisation et de description, mais certaines parties du processus sont imposées soit par l'application d'un opérateur de composition particulier (généralement pour l'élimination des parties cachées, ou la détermination des ombres portées), soit par les primitives proposées à l'utilisateur, soit encore par l'organisation même du système logiciel. Afin d'évaluer l'incidence de ces facteurs sur les processus de visualisation nous examinerons successivement les contraintes dictées par les différents algorithmes d'élimination de parties cachées, et les organisations utilisées dans les principaux systèmes généraux de synthèse d'image.

5.1. Influence des algorithmes d'élimination des parties cachées

L'étude des algorithmes d'élimination des parties cachées dont le premier date de 1963 (Rob63), a suscité des travaux de synthèse (SSS74), (Luc77) visant à classifier les divers algorithmes et à mettre en valeur les opérations élémentaires qu'ils nécessitent. Une étude complète réalisée par Ph. Boulle dans le cadre d'une thèse de docteur-ingénieur (Bou80), a permis, non seulement de disposer de quatre algorithmes opérationnels, mais également de mettre en valeur les opérations élémentaires présentes dans les divers algorithmes publiés, de les évaluer et de les améliorer. Nous reprendrons ci-après une partie de ces travaux.

5.1.1. La classification des algorithmes

Si l'on classe les algorithmes en fonction du type des attributs morphologiques acceptés en entrée et du type de ceux produits en sortie on obtient la répartition illustrée par le tableau de la page suivante.

Cependant ce tableau ne reflète pas uniquement le résultat de l'élimination des parties cachées elle-même en tant qu'opérateur élémentaire, mais généralement tout un processus construit autour de cet opérateur. Ainsi la comparaison de ces algorithmes n'est pas significative si l'on ne tient pas compte du fait que certaines publications ne relatent qu'une technique de tri particulière alors que d'autres présentent tout le processus de visualisation, y compris le traitement des attributs A, E, G, Gv, Ga, voire même des opérations de construction (par exemple le découpage des faces).

5.1.2. Les opérateurs de composition

Il est clair que les points critiques des algorithmes d'élimination sont les opérateurs de composition, notamment parce qu'ils sont les seuls à être appliqués à plusieurs éléments et qu'ils conditionnent directement la suite du processus. Les opérateurs de composition utilisés consistent

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

Tableau comparatif des
principaux algorithmes (Bou80)

généralement à trier des éléments par rapport à une direction donnée en fonction de leurs attributs morphologiques et géométriques. Les différents opérateurs se distinguent notamment par:

- le type des éléments concernés: il peut s'agir
  * de points (cas de Z-buffer)
  * de segments coplanaires (cas de Watkins)
  * de polygones (cas de Newell, Atherton, Warnock)
  * de polyèdres (cas de Schumacker)

- la décomposition éventuelle des éléments à classer: le tri peut en effet nécessiter le découpage des éléments en sous-éléments (cas de Newell, Watkins)

- la nature du résultat:
  * liste ordonnée des éléments
  * sélection d'un élément (maximal ou minimal)

La plupart des algorithmes utilisent des critères de tri de plus en plus fins, de manière à éliminer une grande partie des tests dès les premiers passages, pour n'appliquer la comparaison complète qu'aux cas irréductibles. Les opérations de tri qui peuvent agir dans plusieurs directions peuvent alors être imbriquées par souci d'optimisation.

Exemple
1- tri préliminaire selon ox
2- tri préliminaire selon oz
3- tri final selon ox
4- tri final selon oz

5.1.3. Les processus d'élimination

Nous ne prétendons pas décrire ici les processus internes aux algorithmes, qui ont été étudiés par ailleurs (Bou80), mais plutôt l'influence de ces algorithmes sur le processus général de visualisation.

Nous nous appuierons pour cette étude sur quelques exemples, parmi les plus caractéristiques, en faisant apparaître distinctement les opérateurs de tri/décomposition et les opérateurs de construction et de synthèse permettant, à partir d'un élément initial, d'obtenir l'ensemble des points de sa représentation. Nous supposerons pour tous les algorithmes que l'observateur se situe à l'infini sur l'axe oz.

Algorithme du 'Z-buffer' (Cat74)

Cet algorithme se résume en un opérateur de composition unique permettant de sélecter, en chaque point de l'image, le point le plus proche de l'observateur. La partie du processus concernant la synthèse d'aspect n'est effectuée que pour ce point là. Le reste du processus reste entièrement libre. Cet algorithme est celui qui influe le moins sur le processus de visualisation (figure 28).

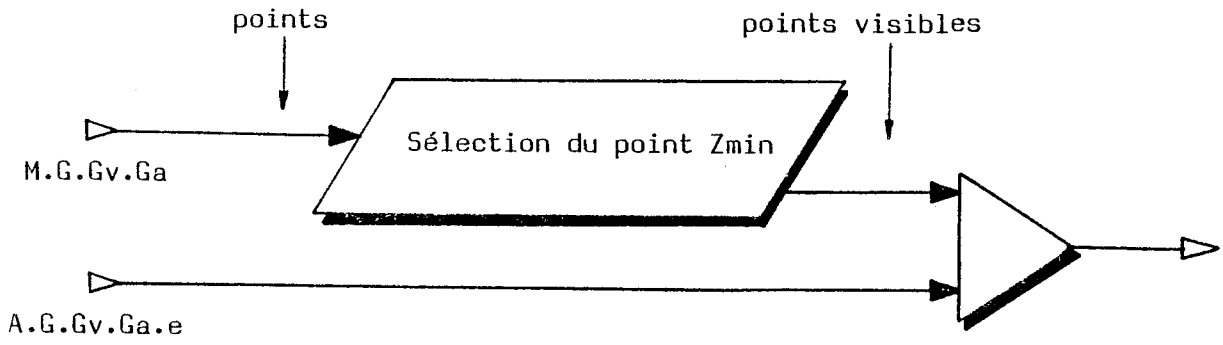


Figure III.28: Algorithme du Z-buffer

Algorithme de Watkins (Wat70) et assimilés

- découpage-Y : intersection des faces avec un plan horizontal

- étude de visibilité des segments ainsi obtenus

- découpage-X : obtention des points d'un segment horizontal visible

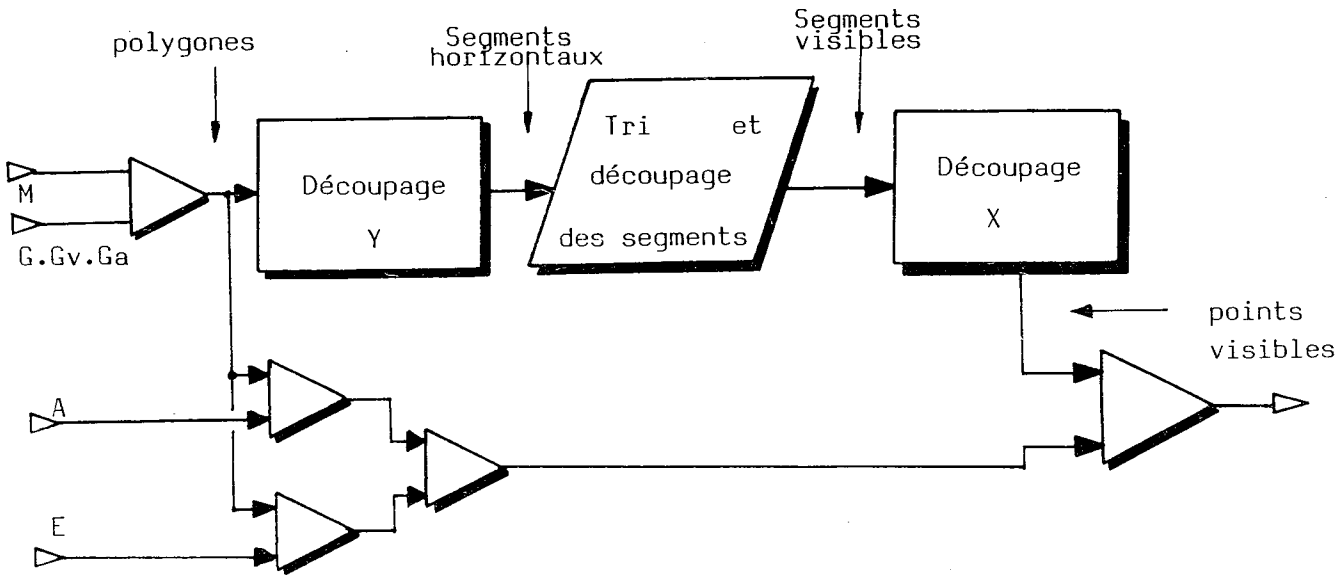


Figure III.29: Algorithme de Watkins

Cette organisation (figure 29) nécessite l'existence de fonctions permettant de déterminer l'aspect et l'éclairage en chaque point, c'est à dire la synthèse M,A,G,E,Gv,Ga. Dans les cas simples (faces planes uniformes) ces fonctions ne dépendent que des attributs initiaux de la face concernée. Dans les cas plus complexes (approximation de surfaces gauches, textures, etc.), les opérateurs de découpage doivent déterminer les attributs des nouveaux éléments crées en fonction des attributs initiaux. Ce processus est illustré par la méthode de lissage proposée par Gouraud (Gou71) (figure 30).

Algorithme d'Atherton et Weiler (AtW77)

Cet algorithme découpe les faces polygonales de la scène pour obtenir l'ensemble des polygones représentant les parties visibles (figure 31).

Du fait de la nature des éléments produits (polygones), la synthèse de l'attribut (Ga) peut être effectuée après l'étude de visibilité. La remarque ci-dessus concernant l'obtention des informations d'aspect est encore valable ici.

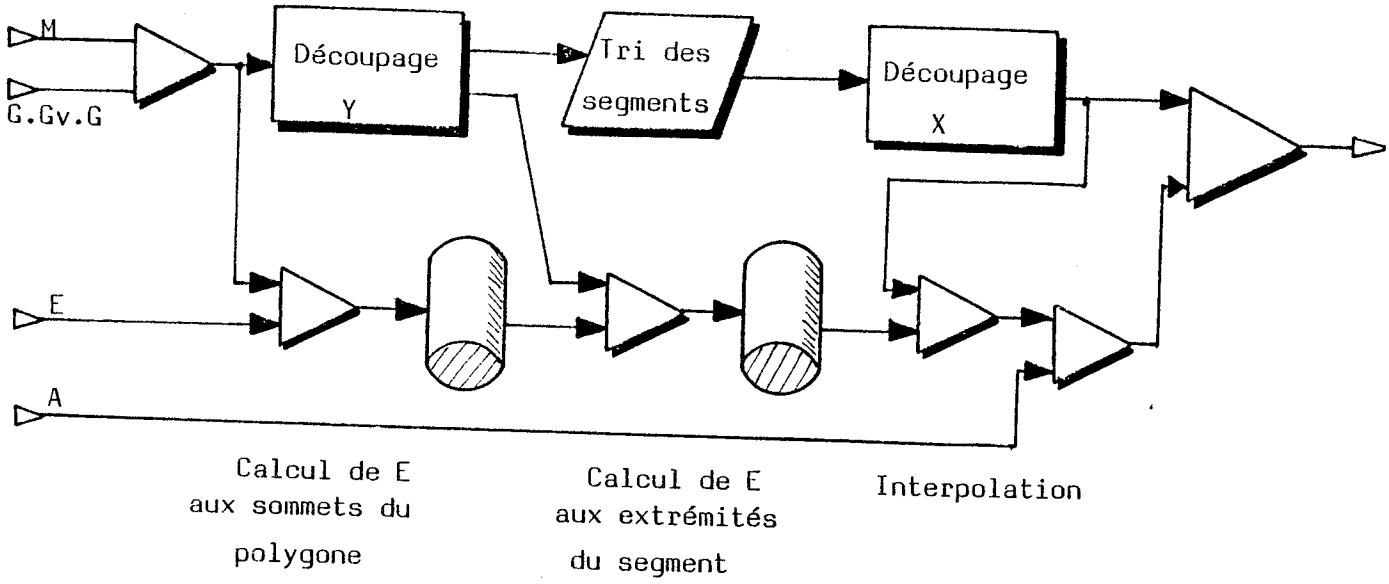


Figure III.30: Lissage par la méthode de gouraud

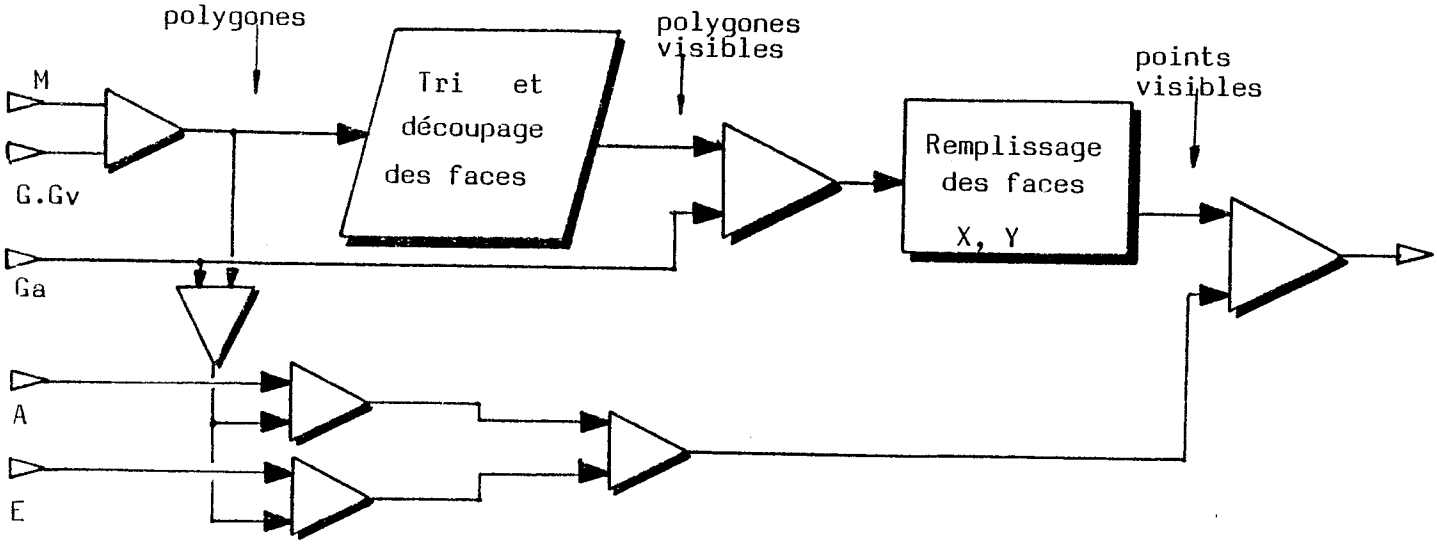


Figure III.31: Algorithme d'Atherton et Weiler

Algorithme de Newell, Newell, et Sancha (NNS72)

Le point intéressant par rapport à l'algorithme précédent est l'introduction d'un opérateur de composition supplémentaire destiné à sélectionner en chaque point le dernier point enregistré (cette fonction est assurée par la mémoire de trame). Il apparaît clairement, sur le processus de la figure 32, que la synthèse de l'aspect et de l'attribut Ga est

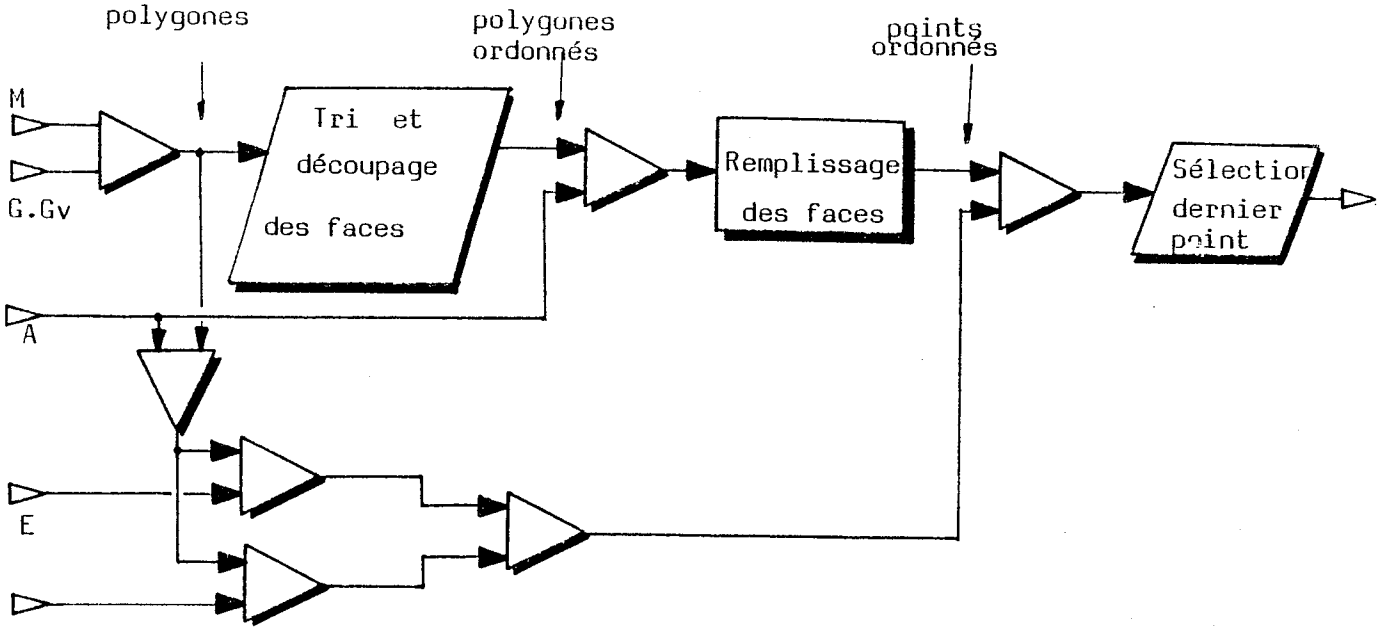


Figure III.32: Algorithme de Newell et al.

effectuée pour tous les points visibles ou invisibles. Il est à noter en outre, que si la même scène est destinée à être vue sous plusieurs angles, le découpage des faces initiales peut être effectué une seule fois, indépendamment du point de vue (figure 33).

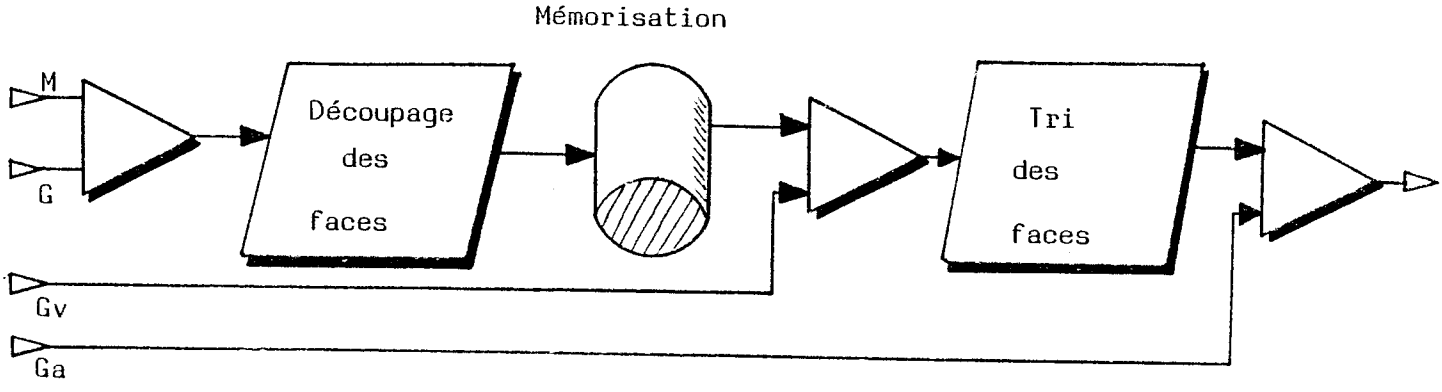


Figure III.33: Découpage préalable des faces

Algorithme de Schumacker (SBG69)

Si l'on ne retient de cette publication que la méthode utilisée pour le classement des faces, on retrouve le processus de l'algorithme ci-dessus dans lequel le tri des faces est effectué en deux temps (figure 34):

- La détermination des priorités statiques (indépendante du point de vue),

- La détermination des priorités dynamiques (fonction du point de vue).

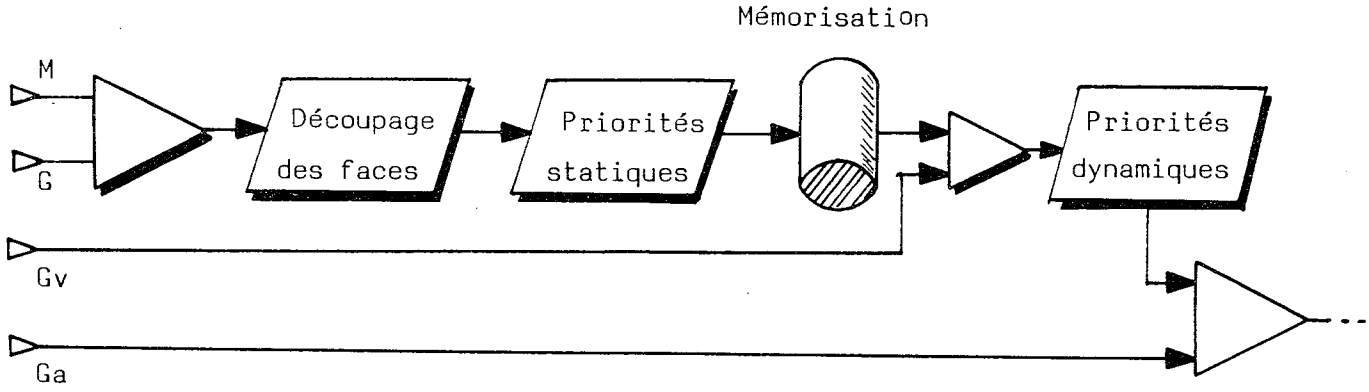


Figure III.34: Algorithme de Schumacker et al.

5.2. Influence des algorithmes de détermination des ombres portées

Les ombres portées (ombrage mutuel des objets d'une scène) fournissent une information non négligeable pour la compréhension d'une image, en particulier en ce qui concerne les positions relatives des objets. En outre la "véracité" de la scène présentée est considérablement accrue par la présence des ombres portées. Nous utiliserons pour présenter les différentes techniques d'ombrage la classification proposée par F.C. Crow (Cro77a).

5.2.1. Ombrage intégré à l'étude de visibilité

Le premier algorithme d'ombrage publié (App68) utilisait cette technique. Appel détecte la frontière des ombres en utilisant la notion de "quantité d'invisibilité" (App67) c'est à dire le nombre de polygones cachant un sommet. Une arête est alors visible si tous ses points ont une

quantité d'invisibilité nulle. Les surfaces ombrées sont détectées durant le balayage horizontal de l'image, en utilisant la quantité d'invisibilité de chaque sommet, vu de la source lumineuse (calculée au préalable) pour déterminer les portions d'arêtes se trouvant dans l'ombre.

Un deuxième algorithme, publié par Bouknight et Kelley (BoK70), utilise une technique similaire, mais présente le gros avantage d'utiliser un algorithme d'élimination de parties cachées s'appuyant déjà sur un balayage horizontal de l'image.

Deux balayages sont effectués (figure 35):

- Le premier est le balayage original qui fait passer de la maquette en trois dimensions à l'image finale.
- Le second transforme, de la même manière, la structure tri-dimensionnelle des ombres, en une information qui sera combinée à la précédente pour donner l'intensité en chaque point de la ligne de balayage.

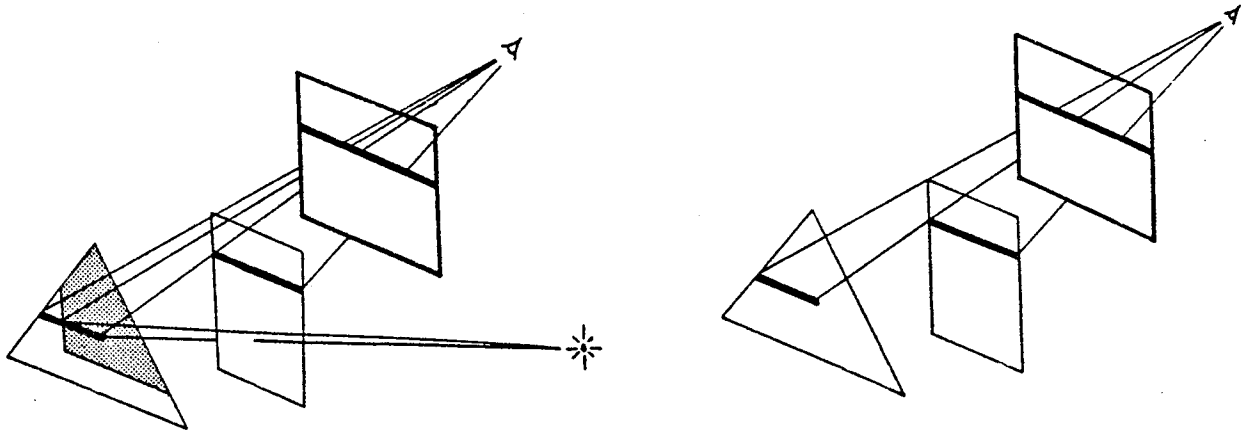


Figure III.35: Méthode du double balayage

L'algorithme est, en outre, notablement accéléré en ne comparant que les paires de polygones pouvant se masquer mutuellement la lumière. Cette détection est effectuée en projetant tous les polygones sur une sphère centrée sur la source, puis en vérifiant leur recouvrement éventuel.

5.2.2. Ombrage par double étude de visibilité

L'idée de base de cette seconde approche est d'utiliser un algorithme d'élimination de parties cachées pour déterminer les portions d'objets "visibles" depuis la source, c'est-à-dire celles qui ne seront pas "ombrées". Néanmoins l'algorithme doit fournir, non pas une image, mais une structure susceptible d'être réintroduite dans l'algorithme (ou dans un autre) pour la détermination des parties réellement visibles depuis le point d'observation. Le nombre d'algorithmes applicables pour la première analyse est donc considérablement restreint.

F.C Crow (Cro77a) propose une solution combinant l'approche récursive de Clark (Cla76) et la méthode de découpage de Sutherland (SpS73).

Nishita et Nakamae (NaN73) proposent pour leur part un algorithme s'appliquant uniquement à des polyèdres convexes et utilisant également une méthode basée sur le découpage (vu de la source) puis le balayage ligne par ligne pour déterminer les surfaces visibles (figure 36).

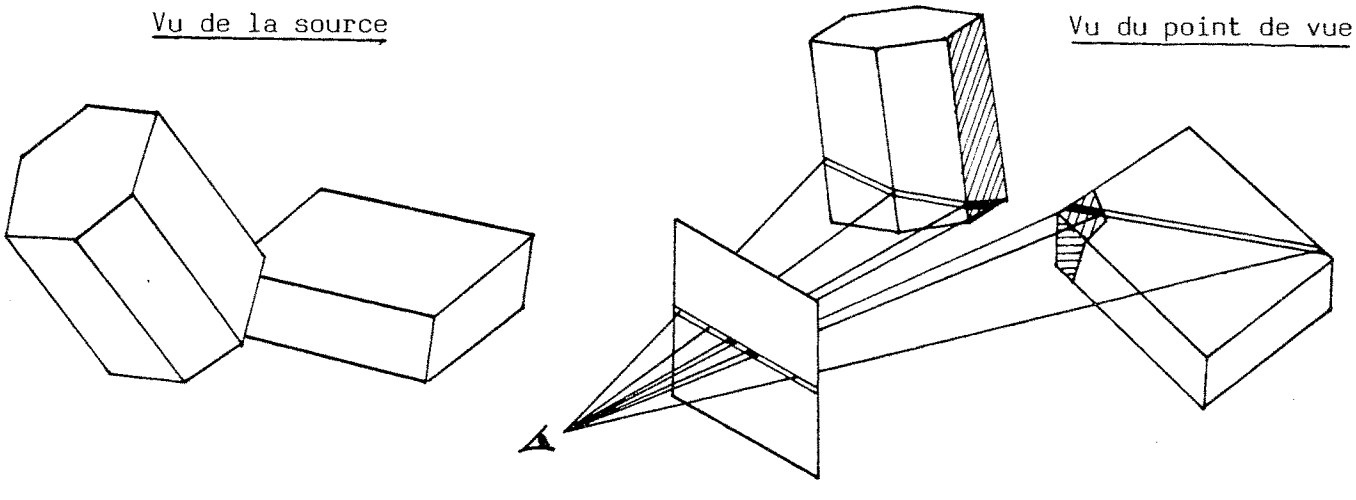


Figure III.36: Découpage des faces ombrées

Toujours basé sur le découpage des polyèdres, l'algorithme publié par Atherton, Weiler et Greenberg (AWG78), s'applique à des polyèdres quelconques. La technique proposée est simple :

- premier passage, vu de la source : détermination des polygones (ou portions de polygones) éclairés et adjonction de cette information à la structure initiale.

- deuxième passage, vu du point d'observation : détermination des polygones (ou portions de polygones) réellement visibles.

Le premier passage est indépendant du point de vue et donc effectué une seule fois dans le cas d'une source immobile. Le cas des sources multiples est traité aisément en effectuant un passage dans l'algorithme pour chacune d'elle.

5.2.3. Ombrage de surfaces gauches

Lorsque les éléments de la maquette ne sont pas représentés à l'aide de faces planes, le besoin d'ombrage soulève des problèmes quasi insolubles d'intersections de volumes dans l'espace. Les temps de calcul prennent alors des proportions inacceptables dans la plupart des applications.

Williams (Wil78) propose, au prix d'une occupation mémoire non négligeable, une méthode permettant ce traitement, aussi bien pour les polyèdres que pour les surfaces gauches, et s'inspirant de la double étude de visibilité.

Le calcul de l'ombrage s'effectue en deux temps à l'aide de deux mémoires de trame, exprimant la profondeur de chaque point : "Z-buffer" (Cat74).

- 1) Obtention dans un "Z-buffer" de l'image vue de la source. Seule la profondeur est calculée.

- 2) Obtention dans un second "Z-buffer" de l'image vue du point d'observation. Chaque point ainsi obtenu est transformé dans les coordonnées (Xp,Yp,Zp) de la première image. Le point est non-éclairé s'il est plus éloigné de la source que le point présent dans le premier "Z-buffer" (Zp > z).

De même que pour les faces polygonales, la première partie de l'algorithme est indépendante du point de vue, et facilement applicable à plusieurs sources lumineuses, en affectant un "Z-buffer" à chacune d'elles.

Le temps de calcul est approximativement le double de celui nécessité par la présentation sans les ombres portées. Le processus schématisé par la figure 37 fait apparaître des possibilités de parallélisme.

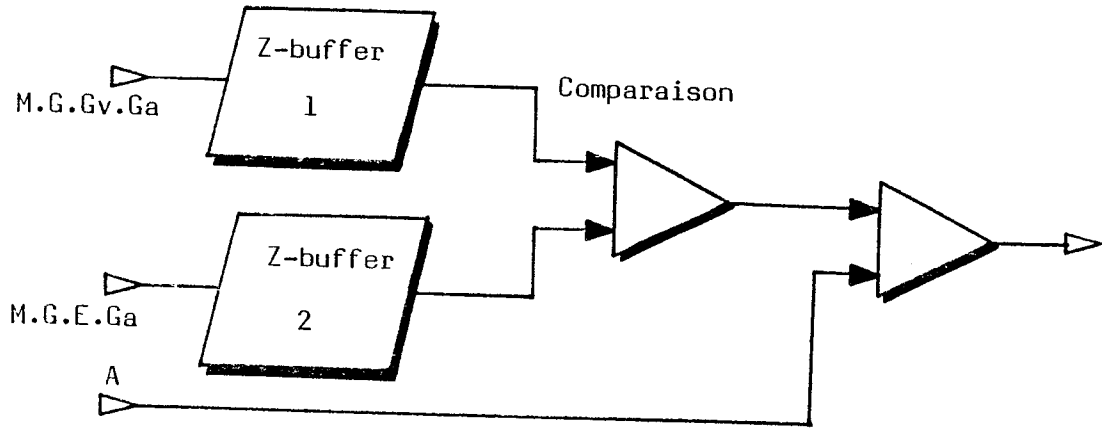


Figure III.37: Utilisation de deux Z-buffers

5.3. Influence de l'organisation interne des systèmes

5.3.1. Organisation générale

On peut classer les systèmes de synthèse d'image existants, en deux catégories distinctes :

- les systèmes généraux permettant d'utiliser divers matériels, pour diverses applications,
- les systèmes spécifiques adaptés à une application et/ou un matériel donné.

Il s'agit là d'une réelle partition, car les systèmes généraux sont incapables de piloter efficacement des matériels sophistiqués tels que ceux

présentés au paragraphe 3 du présent chapitre, et inversement, les systèmes spécifiques sont incapables de piloter d'autres matériels que celui pour lequel ils ont été conçus.

Nous nous intéressons ici uniquement au cas des systèmes dits "généraux" pour évaluer leur influence sur les processus de visualisation, et par suite les facilités offertes à l'utilisateur. Ces systèmes sont généralement partitionnés en trois couches (ou plus) de logiciels (Luc77), assurant l'interface entre les divers matériels et les différentes classes d'applications. La figure 38 ci-dessous illustre cette organisation.

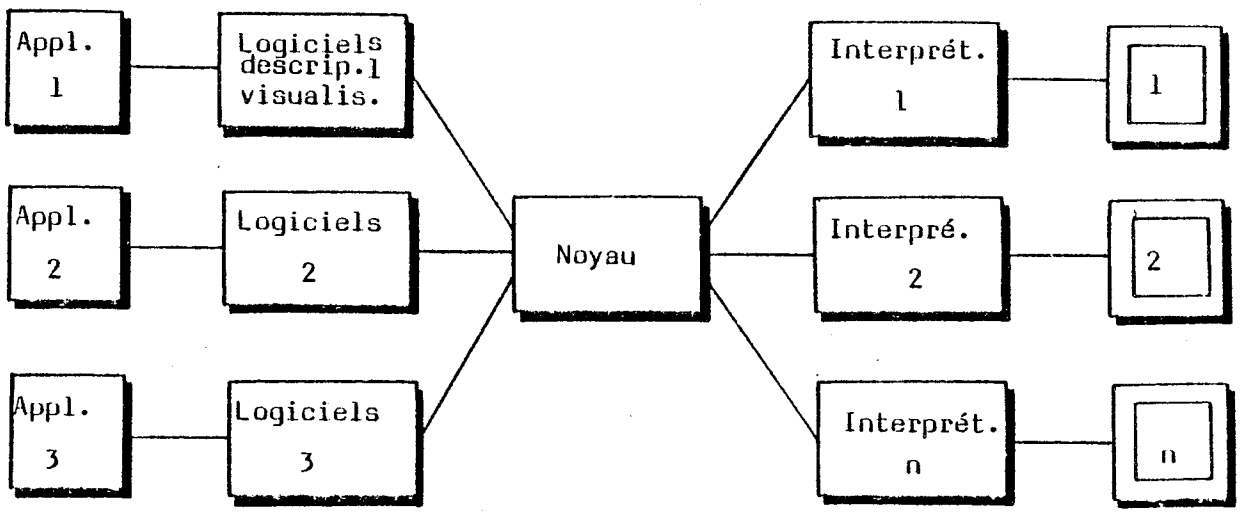


Figure III.38: Organisation en trois couches

Il est clair que tous les processus proposés par ce type de système, possèdent nécessairement une partie commune : celle assurée par le noyau. Il s'agit là, comme nous l'avons vu en II.3.3.3, d'une restriction très importante qui mérite d'être évaluée. Nous examinerons pour ce faire, l'ensemble formé par les deux dernières couches de logiciel; ensemble constituant le logiciel de base graphique.

5.3.2. Les logiciels de base

Le rôle prépondérant du logiciel de base dans l'organisation d'un système, justifie le grand nombre de travaux qui lui ont été consacrés, et les efforts de normalisation dont il fait l'objet. Nous étudierons ici trois logiciels de base qui nous semblent refléter les tendances actuelles dans ce domaine.

- 1) G.K.S. (Graphic Kernel System) (EEK79), (EEK80), en tant que représentant des logiciels indépendants du matériel et configurables. Ce logiciel, issu des travaux du groupe NI-5.9 de l'organisme de standardisation allemand "DIN", diffère de la proposition de norme américaine (GSP77), (GSP79) en ce qui concerne notamment la gestion du poste de travail et les techniques de réalisation utilisées. Nous rangerons également dans cette catégorie, les adaptations directes de la proposition de norme ANSI du GSPG, telles que DIGRAF (WaK79) ou le logiciel réalisé à l'université George Washington (Fow81).

- 2) GRIGRI (LLM76), (Luc77), (LLM78) en tant que représentant des logiciels de base indépendants du matériel mais non configurables. Ce logiciel, conçu et réalisé au laboratoire d'Informatique et de Mathématiques Appliquées de Grenoble, est aujourd'hui commercialisé et distribué au niveau national. La plupart des concepts mis en valeur dans ce logiciel ont servi de base aux travaux présentés dans cette thèse.

- 3) I.G.L. (Interactive Graphics Library) (Hvi79) en tant que représentant des logiciels fournis par les constructeurs. Ce logiciel développé par Tektronix assure une certaine indépendance vis à vis du matériel (à l'intérieur de la gamme Tektronix).

5.3.3. Les primitives du logiciel

Nous ne considérons dans cette étude comparative que les seules primitives ayant un rapport direct avec la synthèse d'image, à l'exclusion de toutes celles servant à gérer le dialogue non graphique et l'interface avec le système (interruptions, fichiers, etc.). Nous tenterons pour chaque

logiciel de classifier ces primitives en fonction du processus concerné : attribution, consultation, description, visualisation.

Les primitives de G.K.S

- Attribution
  * Morphologie : Polygon, Polymarker, String, Pixel-array, Draw
  * Aspect : Fill-polygon, Pixel-array, Visibility, Highlighting, Pen-number, Text-number
  * Géométrie : Transform-segment, Marker-size
  * Eclairage : néant
  * Identité et structure : Insert-segment, Open-segment, Set-pick-identifier, Close-segment
  * Prise de vue : Workstation-window
  * Affichage : Workstation-viewport

- Consultation

  G.K.S. ne permet pas de consulter directement des attributs particuliers, mais autorise la récupération de segments complets : Read-segment, Insert-segment.

- Description
  * Morphologie : Locator, String
  * Identité : Pick

    De nombreuses primitives permettent de configurer le processus de description en spécifiant si un "écho" doit être visualisé, si la saisie doit être effectuée en continu ou avec attente, etc. (Request, Sample, Event, Set-echo,...).

- Visualisation

  G.K.S permet également de configurer dans une certaine mesure le processus de visualisation, en indiquant notamment si la visualisation est :
  * immédiate après chaque primitive,
  * différée jusqu'à la prochaine demande d'interaction,
  * toujours différée, et dans ce cas elle doit être demandée

explicitement à l'aide de la primitive Redraw.

L'utilisateur peut également autoriser ou supprimer la régénération automatique de l'image dans les cas où les changements nécessitent un effacement total de l'écran (tube mémoire).

Bien que ce logiciel soit relativement complet au niveau des types d'éléments 2D proposés et des processus offerts, on peut lui reprocher principalement:

- un manque de cohérence, la plupart des primitives s'adressent en effet à des entités différentes (éléments, segments, "pick-identifiers")
- un manque de symétrie puisque seuls des points et des caractères peuvent être décrits.

D'autres part en ce qui concerne la structuration, la gestion dynamique à l'aide des primitives Open-segment et Close-segment contraint l'utilisateur à structurer ses données graphiques en fonction du déroulement du programme, ce qui va quelque peu à l'encontre des concepts de la programmation structurée, qui tend plus, au contraire, à structurer les programmes en fonctions des données.

Le point fort de ce logiciel réside dans les possibilités de configuration des processus de description et de visualisation.

Les primitives de GRIGRI

- Attribution
  * Morphologie : Points, Texte, Mode
  * Aspect : Mode
  * Géométrie : Mode (pour les caractères seulement)
  * Identité et Structure : Points, Texte, Mode
  * Prise de vue : Fenêtre
  * Affichage : Clôture

- Consultation

Aucune consultation n'est possible, les informations doivent être conservées par le programme d'application.

- Description

* Morphologie : Position
* Identité : Identifier

Les deux processus de description sont prédéfinis lors de la conception du logiciel.

- Visualisation

De même que pour la description, le processus de visualisation est figé et ne dépend que du terminal utilisé. Il s'agit d'une visualisation différée demandée explicitement à l'aide de deux primitives : Afficher, Effacer. Cette dernière combine en fait deux processus:

* attribution d'aspect (invisibilité),
* visualisation proprement dite.

Ce logiciel peut être distingué du précédent par les points suivants:

- Les processus de visualisation et de description sont implicites.

- La structuration offre également deux niveaux, mais "statiques", c'est-à-dire indépendants du déroulement du programme.

- L'ensemble est moins complet du fait que deux types d'éléments seulement sont proposés (sections de points et sections de caractères) et que les transformations géométriques ne sont pas considérées.

- En ce qui concerne la symétrie, on peut regretter l'absence de description au niveau des sections de caractères.

- La cohérence est assez bonne puisque toutes les primitives s'adressent aux mêmes niveaux de structuration (figure, corrélation) pour les deux éléments de base. Seule la primitive Mode présente quelques incohérences:

* elle seule s'adresse au niveau section,
* elle exprime une morphologie pour les points uniquement (segments

disjoints, lignes brisées,etc.)

* elle exprime des informations géométriques pour les caractères uniquement (taille, orientation)
* elle combine structuration (à travers l'affectation du numéro de corrélation) et représentation; deux notions parfois indépendantes.

Les primitives d'I.G.L

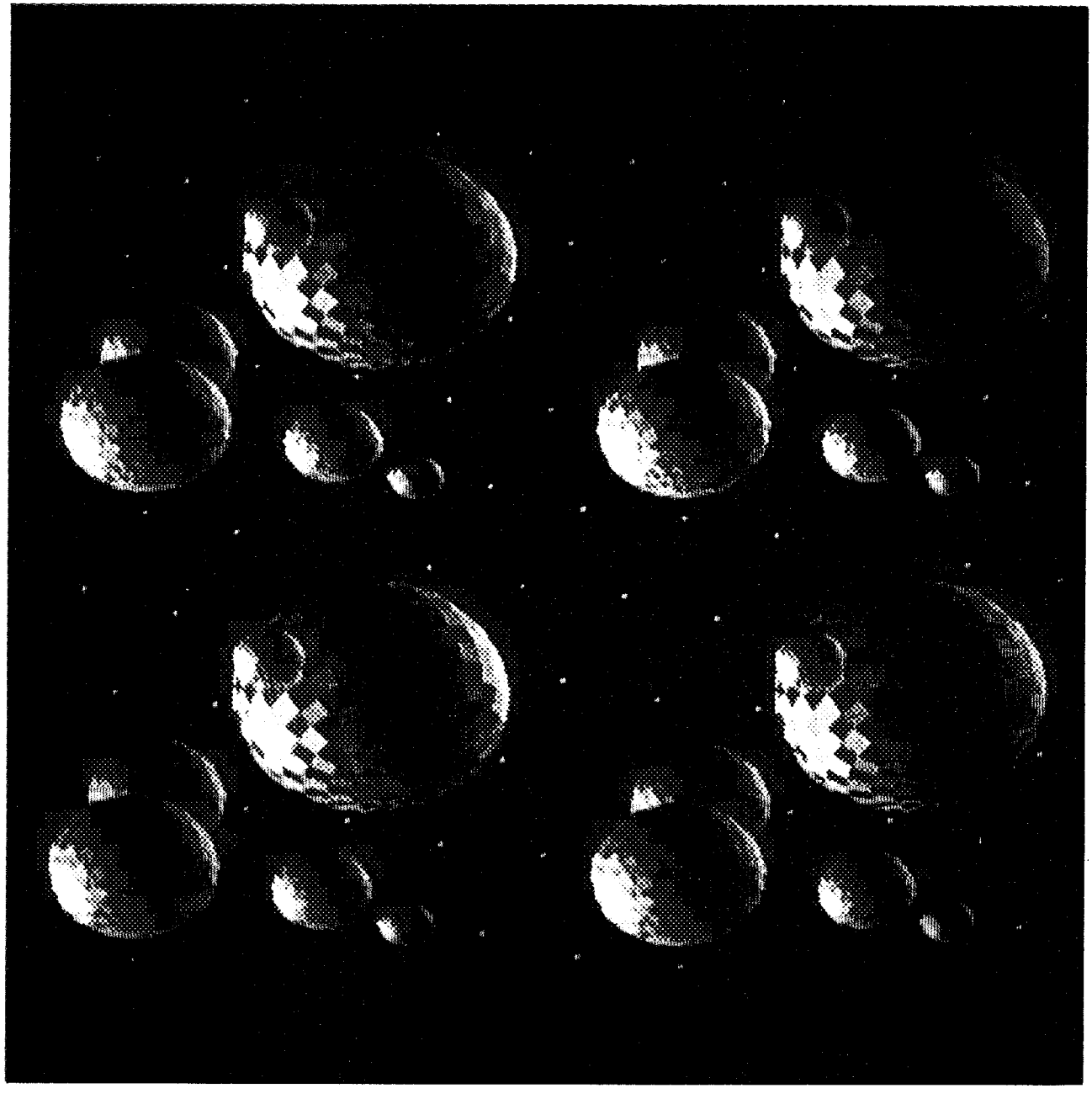
I.G.L. propose aux utilisateurs 91 primitives ayant un rapport direct avec les processus de synthèse d'image et 84 autres primitives servant à la communication avec le système (gestion de fichiers, etc.). Nous nous contenterons d'énoncer quelques règles générales relative à ce logiciel.

Comme son nom l'indique, il s'agit plus d'une bibliothèque de sous programmes que d'un vrai logiciel. La structuration est absente et le processus de visualisation est presque entièrement à la charge du programme d'application. Cette dernière remarque a, d'autre part, une incidence directe sur l'indépendance vis-à-vis du matériel. En effet s'il est bien possible d'utiliser les différents terminaux de la gamme Tektronix, le passage de l'un à l'autre nécessite dans bien des cas la ré-écriture du programme, pour la modification du processus et pour l'invocation de primitives quelque peu différentes. Cette nécessité apparaît notamment lorsque l'option d'affichage rafraîchi (4014, 4114) est utilisée, ou lorsque les terminaux à balayage de trame sont employés (4027, 4112). En outre, l'identification par désignation n'est pas proposée. En revanche, les possibilités d'attribution sont très importantes, particulièrement pour les attributs morphologiques et géométriques pour lesquels de nombreuses primitives sont proposées.

6. Conclusion

Nous avons montré dans ce chapitre les différentes techniques utilisées à l'heure actuelle pour la conception de matériels et de logiciels de synthèse d'images. Cette revue, faite à la lumière des concepts de base présentés dans les chapitres précédents, illustre bien la grande diversité

des approches, nécessaire pour satisfaire le plus grand nombre de situations possibles. Nous avons montré pour notre part, que l'étude des processus de visualisation, de description, d'attribution et de consultation permet de caractériser et d'évaluer toutes ces approches de façon critique. Nous avons en outre soulevé les points faibles et les points forts des réalisations actuelles et nous montrerons dans la deuxième partie de cette thèse de quelle façon nous avons tiré profit de ces enseignements, dans notre expérience personnelle. Bien que très incomplet, ce chapitre nous aura également permis de présenter l'état de l'art de la synthèse d'image ce qui permettra par la suite de situer nos réalisations par rapport à l'ensemble des travaux, et d'en dégager les originalités.



LES PLANETES

DEUXIEME PARTIE

UN EXEMPLE DE REALISATION

ASPECTS LOGICIEL ET MATERIEL

CHAPITRE IV

Présentation générale du système

Ce chapitre présente le point de vue d'ensemble d'un système interactif de synthèse d'image, conçu et réalisé conformément aux concepts et aux notions de base exposés dans la première partie de cet ouvrage. L'étude détaillée des composants logiciels et matériels de ce système fait l'objet des chapitres qui suivent.

1. Les objectifs

1.1. Les applications concernées

Les objectifs qui ont dirigé notre expérience découlent directement du principal domaine d'applications que nous désirions privilégier: celui de la Conception Assistée par Ordinateur.

Ce terme regroupe des applications qui peuvent être très différentes mais qui possèdent quelques points communs:

- les outils graphiques sont généralement à la base du dialogue et doivent par conséquent faire l'objet d'une attention toute particulière.
- L'interaction doit être puissante et rapide pour un maximum d'opérations.
- L'application doit être libérée au maximum des contraintes de modélisation et de structuration inhérentes à la représentation graphique des objets, afin de se consacrer entièrement à l'aspect fonctionnel ou qualitatif.
- Les propriétés, structurelles ou autres, des objets manipulés doivent être conservées par le système de synthèse .

Bien d'autres applications rentrent cependant dans ce cadre là, parmi

lesquelles on peut citer par exemple:

- les applications à tendance pédagogique pour l'illustration de cours, d'algorithmes,etc., ou encore dans le cadre de l'enseignement assisté par ordinateur, pour lequel l'image est un support privilégié.

- les applications à tendance artistique dans lesquelles la finalité du travail est l'image elle-même. Nous incluerons bien sûr dans cette catégorie, la conception de séquences animées, films publicitaires, génériques, etc.

- les applications à tendance scientifique pour la présentation ou la simulation visuelle d'un phénomène quelconque.

1.2. Les performances requises

La liste des applications concernées montre bien la diversité des situations qui peuvent se présenter. Cette diversité se manifeste à tous les niveaux, depuis la nature même de l'image, jusqu'aux types des éléments concernés. La plupart des applications de C.A.O. se contentent d'élaborer des plans ou des schémas, pour lesquels un dessin au trait de très bonne qualité est indispensable mais suffisant. D'un autre côté dans les cas fréquents où l'aspect esthétique est un facteur prépondérant de la conception, la présentation réaliste et anticipée du produit fini procure des possibilités avantageuses. Cette présentation nécessite la synthèse d'images possédant un haut degré de réalisme : prise en compte des textures simulant divers matériaux, des effets de l'éclairage, etc. En ce qui concerne les types d'éléments à considérer, la diversité est encore plus marquée, puisqu'une même application peut nécessiter simultanément des éléments très différents. Dans le cas particulièrement complexe de la présentation de paysages (Mar79b), (AQT79), les objets à considérer sont innombrables et requièrent des modélisations particulières, par exemple:

- les constructions peuvent être modélisées à l'aide de faces planes,

- le terrain peut être modélisé à l'aide de surfaces gauches,

- les éléments "naturels" tels que les arbres peuvent être obtenus à l'aide de régles syntaxiques ou aléatoires,

- les nuages peuvent être simulés par l'utilisation de textures particulières.

La rapidité requise est également très variée, tant par l'ordre de grandeur du temps disponible pour la présentation de l'image que par la nature des opérations à privilégier. Des applications telles que la simulation de conduite, nécessitent le temps réel au niveau de la prise de vue alors que la conception d'un objet demande simplement un temps de réponse "convenable" au niveau des opérations de description.

1.3. Les conséquences

Cette brève revue des performances requises, montre clairement qu'il est illusoire de chercher à satisfaire toutes les situations avec un système mono-processus. Une hypothèse envisageable consiste dans ce cas à laisser à l'application le soin de gérer ses processus. Le système se réduit alors à l'unité de description et de visualisation, c'est à dire qu'il s'agit plus d'une bibliothèque d'actions que d'un système au sens propre du terme. Cette hypothèse est néanmoins assez pénalisante pour l'application puisqu'elle impose, non seulement la détermination et la réalisation du processus, mais encore la connaissance exacte de la modélisation des éléments, de l'organisation des structures de données utilisées, et des spécificités du matériel.

Ces considérations nous ont conduit à réaliser un système se chargeant intégralement de l'exécution des processus. Ce système, présenté dans les pages qui suivent, découle directement des concepts exhibés dans la première partie de cette étude. En vue d'en prouver la validité nous avons choisi de réaliser quelques processus caractéristiques, en appliquant directement ces concepts aux niveaux logiciel et matériel. Nous débuterons ainsi par la détermination des différentes couches de synthétiseurs nécessaires dépendant essentiellement du choix de la configuration matérielle. Après quoi nous envisagerons l'organisation hiérarchique de chacun de ces synthétiseurs,

leur architecture interne et les processus développés ou en cours développement. L'univers de ce système n'est cependant pas limité à ces seuls processus, et nous verrons qu'il est tout à fait envisageable de fournir à l'utilisateur la possibilité d'étendre le système en décrivant de nouveaux processus spécifiques à son application.

2. Les choix fondamentaux

2.1. Le choix de la configuration matérielle

Ce choix a été principalement dicté par le souci d'expérimenter le maximum de situations possibles.

Nous avons cherché à diversifier :

- les technologies d'affichage:
  * balayage cavalier pour le dessin au trait,
  * balayage de trame possédant une bonne définition et une vaste gamme de couleurs pour les images réalistes.

- les configurations:
  * terminal bas de gamme,
  * terminal évolué,
  * calculateur satellite.

- les architectures:
  * pipe-line,
  * parallèle,

- les moyens d'entrée:
  * réticule,
  * tablette à numériser.

Dans le même temps nous voulions prouver l'intérêt d'une forte pénétration des attributs, et démontrer la faisabilité de cette approche dans le cas des images realistes, en intégrant des opérations de synthèse cablées si possible "après la mémoire de trame".

Aucun poste de travail ne présentant, à notre connaissance, ces caractéristiques, nous avons choisi de développer notre propre terminal interactif pour la synthèse d'images réalistes: le prototype HELIOS qui sera présenté en détail au chapitre suivant (MaF81), (Fer81).

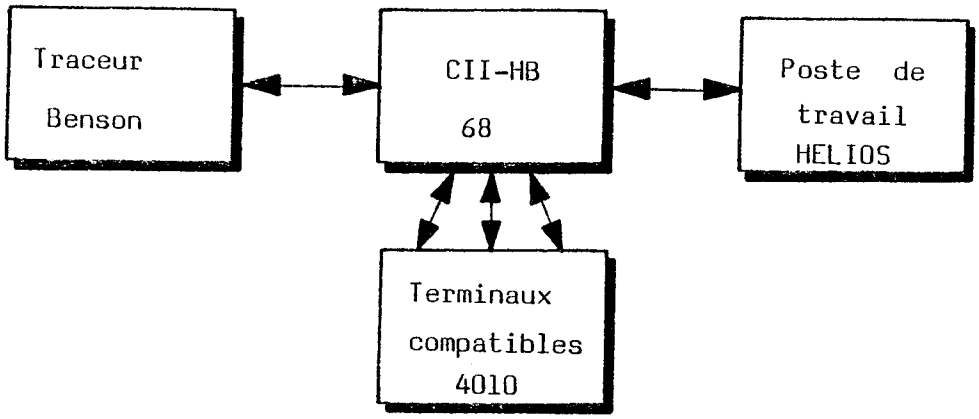


Figure IV.1: Configuration matérielle

L'ensemble de la configuration utilisée, illustrée par la figure 1 se compose des éléments suivants:

- Le calculateur principal: CII HB 68 du Centre Inter-Universitaire de Calcul de Grenoble, pourvu du système Multics opérant en partage de temps.

- Le poste de travail HELIOS: 3 plans de 512*512 points, 4096 couleurs possibles, permettant la modification en temps réel des attributs d'aspect (texture, brillance) et d'éclairage (position source, lumière ambiante etc.).

- Les divers terminaux graphiques disponibles au niveau du laboratoire IMAG (Tektronix 4010, 4014, 4114), DEC Retrographics, Secapa. Tous ces terminaux possèdent la particularité d'être compatibles avec le Tektronix 4010, ce qui nous a permis de ne considérer, dans un premier temps, que ce seul type.

- Un traceur Benson connecté au calculateur principal, accessible à travers le système Multics.

2.2. Le poste de travail HELIOS

La configuration du poste de travail HELIOS est schématisée par la figure 2. Elle regroupe en fait deux synthétiseurs:

- un synthétiseur cablé,

- un synthétiseur programmé sur un calculateur satellite, jouant le rôle de logiciel pilote pour la logique cablée.

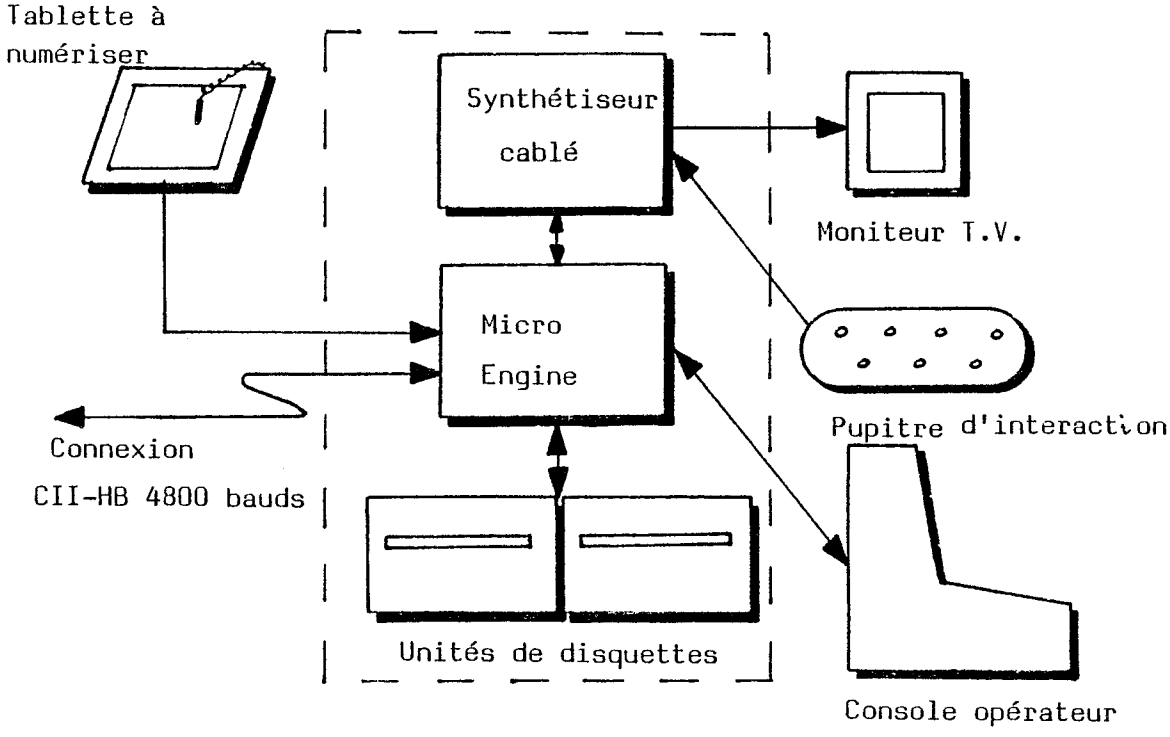


Figure IV.2: Le poste de travail HELIOS

Ces deux synthétiseurs sont rassemblés dans une même unité centrale à laquelle sont connectés divers périphériques que nous examinerons ci-après.

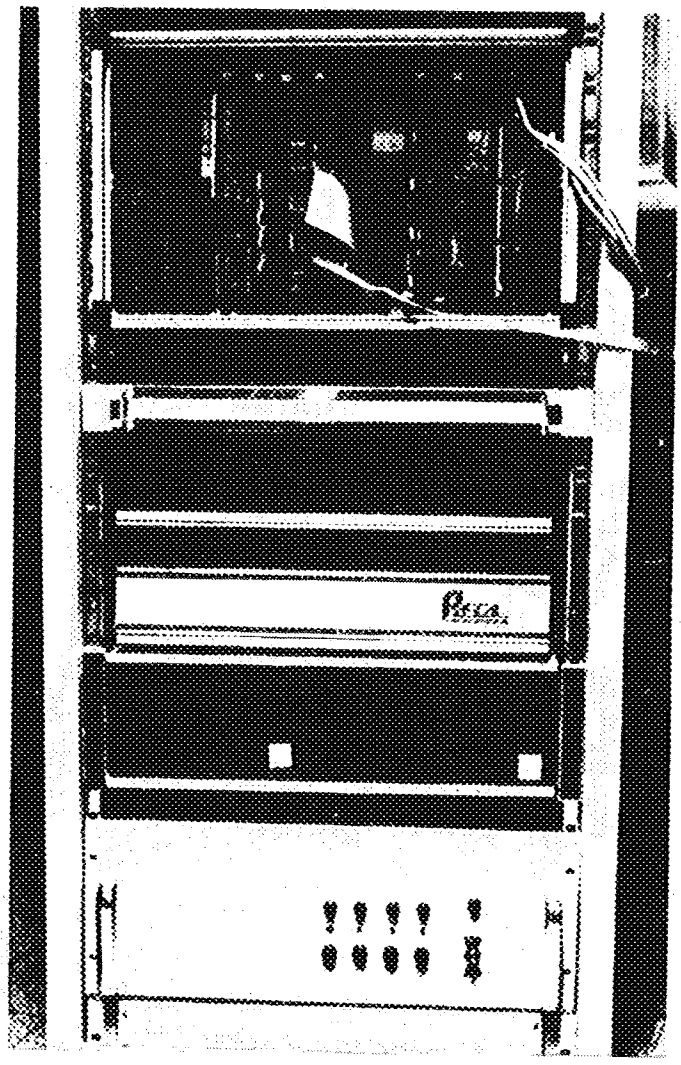


Figure IV.3: Vue du prototype HELIOS

2.2.1 L'unité centrale

L'unité centrale (voir figure 3) rassemble dans une même baie, quatre composants essentiels:

- Le synthétiseur cablé monté dans un rack 19 pouces dans lequel on aperçoit les 13 cartes qui ont été nécessaires à sa réalisation.

- Le calculateur pilote MicroEngine de Western Digital,

- La double unité de disques souples Remex 40 (double face, double densité) permettant de mémoriser 2M octets,

- Le tiroir des alimentations, rassemblant les quatre alimentations nécessaires à la logique cablée.

Le calculateur est une machine à mots de 16 bits interprétant directement le P-code fourni par la compilation de programmes Pascal ou Ada, ce qui a considérablement facilité la mise au point du logiciel. Le système fourni avec le MicroEngine comporte:

- un éditeur de programmes,

- un compilateur Pascal U.C.S.D.,

- une gestion complète des fichiers sur disques,

- un éditeur de liens permettant de rassembler des "unités" compilées séparément,

- des utilitaires divers.

Quatre ports d'entrée-sortie sont également disponibles et sont répartis comme suit:

- Un port série V24 est affecté à la console opérateur ou à la connexion au calculateur principal. Il est configuré à 9600 bauds.

- Un second port série est réservé pour les périphériques locaux.

- Un port spécial est réservé à l'unité de disques souples,

- Un port parallèle est réservé à la communication avec le synthétiseur cablé d'HELIOS.

2.2.2. Les périphériques

Quatre périphériques peuvent être actuellement connectés à l'unité centrale.

- 1) Le moniteur T.V.: Il s'agit d'un moniteur Sony à tube trinitron, comportant quatre entrées haute impédance: rouge, vert, bleu, et synchronisation séparée. Il est possible en outre de régler indépendamment les niveaux et le gain des trois composantes RVB afin d'assurer la "compensation de gamma" et d'obtenir un blanc pur pour la valeur maximale.

- 2) Le pupitre d'interaction: Connecté directement au synthétiseur cablé, ce pupitre permet notamment, le contrôle du réticule, ainsi que quelques

animations locales: déplacement de la source lumineuse, des fenêtres, etc.

- 3) La tablette à numériser Sored : Cette tablette, d'une dimension de 50 cm * 50 cm est connectée directement sur un port série du calculateur local. Elle permet la récupération de points, en continu ou point par point avec une résolution de 0,025 mm. Un curseur et 5 touches de fonction sont également disponibles.

- 4) La console opérateur: Nous avons choisi pour cette console un terminal graphique bas de gamme: Rétrographics VT640 de DEC. Ce choix permet, outre les fonctions alphanumériques d'un TTY classique, le dessin au trait à partir d'un jeu de commandes compatibles avec le Tektronix 4010. Ces possibilités graphiques sont particulièrement intéressantes pour présenter simultanément l'image et le dessin d'une même scène ou encore pour les opérations locales de description utilisant la tablette à numériser.

- 5) L'imprimante Diablo 1641: Cette imprimante permet également la reproduction de dessins avec une résolution assez faible.

2.3. Les différentes configurations

On peut constater que deux types de configuration matérielles peuvent se présenter:

- une configuration comportant deux couches de synthétiseurs : calculateur principal, terminal compatible 4010 ou traceur Benson,

- une configuration comportant trois couches : calculateur principal, calculateur pilote, synthétiseur cablé ou console opérateur (également compatible 4010).

Le schéma de la figure 4 illustre les différentes possibilités:

Il est clair qu'au niveau des deux types de synthétiseurs issus du commerce (traceur Benson, terminal type 4010) les processus sont prédéfinis

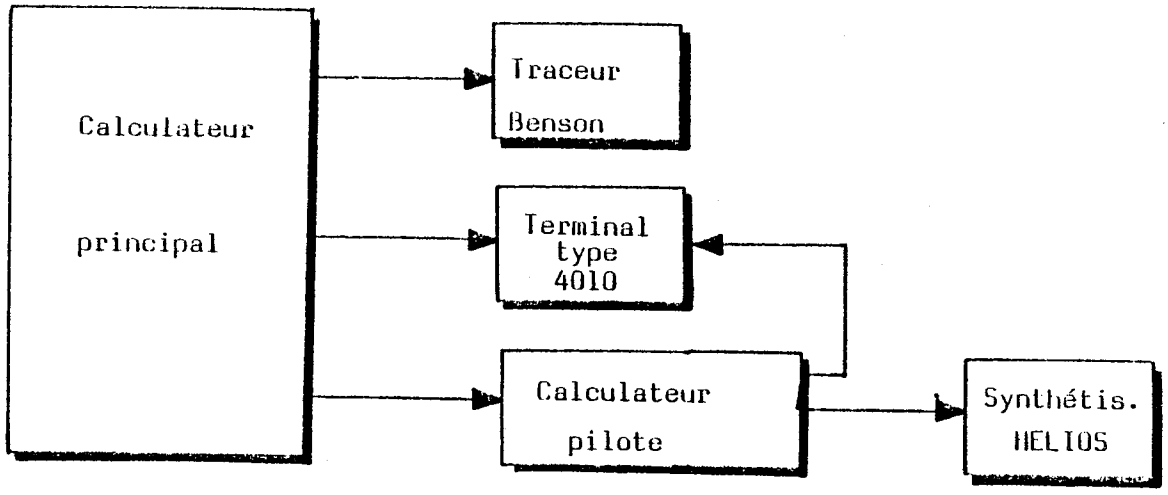


Figure IV.4: Différentes possibilités de connexions

par le constructeur et ne peuvent être modifiés. En revanche, en ce qui concerne les trois autres, aucune contrainte ne vient entraver le choix des processus ou de l'organisation de chaque synthétiseur. La configuration utilisant le synthétiseur cablé d'HELIOS nous a donc permis d'implémenter d'un bout à l'autre nos concepts de base et d'en tester la validité. Nous axerons donc plus spécialement la présentation du système vers cette configuration, qui comporte en outre des particularités originales.

3. L'architecture générale du sytème

3.1. Organisation générale

Nous avons bien entendu opté pour une organisation hiérarchique de chacun des trois synthétiseurs développés. L'organisation générale du système (dans le cas de la configuration HELIOS) est illustrée sur le schéma ci-après (figure 5).

Ces trois synthétiseurs seront détaillés dans les chapitres qui suivent. Nous nous bornerons ici, à énoncer quelques généralités permettant de donner une vue d'ensemble du système et des possibilités de chacun de ses composants.

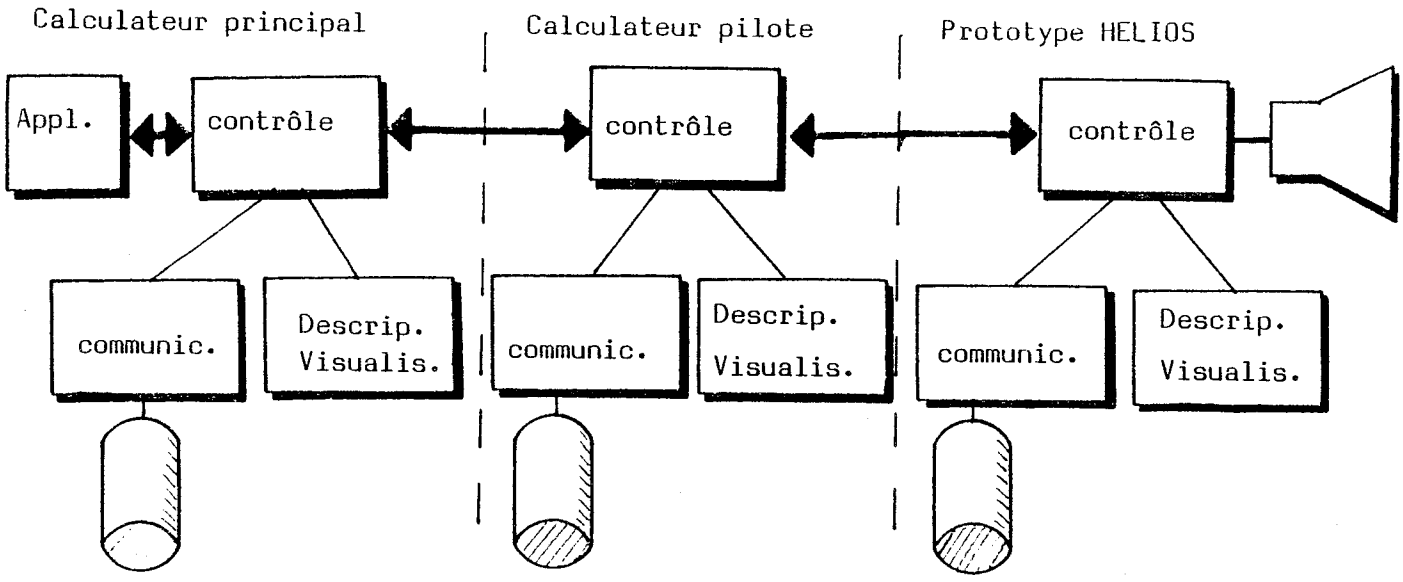


Figure IV.5: Organisation générale du système

3.2. Les synthétiseurs cablés

3.2.1. Le prototype HELIOS

Etant donné qu'un synthétiseur entièrement cablé est limité généralement à un processus unique, le choix de ce dernier est prépondérant pour les performances du synthétiseur et même du système tout entier. Le processus d'HELIOS dont le choix est justifié au chapitre suivant est représenté sur la figure 6.

L'élément de base considéré est la face plane. Les types des attributs manipulés appartiennent aux classes:

- I.M.G.Gv : projection de la face sur le plan de la vue et remplissage des points intérieurs avec l'identité de la face,
- A : texture, visibilité, transparence, et réflexion de la face,
- G.Gv : projection du repère-3D de la face dans le plan de la vue,
- E : description de la source lumineuse (couleur, lumière ambiante, direction,

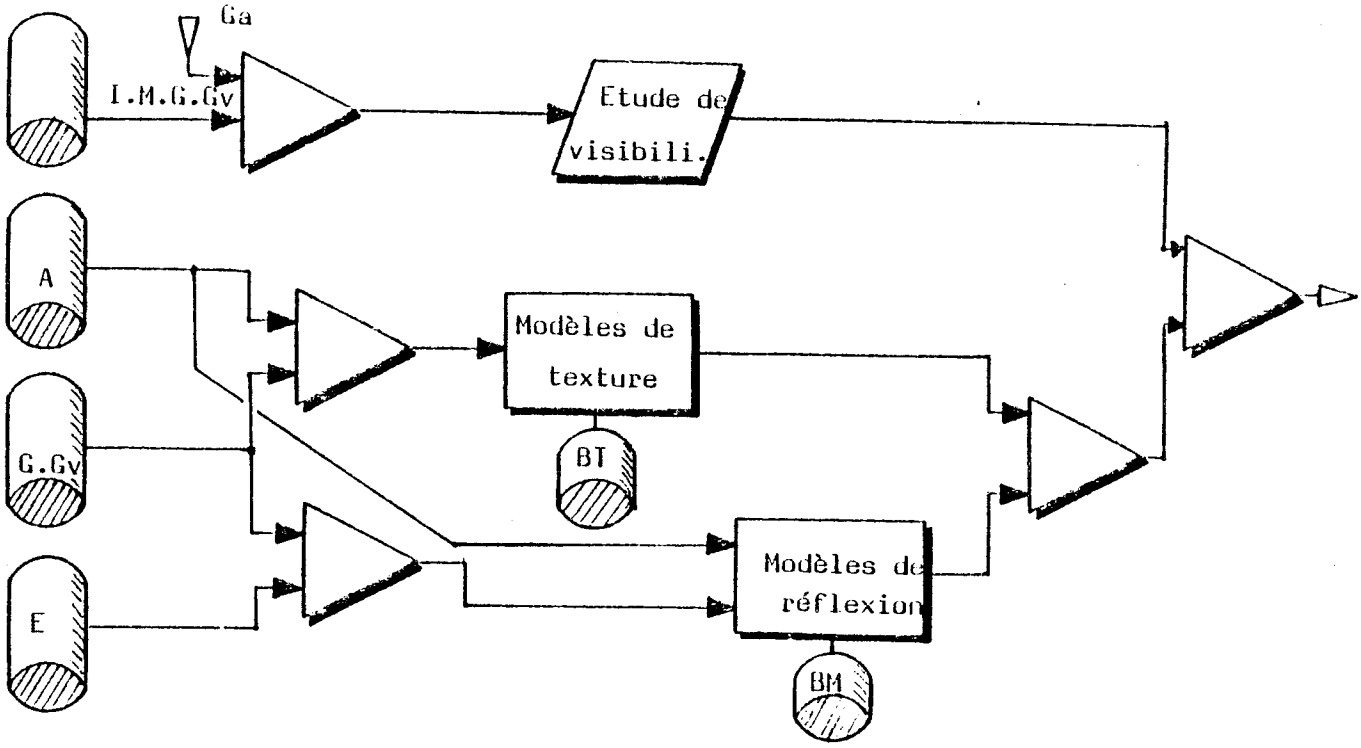


Figure IV.6: Processus cablé

- Ga : description de la fenêtre à projeter sur l'écran.

Deux banques de données sont également intégrées au matériel:

- BT : banque des textures,

- BM : banque des modèles de réflexion.

Les processus cablés permettent :

- l'attribution et la consultation de tous les attributs mentionnés ci-dessus,

- la description et la consultation des coordonnées écran d'un point à l'aide d'un réticule intégré,

- la visualisation en temps réel de toutes les faces enregistrées. Ce processus est implicite et s'exécute continuellement.

3.2.2. Les terminaux compatibles 4010

Le processus des terminaux compatibles avec le Tektronix 4010, est beaucoup plus simple comme en témoigne la figure 7 ci-après.

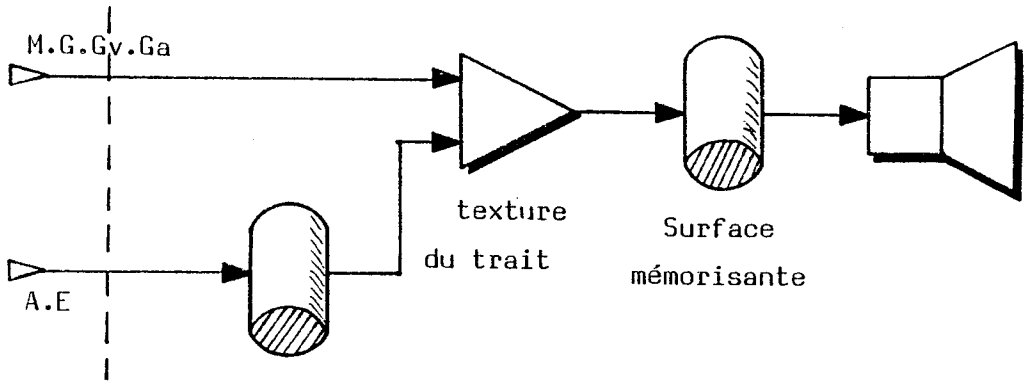


Figure IV.7: Processus cablé des terminaux type 4010

Les types d'attributs acceptés sont :

- dans la classe M.G.Gv.Ga :
  * suite de vecteurs disjoints
  * ligne brisée
  * suite de points
  * suite de caractères

- dans la classe A :

  un type unique indiquant la texture du trait : trait plein, tireté court, tireté long, pointillé.

Les processus proposés sont :

- attribution des types ci-dessus,

- destruction de tous les attributs enregistrés (effacement complet de l'écran),

- description et consultation des coordonnées d'un point à l'aide du réticule,

- visualisation implicite du contenu de la mémoire.

Ces processus ne permettent ni l'effacement selectif d'un élément ni l'identification par désignation.

3.3. Les synthétiseurs programmables : CLOVIS

Le système comporte deux synthétiseurs programmables simulés respectivement par le logiciel du calculateur principal et celui du calculateur pilote. Ces deux logiciels se partagent la partie du processus de visualisation qui n'est pas prise en charge par les synthétiseurs cablés. La détermination optimale du partitionnement des processus est liée à la puissance de calcul et de mémorisation du calculateur pilote.

Nous avons choisi de réaliser deux couches de logiciel entièrement symétriques et cohérentes, organisées toutes deux de manière hiérarchique et prenant chacune en charge une partie des divers processus de visualisation développés. Les deux logiciels acceptent les mêmes types de primitives:

- attribuer (identite-éléments,type-attribut,info)
  qui permet d'associer un attribut à un ou plusieurs éléments

- collecter (identité-éléments,type-attribut,retour-info)
  qui permet de récupérer l'attribut d'un élément

- visualiser (identité-éléments,processus)
  qui permet d'appliquer un processus de visualisation aux éléments considérés

- décrire (identité-éléments,type-attribut,processus)
  qui permet d'appliquer un processus de description à l'attribut 'type-attribut' des éléments considérés.

Les différences essentielles entre les deux couches de logiciel résident dans :

- la structuration des éléments : nous avons opté pour une structure arborescente au niveau du synthétiseur principal alors qu'une structure linéaire a été retenue dans l'unité de communication du synthétiseur pilote.

- le support physique des attributs mémorisés : mémoire centrale au niveau principal, disque souple au niveau pilote.

- les types d'attributs manipulés et les opérateurs élémentaires qui dépendent du processus et du partitionnement choisi.

- le mode d'invocation des processus : véritables primitives au niveau principal; commandes codées au niveau pilote.

- les opérations locales proposées. Seul le logiciel pilote propose à l'heure actuelle des opérations permettant la description et la gestion de banques de données locales.

Les unités de ces deux logiciels ont été conçues et réalisées dans le cadre d'un projet d'étude baptisé CLOVIS (Complexe Logiciel pour la Visualisation Interactive Structurée). Ce projet, qui a débuté en 1979 (Led79), (Mar80), (Tar80), est lui même issu de réflexions menées grâce aux enseignements apportés par la conception et l'utilisation du logiciel GRIGRI.

4. Les éléments de base du système

Nous présenterons dans ce paragraphe les types des attributs et des éléments considérés par le système, tel qu'il est vu de l'application. Ces éléments sont donc ceux qui sont proposés à l'utilisateur, et pour lesquels les quatre processus de base (attribution, consultation, visualisation, description) sont disponibles (ou en cours de réalisation).

4.1. Les types d'attributs

Les types d'attributs actuellement considérés découlent des possibilités offertes par les postes de travail disponibles (Tektronix 4010 et HELIOS), et également des applications envisagées dans le cadre de la C.A.O.

4.1.1. Morphologie

Outre la forme elle même, les attributs morphologiques permettent de différencier les éléments "pleins" des éléments "fil de fer", les éléments bi-dimensionnels des tri-dimensionnels, les éléments plans des éléments gauches. Les types d'attributs morphologiques considérés à l'heure actuelle sont les suivants:

Les éléments "fil de fer" exprimés en 2D

Mpoints-2D : suite de points
Msegments-2D : suite de segments exprimés par leur origine et leur extrémité
Mligne-2D : ligne brisée exprimée par ses sommets
Mcercle-2D : cercle exprimé par son centre et son rayon
Mtexte : suite de caractères (code ASCII)

Les éléments "fil de fer" exprimés en 3D

Mpoints-3D : -
Msegments-3D : -
Mligne-3D : -

Les éléments "pleins" exprimés en 2D

Mface-2D : surface exprimée par le ou les polygones qui composent son contour
Mdisque : surface circulaire exprimée par son centre et son rayon

Les éléments "pleins" exprimés en 3D

Mface-3D : surface plane exprimée dans l'espace par le ou les polygones qui composent son contour

Mfonction : surface gauche exprimée par une fonction explicite à deux variables.

4.1.2. Aspect

La modélisation de l'aspect étant extrèmement dépendante du matériel, la plupart des paramètres d'aspect sont exprimés à l'aide de mots clés référençant des possibilités locales du poste de travail. Deux types d'aspect sont acceptés :

- Asurf : Il exprime l'aspect d'une surface plane et regroupe
  * le nom de la texture plane utilisée,
  * la visibilité, la transparence et le clignotement,
  * le nom du modèle de réflexion.

- Atrait : Ce type exprime l'aspect d'un élément à représenter à l'aide de traits de points ou de caractères. Il rassemble:
  * le nom de la couleur,
  * la visibilité et le clignotement,
  * le nom de la texture du trait (plein, tireté court, tireté long, pointillé) ou de la police des caractères.

4.1.3. Géométrie

Deux types d'attributs géométriques sont proposés, dépendant essentiellement du nombre de dimensions du référentiel absolu (2D ou 3D)

- G3d : Exprime la géométrie d'un élément tri-dimensionnel. Il comprend essentiellement la matrice de transformation appliquée à l'élément.
  * Matrice 4* 4 exprimée en coordonnées homogènes

- G2d : Exprime la géométrie d'un élément bi-dimensionnel.
  * Matrice 3 * 3 exprimée en coordonnées homogènes
    La troisième composante peut être utilisée pour exprimer éventuellement la "profondeur" de l'élément.

4.1.4. Eclairage

Cet attribut utilisé uniquement par le terminal HELIOS ne comporte qu'un seul type.

- E : Caractérisation des conditions d'éclairage
  * couleur de la source exprimée en (RVB),
  * inte,nsité de la lumière ambiante,
  * vecteur directionnel unitaire de la source lumineuse exprimé par ses composantes (x, y, z). dans le repère absolu associé à l'écran.

4.1.5. Géométrie de la prise de vue

De même que pour la géométrie des éléments, deux types sont proposés, dépendants du nombre de dimensions:

- Gv3D: Exprime la prise de vue d'une maquette tri-dimensionnelle.
  * matrice 4 * 4 exprimant en coordonnées homogènes les projections, les rotations, les translations et les homothéties à appliquer pour la prise de vue.
  * limites du domaine visible.

- Gv2D: Exprime la prise de vue d'une maquette bi-dimensionnelle.

- matrice 3 * 3 exprimant en coordonnées homogènes les transformations à faire subir aux éléments.

- limites de la fenêtre visible.

4.1.6. Géométrie de l'affichage

Un seul type d'attribut est considéré ici.

- Ga: Exprime la clôture dans laquelle l'image doit être affichée, en indiquant
  * les coordonnées (x, y) de l'origine du repère attaché à la clôture,
  * les limites de la clôture: x min, x max, y min, y max

    Toutes ces coordonnées sont exprimées en (1/512)ème d'écran.

4.2. Les types d'éléments de CLOVIS

Nous avons vu qu'un type d'élément résulte de l'association: d'une morphologie, d'un aspect, d'une géométrie. On trouvera ci-après pour chaque type d'élément, les types d'attributs qui le composent.

4.2.1. Les éléments 'fil de fer'

- 1) Dans le plan (2D)
  * Points-2D : (Mpoints-2D, Atrait, G2d)
  * Ligne-2D : (Mligne-2D, Atrait, G2d)
  * Segments-2D : (Msegments-2D, Atrait, G2d)
  * Cercle-2D : (Mcercle-2D, Atrait, G2d)
  * Texte : (Mtexte, Atrait, G2d)

- 2) Dans l'espace 3D
  * Points-3D : (Mpoints-3D ou Mpoints-2D, Atrait, G3d)
  * Ligne-3D : (Mligne-3D ou Mligne-2D, Atrait, G3d)
  * Segments-3D : (Msegments-3D ou Msegments-2D, Atrait, G3d) Les éléments exprimés en 2D sont situés dans l'espace à l'aide de l'attribut G3D. Tous ces éléments seront représentés à l'aide de traits, quel que soit le terminal et le processus utilisés.

4.2.2. Les éléments pleins

- 1) Dans l'espace 2D 1/2 (succession de plans parallèles)
  * Face-2D *: (Mface-2D, Atrait ou Asurf, G2d)
  * Disque-2D : (Mdisques-2D, Atrait ou Asurf, G2d)

- 2) Dans l'espace 3D
  * Face-3D *: (Mface-3D ou Mface-2D, Atrait ou Asurf, G3d) (face polygonale plane)
  * Disque-3D : (Mdisque-3D Atrait ou Asurf, G3d) (disque plan)

  Ces éléments sont représentés sous forme de taches ou de traits suivant le processus employé et le type de l'attribut d'aspect (Atrait ou Asurf).

4.2.3. Les éléments gauches

Un seul type de surface gauche est accepté dans cette première version. Il s'agit des surfaces obtenues à l'aide de fonctions explicites à deux variables $z = f(x,y)$, définies sur un domaine rectangulaire parallèle aux axes ox et oy. Cet élément est défini comme suit:

- Fonction : (Mfonction, Atrait ou Asurf, G3d)

  où Mfonction exprime les limites du domaine de définition et l'adresse de la procédure de calcul de la fonction. La représentation de la fonction sous forme de traits ou de surface, dépend du processus et du type d'aspect (Atrait ou Asurf).

4.3. Les processus de visualisation

Le processus de visualisation, qui est mentionné par l'utilisateur dans la primitive "visualiser", tient compte:

- de la configuration matérielle utilisée, c'est-à-dire des processus offerts par le terminal et éventuellement par le logiciel pilote.

- du type de représentation souhaité : dessin au trait ou par tache pour les éléments d'attribut "Asurf", élimination ou non des parties cachées

pour les éléments plans ou gauches.

- des propriétés globales de la maquette : par exemple la présence simultanée d'éléments plans et gauches, ou encore des propriétés géométriques particulières, assurant par exemple que les différents éléments plans et gauches ne se coupent jamais.

Nous donnerons ci-après quelques exemples de processus proposés par CLOVIS, en utilisant leur représentation schématique.

4.3.1. Les éléments fil de fer sur terminal bas de gamme

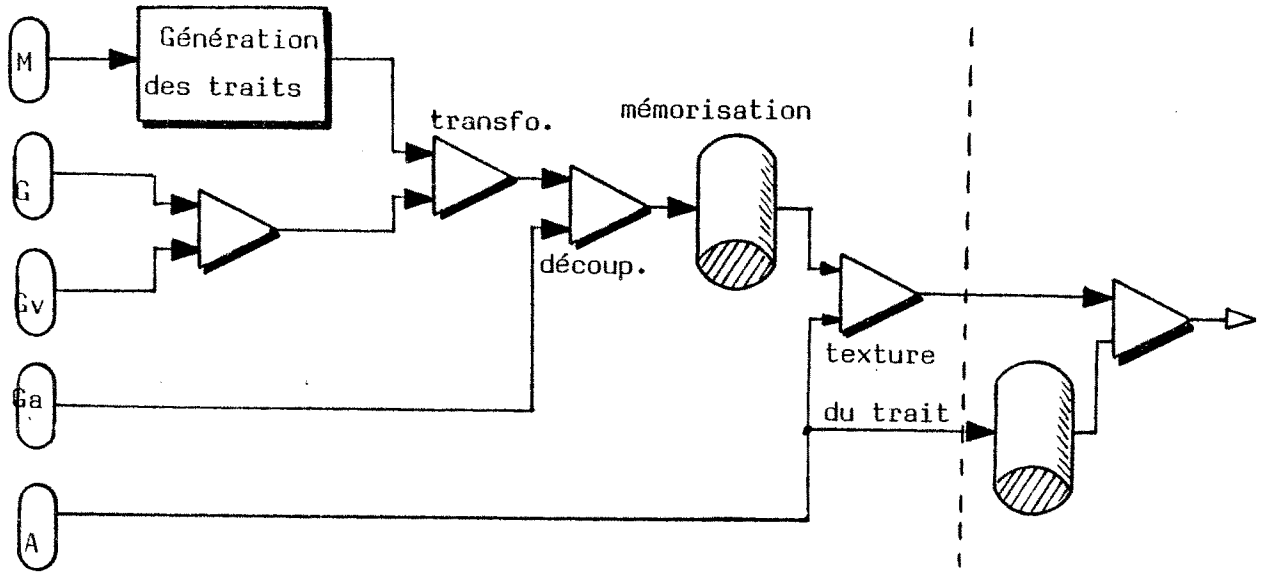


Figure IV.8: Eléments fil de fer sur terminal bas de gamme

Ce processus s'applique à une maquette ne comportant que ces éléments "fil de fer". On peut noter la mémorisation intermédiaire, nécessaire pour permettre l'effacement sélectif et l'identification par désignation. Ces opérations seront détaillées en VI.5. L'opérateur de synthèse d'aspect du logiciel prend en charge la génération des textures de trait non assumées par le matériel (traits mixtes, tiretés longs, etc.), et la génération des caractères non standards.

4.3.2. Les éléments fil de fer sur HELIOS

Comme le précédent, ce processus (figure 9) ne s'applique qu'à une maquette ne comportant que des éléments "fil de fer".

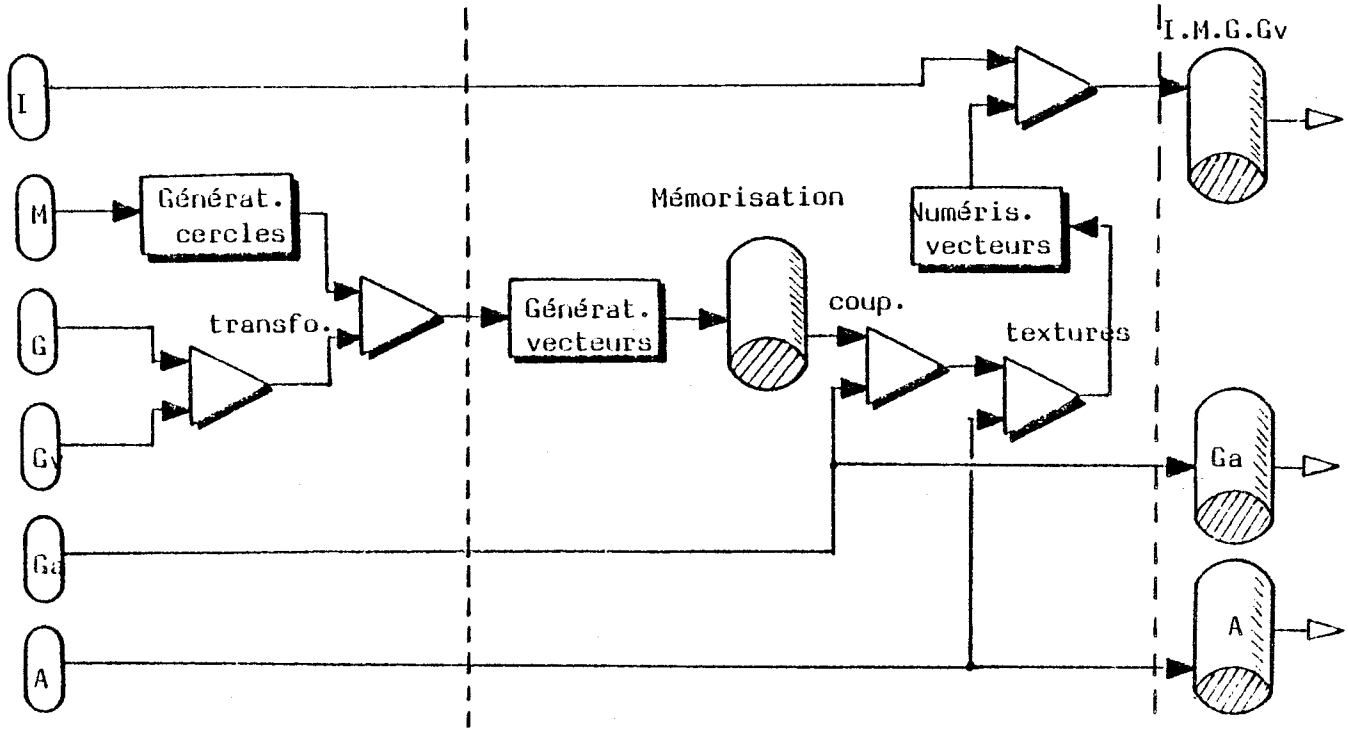


Figure IV.9: Les éléments fil de fer sur HELIOS

Le logiciel pilote assume une grande partie des opérations. La mémorisation intermédiaire s'effectue au sein du synthétiseur cablé et du logiciel pilote. L'effacement sélectif et la désignation sont directement gérés au niveau du synthétiseur cablé.

4.3.3. Les éléments pleins sur terminal bas de gamme

Ce processus (figure 10) accepte à la fois des éléments pleins et des éléments "fil de fer". La représentation est faite sous forme de dessin au trait. A noter l'opération d'élimination des lignes cachées qui impose la ré-exécution du processus complet pour tous les éléments, quelle que soit la modification effectuée. Seule, la modification de la texture du trait est privilégiée et peut être effectuée indépendamment.

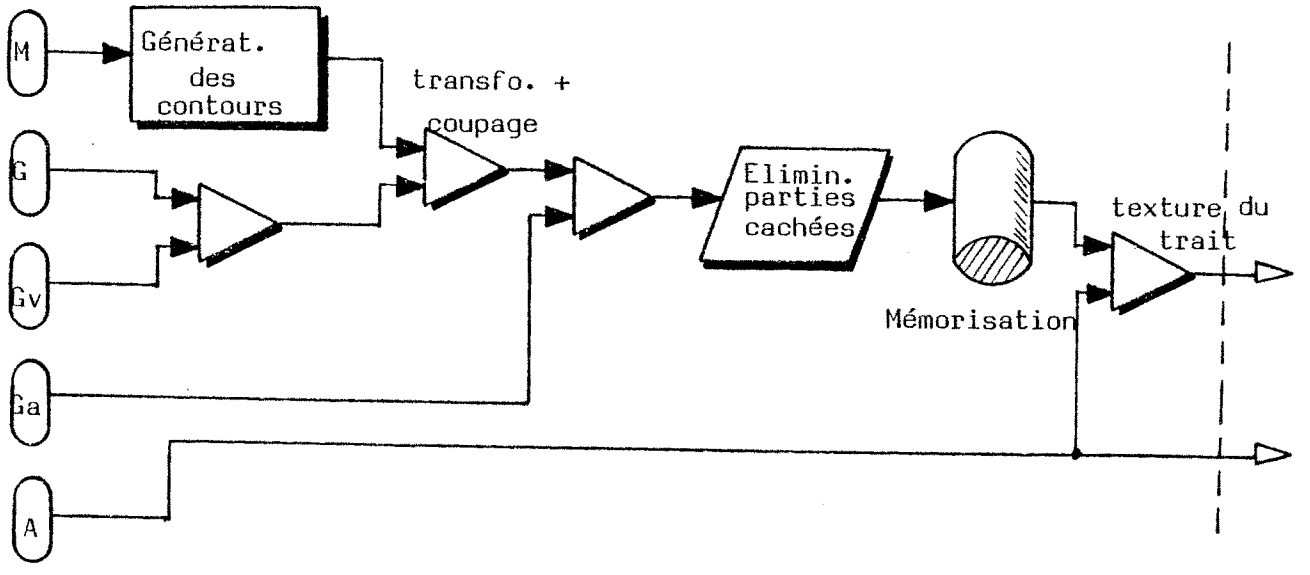


Figure IV.10: Les éléments pleins sur terminal bas de gamme

4.3.4. Les éléments pleins sur HELIOS

Ce processus (figure 11) ne considère que des faces planes qui ne s'inter-pénètrent pas. La présentation est faite sous forme d'image réaliste.

Ce processus utilise un algorithme d'élimination de parties cachées original inspiré de celui de R.A. Schumaker (SBG69) qui travaille en deux temps, en considérant tout d'abord les priorités statiques dépendant de la morphologie et de la géométrie des éléments, puis les priorités dynamiques dépendant du point de vue. Cet algorithme sera détaillé au chapitre VII.

Dans le cas où les faces s'inter-pénétrent, nous avons choisi de nous ramener au cas précédent en faisant précéder le processus de visualisation par une opération de découpage des faces, analogue à celle utilisée dans l'algorithme de Newell, Newell et Sancha (NNS72). Cette opération ne s'appliquant qu'aux faces polygonales, le processus de visualisation est modifié de façon à ce que la génération des contours soit effectuée avant le découpage (figure 12).

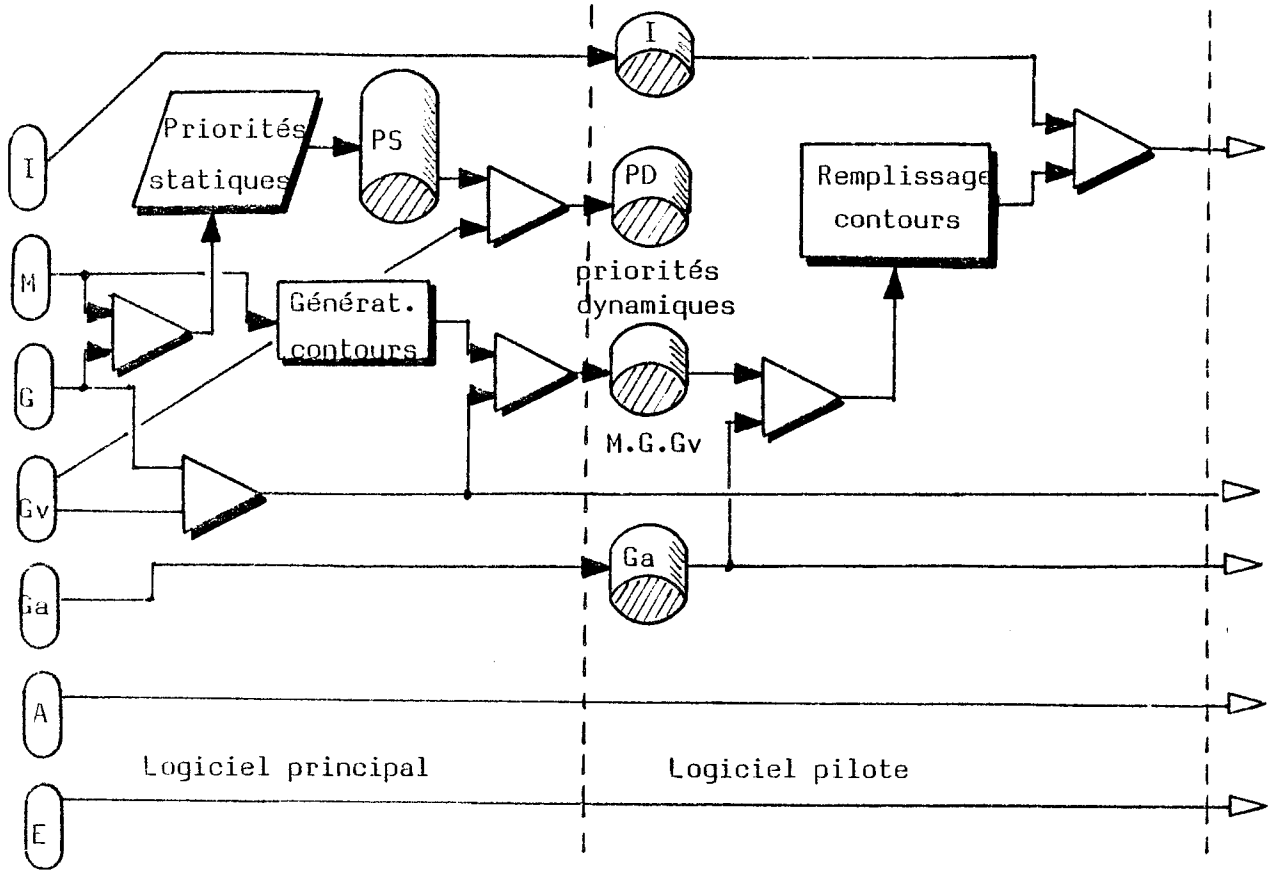


Figure IV.11: Les éléments pleins sur HELIOS

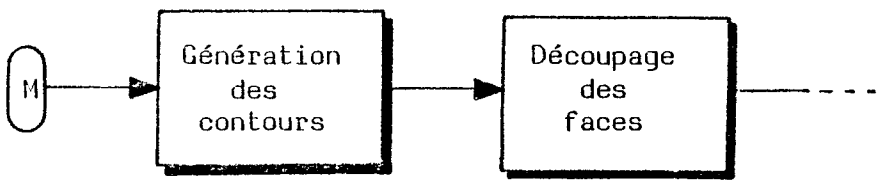


Figure IV.12: Cas où les faces s'inter-pénètrent

4.3.5. Les éléments gauches sur terminal bas de gamme

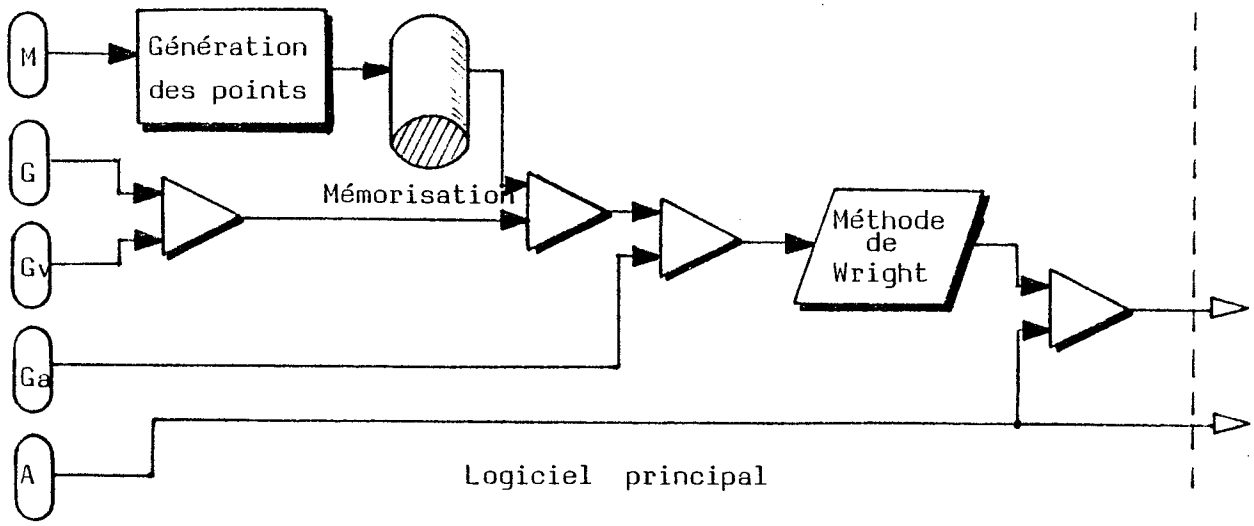


Figure IV.13: Les éléments gauches sur terminal bas de gamme

Nous avons choisi le processus de visualisation proposé par Wright (Wri73), permettant une présentation rapide des fonctions, sous forme de traits (figure 13).

4.3.6. Les éléments gauches sur HELIOS

Nous utilisons pour ce processus (figure 14) une opération implantée dans le logiciel pilote et permettant notamment le "remplissage" de faces quadrilatères gauches. Cette action, détaillée en VII, détermine le vecteur normal en chaque point de la face en fonction des quatre vecteurs normaux aux sommets du quadrilatère. Le processus utilise également la technique des priorités statiques et dynamiques pour le tri des faces.

On notera dans ce processus, la perte des attributs d'identité et la synthèse des attributs géométriques effectuée par le logiciel pilote (détermination des normales).

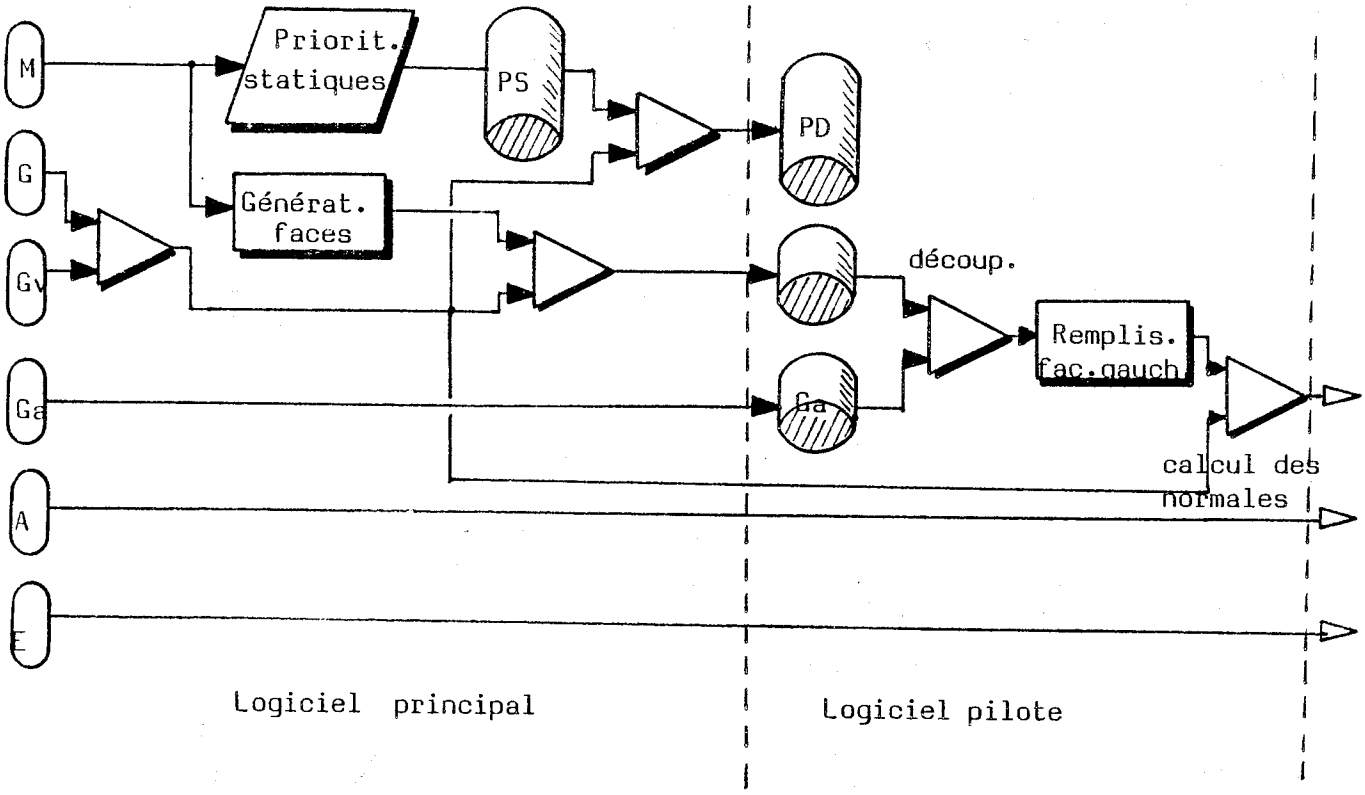
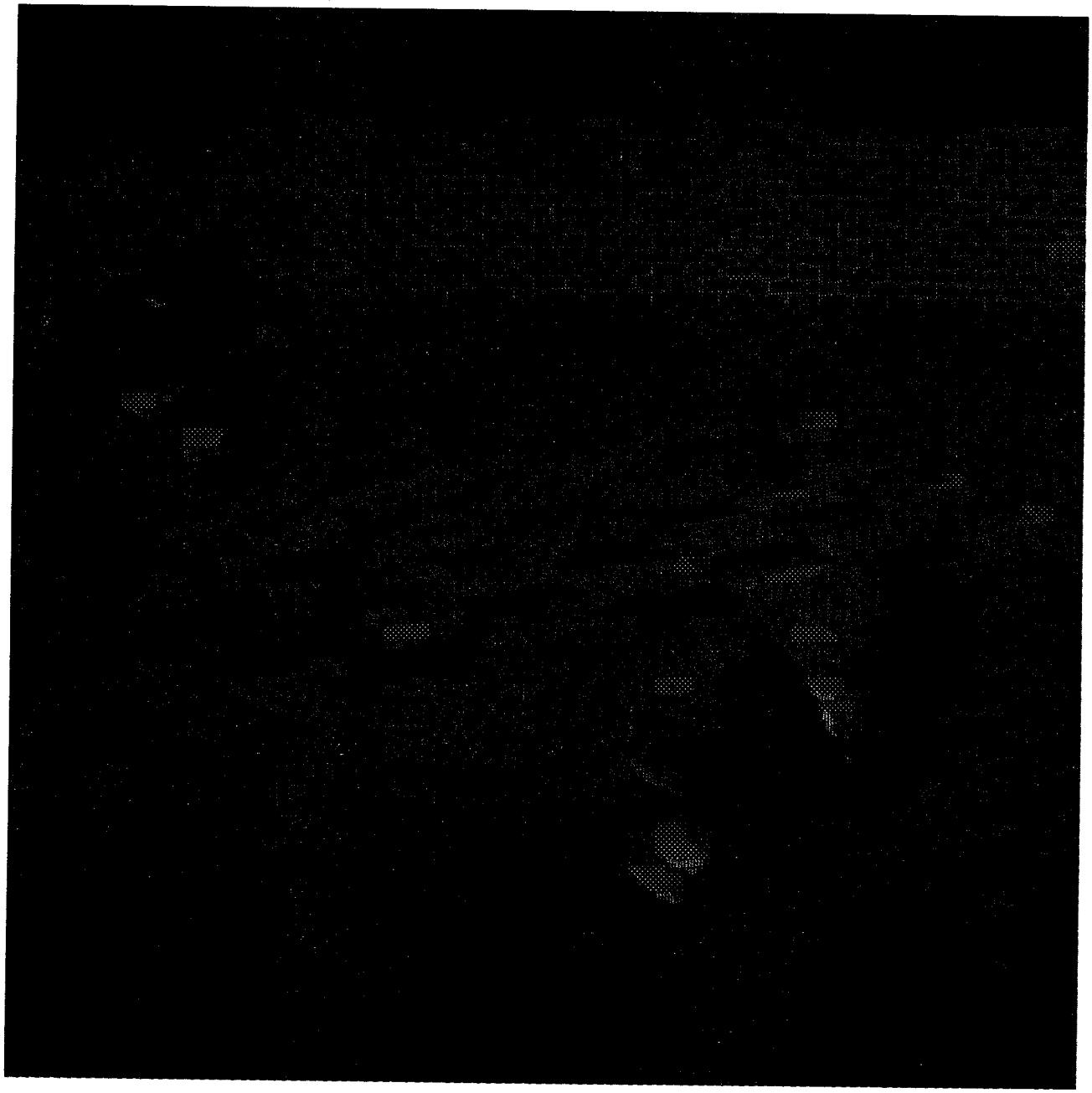


Figure IV.14: les éléments gauches sur HELIOS

5. Conclusion

Cette présentation générale du système illustre bien la diversité et la complexité des situations possibles, pour lesquelles il serait très fastidieux de faire la revue exhaustive des processus et opérateurs réalisés, en cours de réalisation ou envisagés. Nous nous bornerons dans les chapitres qui suivent à présenter les techniques de réalisation les plus originales, tant au niveau logiciel qu'au niveau matériel. Nous ferons référence plus spécialement à l'utilisation du terminal HELIOS qui implique des processus peu classiques et soulève des problèmes nouveaux et intéressants par les voies de recherche qu'ils ouvrent. On trouvera au chapitre VII des exemples permettant d'apprécier la facilité d'utilisation et les résultats obtenus.



SOIREE D'ECHECS

CHAPITRE V

Aspect matériel

Le synthétiseur cablé HELIOS

1. Présentation générale

Ce chapitre est consacré à la présentation d'un prototype de synthétiseur cablé, dont la conception découle directement de l'approche choisie, justifiée précédemment. Ce terminal orienté vers la synthèse d'images réalistes, présente un haut degré d'interactivité, et intègre des fonctions évoluées permettant en particulier:

- la modification en temps réel de l'aspect des objets (texture, brillance),
- la modification en temps réel des attributs d'éclairage (position, couleur, et intensité de la source lumineuse, lumière ambiante, etc),
- la modification en temps réel de quelques attributs géométriques (fenêtre),
- l'identification instantanée d'un objet désigné sur l'écran à l'aide d'un réticule intégré,
- le calcul en temps réel, pour chaque face, de la projection des textures et de la réflexion diffuse et spéculaire,
- le compression des données échangées avec le calculateur pilote,
- la prise en compte d'une partie de l'élimination des parties cachées.

Il s'agit bien entendu d'un choix délibéré visant à privilégier les applications de C.A.O. et plus particulièrement celles où l'aspect esthétique intervient directement dans le processus de conception. Ces applications sont relativement nombreuses, et nous ne citerons ici que quelques cas typiques:

- construction d'objets divers (meubles, bijoux, ustensiles ménagers) en particulier pour le choix des revêtements (bois, tissus d'ameublement, métaux, peinture, etc.),
- décoration, pour juger de l'harmonie des formes et des couleurs,
- architecture, pour juger de l'aspect final des constructions,
- urbanisme, pour apprécier "l'intégration" d'un ouvrage dans un paysage,
- etc.

La conception de ce terminal est le résultat d'une longue évolution de l'Equipe de Communication Graphique, initialement concernée par l'aspect logiciel, vers les techniques du matériel. Ayant eu l'opportunité de réunir des compétences dans ces deux domaines, nous avons choisi de mener parallèlement notre progression au niveau logiciel et la conception d'un terminal puissant nous permettant de tester la validité de nos concepts et également de nous doter d'un outil de travail indispensable.

La réalisation du premier prototype a fait l'objet de la thèse de docteur-ingénieur de Fernando Nunes Ferreira (Fer81), et a bénéficié de l'aide efficace du Service d'Electronique du Laboratoire IMAG. Ces travaux ont abouti, au cours de l'année 1981, à la mise en service d'un premier prototype qui, bien que réduit, comporte toutes les fonctions initialement prévues. Ce prototype est à l'heure actuelle entièrement opérationnel et continue d'être étendu, tant au niveau matériel qu'au niveau logiciel, comme nous le verrons ci-après. Nous débuterons cette étude par une présentation générale du processus de visualisation et de l'architecture qui sont à la base d'HELIOS, puis nous évoquerons les solutions techniques originales qui ont dû être utilisées pour mener à bien cette réalisation. Nous terminerons cette présentation par la revue des extensions cablées et micro-programmées en cours ou prévues.

1.1. Le processus de visualisation

Le terminal HELIOS est un synthétiseur mono-processus destiné à former la couche terminale de processus de visualisation complets comportant une partie logicielle et une partie matérielle. Nous examinerons successivement:

* l'élément de base à synthétiser,
* les opérations privilégiées,
* les structures de données.

1.1.1. La notion de face plane dans HELIOS

L'élément unique manipulé par le terminal est la face plane. cette notion de face diffère cependant sensiblement de la notion habituelle, et nous nous appuierons sur la définition suivante.

Nous nommerons face, tout ensemble homogène de points c'est-à-dire possédant la même identité, le même aspect et les mêmes attributs géométriques (même repère).

Les attributs d'aspect acceptés sont au nombre de trois:

- la texture qui représente la distribution intrinsèque des couleurs du matériau dont la face est constituée, indépendamment de la façon dont il est vu ou éclairé,

- la réflexion qui exprime la faculté du matériau à réfléchir la lumière qu'il reçoit,

- la visibilité ou l'invisibilité, la transparence ou l'opacité.

Les attributs géométriques expriment pour leur part le repère tri-dimensionnel (U,V,N) attaché à la face, exprimé dans le repère (x,y,z) associé à l'écran.

U,V représentent les vecteurs de base du plan de la face, N représente le vecteur normal à ce plan.

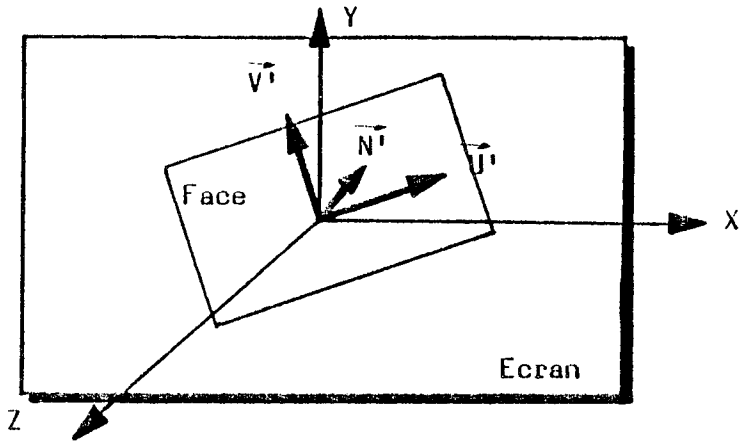


Figure V.1: Repère associé à une face

Le nombre maximum de faces ayant été fixé à 1024 pour ce premier prototype, l'identité d'une face est représentée par son numéro, compris entre 0 et 1023. C'est ce numéro d'identification qui sera récupéré après une désignation à l'aide du réticule.

Il est à noter que si le nombre de 1024 faces semble un peu limitatif, il est en fait possible de présenter des scènes comportant un nombre de faces beaucoup plus important. Ainsi des faces situées dans des plans parallèles et constituées d'un même matériau, sont représentées par une seule "face HELIOS". La figure 2, représente un objet composé de trois faces seulement. Bien entendu, ces faces communes ne peuvent plus être identifiées séparément.

1.1.2. L'ordonnancement du processus

Outre l'identification par désignation nous avons choisi de favoriser les modifications ayant trait aux attributs visuels, c'est-à-dire l'aspect et l'éclairage. Comme nous l'avons vu en II.2.1, le choix du processus de visualisation joue ici un rôle déterminant, et assure:

* la pénétration des attributs concernés (A) et (E),
* la mémorisation des autres attributs synthétisés M,G,Gv et I,
* les opérations de synthèse des attributs de base,
* les opérations de composition inter-éléments.

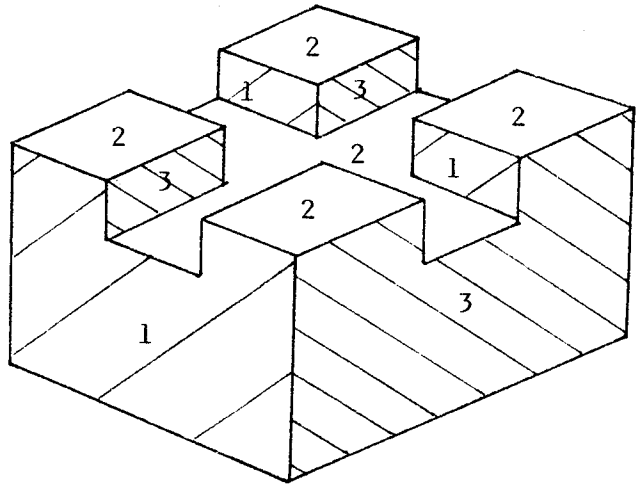


Figure V.2: Objet représenté à l'aide de trois faces

Le processus de base sur lequel HELIOS a été conçu est représenté sur la figure 3. On remarque le grand nombre d'opérateurs de synthèse intégrés au matériel et la forte pénétration des attributs (A) et (E). Nous détaillerons ci-après les différents opérateurs :

- 1) Synthèse des attributs (G) situant les faces les unes par rapport aux autres (description), et des attributs (Gv) exprimant les paramètres de prise de vue. Cet opérateur consiste essentiellement à effectuer les produits des matrices de transformations, en vue d'obtenir le repère 3D associé à chaque face, exprimé dans celui de la vue (mémoire de trame).

- 2) Les points exprimant le ou les contours de la face sont transformés et projetés dans le repère de la vue par le résultat de (G.Gv); le découpage est effectué si besoin est.

- 3) Les points intérieurs à la face projetée sont remplis dans la mémoire de trame avec le numéro d'identification de la face. Cette mémoire de trame constitue un plan nommé plan d'identification.

Nous entrons ici dans la catégorie des opérateurs cablés, intégrés au terminal, qui sont exécutés pour chaque pixel au rythme du balayage de

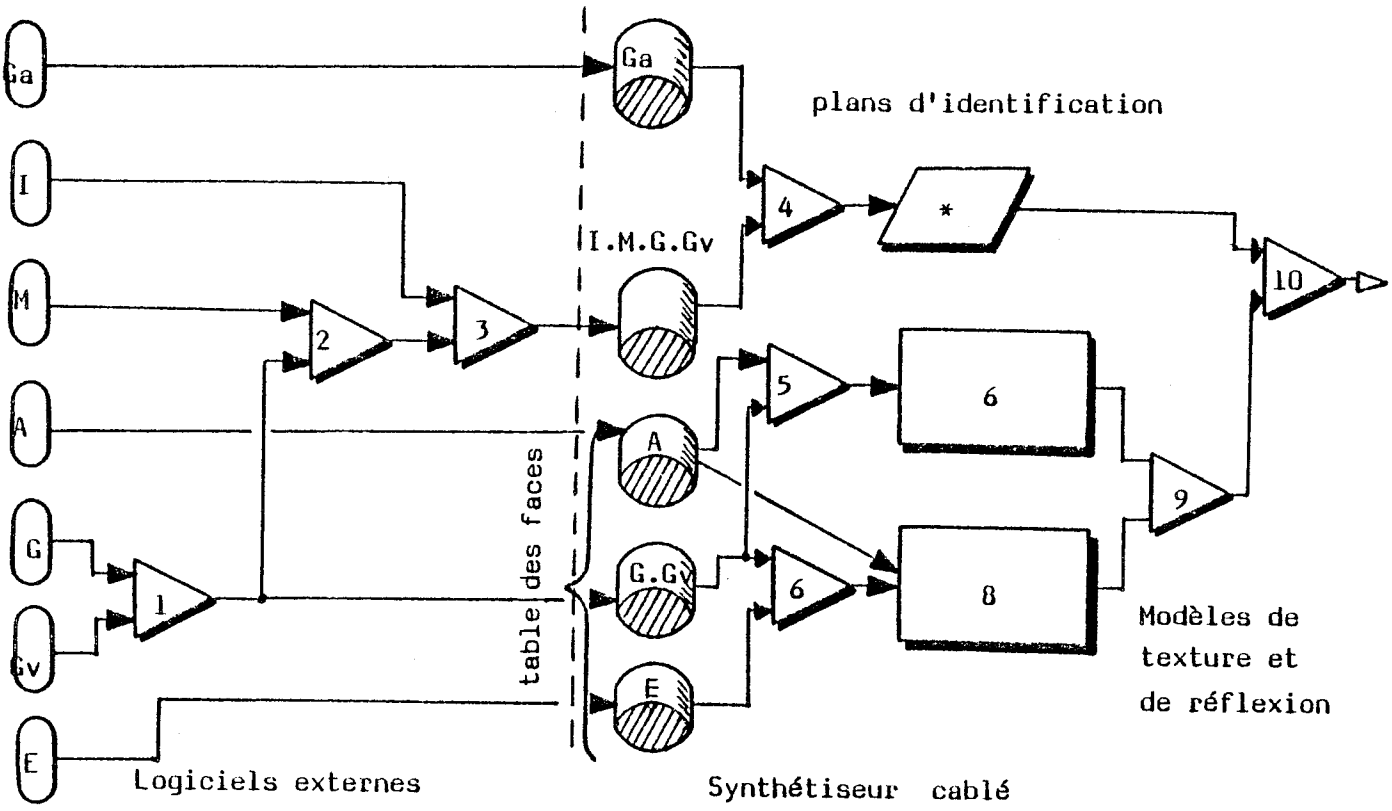


Figure V.3: Processus de base d'HELIOS

trame.

- 4) Détermination de la portion de la vue visible sur l'écran. Cette fonction de "fenêtrage" permet également d'exprimer des translations et des zooms. Le résultat de cet opérateur est le numéro d'identification de la face visible en chaque point de l'image finale.

- 5) Projection de la texture affectée à chaque face sur le plan de celle-ci, et "pavage" de l'intérieur de la face.

- 6) Tansformation éventuelle des textures. Cet opérateur permet d'exprimer des modifications de matériaux, valables pour l'ensemble des faces.

- 7) Calcul des angles entre la normale à la face et la direction de la source lumineuse, et entre la normale et la direction médiane entre la source et l'observateur, afin de déterminer les coefficients de réflexion diffuse et spéculaire.

- 8) Transposition éventuelle des phénomènes de réflexion. Cet opérateur permet de modifier globalement les conditions d'éclairage pour toutes les faces (effets de brume, contrastes, etc.).

- 9) Synthèse (A.E). Multiplication de la couleur intrinsèque déduite de la texture par les coefficients de réflexion issus des opérateurs 7 et 8.

- 10) Synthèse finale: affichage du point courant fourni par l'opérateur 4 avec la couleur issue de 9.

1.2. Parallélisme

1.2.1. Parallélisme inter-processus

HELIOS propose un certain degré de parallélisme inter-processus en acceptant jusqu'à 8 plans d'identification superposés.

L'opération de composition (notée '*') détermine la face visible au point concerné en fonction de la priorité des plans, de la visibilité et de la transparence des faces.

Les opérateurs '4' sont présents dans chaque plan et s'exécutent en parallèle, ce qui permet de définir une fenêtre différente pour chaque plan, comme on peut le voir sur la figure 4.

Il va sans dire que cette organisation nécessite au niveau du logiciel un algorithme d'élimination de parties cachées susceptible de répartir convenablement les faces initiales dans les 8 plans d'identification. Ce problème sera évoqué au chapitre VII.

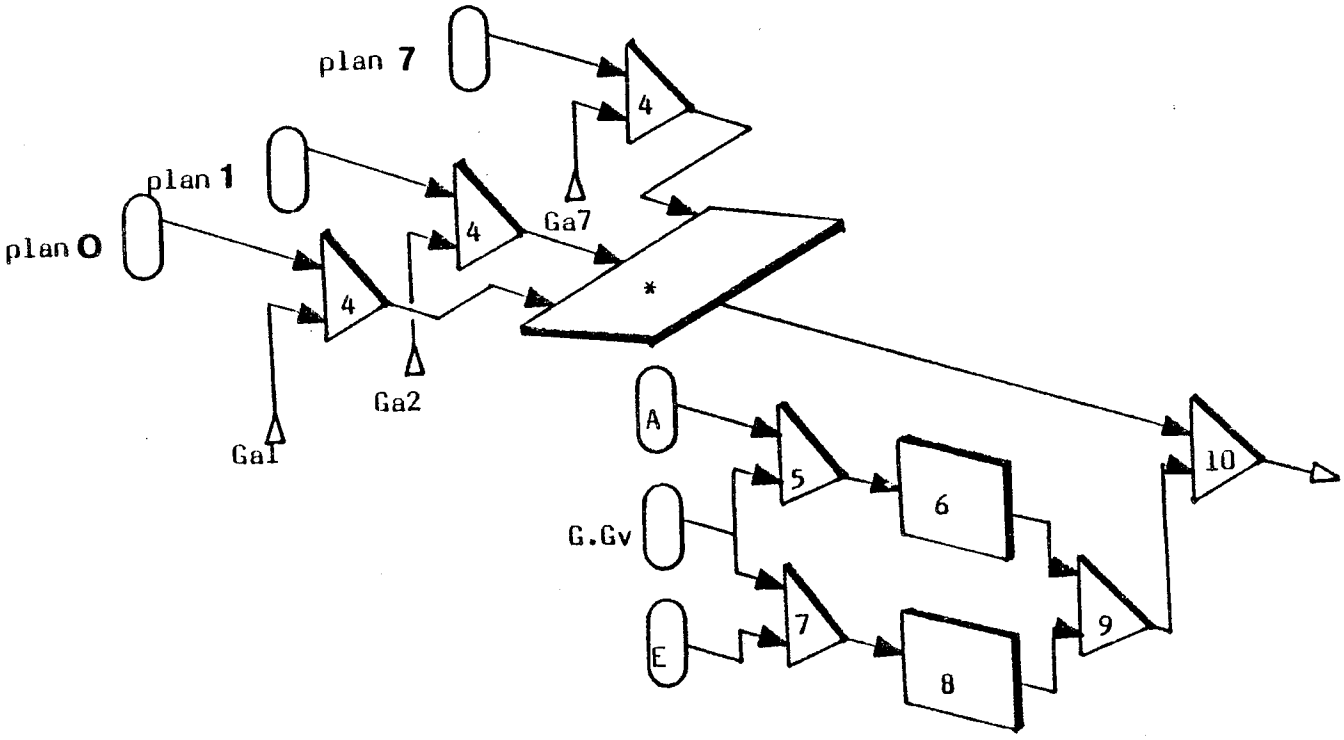


Figure V.4: Parallélisme au niveau des plans d'identification

1.2.2. Parallélisme intra-processus

Le processus de la figure 3 fait nettement apparaître une possibilité de parallélisme entre les opérateurs (5,6) d'une part et (7,8) d'autre part, ce qui revient à considérer séparément la teinte et la luminance en chaque point.

1.3. L'architecture générale

1.3.1. Organisation hiérarchique

Nous avons conçu le terminal HELIOS en utilisant directement les concepts dégagés aux chapitres I et II de cette étude. Nous avons par conséquent organisé ce synthétiseur de manière hiérarchique autour des trois unités de contrôle, de communication, et de visualisation description.

L'unité de contrôle

Un processeur cablé est chargé des échanges avec le calculateur pilote, et de l'ordonnancement des opérateurs. Cette dernière tâche consiste à engendrer et à distribuer les signaux de synchronisation nécessaires aux autres processus et au moniteur T.V.

L'unité de communication

Cette unité gère à l'aide d'un processeur cablé, les processus d'attribution et de consultation des différentes mémoires intégrées. Elle gère également les conflits qui peuvent survenir entre les accès externes, les accès des processus de visualisation et le rafraîchissement des mémoires dynamiques.

L'unité de description-visualisation

Les opérateurs du processus de visualisation ont été regroupés dans quatre processeurs, affectés respectivement:

* au calcul de visibilité,
* au calcul des textures,
* au calcul de la réflexion,
* au calcul de l'éclairage et à l'affichage final.

Un processeur supplémentaire est dévolu à l'unique processus de description intégré, permettant la saisie de coordonnées dans l'espace écran à l'aide d'un réticule digital.

1.3.2. Les structures de données

Les informations traitées par ces processeurs sont enregistrées dans six mémoires distinctes, par le logiciel pilote.

* mémoire de trame ou plans d'identification (attribut M.G.Gv.I),
* table des faces (attributs A, G.Gv),
* banque des textures,

* banque des modèles de réflexion,
* paramètres d'éclairage (attribut E),
* paramètres de fenêtrage (attribut Ga).

La table des faces est la seule structure partagée par les processeurs de calcul. C'est elle qui effectue la correspondance entre les différents attributs associés à une face. Elle comporte pour chacune des 1024 faces simultanément synthétisables:

* un indicateur de visibilité/invisibilité, transparence/opacité, pour chaque plan d'identification,
* le pointeur d'adresse de la texture affectée à la face,
* le numéro du modèle de réflexion,
* le vecteur normal à la face,
* les vecteurs de base du plan de la face.

Ces attributs sont modélisés de façon originale en vue de faciliter le travail des processeurs de synthèse. La recherche de cette modèlisation a été l'une des étapes les plus délicates de la conception de ce terminal, et nous en présenterons les détails dans le paragraphe suivant.

La figure 5 schématise l'architecture générale d'HELIOS et la figure 6 illustre les liens entre les sept processeurs.

2. Les processeurs de visualisation

Nous donnerons ici une description des fonctions exécutées par les quatre processeurs de visualisation, ainsi que des différentes mémoires utilisées. Cette présentation sera faite sans entrer dans les détails techniques qui seront évoqués dans le paragraphe suivant.

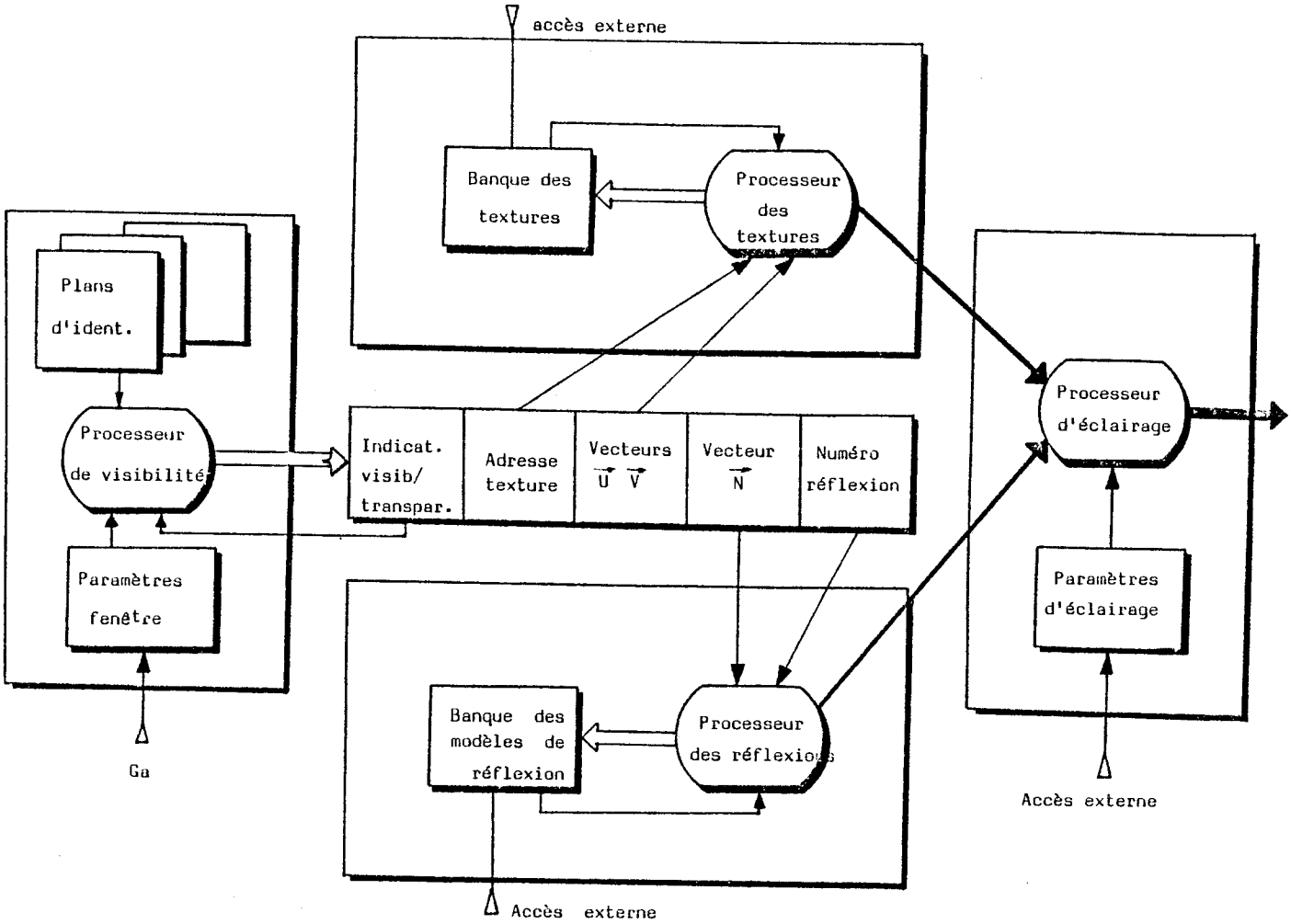


Figure V.5: Architecture générale d'HELIOS

2.1. Le processeur de visibilité

Deux opérateurs élémentaires sont à la charge de ce processeur:

- la fonction fenêtre, (opérateur de synthèse Ga)

- l'étude de visibilité, (opérateur de composition)

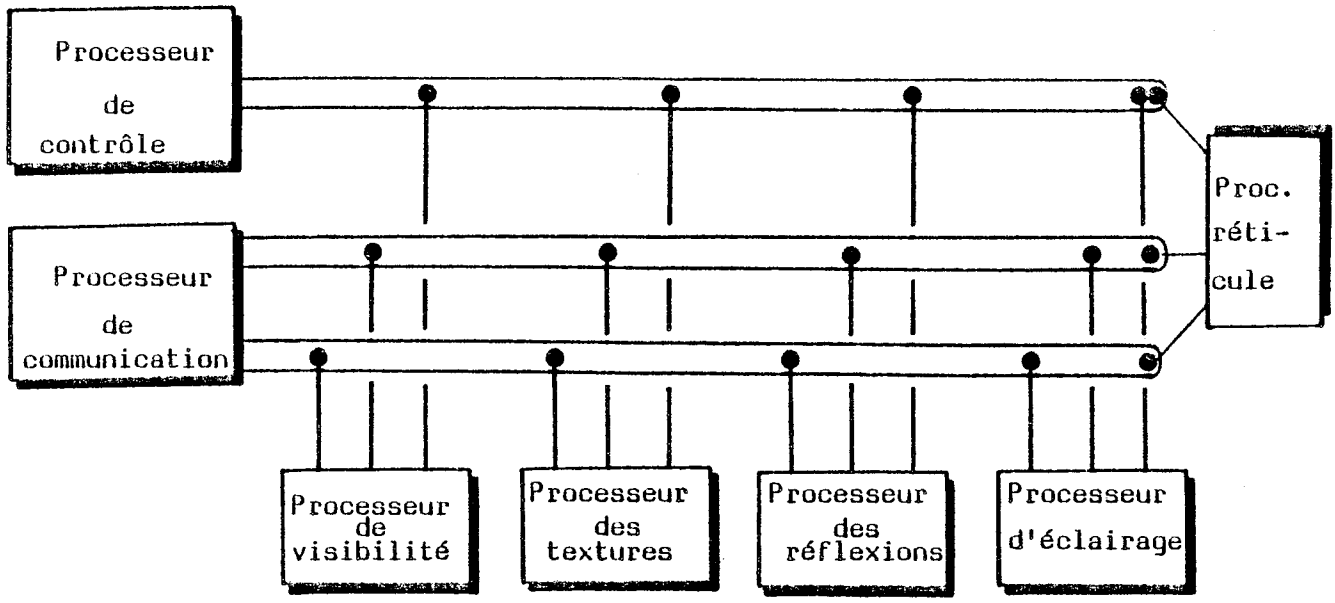


Figure V.6: Les liens entre les divers processeurs cablés

2.1.1. Les plans d'identification

HELIOS permet de gérer 8 plans d'identification dont le dernier, nommé plan de fond possède quelques particularités.

Chaque plan est une mémoire de trame de 512*512 points de 10 bits destinés à contenir le numéro d'identification (0 à 1023) des faces à visualiser (cf. figure 7). Ces plans mémorisent une vue, et non l'image finale.

Le plan de fond quant à lui ne mémorise qu'un seul numéro d'identification, qui affecte tous les points de la vue. Le grand avantage du plan de fond réside dans la possibilité d'associer une texture à toute la vue par l'intermédiaire d'un seul registre. Il sert également à garantir qu'il y aura toujours une face visible en chaque point de l'écran (voir l'étude de visibilité).

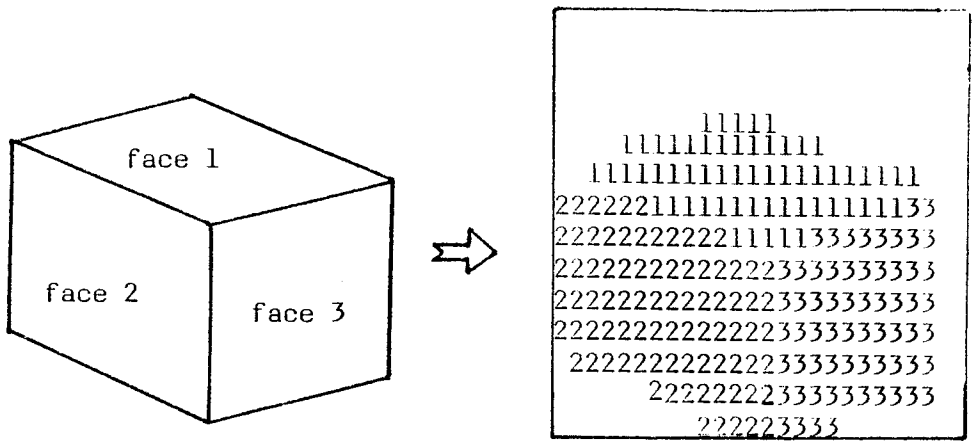


Figure V.7: Mémorisation de l'identité des faces

2.1.2. La fonction fenêtre

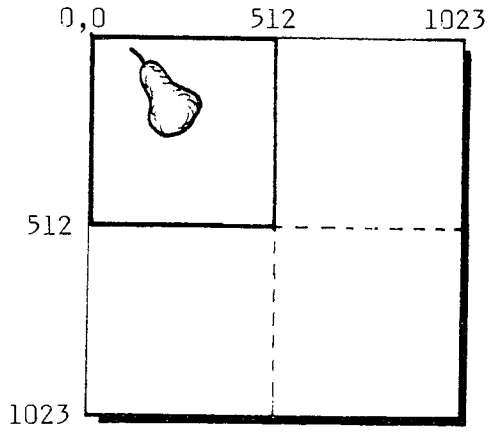
Cette fonction (opérateur 4 du processus) existe pour chacun des plans d'identification, y compris le plan de fond. Elle permet de décrire la portion carrée de la vue qui doit être affichée sur l'écran, en indiquant:

* son coin supérieur gauche,
* sa taille,
* le mode de découpage.

L'espace d'adressage de chaque vue peut s'étendre de 0 à 1023. Cependant selon les besoins, l'espace utile des plans qui est normalement de 512*512 peut être réduit à 256*256 voire même moins. L'une des premières raisons d'être de cette fonction est donc de permettre le fonctionnement en configuration réduite. Quatre tailles sont disponibles sur le prototype (figure 8).

- 512*512, 256*256, 128*128, 64*64.

En déplaçant le coin de la fenêtre on peut faire défiler celle-ci sur le plan (en temps réel) afin de "centrer" une zone intéressante. Un zoom peut alors être effectué en modifiant la taille de la fenêtre, comme le montre la figure 9.



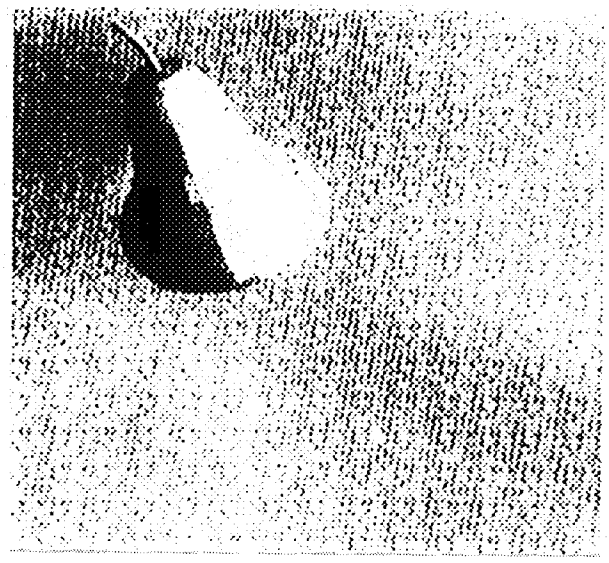
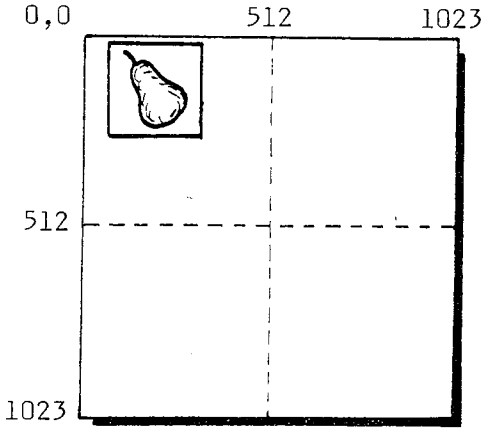


Figure V.8:

La fonction fenêtre



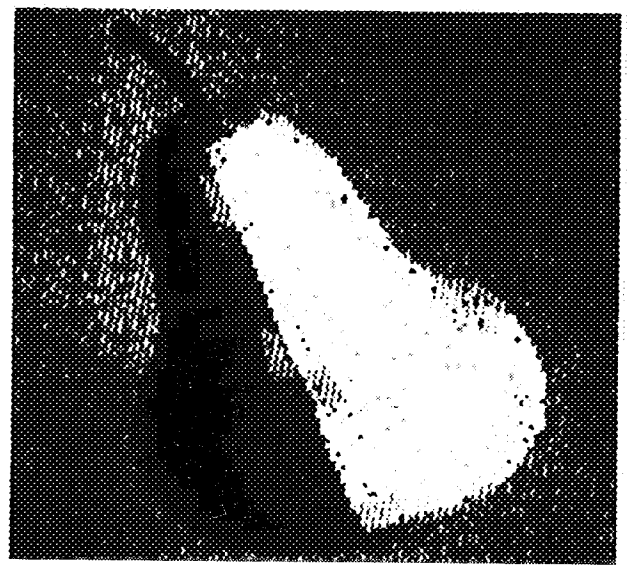


Figure V.9:

Translation de la fenêtre

Le mode de découpage permet d'exprimer la visibilité ou l'invisibilité de la vue en dehors de la zone utile. dans le cas de la visibilité, la zone utile se répète sur l'espace adressable de la vue (figure 10).

En combinant ces possibilités il est possible de translater les plans les uns par rapport aux autres, de les superposer ou encore de les juxtaposer. Nous verrons au chapitre VII que cette possibilité permet d'envisager une certaine animation en temps réel.

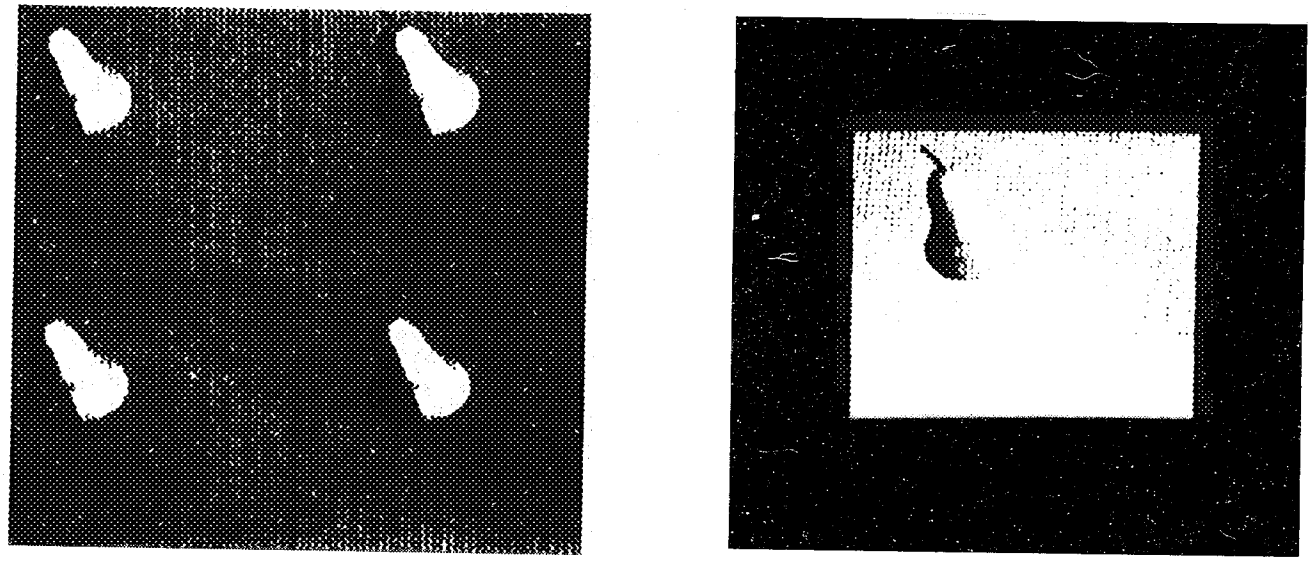


Figure V.10: Illustration des modes de découpage

2.1.3. L'étude de visibilité

Cette opération de composition est le point de convergence des processus parallèles précédents. Elle a pour rôle la détermination de la face visible en chaque point de l'image. Cette étude tient compte:

* de la priorité des plans d'identification,
* du mode de découpage de la fenêtre,
* de la visibilité et de la transparence des faces.

La priorité des plans est fixée par construction (figure 11). Le plan 0 est celui qui possède la plus forte priorité, le plan de fond la plus faible. La table des faces peut contenir jusqu'à 7 indicateurs de visibilité exprimant si la face est visible ou invisible sur chacun des plans. L'unique face contenue dans le plan de fond est, quant à elle, toujours visible.

Le calcul de visibilité s'effectue à l'aide d'un processus pipe-line transversal permettant de déterminer quelle est la première face visible, dans l'ordre croissant. On notera qu'une face peut être rendue visible ou invisible instantanément en changeant simplement le bit de son indicateur de visibilité. De la même manière, un plan tout entier peut être rendu visible ou invisible grâce à la fonction fenêtre, en effectuant une translation en

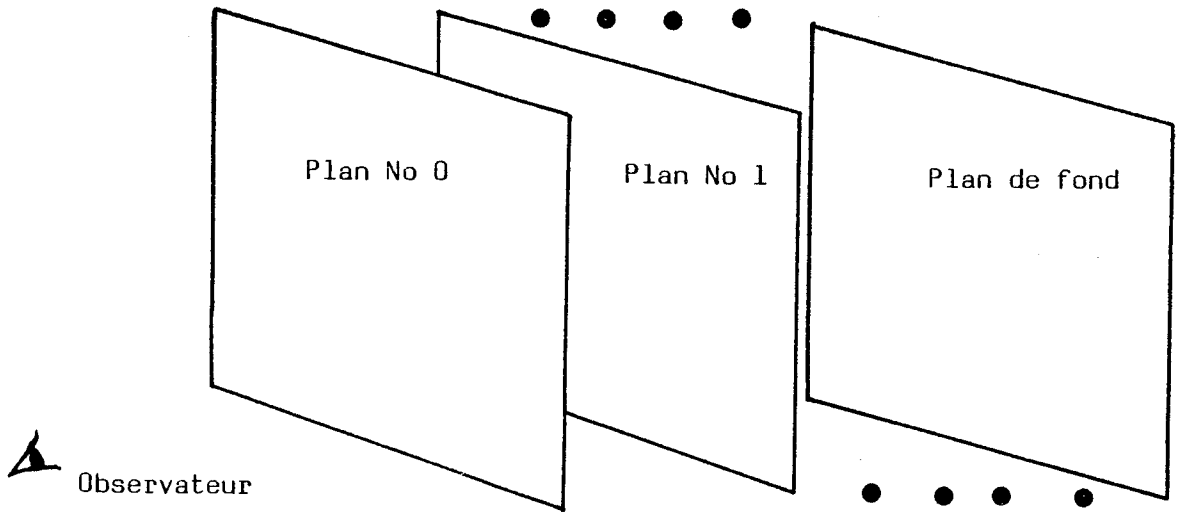


Figure V.11: Priorités des plans d'identification

dehors de la zone utile.

2.1.4. La transparence

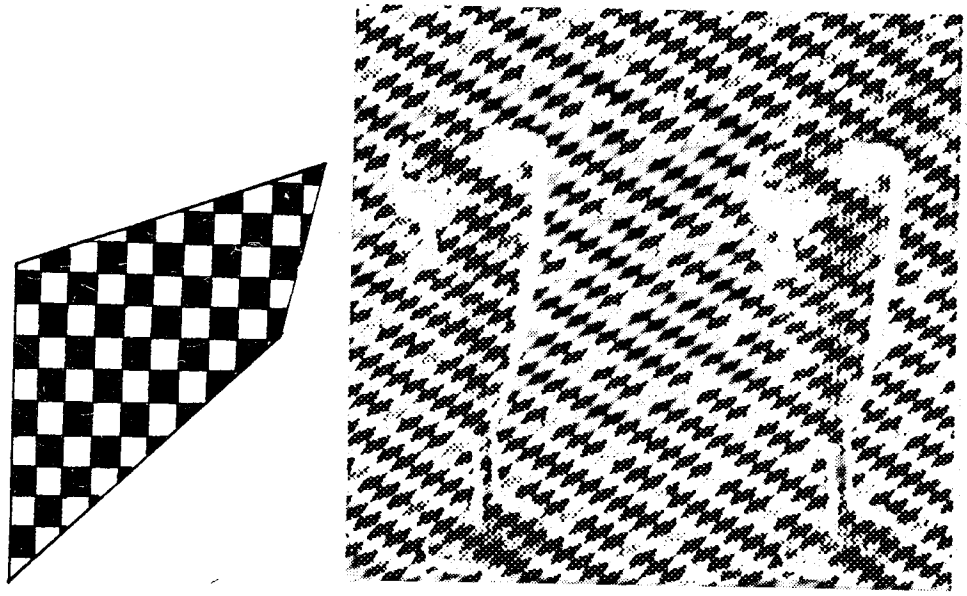
Bien que cette organisation se prête mal à la prise en compte de la transparence au niveau du matériel, nous avons introduit récemment un artifice permettant d'obtenir des résultats acceptables.

Seul le plan 0 accepte des faces transparentes pour lesquelles un indicateur supplémentaire est prévu. Lorsqu'une face est visible et transparente sur ce plan, seul un point sur deux est effectivement rendu visible, alors que l'autre (invisible) laisse apparaître les plans supérieurs. Afin d'éviter l'apparition de bandes verticales sur l'écran, l'ordre des points visibles et invisibles est inversé à chaque ligne (cf. figure 12).

2.2. Le processeur des textures

Deux opérateurs élémentaires sont à la charge de ce processeur:

- La projection des textures sur le plan de la face (opérateur 5 du processus: synthèse A.G)



Face transparente

Figure V.12: Le principe de la transparence des faces

- L'obtention de la texture dans la banque intégrée au matériel (opérateur 6 du processus: construction)

2.2.1. La notion de texture

La notion de texture intégrée dans HELIOS, comporte trois originalités importantes:

- Chaque point exprime la couleur intrinsèque de la face indépendamment des conditions d'éclairage (4096 couleurs pour chaque point).

- Chaque texture est supposée définie dans le plan de la face et non dans le plan de l'écran. La projection sur l'écran est effectuée pour chaque point affiché, en fonction du repère associé à la face.

- Des textures de tailles différentes (variant entre 16*16 points et 512*512 points) peuvent être utilisées simultanément. La mémoire disponible sur le prototype actuel permet de mémoriser 64k points de texture à raison de 12 bits par point. Les couleurs sont modélisées en trichromie rouge, vert, bleu de la manière suivante:

11 10 9 3	7 6 5 4	3 2 1 0
VERT	ROUGE	BLEU

Ces trois propriètés permettent de considérer chaque texture comme la simulation visuelle de l'aspect d'un matériau donné (bois, briques, tissu, etc.) indépendamment du repère et des conditions d'éclairage. Une seule texture est donc nécessaire pour toutes les faces constituées d'un même matériau. Il faut noter que cette projection, qui tient compte également de l'éloignement des faces par rapport au point de vue, ne peut être effectuée par logiciel si le pavage est laissé au soin du matériel, comme l'illustre l'exemple de la figure 13.

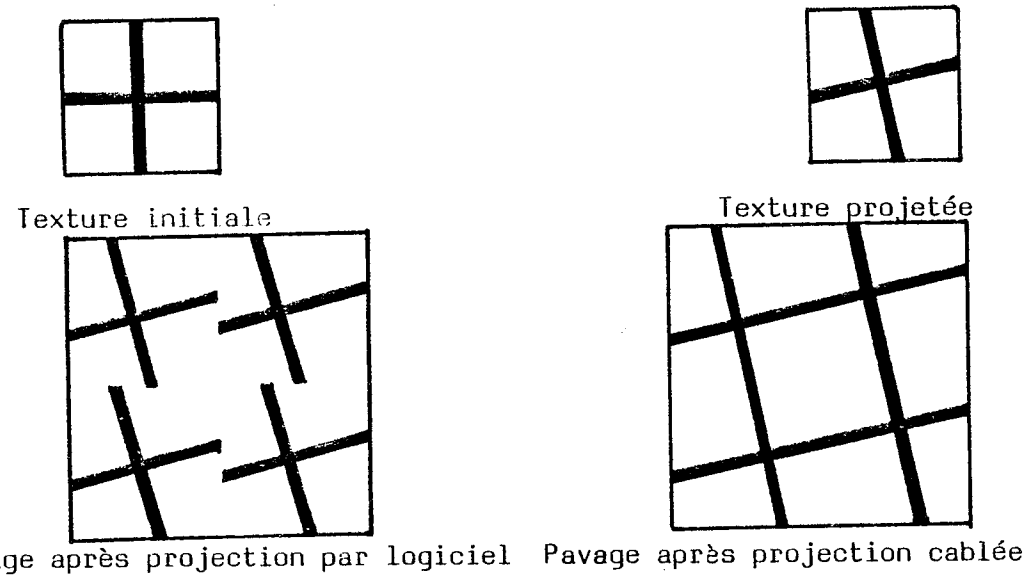
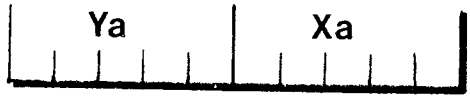


Figure V.13: Le problème de la projection des textures

2.2.2. La banque des textures

Le terminal HELIOS comporte une banque permettant de mémoriser des textures de tailles différentes. Afin de réduire l'espace nécessaire à la mémorisation dans la table des faces, nous avons choisi d'organiser la banque des textures de telle façon que la taille soit implicitement donnée par l'adresse de la texture utilisée.

Cette adresse (Xa,Ya) est celle du coin supérieur gauche de la texture, exprimée sur 10 bits (5 pour l'abscisse, 5 pour l'ordonnée).



La taille est déduite de l'adresse selon l'algorithme ci-après:

```
si (y4,y0)=0 et (x4,x0)=0 alors taille 512*512
sinon
  si (y3,y0)=0 et (x3,x0)=0 alors taille 256*256
  sinon
    si (y2,y0)=0 et (x2,x0)=0 alors taille 128*128
    sinon
      si (y1,y0)=0 et (x1,x0)=0 alors taille 64*64
      sinon
        si y0=0 et x0=0 alors taille 32*32
        sinon taille 16*16
        finsi
      finsi
    finsi
  finsi
finsi
```

La figure 14 montre un exemple de configuration de la banque des textures

Le nombre maximum de textures de même taille est:

1 texture 512*512
3 textures 256*256
12 textures 128*128
48 textures 64* 64
192 textures 32* 32

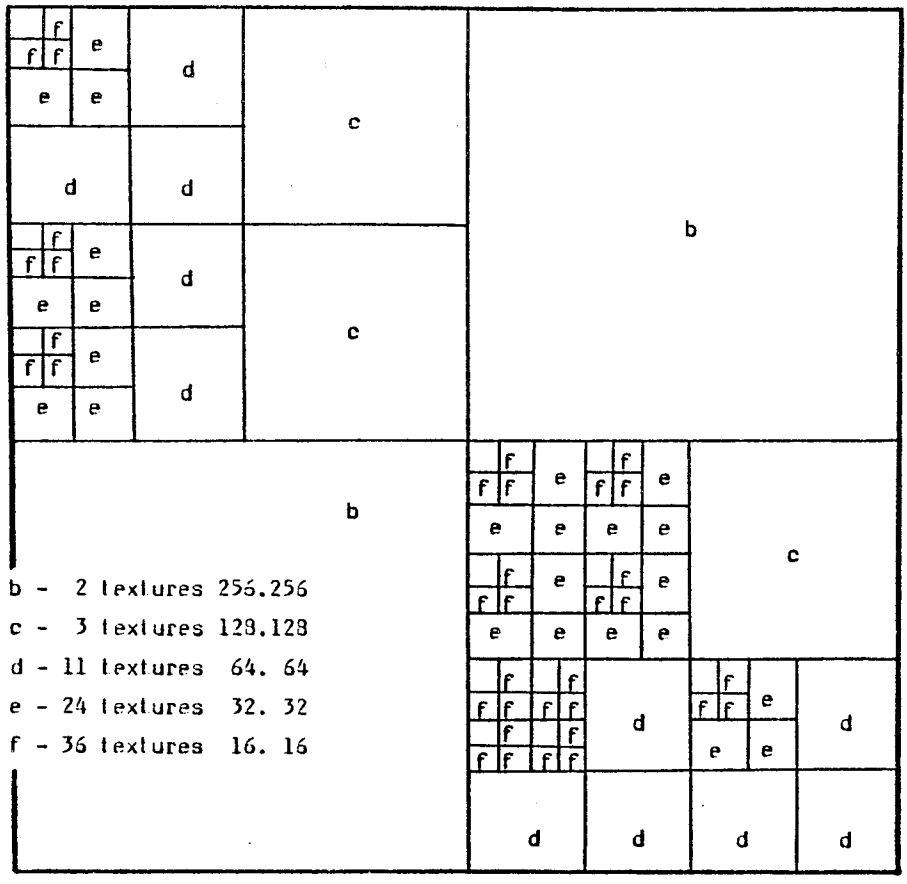


Figure V.14: Exemple de configuration de la banque des textures

Afin de limiter la place perdue, le logiciel pilote effectue une gestion particulière de cette banque de texture (cf VII.2.4).

2.2.3. La projection des textures

Comme nous l'avons vu, la texture est supposée définie dans le plan de la face, et doit être par conséquent projetée sur celui de l'écran. Afin de simplifier le calcul cablé, nous n'avons envisagé que les projections axonométriques, c'est à dire en considérant le point de vue à l'infini. On obtient ainsi, à partir du repère 3D, $(\vec{U}',\vec{V}',\vec{N}')$ associé à la face, la projection des vecteurs de base $(\vec{U},\vec{V})$ du plan de la face, dans le repère de la vue (opérateur 1 du processus) (figure 15).

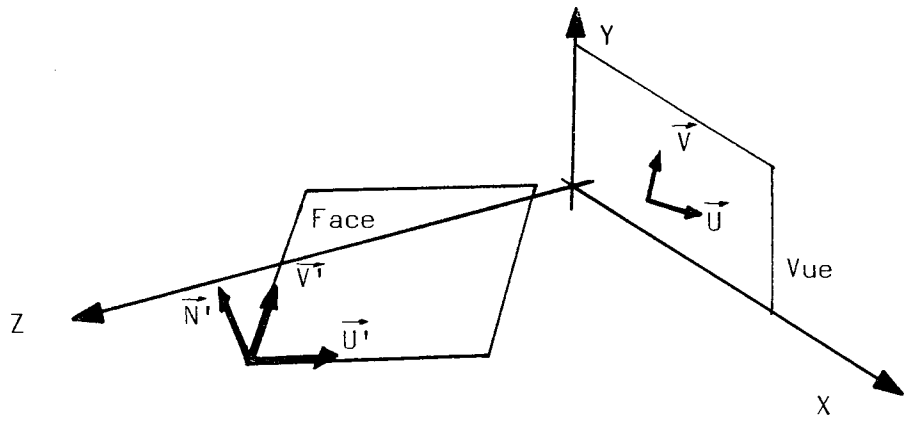


Figure V.15: Projection du repère d'une face

Cependant le problème est ici inverse puisqu'il s'agit de déterminer pour chaque point (Xe,Ye) de la vue, le point (Xf,Yf) correspondant dans le plan de la face, ce qui revient à écrire:

$$\begin{vmatrix} Xf \\ Yf \end{vmatrix} = M^{-1} \times \begin{vmatrix} Xe \\ Ye \end{vmatrix}$$

L'inverse (M^{-1}) de la matrice est précalculé par le logiciel pour chaque face, puis enregistré dans la table des faces.
On obtient:

$$M^{-1} = \frac{1}{Ux*Vy - Uy*Vx} \begin{vmatrix} Vy & -Vx \\ -Uy & Ux \end{vmatrix} = \begin{vmatrix} a & b \\ c & d \end{vmatrix}$$

A noter que les cas où le déterminant s'annule sont ceux où les vecteurs $\vec{U}$ et $\vec{V}$ sont colinéaires, c'est à dire lorsque la face est vue de profil. Le résultat dans ce cas là est sans signification.

La codification des coefficients (a,b,c,d) de la matrice M dans la table des faces soulève un certain nombre de problèmes:

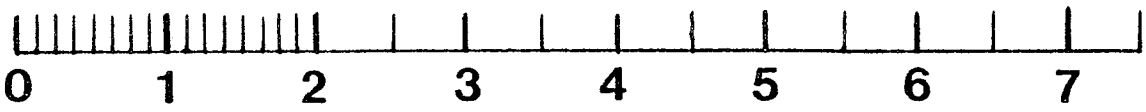
- les coefficients sont à valeur réelle et ne peuvent être approchés par des entiers,

- les valeurs peuvent être positives ou négatives.

Afin de réduire l'encombrement nécessité par une représentation conventionnelle, la valeur absolue de chaque coefficient est représentée sur 5 bits de la façon suivante:

La valeur V est exprimée par V = m * 2 exp (2e-3)

Cette représentation permet de coder 28 valeurs absolues comprises entre 0 et 7,5.



D'autre part, seuls les coefficients b et c sont signés. Cette limitation implique qu'il ne peut y avoir de retournement complet de la texture. Lorsque a ou d sont négatifs, le logiciel pilote effectue une symétrie des vecteurs de base.

On remarque en outre, que les valeurs élevées de a,b,c,d correspondent aux vues de profil pour lesquelles le déterminant tend vers 0. L'expérience a montré que la limitation à 7,5 des valeurs absolues des coefficients est largement suffisante compte tenu des erreurs importantes provoquées par l'échantillonnage de la texture dans ces cas limites.

2.2.4. Le pavage

Après avoir obtenu les coordonnées (Xf, Yf) du point courant dans le plan de la face, l'étape suivante consiste à calculer le point résultant dans la banque des textures, compte tenu de son adresse et de sa taille implicitement déduite.

Nous avons simplifié cette phase du calcul en supposant que le point (0,0) de la banque des textures coincide avec l'origine du repère de la face, et également avec celle du repère de la vue. Les coordonnées (Xb, Yb) dans la banque des textures sont alors calculées à l'aide de l'expression ci-dessous:

$$\begin{aligned} Xb &= Xa * taille + Xf \ \underline{modulo}\ taille \\ Yb &= Ya * taille + Yf \ \underline{modulo}\ taille \end{aligned}$$

où (Xa,Ya) représentent le pointeur de texture (adresse du coin supérieur gauche).

A noter que si Xa = Ya = 0 (taille 512) on a

$$\begin{aligned} Xb &= Xf \\ Yb &= Yf \end{aligned}$$

La texture unique est alors assimilée à une mémoire d'image dont chaque pixel contient la couleur du point considéré. Si la texture 0 est affectée à toutes les faces et si les projections sont neutres, le terminal HELIOS se comporte alors comme un terminal classique et peut présenter des images réelles saisies à l'aide de caméras ou encore des images entièrement calculées par logiciel.

2.3. Le processeur de réflexion

Ce processeur a pour rôle la détermination des coefficients de réflexion diffuse et spéculaire en chaque point de l'écran. Les données disponibles pour ce calcul sont:

* le vecteur normal à la face visible, (attribut G)
* les paramètres régissant les conditions d'éclairage (attribut E)
* les modèles de réflexion.

Les opérateurs élémentaires exécutés par ce processeur sont:

* Le calcul des angles (synthèse (E.G): opérateur 7)
* La modélisation de la réflexion (construction (E): opérateur 8)

2.3.1. Le calcul des angles

Le calcul par un processus cablé, de l'angle existant entre deux directions de l'espace, a sans doute été le point le plus délicat de notre étude. N'ayant trouvé dans la littérature aucune publication sur ce sujet, nous avons proposé une méthode permettant de résoudre ce problème sans utiliser de tables importantes, comme c'est le cas dans la technique proposée par D. Bass (Bas81).

L'originalité de cette approche réside principalement dans l'utilisation de coordonnées sphériques, pour exprimer les directions par rapport au repère de l'écran (figure 16).

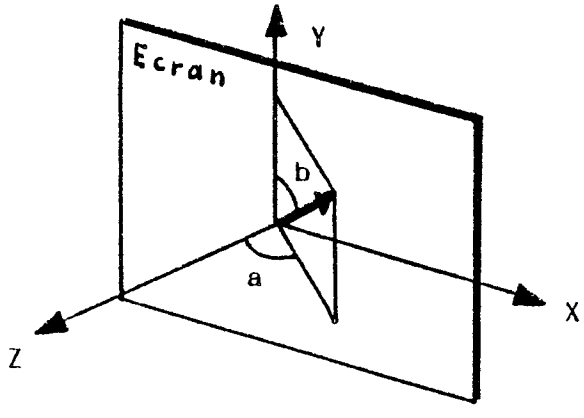


Figure V.16: Représentation de la normale en coordonnées sphériques

Cinq bits sont réservés pour représenter chacun des deux angles (a et b), ce qui autorise 1024 directions dans le demi espace avant de l'écran. A noter que a varie entre -pi/2 et pi/2 alors que b varie entre 0 et pi.

La codification binaire de ces deux angles limite en fait a à (-pi/2, 31/64 pi) et b à (0, 31/32 pi)

Soit à calculer l'angle d compris entre deux vecteurs unitaires $\vec{j}$ et $\vec{k}$ exprimés en coordonnées sphériques $\vec{j}$ = (aj, bj), $\vec{k}$ = (ak, bk).

Les coordonnées cartésiennes des deux vecteurs sont:

xj = sin(aj) sin(bj)	xk = sin(ak) sin(bk)
yj = cos(bj)	yk = cos(bk)
zj = cos(aj) sin(bj)	zk = cos(ak) sin(bk)

ce qui nous permet d'obtenir après calcul:

$$\cos(d) = \sin(bj)*\sin(bk)*(\sin(aj)*\sin(ak) + \cos(aj)*\cos(ak)) + \cos(bj)*\cos(bk) \qquad (1)$$

On peut remarquer que cette équation se prête mal à une réalisation cablée pour trois raisons principales:

* 5 multiplications sont nécessaires,
* 8 tables trigonométriques sont également nécessaires
* tous les termes peuvent être positifs ou négatifs ce qui complique toutes les opérations.

Nous avons utilisé deux artifices pour transformer cette expression:

- 1) Transformations trigonométriques pour faire apparaître la somme et la différence des angles.

  Il vient:

$$\cos(d) = 1/2\ ((\cos(bj-bk) - \cos(bj+bk)) * \cos(aj-ak) + \cos(bj+bk) + \cos(bj-bk)) \qquad (2)$$

Cette expression ne nécessite plus que 3 tables et 1 seule multiplication.

- 2) Transformation pour obtenir des termes positifs
  Soit h une fonction telle que:

$$h(x) = \cos(x) + 1$$

On obtient

$$h(d) = 1/2\ (h(bj-bk) - h(bj+bk)) * h(aj-ak) + h(bj+bk) \qquad (3)$$

Dans cette dernière expression tous les termes, y compris le résultat sont positifs et varient entre 0 et 2, ce qui ne pose plus aucun problème de cablage, comme nous le verrons au cours du paragraphe suivant.

2.3.2. Le calcul de la réflexion

Les méthodes de réflexion que nous avons choisies tiennent compte:

- du niveau de lumière ambiante (16 niveaux),
- de la réflexion diffuse, fonction de l'angle entre la normale et la direction de la source lumineuse,
- de la réflexion spéculaire, fonction de l'angle entre la normale et la direction médiane entre la source lumineuse et l'observateur.

La géométrie de la réflexion est représentée figure 17.

L'observateur est situé sur l'axe oz à l'infini.

- $\vec{N}$ = (an, bn) est le vecteur normal à la face,

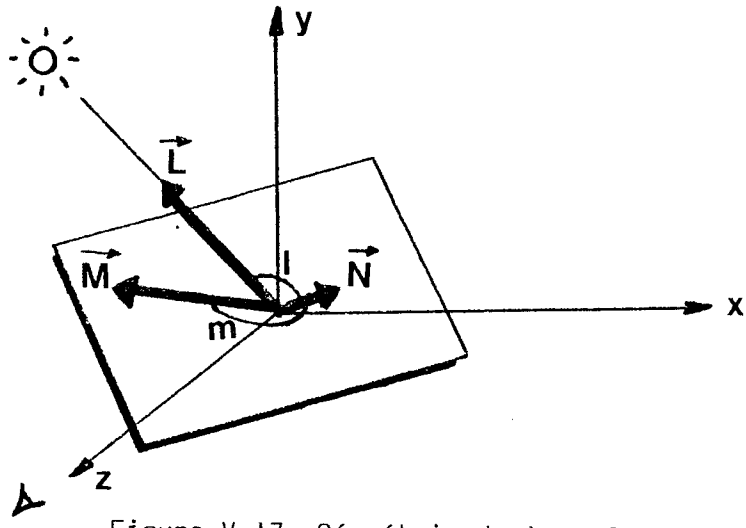


Figure V.17: Géométrie de la réflexion

- $\vec{L}$ = (al, bl) est un vecteur dans la direction de la source lumineuse qui est située à l'infini,

- $\vec{M}$ = (am, bm) est un vecteur dans la direction médiane entre la source et l'observateur (axe oz).

Les coefficients de réflexion diffuse Rd et spéculaire Rs sont obtenus par les expressions

$$Rd = D(l)$$
$$Rs = S(m)$$

où

l est l'angle entre $\vec{N}$ et $\vec{L}$,

m est l'angle entre $\vec{N}$ et $\vec{M}$,

D est un modèle de réflexion diffuse,

S est un modèle de réflexion spéculaire.

A partir de l'expression (3) on obtient:

$$D(l) = Doh^{-1}\,(1/2*h(an-al)*(h(bn-bl)-h(bn+bl))+h(bn+bl))$$
$$S(m) = Soh^{-1}\,(1/2*h(an-am)*(h(bn-bm)-h(bn+bm))+h(bn+bm))$$

- an, bn sont issus de la table des faces (vecteur normal),
- al, bl et am, bm sont calculés par le logiciel pilote et enregistrés dans les registres spécialisés.

2.3.3. La banque des modèles

HELIOS comporte une mémoire pouvant contenir quatre modèles de réflexion. Le choix du modèle est indiqué dans la table des faces pour chacune d'elles. Des modèles standards peuvent être enregistrés lors de l'initialisation du système afin de permettre, par exemple, différents niveaux de "brillance".

Chaque modèle est une table exprimant à la fois $Doh^{-1}(x)$ et $Soh^{-1}(x)$ pour x variant entre 0 et 2. Les valeurs sont exprimées entre 0 et 1 et donnent la valeur du coefficient en fonction de l'angle (figure 18). Ces modèles seront étudiés en détail au chapitre VII.

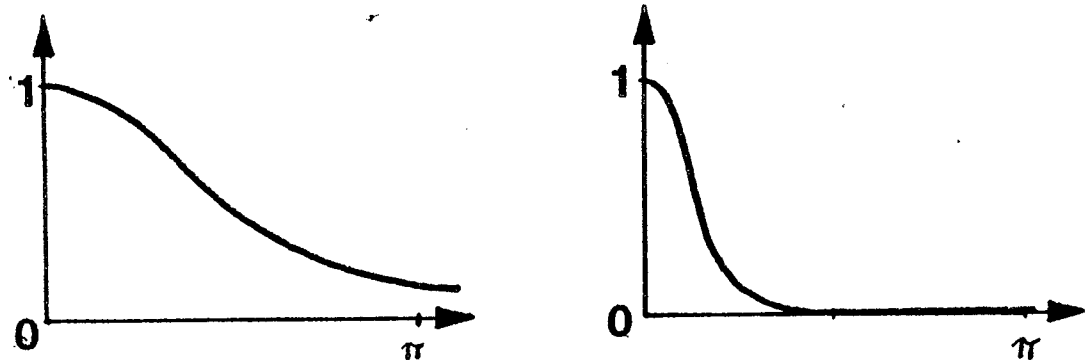


Figure V.18: Modèles de réflexion d'un matériau

2.4. Le processeur d'éclairage et d'affichage

Ce processeur est celui qui est affecté à la synthèse finale entre l'aspect et l'éclairage (opérateurs 9 et 10 du processus). Les données dont il dispose sont:

- la couleur du point courant issu du processeur des textures (4096 couleurs possibles),
- les coefficients de réflexion diffuse et spéculaire, issus du processeur de réflexion,
- la couleur de la source lumineuse mémorisée dans un registre du matériel,
- l'intensité de la lumière ambiante.

2.4.1. Le choix du modèle

De nombreux travaux ont été effectués pour proposer diverses approximations permettant de modéliser plus ou moins exactement les phénomènes d'éclairage. L'un des modéles les plus couramment utilisé pendant ces dernières années est celui proposé par B.T. Phong (Pho75) et repris par Blinn (Bli77), qui s'appuie sur les suppositions suivantes:

* La couleur de la réflexion diffuse est celle du matériau de l'objet,
* La couleur de la réflexion spéculaire est celle de la source lumineuse,
* Ces deux couleurs sont indépendantes de l'angle d'incidence.

Une étude récente de R.L. Cook et T.E. Torrance (CoT81) montre qu'il ne s'agit là que d'un cas particulier, incapable de rendre l'aspect réaliste de matériaux métalliques par exemple. Ils expliquent du même coup l'aspect "plastique" de la plupart des objets synthétisés à l'aide d'un ordinateur, en démontrant que la nature physique des matières plastiques correspond précisément au cas particulier du modèle de Phong.
En conclusion:

- la réflexion spéculaire est normalement de la couleur du matériau, et non de la couleur de la source,

- les réflexions diffuses et spéculaires peuvent engendrer des couleurs différentes si le matériau n'est pas homogène (cas des matières plastiques et bien d'autres),

- la couleur fondamentale du matériau peut-être alterée si l'angle d'incidence est important,

- des matériaux tels que les métaux ont une réflexion diffuse quasiment nulle.

Il est clair que ces améliorations se payent par un accroissement très important des calculs nécessaires, incompatible avec une réalisation cablée. Nous avons donc opté pour une méthode intermédiaire s'appuyant sur le modèle de Phong pour le cas général, mais permettant d'utiliser la couleur de la source pour simuler avec plus de réalisme certains objets particuliers.

2.4.2. La synthèse finale

Le modèle d'éclairage utilisé pour la synthèse finale est donnée, pour chaque point, par l'expression:

$$C = ((A + Rd) * T + Rs * B) * S \qquad (1)$$

où

C représente la couleur finale obtenue,
A représente l'intensité de la lumière ambiante,
T la couleur en ce point issue de la texture,
B le blanc pur,
S la couleur de la source lumineuse.

Dans le cas général la composante spéculaire est donc de la couleur de la source, et non pas systématiquement blanche.

Si l'on exprime l'équation (1) en fonction des composantes rouge,

verte et bleue, on obtient:

$$\begin{aligned} Cr &= ((A+Rd) * Tr + Rs) * Sr \\ Cv &= ((A+Rd) * Tv + Rs) * Sv \\ Cb &= ((A+Rd) * Tb + Rs) * Sb \end{aligned} \qquad (2)$$

Cette transformation n'est, bien entendu, qu'une approximation des phénomènes réels, mais les résultats obtenus, nous semblent suffisants pour la plupart des applications envisagées.

On peut noter en outre, que si tous les paramètres des équations (2) varient entre 0 et 1, les coefficients Cr, Cv, Cb prennent leurs valeurs entre 0 et 2. Nous avons choisi de limiter ces coefficients à 1 en introduisant une "dé-saturation" des couleurs vers le blanc, phénomène fréquent pour bon nombre de matériaux sur-éclairés.

$$\begin{aligned} Cr &= Max\ (Max(1,A+Rd) * Tr + Rs, 1) * Sr \\ Cv &= Max\ (Max(1,A+Rd) * Tv + Rs, 1) * Sv \\ Cb &= Max\ (Max(1,A+Rd) * Tb + Rs, 1) * Sb \end{aligned} \qquad (3)$$

Les photographies présentées dans cette thèse illustrent les résultats obtenus à l'aide de ce modèle.

3. Réalisation du prototype

Nous évoquerons ici quelques-uns des points intéressants abordés lors de la réalisation effective du prototype. Le lecteur désireux d'approfondir les détails techniques de la construction, trouvera dans (Fer81) tous les renseignements et schémas utiles. Notre propos se limitera, dans les paragraphes qui suivent, à exhiber les insuffisances de la technologie actuelle, et les artifices employés pour pallier cette carence.

3.1. Considérations générales

La réalisation du prototype a été décidée à partir d'une étude de faisabilité préliminaire, et d'une simulation par logiciel de l'ensemble des processeurs présentés ci-dessus. La construction effective s'est ensuite déroulée selon un processus évolutif, nous permettant de présenter des images, a chaque étape de la construction, ce qui s'est traduit par de multiples avantages:

- facilité des tests, pour lesquels l'image affichée joue le rôle d'analyseur logique,
- visualisation des résultats des simulations par logiciel qui ont précédé le cablage des processeurs, ce qui nous a permis de rectifier quelques erreurs.

3.1.1. Le standard vidéo

Afin de bénéficier de l'appareillage vidéo du marché, nous avons choisi d'utiliser le standard 625 lignes conventionnel. Ce choix permet d'utiliser les magnétoscopes, caméras, et moniteurs T.V. du commerce. Cependant plusieurs difficultés découlent de cette option.

- En premier lieu, sur les 625 lignes disponibles dans le standard, 512 seulement sont utilisées pour l'affichage. Dans le cas du traitement point par point d'HELIOS la perte de temps est de (625-512) * 0,064 = 7,23 ms par images (40ms).
- D'autre part pour chaque ligne visible, une partie du temps de balayage est nécessaire à la synchronisation et aux retours de lignes, et une autre partie est perdue afin d'obtenir une image carrée sur l'écran ce qui porte à 36 ns le temps utile d'une ligne. La perte sur la partie visible est donc de (0,064 - 0,036) * 512 = 14,33 ms.

Ainsi plus de la moitié du temps disponible de (40ms pour l'affichage d'une image), est perdu (exactement 21,5 ms) (figure 19). Le temps disponible pour le calcul d'un point, se réduit ainsi à 36000/512 $\simeq$ 70 ns,

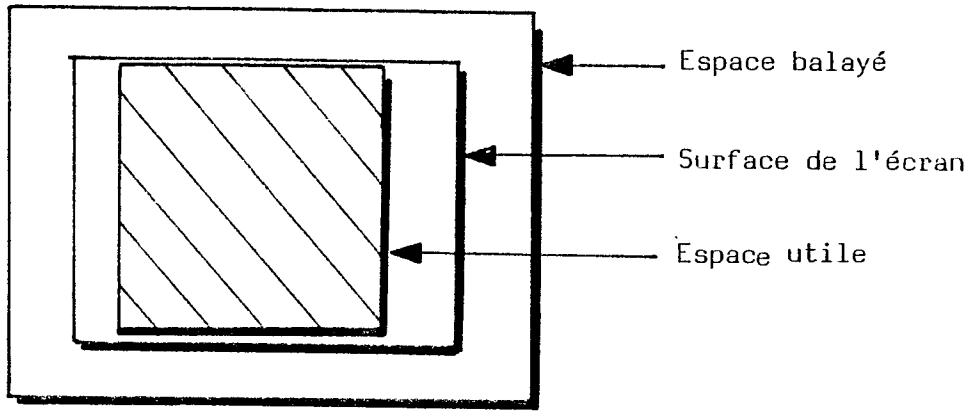


Figure V.19: Espace utile pour la visualisation

ce qui est nettement insuffisant face à la complexité des calculs nécessaires.

Trois artifices sont utilisés pour réaliser le processus de synthèse:

- l'asynchronisme de l'affichage, et des processus de visualisation,
- l'utilisation d'un pipe-line synchrone, pour la réalisation des processus,
- l'utilisation de processeurs parallèles.

3.1.2. L'affichage asynchrone

Afin d'utiliser entièrement les 64 micro-secondes disponibles sur une ligne, l'affichage réel sur le moniteur est désynchronisé du reste du processus. Cet asynchronisme est assuré à travers un double tampon de ligne alternatif. Ainsi les processeurs de visualisation disposent de

$$64000 / 512 = 125 \text{ ns}$$

pour le calcul d'un point et le remplissage d'un tampon, tandis que le second se "vide" vers le moniteur via les convertisseurs numériques/analogiques au rythme de 70ns par point. La figure 20 illustre ce principe.

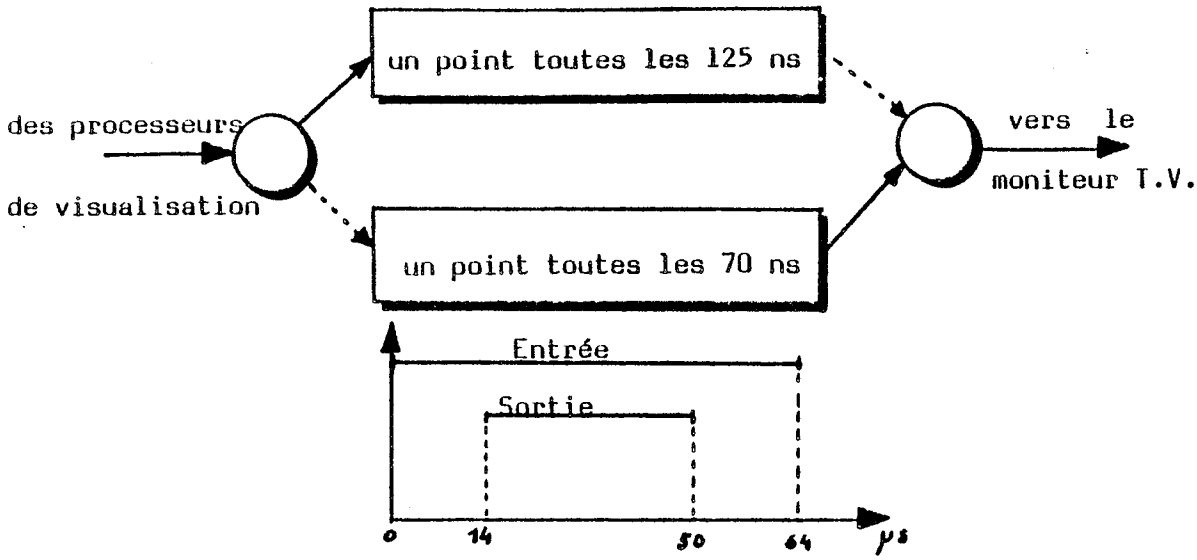


Figure V.20: Double tampon asynchrone

On remarque que le processus de visualisation est en avance d'une ligne sur l'affichage.

3.1.3. L'architecture pipe-line

Bien que le temps utile pour le calcul d'un point ait pratiquement doublé, grâce à l'artifice ci-dessus, nous sommes encore bien loin du temps nécessaire au processus complet. Une solution classique dans ce type de situation consiste à utiliser une architecture pipe-line synchrone dans laquelle le traitement est décomposé en étapes élémentaires nécessitant chacune moins de 125ns.

L'inconvénient majeur de cette technique réside dans l'utilisation obligatoire d'un grand nombre de registres mémorisant les résultats intermédiaires des étapes successives (figure 21).

Un autre problème subsiste encore, pour les étapes élémentaires qui ne peuvent être décomposées et dont le temps est supérieur à 125 ns. Cette situation se présente dans le cas d'HELIOS, pour l'accès aux plans d'identification et à la banque des textures utilisant des boitiers de

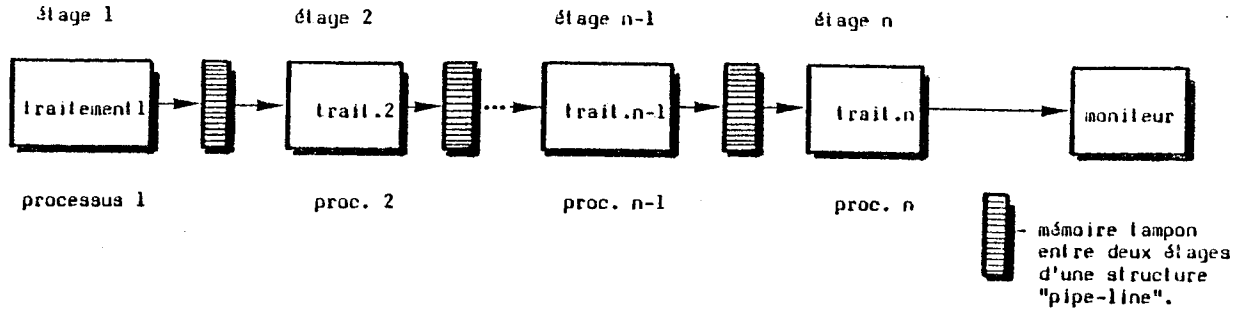


Figure V.21: Pipe-line synchrone

mémoire RAM dynamiques de technologie MOS, et possédant un cycle d'accès assez long (300ns).

L'emploi de ces boitiers, dicté principalement par la taille des mémoires nécessaires et par leur faible coût, complique passablement les étapes d'accès pour lesquelles quatre points sont lus ou écrits parallèlement. Ce parallélisme implique par conséquent un mécanisme de conversion série-parallèle avant l'accès puis une conversion parallèle-série après l'accès (figure 22). Nous verrons qu'un grand nombre de difficultés résultent de ce parallélisme "forcé".

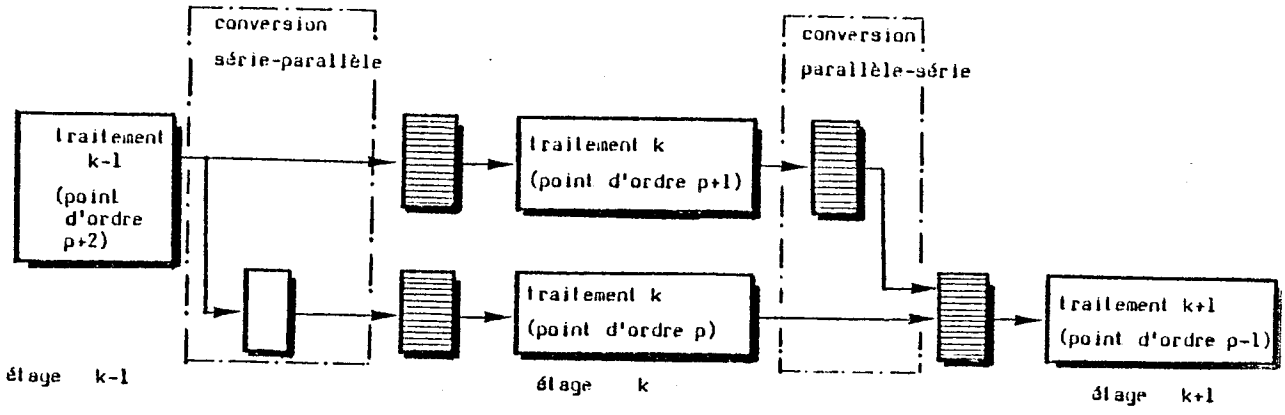
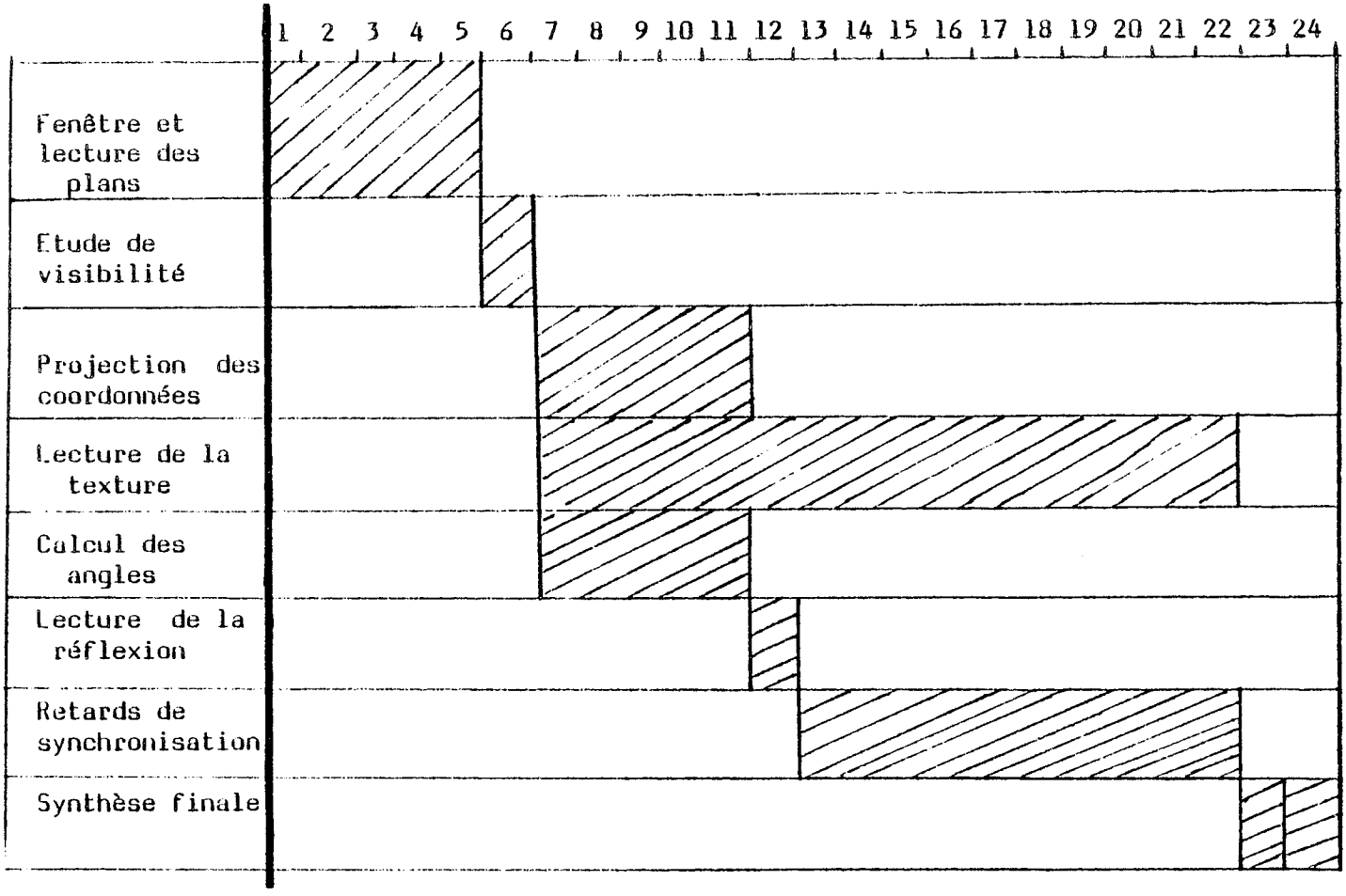


Figure V.22: Traitement parallèle de plusieurs points

3.1.4. Le parallélisme

Outre le parallélisme imposé par les mémoires MOS, le nombre total d'étapes peut être réduit en exploitant le parallélisme intra-processus évident entre le processeur des textures et celui de la réflexion.

Nous donnons ci-après le diagramme des temps du processus complet, faisant apparaître le nombre d'étapes pipe-line nécessaires pour chaque opérateur ainsi que le parallélisme entre aspect et éclairage.



Ce diagramme suscite quelques commentaires:

- Le parallélisme entre les opérateurs (5,6) et (7,8) s'accompagne d'un certain parallélisme entre 5 et 6.

- L'opérateur 6 est celui qui nécessite le plus d'étapes. Ceci est dû à l'accès à la banque des textures pour laquelle doivent avoir lieu des

conversions série-parallèle et parallèle-série.

3.2. Réalisations particulières

Nous examinerons dans ce paragraphe, les principales originalités de la réalisation du prototype HELIOS, en particulier:

* le processeur des textures,
* le processeur des réflexions et de l'éclairage,
* le processeur de gestion du réticule,
* le processeur de contrôle et d'interface avec le calculateur.

3.2.1. Le processeur des textures

Le cablage de ce processeur, dont le schéma est représenté par la figure 23, occupe à l'heure actuelle trois cartes de 80 circuits TTL. Il est clair qu'il s'agit là d'un cas typique pour lequel l'emploi de VLSI "sur mesure" permettrait de réduire considérablement l'espace occupé et la consommation.

La lecture de quatre points en parallèle entraîne une limitation importante au niveau de la projection. En effet, quelle que soit l'adresse du point à lire dans la banque des textures, le premier point de la ligne est toujours lu dans la colonne 1, le second dans la colonne 2, le cinquième à nouveau dans la colonne 1 et ainsi de suite. La figure 24 ci-après montre l'erreur commise dans le cas simple de rayures verticales noires et blanches.

Deux solutions peuvent être envisagées pour réduire cet inconvénient:

- 1) Utiliser des textures possédant toujours quatre points consécutifs identiques (en abscisse seulement).

- 2) Ré-organiser la banque des textures de telle manière que chaque texture apparaisse quatre fois, une fois dans chaque colonne. La capacité se trouve dans cette hypothèse, divisée par quatre.

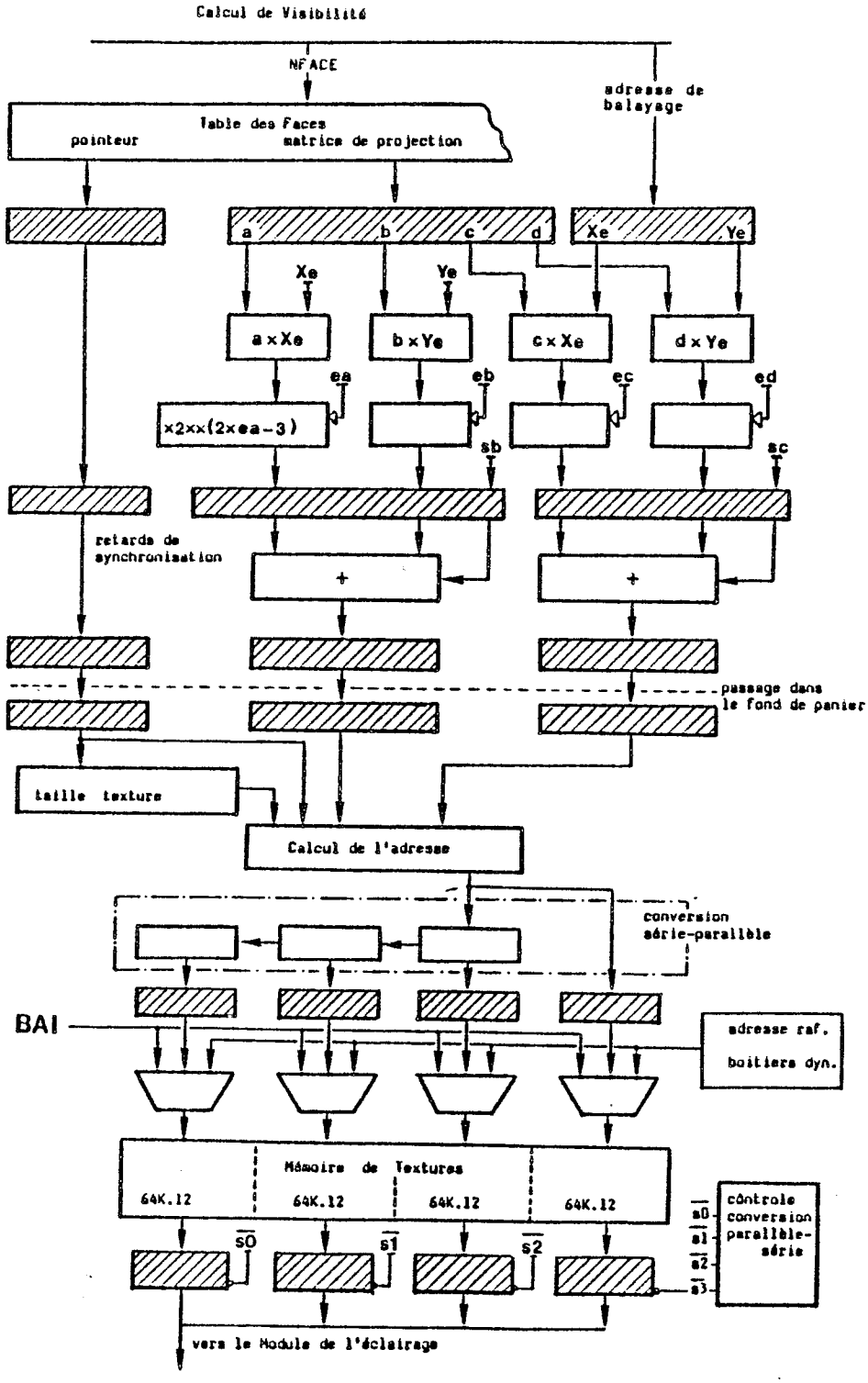


Figure V.23: Le processeur des textures

La solution retenue permet de choisir entre ces deux configurations, en fonction des exigences de l'application concernée.

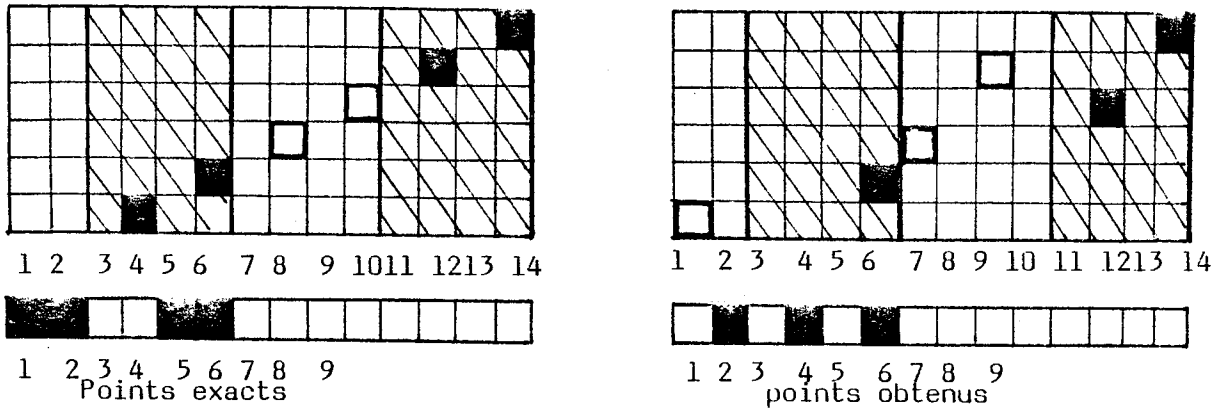


Figure V.24: Erreur commise pour la projection des textures

3.2.2. Les processeurs des réflexions et de l'éclairage

Le synoptique du calcul de l'éclairage est représenté sur la figure 25.

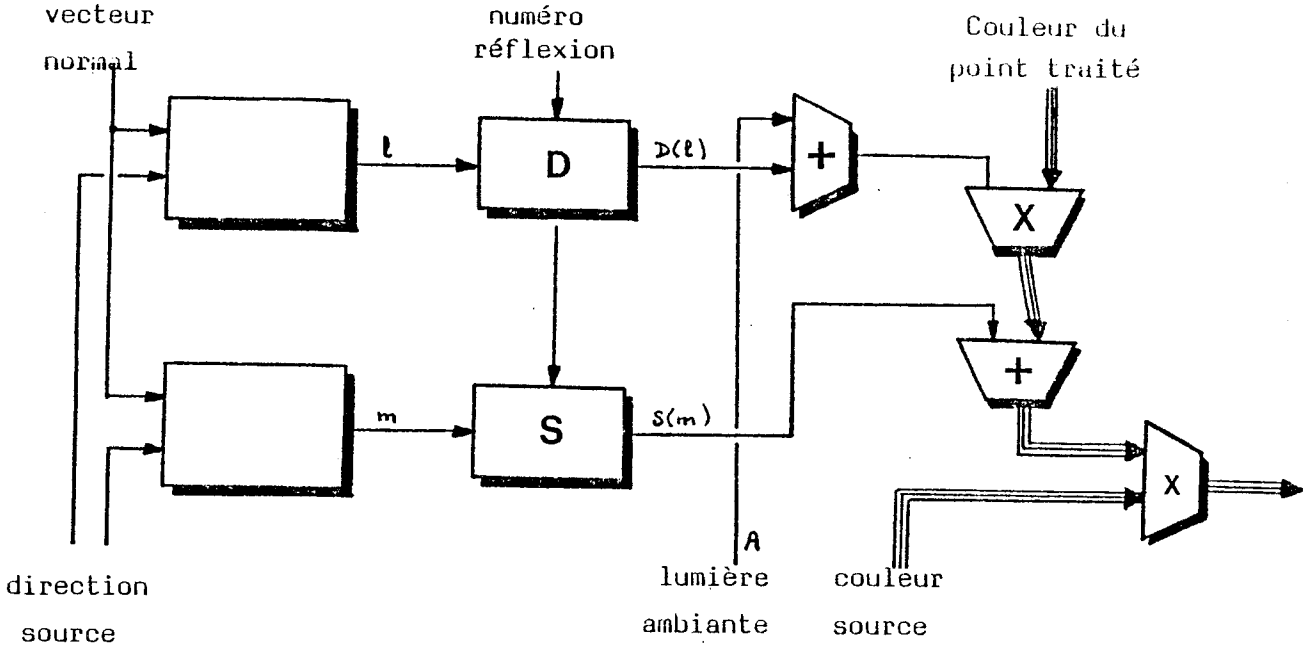


Figure V.25: Synoptique du calcul de l'éclairage

La banque des modèles est constituée de deux RAM de 256*4 bits accéssibles de l'extérieur.

Le calcul des angles $\underline{l}$ et $\underline{m}$ est effectué à partir de l'expression (3) du paragraphe 2.4.2, cablée selon le schéma de la figure 26.

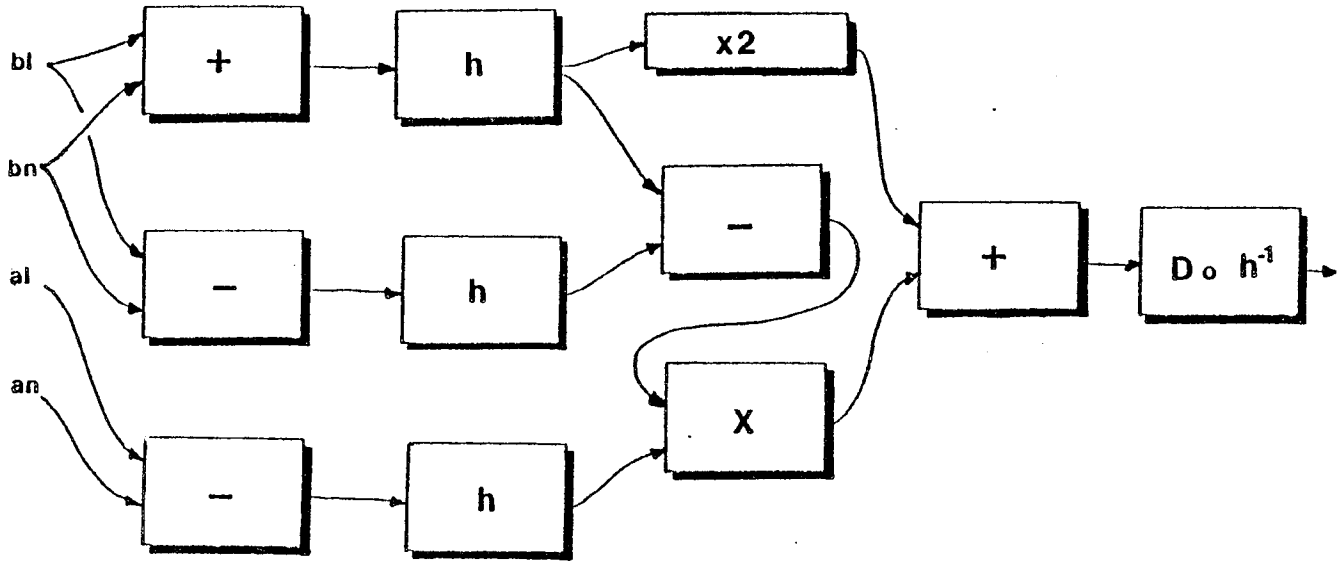


Figure V.26: Schéma de principe du calcul des angles

Les fonctions $\underline{h}$ sont enregistrées dans les PROM bipolaires de taille réduite (32 mots de 8 bits).

La réalisation complète montrant les étapes de pipe-line est représentée sur la figure 27.

3.2.3. Le processeur de gestion du réticule

Afin de permettre la désignation d'une face sur l'écran, ou de collecter des coordonnées, un réticule digital est géré par le terminal HELIOS. Ce réticule peut être, soit commandé à partir d'un ensemble de touches, soit commandé à partir du calculateur pilote pour être asservi, par

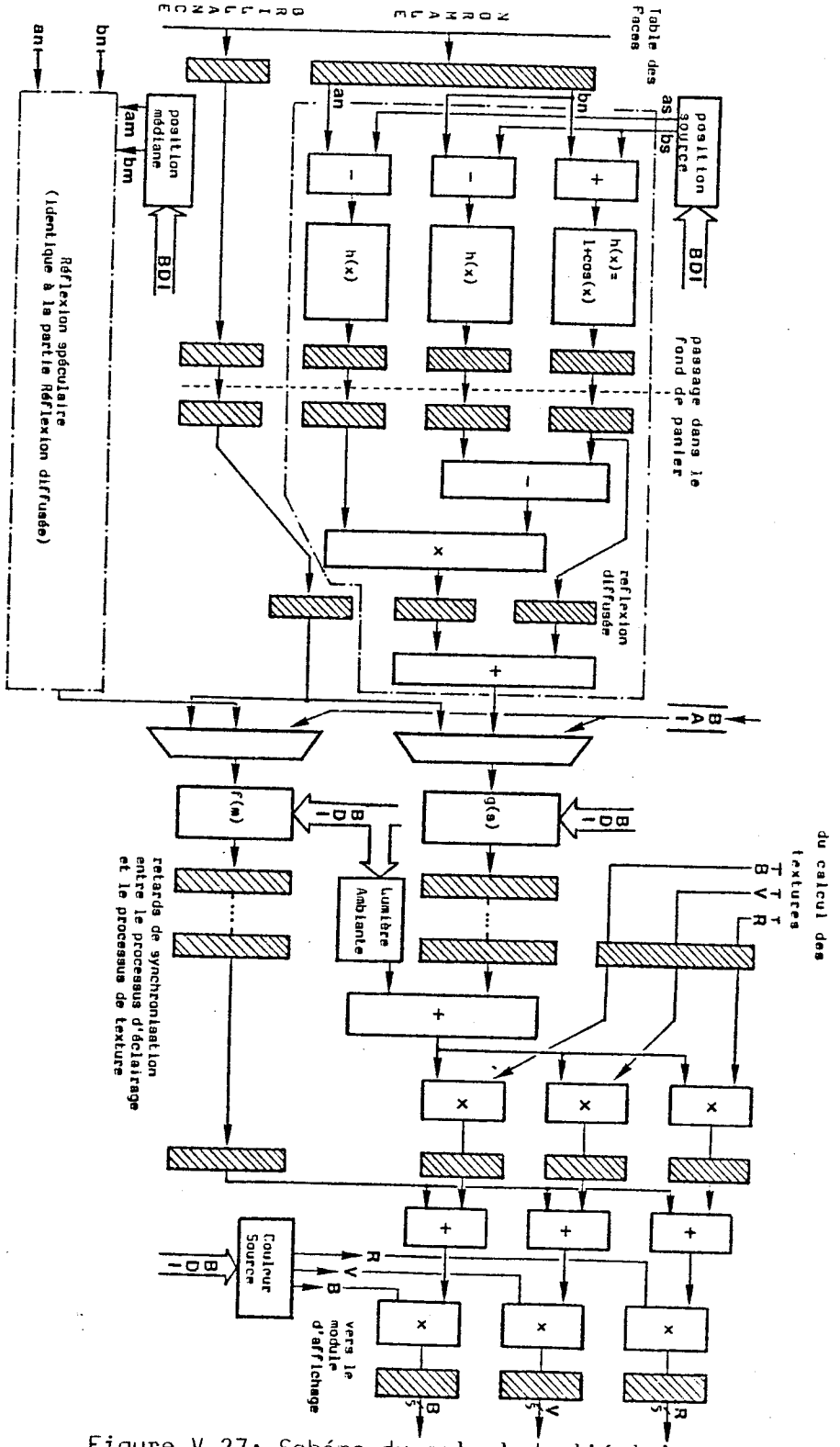


Figure V.27: Schéma du calcul de l'éclairage

exemple aux déplacements d'une tablette à numériser. Les fonctions concernant le réticule sont:

- l'affectation d'une couleur parmi la palette complète de 4096 nuances,
- l'affectation d'une position dans l'espace adressable de l'écran 512*512,
- l'affectation d'un état de visibilité (réticule visible ou invisible),
- la consultation de la position courante,
- la consultation de la visibilité,
- le déplacement du réticule à l'aide de six touches de contrôle qui sont proposées à l'opérateur (figure 28).

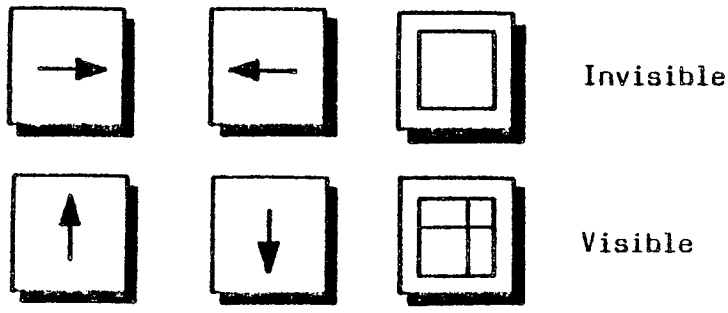


Figure V.28: Les touches de contrôle du réticule

3.2.4. Les processeurs de contrôle et de communication

Les échanges avec le calculateur sont assurés par deux processeurs spécialisés:

* le processeur de contrôle et d'interface,
* le processeur de communication.

Le processeur d'interface assure la compatibilité avec le calculateur pilote. Compte-tenu des fortes exigences d'un tel matériel, nous n'avons envisagé qu'une connexion parallèle. Toutefois, afin de permettre la liaison avec tout type de calculateur, l'interface assure l'adaptation du bus de données d'HELIOS (24 bits) avec celui du calculateur, quelle que soit sa largeur (8, 16, 24, ou 32 bits).

L'accès aux différents registres et mémoires est effectué de manière indirecte à travers un processeur de communication dont les principaux registres permettent d'indiquer la nature et la destination de l'accès. Ces registres, au nombre de sept, sont représentés sur la figure 29.

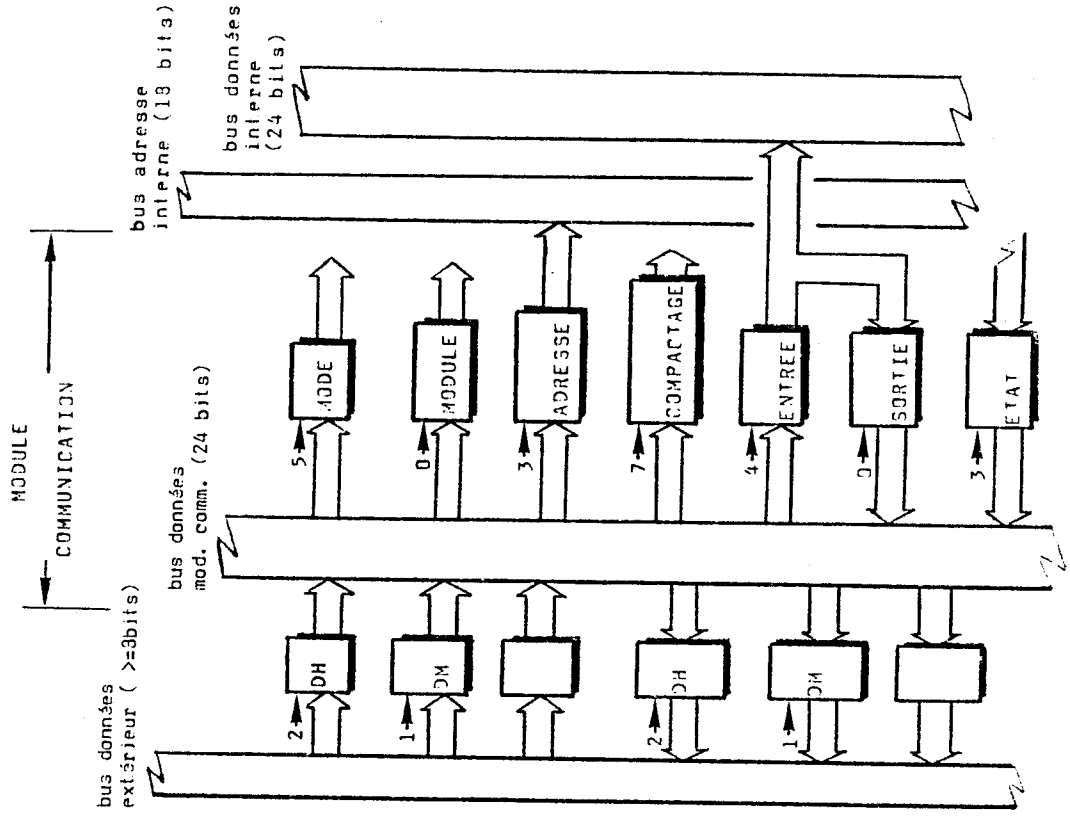


Figure V.29: Les registres de contrôle d'HELIOS

- 1) Registre état:

il permet d'indiquer l'état du terminal, en particulier

* la validité de l'attribut contenu dans le registre réticule,
* la disponibilité du registre d'entrée,
* la disponibilité du registre de sortie.

- 2) Registre module:

Il indique:

* le processeur concerné par l'accès,
* la mémoire ou le registre à accéder,
* la nature de l'accès (lecture ou écriture).

Le tableau ci-dessous récapitule pour chaque processeur, les mémoires concernées.

processeur	Mémoire ou registre	mot	adressage
	plan d'identification	10 bits	0..511,0..511
visibilité	indicateur de visibilité	1 ou 2 bits	0..1023
plans (0 à 6)	coin fenêtre (x,y)	20 bits	
	taille fenêtre + découpage	4 bits	
textures	banque des textures	12 bits	0..511,0..511
	pointeur d'adresse texture	10 bits	0..1023
	matrice de prOjectiOn/face	22 bits	0..1023
réflexion	banque desmodèles	8 bits	0..255
	pointeur numéro de modèle	2bits	0..1023
	vecteur normal/face	10 bits	0..1023
	direction source	22 bits	
éclairage	lumière ambiante	4 bits	
	couleur source	12 bits	
réticule	position	18 bits	
	couleur	12 bits	
	visibilité	1 bit	

- 3) Registre mode: Ce registre permet de configurer le mode d'accès, il définit notamment:
  * l'espace d'adressage des mémoires, par exemple pour les cas ou l'on travaille en 256*256.
  * les instants auxquels les accès extérieurs peuvent prendre place, ce

qui permet par exemple de synchroniser les accès avec le balayage de l'image. Cette facilité permet d'effectuer des changements pendant les "retours de trame" afin d'animer une image de façon continue (cf. chapitre VII).

- 4) Registre répétition: HELIOS offre, pour l'écriture, une possibilité de compresser considérablement les informations. L'opération coûteuse du remplissage des plans d'identification est ainsi grandement accélérée en indiquant simplement le numéro de la face à écrire, et le nombre de points consécutifs. Ce facteur de répétition peut atteindre 512 et permet ainsi de remplir tout le plan d'identification en une seule trame, à condition que le logiciel pilote en soit capable.

Il est à noter que le processus de synthèse utilisé favorise la compression, beaucoup plus que dans le cas d'une mémoire d'image conventionnelle, dans laquelle les points consécutifs sont rarement identiques.

- 5) Registre adresse: Ce registre sert de tampon au bus d'adresse interne du terminal, il est incrémenté automatiquement à chaque accès, ce qui évite un grand nombre d'échanges entre le calculateur et la logique cablée. Cette adresse est sans signification pour les registres marqués 'x' dans le tableau précédent.

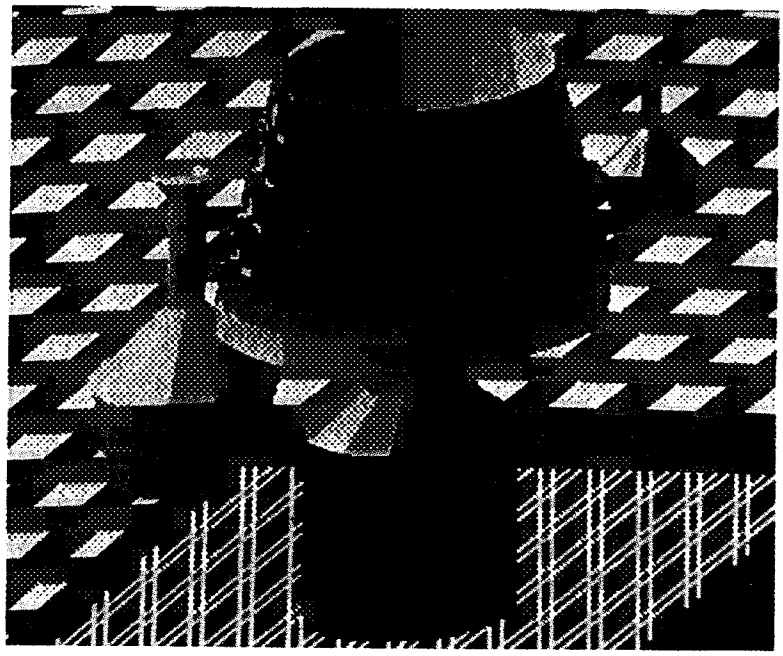
- 6) Registre entrée: Le contenu de ce registre définit l'information à enregistrer, dans le format compatible avec la mémoire concernée. Le registre d'entrée sert de tampon au bus de données interne, il conserve l'information.

- 7) Registre sortie: Ce registre contient l'information lue dans l'une des mémoires du matériel. La lecture est effectué de manière anticipée dès que l'adresse est communiquée au registre-adresse. La lecture du registre-sortie ne nécessite pas d'attente, et provoque la lecture suivante (adresse incrémentée d'une unité).

4. Conclusion

Nous avons présenté, au cours de ce chapitre, les caractéristiques actuelles du prototype de synthétiseur cablé HELIOS. Cette réalisation nous a permis de prouver la validité des concepts de base et des techniques de réalisation proposées, tout en nous dotant d'un outil de travail performant. Les photographies présentées dans cette thèse illustrent le niveau satisfaisant de réalisme que l'on peut obtenir tout en conservant des possibilités d'interaction en temps réel et d'identification directe. Le chapitre VII nous permettra d'insister d'avantage sur l'aspct dynamique des processus et sur les utilisations particulières de ce matèriel.

Il faut cependant noter que, comme tout premier prototype, HELIOS nous a permis de mettre en évidence quelques lacunes relatives aux techniques de réalisation. A l'inverse, quelques fonctions offrent un luxe de possibilités quelquefois discutables. Ces enseignements seront mis à profit puisque, d'ores et déjà, un second prototype est à l'étude et comprendra un micro-processeur chargé des opérations réalisées à l'heure présente par le logiciel pilote ainsi que quelques fonctions actuellement cablées (gestion du réticule et de l'interface avec le calculateur).



LAMPE ET BOUTEILLES

CHAPITRE VI

Aspect logiciel

Les synthétiseurs programmables : CLOVIS

Ce chapitre présente les principaux composants logiciels du système interactif de synthèse d'image présenté au chapitre IV de cette thèse. La réalisation du synthétiseur cablé HELIOS nous a permis d'élargir notablemment l'éventail des éléments concernés, en considérant les images de type "réaliste". La prise en compte d'attributs d'aspect tel que les textures, ou d'attributs d'éclairage, ou encore de la structure des objets remet en cause l'organisation rigide des logiciels à vocation "graphique", pour lesquels les objectifs sont plus restreints.

Ces problèmes ont été étudiés dans le cadre d'un projet de recherche nommé CLOVIS ayant pour objectif principal la conception et la réalisation d'un Complexe Logiciel pour la Visualisation Interactive Structurée. Le terme "complexe" illustre les liens étroits existant entre les logiciels développés, en vue de former un tout cohérent. Nous limiterons ce chapitre aux deux logiciels développés à l'heure actuelle: le logiciel pilote et le logiciel principal, que nous présenterons individuellement sous leur aspect général.

1. Le logiciel pilote

Nous consacrerons le premier paragraphe de ce chapitre à la présentation du logiciel pilote implanté dans le poste de travail HELIOS. Nous insisterons plus particulièrement sur les problèmes d'organisation générale, et la prise en charge des fonctions locales qui confèrent à ce poste de travail toute son originalité. Nous passerons sous silence la réalisation des opérateurs élémentaires de visualisation et de description ainsi que la définition exacte des types d'éléments manipulés. Le lecteur interessé pourra trouver tous ces détails en (Sar82) et (Pay82).

1.1. Organisation générale

L'organisation hiérarchique du logiciel pilote peut être complétée en faisant apparaître les unités de gestion des périphériques connectés au poste de travail (voir figure 1)

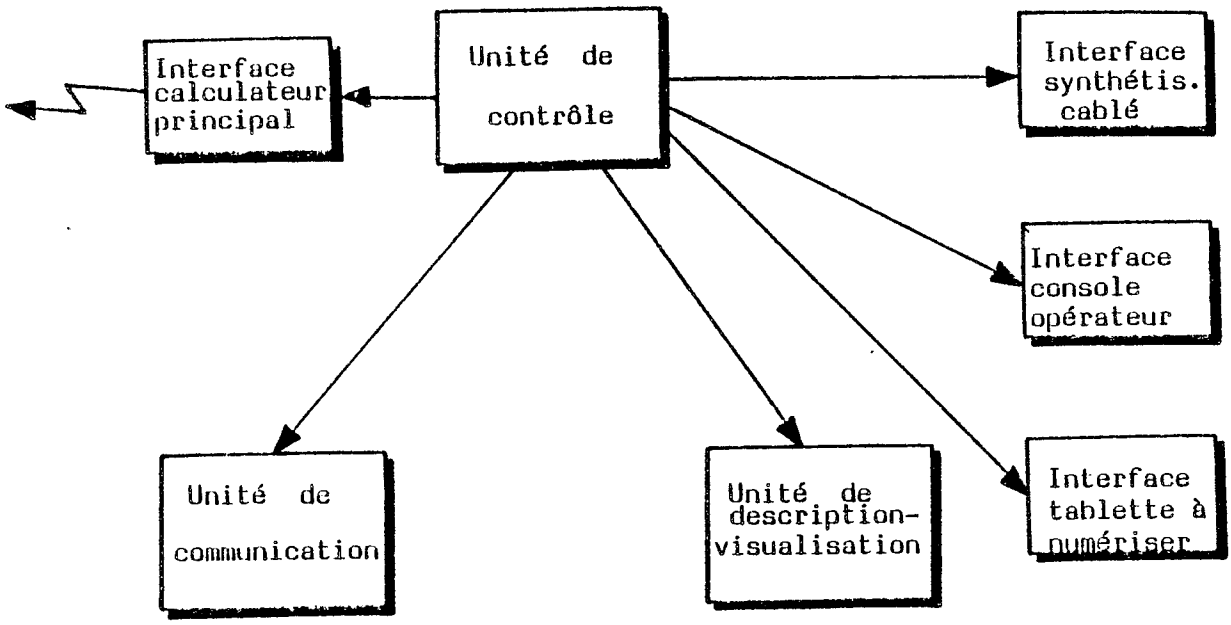


Figure VI.1: Organisation hiérarchique du logiciel pilote

L'unité de contrôle a la charge d'exécuter et d'aiguiller les processus d'attribution, de consultation, de visualisation et de description, invoqués à partir du calculateur principal ou localement à partir de la console-opérateur, selon le mode de fonctionnement.

1.1.1. Les modes de fonctionnement

L'unité de contrôle du logiciel pilote propose trois modes de fonctionnement correspondant à des phases distinctes du travail.

- 1) Mode "connexion": dans ce mode le logiciel est transparent et assure simplement l'échange entre la console opérateur et le calculateur principal. Tout se passe comme si la connexion était directe, ce qui

permet de dialoguer avec le système Multics afin de lancer les travaux.

- 2) Mode "opération": Ce mode est le mode normal de fonctionnement dans lequel le terminal est commandé par les programmes et les logiciels du calculateur principal. Tous les dispositifs, y compris la console opérateur sont alors sous la responsabilité du logiciel pilote. Pour chaque commande émise par le calculateur principal, le logiciel pilote émet un acquittement indiquant la fin (correcte ou érronée) de l'opération, et éventuellement les résultats demandés.

- 3) Mode "local": Les commandes sont émises de manière interactive à partir de la console opérateur. Tout se passe comme si la ligne entre le poste de travail et le calculateur principal était coupée.

Le passage d'un mode à un autre s'effectue soit à partir de la console opérateur à l'aide de touches particulières, soit à partir du calculateur principal à l'aide de commandes spécialisées.

1.1.2. La gestion des processus

Les processus gérés par l'unité de contrôle sont ceux qui ont été présentés en IV.4.3. En mode "opération" ces processus sont invoqués par le logiciel principal à l'aide de commandes codées sous forme de suites de caractères, pour les besoins de la transmission série entre les deux calculateurs. Ces commandes assurent l'attribution et la consultation de tous les attributs pour les types d'éléments traités par le logiciel pilote. En ce qui concerne les processus de description et de visualisation, le choix du processus (inclus dans la commande), indique non seulement l'ordonnancement et la nature des opérateurs à appliquer, mais encore le périphérique concerné (synthétiseur cablé, console-opérateur ou tablette à numériser). Il est également possible, si le processus n'est pas mentionné, d'utiliser un processus par défaut, ou de laisser le choix à l'opérateur. Ces divers processus permettent notamment:

- la visualisation sur le synthétiseur cablé (sous forme d'images réalistes),

- la visualisation sur la console-opérateur (sous forme de dessins au trait),

- la description de quelques types d'attributs sur l'un des trois périphériques (console-opérateur, synthétiseur cablé, ou tablette à numériser).

Quelques opérateurs élémentaires de description, parmi les plus caractéristiques seront présentés au chapitre VII. En mode "local" tous les processus peuvent être invoqués de manière interactive à l'aide de commandes émises à partir de la console-opérateur. Ce dialogue, effectué principalement à l'aide de menus et de désignations sur l'écran, permet également de configurer le poste de travail à l'aide de quelques fonctions locales qui seront présentées en 1.4.

1.2. L'unité de communication

1.2.1. L'identité des éléments

Cette unité qui est chargée de gerer la structure de données locale, est notablement plus simple que celle du logiciel principal qui sera présentée au paragraphe suivant. Cette simplicité s'explique par le niveau unique de structuration proposé. Chaque élément, quel qu'il soit (ligne brisée, texte, face-3D, etc.), est identifié par un entier exprimé sur quatre octets. La liste des numéros d'éléments concernés par un processus donné, peut être communiquée en utilisant les conventions suivantes:

- un numéro positif signifie: ajouter l'élément à la liste

- un numéro négatif signifie: ajouter tous les éléments compris entre le numéro précédemment communiqué et la valeur absolue du numéro négatif.

- un numéro nul signifie: supprimer tous les éléments de la liste.

Tous les processus invoqués s'appliquent sur tous les éléments de la liste courante. Cette organisation permet de minimiser les échanges avec le

calculateur principal.

1.2.2. Le support d'information

L'espace insuffisant dans la mémoire centrale du calculateur pilote (64K octets) nous a conduit à utiliser les unités de disques souples pour la mémorisation des éléments et leurs attributs. Cette contrainte, qui provoque un ralentissement sensible des processus, conduit à reconsidérer l'intérêt d'une mémorisation intermédiaire au niveau du logiciel pilote. Il est envisageable, effectivement de transformer les processus présentés au chapitre IV de manière à reporter cette mémorisation au niveau du logiciel principal, voire même de la supprimer. Ces nouveaux processus imposent alors un transfert de tous les attributs des éléments concernés à chaque fois qu'une reconstitution de l'image est nécessaire. Ce transfert, effectué depuis le calculateur principal, nécessite :

* un codage de chaque attribut sous forme de caractères,
* la transmission de ces caractères sur la ligne de connexion reliée au calculateur pilote (960 caractères par seconde),
* le décodage des attributs par ce dernier.

Il va sans dire que le temps de transfert devient très vite prohibitif et pénalise les performances de l'ensemble de façon beaucoup plus conséquente que l'utilisation des disques souples. Nous avons observé un rapport de performances supérieur à dix entre ces deux options. En outre la présence des éléments au niveau local permet de proposer des fonctions locales beaucoup plus puissantes, déchargeant ainsi d'autant le rôle du programme d'application ou du logiciel.

1.2.3. La gestion de la structure de données locale

Sur les deux unités de disques souples disponibles, l'une est réservée en permanance au système et aux programmes effectuant les divers processus, l'autre est dévolue aux structures de données locales. Outre le fichier graphique proprement dit, cette unité contient également les banques de textures et de modèles de réflexion nécessaires à la configuration du

matériel (cf. 1.4).

Afin d'optimiser le temps d'affichage, le fichier graphique est organisé pour l'accès direct sur le numéro d'élément. Un enregistrement de 256 octets est réservé pour mémoriser les attributs d'un élément, ce qui permet d'enregistrer 3000 éléments en moyenne sur une unité de disque. Les monotonies fréquentes de la liste courante des éléments permettent de minimiser les déplacements de la tête de lecture et d'atteindre ainsi des temps de réponse acceptables.

Les primitives disponibles au niveau de l'unité de communication, chargée de la gestion de cette structure de données sont:

- Parcours-Liste : Cette primitive permet de parcourir la liste des éléments courants, de la décoder, de vérifier sa validité et de communiquer les numéros d'éléments aux autres primitives.

- Affecter-Pilote (type-attribut, adresse-info)
  l'attribut de type 'type-attribut' est affecté aux éléments courants.

- Info-Pilote (type-attribut) : adresse-info
  permet de récupérer l'attribut de type 'type-attribut' affecté à l'élément courant.

- Détruire-Pilote :
  permet de détruire les éléments courants. Tous les attributs qui leur étaient affectés deviennent inaccessibles.

1.3. Les unités d'interface

1.3.1. Les échanges avec le calculateur principal

Nous examinerons les grandes lignes du protocole d'échange entre les deux logiciels pour les quatre types de processus.

- 1) Attribution: Le logiciel principal émet:
  * le code du processus (qui comprend le type d'attribut),

* la liste des éléments concernés (cf 1.2.1),
* l'attribut codé.

Lorsque le logiciel pilote a exécuté la commande il émet un message d'acquittement indiquant éventuellement les erreurs détectées (éléments inconnus, valeurs extérieures aux domaines de définition, etc.)

- 2) Visualisation

Le logiciel principal émet:

* le code du processus
* les éléments concernés

Après exécution le logiciel principal émet également un message d'acquitement.

- 3) Consultation

Le logiciel principal émet:

* le code processus (qui comprend le type d'attribut)
* l'élément concerné

Le logiciel pilote émet en retour l'attribut codé ou un message d'acquitement si une erreur a été rencontrée.

- 4) Description

Le logiciel principal émet:

* le code du processus
* les éléments concernés.

Le logiciel pilote envoi un message d'acquittement indiquant la fin du processus ou les erreurs détectées.

1.3.2. Les échanges avec le synthétiseur cablé

Cette unité prend en charge le codage particulier nécessité par la logique cablée d'HELIOS. Elle accède directement aux 7 registres du processeur d'interface (cf. V.3.2.4). Les opérations proposées par cette unité constituent un "habillage fonctionnel" des processus offerts par le matériel. Il s'agit donc de l'unité de contrôle du synthétiseur cablé, reproduite au niveau du logiciel pour faciliter les échanges et accroître la modularité de l'ensemble. La visualisation étant implicite nous trouverons trois types de primitives concernant: l'attribution, la consultation et la description pour l'unique type d'élément réellement accepté par le matériel: "le segment horizontal appartenant à une face 3D plane". Il est à noter qu'au niveau du synthétiseur cablé, l'identification des éléments (que nous nommerons "faces") s'effectue à l'aide de leur numéro compris entre 0 et 1023.

- 1) L'attribution des faces
  * Morphologie

    Segment (numéro-face, Y, X-gauche, X-droit)

Cette action provoque le remplissage du segment horizontal concerné dans le plan d'identification courant.

  * Aspect

    Texture (prem-face, der-face, nom-texture)

Attribue la texture de nom "nom-texture" aux faces dont les numéros sont compris entre "prem-face" et "der-face". La correspondance entre le nom de la texture et son adresse dans la banque intégrée des textures, est effectuée à l'aide d'une table initialisée lors de la configuration du poste de travail (cf 1.4.2).

    Réflexion (prem-face, der-face, nom-modèle)

Attribue un modèle de réflexion aux faces comprises entre "prem-face" et "der-face". La correspondance entre le nom du modèle et son adresse dans la banque des modèles intégrée au synthétiseur cablé HELIOS, est effectuée de la même façon que pour les textures.

Visibilité (prem-face, der-face, visibilité/transparence)

Attribue la visibilité et/ou la transparence des faces comprises entre "prem-face" et "der-face".

* Géométrie

Normale (prem-face, der-face, alpha, béta)

Attribue la direction perpendiculaire à la face exprimée en coordonnées sphériques par rapport au repère de la vue,

Projection (prem-face, der-face, Ux, Uy, Vx, Vy)

Cette primitive calcule l'inverse de la matrice si le déterminant n'est pas nul, et se charge également de la symétrie du repère si les coefficients diagonaux sont négatifs. Les coefficients obtenus sont codés selon le format désiré (cf V.2.2.3)

```
si valeur > 7.5 alors valeur ← 7.5 finsi
si valeur > 1.875 alors
|   e ← 1
|   m ← arrondi (valeur * 2)
sinon
|   e ← 0
|   m ← arrondi (valeur * 8)
finsi
```

Plan (numéro-plan)

Cette primitive permet d'indiquer un plan d'identification implicite

pour les primitives "segment", "visibilité" et "fenêtre". Le plan 0 est le plus proche de l'observateur, le plan 7 est le plan de fond (cf V.2.1.1).

* Eclairage

Eclairage (alpha, béta, couleur, ambiant)

Cette action réalise le codage de la direction de la source lumineuse, ainsi que le calcul de la direction médiane entre la source et l'observateur, et son codage. Elle transmet également la couleur de la source exprimée en R.V.B. et l'intensité de la lumière ambiante.

* Géométrie de l'affichage

Fenêtre (X-sup-gauche, Y-sup-gauche, taille, découpage)

Positionne la fenêtre du plan d'identification courant. Comme pour toutes les autres primitives, l'effet est immédiat.

- 2) La consultation: Les mémoires d'HELIOS étant accessibles en lecture tous les attributs communiqués au matériel peuvent, sans exception, être consultés, à l'aide de primitives duales de celles présentées ci-dessus. Nous n'en donnerons pas la liste qui figure en (Sar82).

- 3) Description: Aucun processus de description n'est disponible pour les attributs des faces, au niveau du synthétiseur cablé. Ces processus sont à la charge de l'unité de contrôle du logiciel pilote.

1.3.3. Les échanges avec les dispositifs de dialogue

Trois dispositifs de dialogue sont gérés actuellement par le logiciel pilote:

* le réticule intégré au synthétiseur cablé d'HELIOS,
* le réticule intégré à la console-opérateur,
* la tablette à numériser.

Ces dispositifs sont considérés comme des éléments particuliers sur

lesquels des processus spéciaux sont défini.

- 1) L'attribution: Seuls les deux réticules peuvent être attribués
  * Morphologie: implicite
  * Géométrie:

    Coord-rétic (nom-dispositif, x, y)

    Cette primitive permet de centrer l'un des deux réticules sur un point exprimé dans le repère écran. Cette action n'est pas définie pour la tablette.
  * Aspect

    Aspect-rétic (nom-dispositif, visibilité, couleur)

    Permet de faire apparaître ou disparaître l'un des deux réticules. Dans le cas du réticule HELIOS, une couleur quelconque peut être attribuée.

- 2) La description: Ce processus permet de décrire les coordonnées d'un point à l'aide des touches disponibles sur la console-opérateur et sur le pupitre d'interaction d'HELIOS, et également à l'aide du crayon associé à la tablette à numériser.

  Décrire-coord (nom-dispositif, processus)

  Le processus permet d'indiquer si une attente de visualisation doit être effectuée ou si, au contraire, les coordonnées sont à récupérer en continu.

- 3) La consultation: Deux attributs peuvent être consultés pour chacun des trois dispositifs:
  * La géométrie exprimant les coordonnées des points récupérés par le processus de description.

    Consulter-coord (nom-dispositif, x, y)

    Les coordonnées courantes du dispositif mentionné sont accessibles dans les variables x,y.
  * La visibilité exprimant également la validité des coordonnées. Lorsque

l'opérateur a introduit toutes ses coordonnées, il dispose de touches permettant de faire disparaître les réticules ou d'invalider la tablette. L'état du dispositif peut être testé par la primitive suivante:

Consulter-état (nom-dispositif, état)

1.4. Les fonctions locales

1.4.1. Initialisation

Lorsque le poste de travail est mis sous tension, les registres et mémoires du synthétiseur cablé sont dans un état indéterminé. La première tâche du logiciel pilote est de configurer la logique cablée selon les options standards suivantes:

- remplissage de tous les plans d'identification avec le numéro de la face 1023 (face réservée au système).
- initialisation du plan de fond avec la face 1023 également.
- initialisation des attributs de toutes les faces:
  * normale: perpendiculaire à l'écran,
  * projection: matrice identité,
  * invisibilité sur tous les plans,
  * texture numéro 0 (texture 512*512),
  * réflexion numéro 0.
- initialisation des fenêtres de telle façon qu'elles couvrent l'ensemble des plans d'identification (512*512), pas de découpage.
- initialisation de la source lumineuse:
  * direction perpendiculaire à l'écran,
  * couleur blanche,
  * luminosité ambiante 0.

- initialisation de la banque des textures à zéro (texture noire).

- initialisation de la banque des modèles de réflexion avec un modèle neutre (insensible à la position de la source, cf. chapitre VII). Ces initialisations sont effectuées automatiquement lors du lancement du système, mais peuvent être provoqués en mode local par l'appui sur une touche de "reset".

1.4.2. Configuration

Lorsque le poste de travail a été initialisé, aucune image n'est visible sur l'écran. Le plan de fond, seul visible, contient la face 1023 à laquelle la texture 0 est associée (texture noire couvrant tout l'écran). Dans cette configuration le terminal se comporte comme un terminal classique. En effet, la mémoire de textures joue le rôle de mémoire d'image, et toute image enregitrée (en RVB) apparaît sur l'écran.

La phase suivante consiste donc à configurer le terminal en fonction de l'application. Deux opérations sont effectuées:

- Remplissage de la banque des textures à partir du fichier 'textures' disponible sur l'unité de disque. Les textures récupérées sont enregistrées dans la mémoire (et apparaissent donc sur l'écran), et les correspondances entre le nom des textures et leur adresse en mémoire sont initialisées.

- Remplissage de la banque des modèles de réflexion à partir du fichier 'réflexions' situé sur l'unité de disque. La correspondance entre les nom des modèles et leur pointeur est également initialisée.

Ces deux opérations sont réalisées en mode local par l'opérateur qui peut ainsi personnaliser le poste de travail en fonction de son application, en faisant appel à un jeu de textures "naturelles" pour la présentation de paysages, ou à un jeu de textures simulant divers matériaux de construction pour la présentation de batiments, etc.

Une autre configuration peut être éventuellement effectuée dans le cas où l'application nécessite la visualisation de surfaces gauches. Cette visualisation est possible sur HELIOS si les plans d'identification contiennent pour chaque point, la normale à la face au lieu du numéro de face (modélisés tous deux sur dix bits). Il suffit pour celà d'initialiser la table des faces de façon à associer à chaque face le vecteur normal dont la représentation binaire correspond à celle du numéro de face. Le plan d'identification se transforme alors en plan de normales (cf. figure 2). La photographie de la page a été obtenue avec ce procédé.

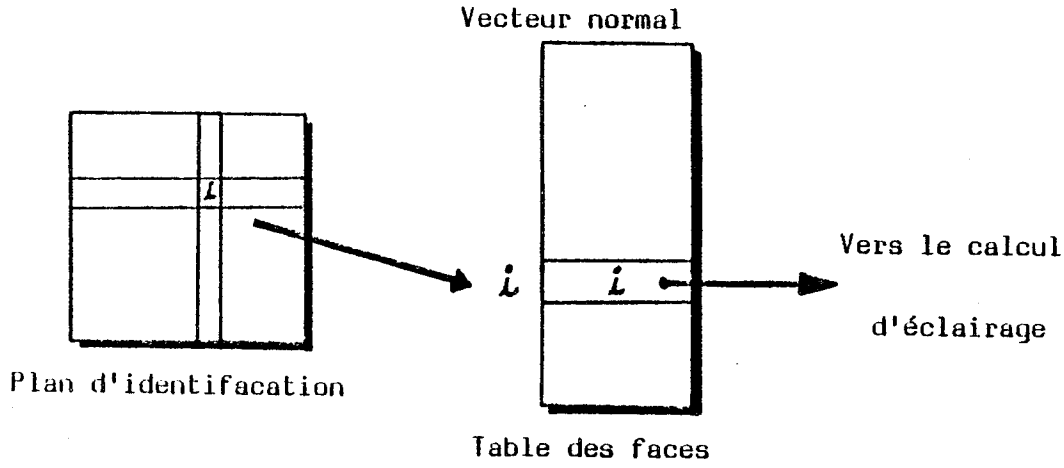


Figure VI.2: Configuration pour les surfaces gauches

1.4.3. L'archivage local

En plus des fonctions d'initialisation et de configuration, le logiciel propose deux fonctions permettant d'archiver et de récupérer une image. L'opérateur peut ainsi mémoriser l'état courant du poste de travail c'est-à-dire aussi bien le contenu des plans d'identification que des banques intégrées, et également les tables de correspondances du logiciel. Une fonction de récupération permet de restituer tous ces attributs en vue d'une nouvelle visualisation ou de fonctions locales de description. Cette facilité est plus qu'une simple "hard-copy" puisqu'elle considère également les mémorisations intermédiaires effectuées.

1.5. Conclusion

Cette présentation succinte du logiciel pilote permet d'apprécier l'importance des fonctions locales assumées. Il va sans dire que pour des applications pour lesquelles la finalité est la description pure et simple d'une image, les possibilités locales sont suffisantes et permettent un gain appréciable au niveau du coût et des temps de réponse. Nous avons pour notre part, utilisé le mode local pour toute la phase des tests. Le logiciel pilote, qui continue d'être développé à l'heure actuelle a été réalisé dans le cadre de quatre études (BaH81), (Bag81), (Pay82), (Sar82), dans lesquelles le lecteur intéressé trouvera de plus amples renseignements.

Nous consacrerons la suite de ce chapitre à la présentation du logiciel principal, à travers la réalisation de ses unités de communication et de contrôle, avant d'étudier quelques opérateurs élémentaires particuliers, dont certains sont en cours de transfert du logiciel principal vers le logiciel pilote, et même vers le matériel.

2. Le logiciel principal

Il s'agit là du logiciel implanté sur le calculateur principal (CII-HB 68 du Centre Inter-Universitaire de Calcul de Grenoble). Ce logiciel est directement accessible par les utilisateurs, c'est à dire que les primitives proposées par l'unité de contrôle (cf 2.2) constituent le jeu de commandes accepté par l'ensemble du système. Nous avons volontairement éludé toutes les primitives de modélisation particulière des données qui sont (et seront) développées au fur et à mesure des besoins, pour nous consacrer entièrement aux seules primitives ayant un rapport direct avec les quatre processus de base de la synthèse.

Nous avons pu ainsi réaliser le noyau minimal du logiciel, qui est actuellement achevé, et proposé aux utilisateurs pour une première période d'observation. La nature extensible de ce système nous a déjà permis de répondre dans un temps très bref aux demandes spécifiques de quelques utilisateurs. Cette adaptation ne nécessite dans la plupart des cas que

l'écriture d'une primitive de modélisation des données, permettant de modéliser un attribut non considéré jusqu'alors, et l'insertion dans le processus de visualisation d'un opérateur de synthèse ou de construction adapté à cet attribut.

Les pages qui suivent présentent les solutions retenues pour la réalisation des unités de communication et de contrôle.

2.1. L'unité de communication

Nous nous interesserons dans ce paragraphe, aux propriétés du fichier graphique utilisé par CLOVIS. Cette structure banalisée est entièrement gérée par une unité spécialisée: l'unité de communication. Comme pour toute structure, la gestion de ce "fichier graphique" comporte des actions permettant

* la structuration,
* le parcours,
* l'affectation,
* la recherche.

L'unité de communication doit également assurer:

- la protection des attributs qui lui sont confiés,

- la maîtrise totale de l'application sur tous les éléments, y compris ceux qui sont engendrés par les logiciels et le matériel.

2.1.1. La structure arborescente

De la structure utilisée, dépendent pour une grande part, la puissance et la souplesse d'utilisation de l'ensemble du système. Le choix de la structure mérite par conséquent l'attention toute particulière qui lui est accordée ici.

Deux niveaux de structuration sont apparus aux cours des chapitres précédents.

* le niveau maquette,
* le niveau élément.

Ainsi en première approche, le fichier graphique peut se présenter sous la forme de l'arborescence, schématisée par la figure 3.

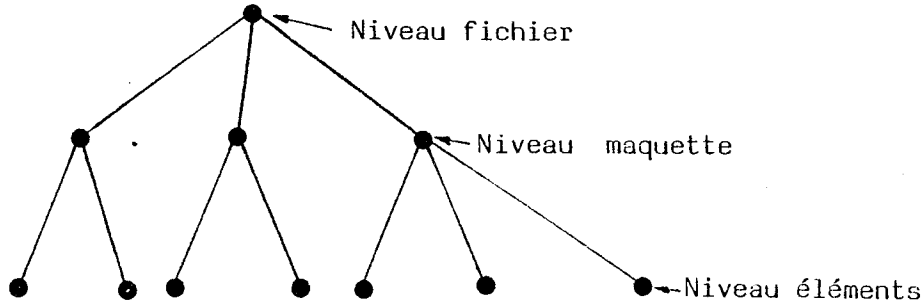


Figure VI.3: Structuration du fichier graphique

On peut noter qu'il s'agit bien d'une arborescence, car si un même objet apparaît dans plusieurs maquettes, il y aura création, dans chacune d'elles, d'un élément indépendant.

Ces deux niveaux de structuration ne suffisent pas, dans la majorité des cas, à exprimer les relations hiérarchiques existant dans la scène initiale. Il est en effet fréquent, en particulier pour les scènes "réalistes", que les objets soient "naturellement" structurés, par des considérations géométriques telles que

* la proximité,
* l'assemblage,
* l'inclusion,

ou encore par des considérations fonctionnelles telles que

* l'unité d'aspect,
* l'unité de mouvement, etc.

Nous avons choisi pour exprimer ces relations de prolonger la structure arborescente au niveau des éléments eux-mêmes (figure 4).

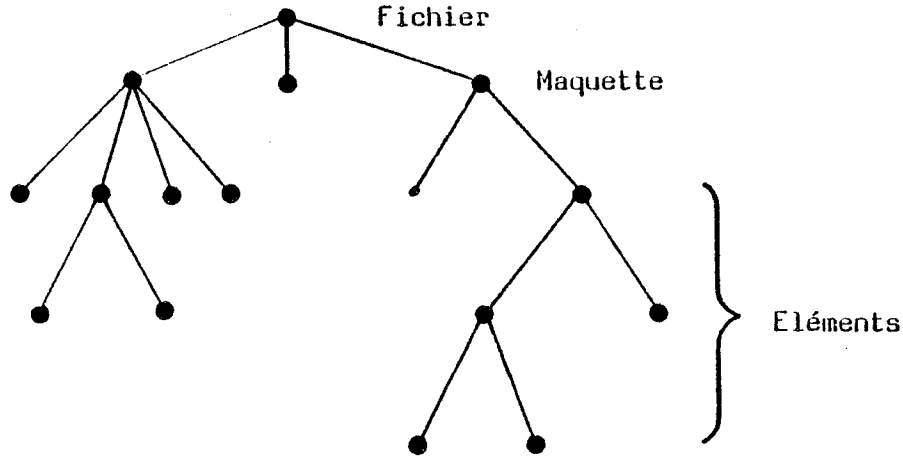


Figure VI.4: Structuration des éléments

Le principal reproche que l'on puisse faire à ce choix est précisément qu'il n'est qu'un "choix", qui ne répond pas nécessairement aux besoins de toutes les classes d'applications.

Il faut cependant se garder d'assimiler cette structure à celle qui est manipulée par l'application pour exprimer ses propres relations fonctionnelles. Il s'agit là de la structure graphique de la maquette et non de celle de la scène pour laquelle l'application reste, bien entendu, maître du choix.

2.1.2. Les avantages de la structure arborescente

L'utilisation d'une structure arborescente n'est pas nouvelle et a suscité de nombreux travaux, ces dernières années. On peut remarquer que ces publications concernent presque exclusivement l'exploitation des propriétés des arbres pour optimiser les processus de visualisation, notamment pour l'élimination des parties cachées et le découpage (Cla76), (FKN80), (RuW80).

En revanche les problèmes relatifs à la description et à la gestion de la structure sont éludés, ainsi que les avantages qu'elle procure pour l'ensemble du système. Il semble en effet, qu'il soit possible d'utiliser efficacement les propriétés de l'arborescence pour tous les processus du système.

Nous donnerons ci-dessous une liste des principaux avantages qui nous sont apparus à travers les processus réalisés. Nous pensons cependant que chaque nouveau processus peut ouvrir de nouvelles voies dans cette direction, en utilisant judicieusement les possibilités de mémorisation banalisée qui lui sont offertes.

- 1) Le nombre de niveaux n'est pas limité, ce qui permet de répondre aux besoins des applications manipulant des maquettes complexes. La plupart des logiciels graphiques ne proposent que deux niveaux, laissant ainsi à l'application la charge de la structuration graphique et l'exploitation judicieuse de celle-ci. Nous pensons pour notre part que cette contrainte est très forte et dépasse généralement la compétence du programme d'application pour lequel l'aspect graphique est souvent accessoire.

- 2) Les opérations quelles qu'elles soient, peuvent être appliquées à tous les noeuds de l'arbre, assurant ainsi une cohérence parfaite du système. Cette cohérence, absente des logiciels graphiques "conventionnels" (cf. III.4), procure une grande puissance aux opérations qui peuvent être invoquées sans avoir connaissance ni de la structure du sous-arbre dont le noeud concerné est la racine, ni du type des éléments qui s'y trouvent.

  La "portée" des différentes opérations sera évoquée dans les paragraphes qui suivent.

- 3) Les attributs contenus dans le fichier graphique peuvent être "composés" en fonction de la structure même. Ainsi les transformations géométriques telles que translation, rotation, etc., peuvent concerner tout le sous-arbre issu du noeud auquel elles sont affectées. Les lois de composition feront également l'objet des paragraphes suivants.

- 4) La puissance de l'interaction est considérablement accrue, puisque par simple désignation sur l'écran, on peut communiquer à l'application, non seulement l'identité de l'élément désigné, mais aussi la sous-structure complète à laquelle il appartient. Ces attributs peuvent alors être réutilisés directement pour invoquer les processus à appliquer sur cette sous-structure.

- 5) La protection des données graphiques peut également être assurée à travers la structure arborescente. Dans le cas fréquent où l'application est partitionnée en modules ayant chacun des besoins graphiques, il est possible d'affecter à chacun de ces modules un sous-arbre propre, qu'il gère à sa guise, mais auquel sa portée est limitée. Cette particularité évite les effets de bords classiques résultant de l'utilisation d'une structure linéaire.

- 6) Les processus peuvent être considérablement optimisés en exploitant judicieusement la structure arborescente. Quelques unes de ces possibilités seront évoquées lors de la présentation des processus de description et de visualisation.

2.1.3. Le mécanisme de dénomination

2.1.3.1. Le codage des noms

Le premier problème qui se pose, lors de la conception de l'unité de communication, est l'accès aux différents noeuds de la structure. Afin qu'ils puissent être reconnus, ces noeuds doivent être nommés à l'aide de noms codés. La structure elle même peut alors s'exprimer à travers les règles syntaxiques utilisées pour la reconnaissance des noms. Deux types de codages peuvent être utilisés:

- le codage numérique: Les noeuds peuvent être identifiés par un numéro. Ce type de codage (utilisé par exemple dans GRIGRI, et dans le logiciel pilote) s'adapte parfaitement aux systèmes à un ou deux niveaux de structuration, et permet un décodage très rapide. Pour les systèmes structurés au delà de deux niveaux, le codage numérique offre peu de lisibilité et limite les facilités offertes par l'utilisation de règles syntaxiques.

- Le codage alphanumérique: Il nécessite une place plus importante et une analyse syntaxique plus coûteuse que le précédent, mais procure une bonne lisibilité et une grande souplesse d'emploi sans limiter le nombre de niveaux autorisés.

Ces trois derniers arguments nous ont porté naturellement vers ce type de codage. Quant aux reproches qui lui sont faits, on peut arguer qu'il s'agit, en fait, du simple report au niveau du système de synthèse, de traitements qui auraient pris place, sous une forme ou sous une autre, dans l'application.

Chaque noeud sera identifié par un identificateur de 8 caractères au plus. La racine est repérée par un identificateur vide.

2.1.3.2. Les règles syntaxiques

La réalisation d'un premier prototype de l'unité de communication (Led79) nous a permis de constater les faiblesses d'une définition syntaxique trop pauvre, et de proposer des extensions (Mar80).

La syntaxe utilisée à l'heure actuelle permet, pour identifier un noeud, l'utilisation de noms

- relatifs: expression du chemin à suivre à partir d'un noeud quelconque utilisé comme racine,
- absolus: expression du chemin à suivre à partir de la racine du fichier graphique,
- globaux: expression simultanée de plusieurs chemins,
- indicés: expression de sous-arbres identiques différenciés par un indice.

Les règles syntaxiques autorisées dépendent des opérations concernées. Différentes syntaxes sont possibles selon qu'il s'agit d'exprimer la construction de la structure ou son parcours.

Construction

Syntaxe:

```
<structure> ::= <ss-structure> ("," <ss-structure>)*
<ss=structure> ::= <identif.> ( "[" <indices> "]" ) ( "(" <structure> ")" )
<indices> ::= <borne> ( ":" <borne> )
<borne> ::= <-99..99> || "!"
```

Premier exemple

* moulin [1:2] (corps (murs, toit), ailes [1:4])

  Cette chaîne exprime la construction de deux moulins structurés ainsi que le montre la figure 5 ci-après.

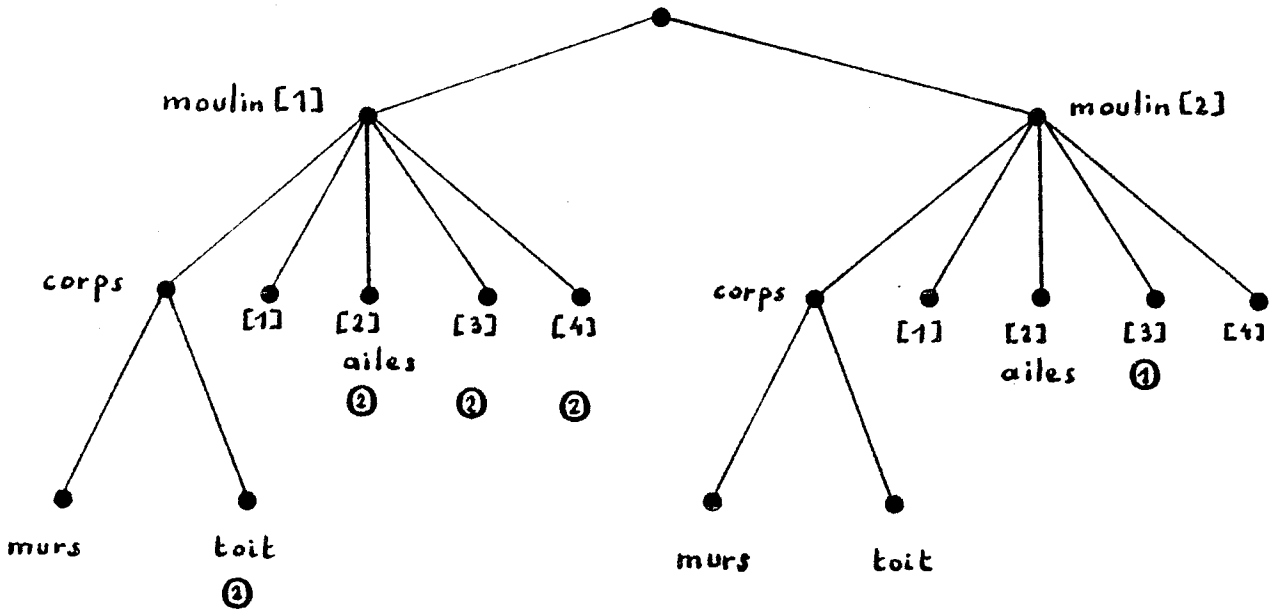


Figure VI.5: Premier exemple

Deuxième exemple

* vélo [1*!] (cadre (selle, guidon), roue [1:2] (jante, rayons))

Cette chaîne exprime la construction d'un nombre indéterminé de vélos (indiçables entre 1 et +99), dont la structure est illustrée par la figure 6.

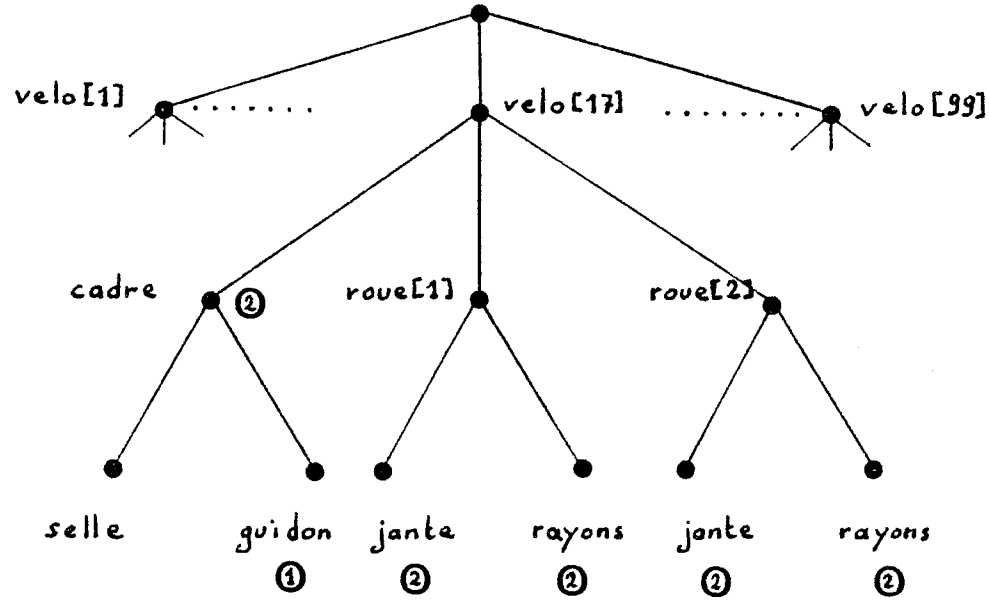


Figure VI.6: Deuxième exemple

Référence à un noeud unique

Syntaxe:

```
<chaîne-simple> ::= (<branche>)
<branche> ::= <identificateur> ( "[" <indice> "]" ) ( "(" <branche> ")" )
<indice> ::= <-99..99>
```

Exemple

* moulin [2] (aile [3]) identifie le noeud (1) dans la figure 5
* vélo [17] (cadre (guidon)) identifie le noeud (1) dans la figure 6

Référence à plusieurs noeuds

Syntaxe:

<chaîne-multiple> ::= ">" | "*" | <branche> ("," <branches>)*
<branche> ::= <identif.> ("["<indices>"]") ("(" <chaîne-multiple>")")
<indices> ::= <borne> (":" <borne>)
<borne> ::= <-99..99 > | "*"

Le symbole "*" référence toutes les occurences (cas des indices) ou toutes les branches issues d'un noeud.

Le symbole ">" référence tout le sous-arbre issu d'un noeud.

Exemples

* moulin [1] (corps (toit), ailes [2:*])
  identifie les noeuds marqués (2) dans la figure 5.
* vélo [*:17] (roue [*] (*), cadre)
  identifie les noeuds marqués (2) dans la figure 6, pour tous les vélos indicés de 1 à 17.

2.1.4. Les primitives de structuration

Les primitives de l'unité de communication, qui sont utilisées par l'unité de contrôle, sont également proposées à l'application. Outre les primitives d'affectation et de consultation, cette unité propose également deux fonctions permettant à l'utilisateur de modéliser plus aisément l'attribut de structure de la maquette.

2.1.4.1. Construction d'une structure

La construction d'une structure s'effectue au moyen de la fonction "STRUCTURE" donnant en retour l'adresse de la structure créée.

STRUCTURE (<structure>) : <référence>

<référence> est un pointeur qui contient l'adresse de la racine de la

Référence à plusieurs noeuds

Syntaxe:

```
<chaîne-multiple> ::= ">" | "*" | <branche> ("," <branches> )*
<branche> ::= <identif.> ("["<indices>"]") ("(" <chaîne-multiple>")")
<indices> ::= <borne> (":" <borne> )
<borne> ::= <-99..99 > | "*"
```

Le symbole "*" référence toutes les occurences (cas des indices) ou toutes les branches issues d'un noeud.

Le symbole ">" référence tout le sous-arbre issu d'un noeud.

Exemples

* moulin [1] (corps (toit), ailes [2:*])
  identifie les noeuds marqués (2) dans la figure 5.
* vélo [*:17] (roue [*] (*), cadre)
  identifie les noeuds marqués (2) dans la figure 6, pour tous les vélos indicés de 1 à 17.

2.1.4. Les primitives de structuration

Les primitives de l'unité de communication, qui sont utilisées par l'unité de contrôle, sont également proposées à l'application. Outre les primitives d'affectation et de consultation, cette unité propose également deux fonctions permettant à l'utilisateur de modéliser plus aisément l'attribut de structure de la maquette.

2.1.4.1. Construction d'une structure

La construction d'une structure s'effectue au moyen de la fonction "STRUCTURE" donnant en retour l'adresse de la structure créée.

```
STRUCTURE ( <structure> ) : <référence>
```

<référence> est un pointeur qui contient l'adresse de la racine de la

structure.

Exemples

* A ← STRUCTURE ("moulin [1:2] (corps (murs, toit), ailes [1:4])") crée la structure de la figure 8 et affecte son pointeur à la variable "A".
* B ← STRUCTURE ("moulin [*]) (aile [*])") crée un nombre indéterminé de moulins, contenant chacun un nombre indéterminé d'ailes, et affecte le pointeur à 'B'.

Dans le cas des noeuds indicés, un seul noeud est effectivement créé par la fonction STRUCTURE. Les bornes inférieure et supérieure sont mémorisées pour chaque noeud.

2.1.4.2. Définition d'une identité

L'identification d'une sous-structure du fichier graphique est effectuée par une fonction récursive basée sur la syntaxe des chaînes de caractères à analyser. Deux situations sont donc à considérer selon qu'il s'agit d'une référence simple ou d'une référence multiple.

L'invocation de l'analyse s'effectue à l'aide de la fonction ci-dessous, produisant en retour l'adresse de la racine de la sous-structure décrite.

IDENTITE (<chaîne-simple>) : ref-simple
IDENTITE (<chaîne-multiple>) : ref-multiple

Cette action vérifie les erreurs de syntaxe et de sémantique. Si une erreur est détectée, un message est envoyé sur la console-opérateur, en vue d'une correction immédiate. Le sous-arbre correspondant à la chaîne de caractères est engendré au fur et à mesure de l'analyse syntaxique, sur

l'arbre initial lui-même en ajoutant les chaînages nécessaires. L'adresse de la racine est le résultat de la fonction.

Exemple

A <- IDENTITE ("moulin [*] (corps, aile [2:3])")

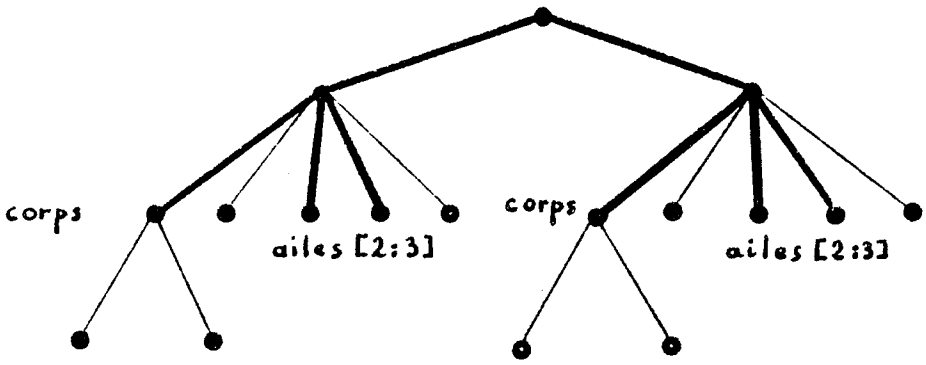


Figure VI.7: Sous-arbre extrait d'une structure initiale

Le sous-arbre extrait de la structure initiale est représenté en gras sur la figure 7.

Lorsque les références concernent des noeuds indicés, l'analyse peut provoquer un éclatement de l'arbre initial en sous arbres identiques.

Exemple:

- 1) A <- STRUCTURE ("vélo [1:10] (roue [1:2] (jante, rayon))") Après (1) l'arbre se présente comme suit (figure 8).
- 2) B <- IDENTITE ("vélo [3:6] (roue [1] (jante), roue [2] (rayon))") Après (2) l'arbre est éclaté et le sous-arbre extrait est celui représenté en gras sur la figure 9.

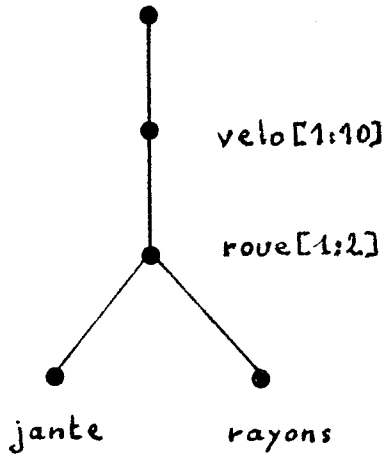


Figure VI.8: Arbre initial

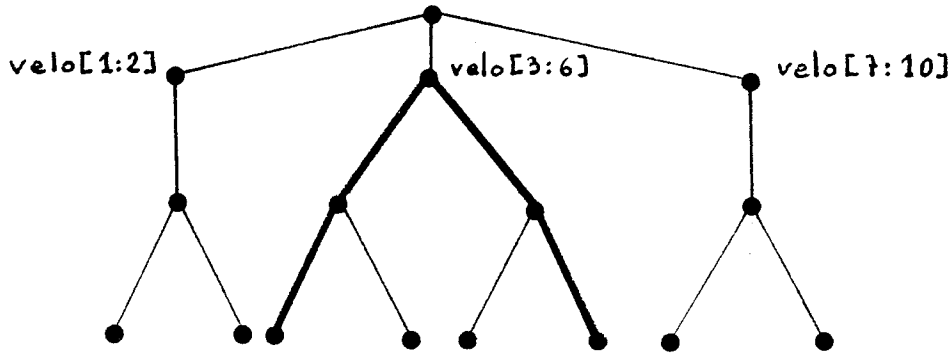


Figure VI.9: Arbre final après éclatement

2.1.4.3. Déclaration de contexte

CONTEXTE (ref-simple)

Cette primitive permet de définir un nouveau contexte implicite pour toutes les primitives invoquées ultérieurement. La définition d'un contexte s'effectue en indiquant le noeud qui deviendra la nouvelle racine implicite <ref-simple>. Le nouveau contexte est empilé, on revient au précédent par la primitive:

FINCONTEXTE

De nombreux avantages résultent de cette facilité.

- 1) Les chaînes de caractères servant à identifier les noeuds peuvent être réduites. Ainsi si l'on s'intéresse uniquement au sous-arbre issu du cadre du 3ème vélo dans l'exemple de la figure 6, il suffit de déclarer le contexte suivant:

CONTEXTE ("vélo[3] (cadre)")

L'accès au guidon se fait ensuite en mentionnant simplement "guidon". FINCONTEXTE permet de retrouver le contexte précédent qui dans ce cas est le fichier graphique tout entier.

- 2) L'évaluation des attributs associés aux noeuds "ascendants" de la nouvelle racine est effectuée une seule fois lors de la déclaration de contexte, ce qui réduit considérablement les temps de calcul. Dans l'exemple ci-dessus, les attributs attachés aux noeuds "vélo[3]" et "cadre" sont évalués lors de la déclaration et pour toute la durée du contexte.

- 3) La protection des attributs ainsi que la portée des opérations peuvent êtres contrôlées par l'application (ou par l'unité de contrôle) en astreignant les sous-programmes invoqués à un contexte donné. L'imbrication possible des contextes permet ainsi de calquer la structure graphique sur celle du programme d'application, encourageant du même coup la programmation structurée et modulaire, comme le montre l'exemple ci-après.

Programme principal

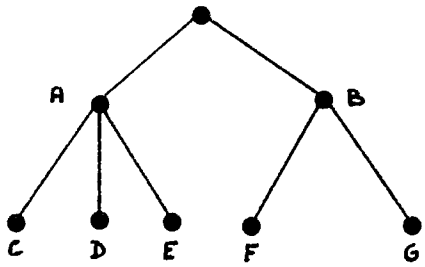
```
debut
|AFFECTER (STRUCTURE ("A,B"))
|SP1(IDENTITE("A"))
|SP2(IDENTITE("B"))
fin
```

Procedures

```
procedure SP1 (param)
| CONTEXTE(param)
| AFFECTER(STRUCTURE("C,D,E"))
| FINCONTEXTE
fin SP1

procedure SP2 (param)
| CONTEXTE(param)
| AFFECTER(STRUCTURE("F,G"))
| FINCONTEXTE
fin SP2
```

Structure resultante



- 4) La réalisation d'opérations récursives est grandement facilitée par l'imbrication des contextes, comme nous le verrons dans les paragraphes qui suivent.

2.1.4.4. Le parcours d'une sous structure

Le parcours d'une structure ou d'une sous-structure extraite après analyse s'effectue de manière récursive à l'aide de la primitive PARCOURS. A noter que cette action n'est pas proposée à l'application, elle est utilisée uniquement par l'unité de contrôle.

```
|| PARCOURS ( ref-multiple, action-f, action-d, action-a)
```

* ref-multiple est l'adresse de la sous-structure à parcourir: elle est définie par la fonction "IDENTITE",
* action-f est l'action à appliquer aux feuilles de l'arbre,
* action-d est l'action à appliquer aux noeuds parcourus dans l'ordre descendant,
* action-a est l'action à appliquer aux noeuds parcourus dans l'ordre ascendant.

Cette primitive parcourt l'arbre et invoque directement pour chaque noeud l'action correspondante en lui communiquant en paramètre l'adresse de ce noeud. Le parcours s'effectue toujours à partir de la racine du contexte.

Exemple:

Sur l'exemple de la figure 9 ci-dessus,
PARCOURS (b, action-f, action-d, action-a)
engengre la séquence d'appels suivante.
action-d (vélo (3:6))
action-d (roue (1))
action-f (jante)
action-a (roue (1))
action-d (roue (2))
action-f (rayon)
action-a (roue (2))
action-a (vélo (3:6)

2.1.5. La gestion interne des attributs

2.1.5.1. Représentation interne

L'utilisation des primitives présentées ci-dessus permet de gérer la structure de manière aisée. Nous nous intéresserons plus particulièrement dans ce paragraphe, à la gestion des attributs associés aux différents noeuds.

Plusieurs problèmes se posent:

- En premier lieu, la structure de données doit être banalisée, et par conséquent aucune modélisation des attributs ne peut être supposée a priori.

- D'autre part, les besoins des processus en mémorisation intermédiaire sont inconnus, ce qui revient à dire que le nombre d'attributs à mémoriser pour chaque noeud est également indéterminé.

- Enfin, certains attributs concernent la maquette (E,Gv,Ga) (c'est-à-dire une sous-structure et non pas un seul noeud), et d'autres peuvent être attachés à plusieurs noeuds simultanément.

Pour faciliter la réalisation du premier prototype de l'unité de communication, nous avons opté pour une limitation du nombre d'attributs attachables simultanément à un même noeud. Six attributs de classe et de type quelconques peuvent être considérés en plus de la structure et de l'identité.

L'organisation générale adoptée est représentée sur la figure 10 ci-après.

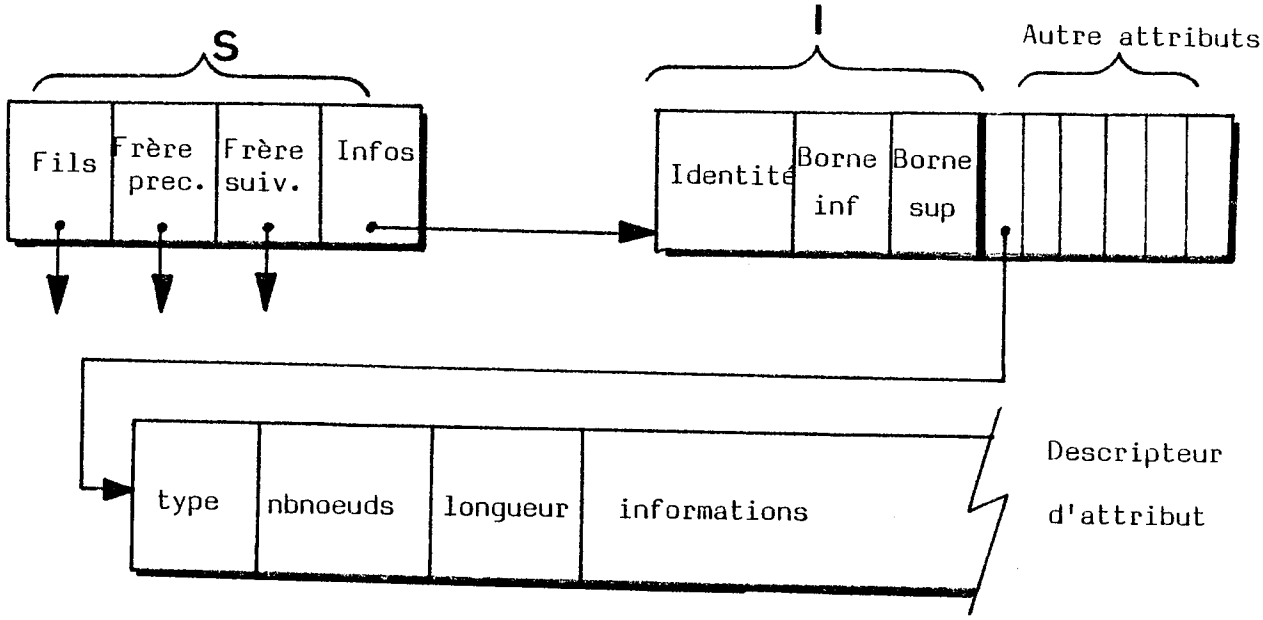


Figure VI.10: Modélisation interne des informations

- type exprime la classe et le type de l'attribut,

- nbnoeuds exprime le nombre de noeuds (c'est-à-dire d'éléments) utilisant cet attribut.

- longueur exprime le nombre de mots utilisés pour modéliser l'information définissant l'attribut.

2.1.5.2. Notions de constantes ou de variables

Comme toute application, la synthèse d'image manipule des éléments variables ou constants, c'est à dire dont tous les attributs restent inchangés tout au long de leur existence. Cependant, certains éléments peuvent être constants du point de vue géométrique, mais variables du point de vue morphologique, ou vice versa.

Afin de minimiser les échanges entre l'application et le système, nous proposons de considérer, pour chaque élément:

- des attributs constants qui sont conservés par le système et libèrent ainsi le programme d'application,

- des attributs variables qui sont conservés par l'application elle-même, et que le système examine à chaque fois que c'est nécessaire. Cette dernière solution est celle qui fut employée pour la réalisation du logiciel GRIGRI.
  Les attributs I et S sont toujours considérés comme des constantes, même s'ils peuvent être modifiés (figure 11).

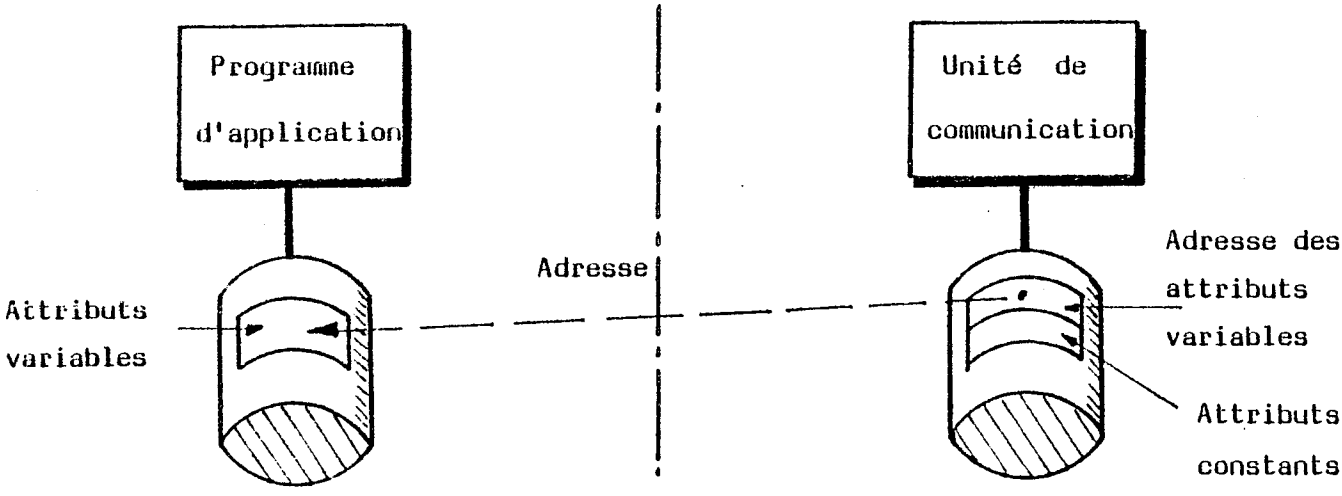


Figure VI.11: Les attributs constants et variables

2.1.6. Les primitives d'affectation, de recherche et de destruction

2.1.6.1. L'affectation

Il est bon de rappeler que les primitives présentées ici ne sont pas celles qui sont proposées aux utilisateurs, mais celles qui sont utilisées directement par l'unité de contrôle pour l'exécution des processus.

L'affectation s'effectue à l'aide de la primitive :

AFFECTER (ref-simple, type-attribut, adresse-info, cte/var)

Les actions suivantes sont réalisées:

- création du descripteur d'attribut et initialisation,
- s'il s'agit d'une constante, recopie des données sinon enregistrement de l'adresse. Le premier mot de l'information doit exprimer sa longueur.
- l'attribut est affecté au noeud déterminé par ref-simple.

Exemple 1:

Si l'attribut est de type 'structure', 'adresse-info' est l'adresse d'une structure à attacher aux feuilles mentionnées par "référence".

```
struct-a <- STRUCTURE ("moulin[*](ailes[1*4])")
struct-b <- STRUCTURE ("armature,voilure")
AFFECTER (struc-a,"structure",struc-b,cte)
```

Exemple 2:

Pour les autres types d'attribut, il existe des fonctions permettant de modéliser chacun des types disponibles afin de décharger l'application de cette tâche.

```
* jante <- CERCLE (xo,yo,ro)    (* Modélise un cercle et communique
  l'adresse du résultat *)
```

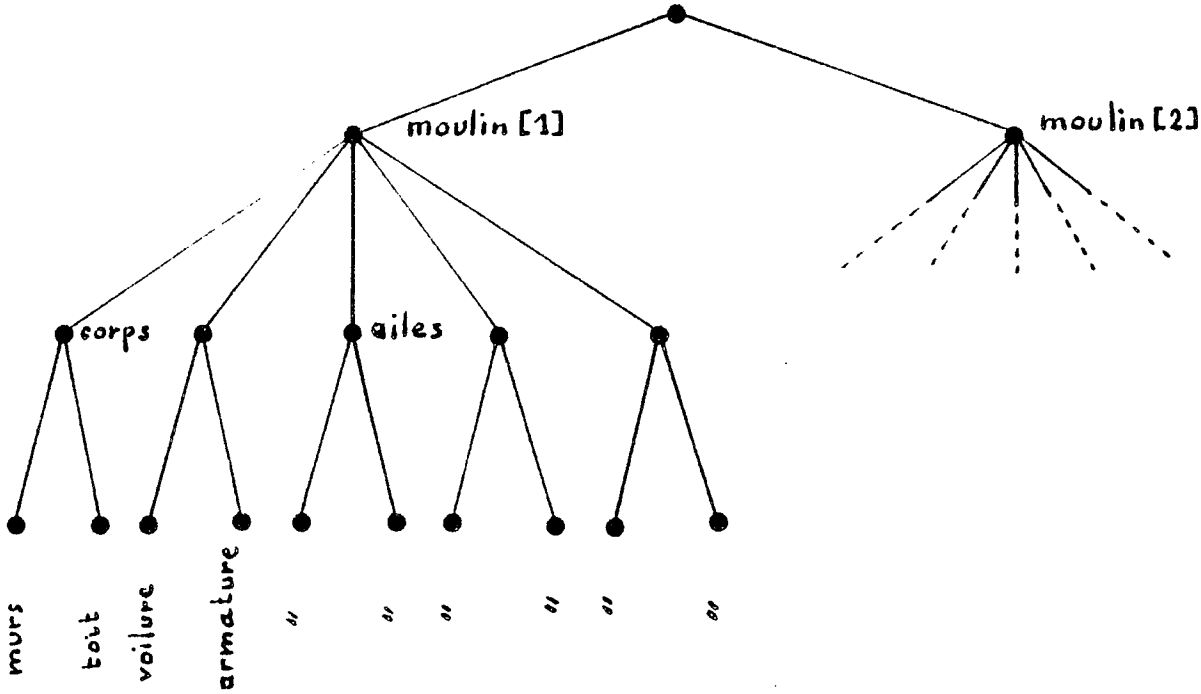


Figure VI.12: Structure de l'exemple 1

* AFFECTER (IDENTITE("velo[3](roue[1](jante))"), "cercle", jante, var)

Cette dernière primitive affecte l'attribut morphologique de type "type-cercle" au noeud considéré. Etant donné qu'il s'agit d'un attribut variable, sa modification peut s'effectuer sans nouvelle affectation.

* jante <- CERCLE (xl, yl, rl)

La modification de l'image aura lieu lors de la prochaine exécution du processus de visualisation, c'est-à-dire instantanément si le processus est du type "immédiat" ou encore lors d'une demande explicite (cf 2.2).

2.1.6.2. La recherche

Cette fonction permet de récupérer l'adresse d'un attribut associé à un élément de la structure.

INFO (ref-simple, type-attribut) : adresse-info

"ref-simple" référence obligatoirement un élément unique.

- Si l'attribut considéré n'existe pas pour cet élément l'adresse retournée est "nil".

2.1.6.3. La destruction des attributs

DETRUIRE (ref-simple, type-attribut)

Cette primitive permet de détruire l'attribut de type "type-attribut" associé au noeud référencé par ref-simple. L'adresse de l'attribut dans le descripteur du noeud est détruite (nil). Le champ "Nbnoeud" du descripteur d'attribut est décrémenté; s'il devient nul l'attribut est également détruit et l'espace mémoire récupéré.

Dans le cas où l'attribut est du type "structure" tout le sous arbre issu du noeud indiqué par ref-simple est détruit.

2.1.6.4. Le marquage des noeuds

L'optimisation des processus nécessite comme nous le verrons, de déterminer quelles sont les modifications ayant eu lieu dans le contexte depuis le précédent examen. A cet usage l'unité de communication tient à jour pour chaque noeud, un ensemble d'indicateurs, permettant d'exprimer l'état de chaque attribut.

* 1: inexistant
* 2: inchangé
* 3: modifié
* 4: modifié dans le sous arbre descendant

Des fonctions logiques permettent de tester l'état de chaque attribut pour un noeud donné.

* EXISTE : indique si l'attribut existe ou non,
* MODIF : indique si l'attribut a été modifié (directement ou indirectement).

2.2. L'unité de contrôle

Le rôle principal de cette unité est de gérer l'ordonnancement des processus associés aux primitives proposées par le logiciel principal.

Etant donné que l'étude des processus correspondant à toutes les situations envisagées dépasserait largement le cadre de cet ouvrage, nous fonderons cet exposé sur l'exemple schématisé par la figure 13, reprenant le processus présenté en IV.4.3.4. Les caractéristiques intéressantes de ce processus nous permettent d'illustrer les possibilités offertes par l'organisation proposée.

La partie du processus située sur le calculateur principal est pilotée par une unité de contrôle un peu particulière du fait de la structure arborescente utilisée à ce niveau. Elle se doit, en effet, d'exploiter au maximum les possibilités offertes par l'unité de communication.

2.2.1. Les primitives disponibles et proposées

Les opérateurs élémentaires dont dispose cette unité pour l'élaboration des quatre processus fondamentaux sont:

- Les opérateurs de l'unité de communication représentés par les fonctions et primitives, qu'elle propose.
  * STRUCTURE, IDENTITE, CONTEXTE, PARCOURS,
  * AFFECTER, INFO, DETRUIRE,
  * EXISTE, MODIF.

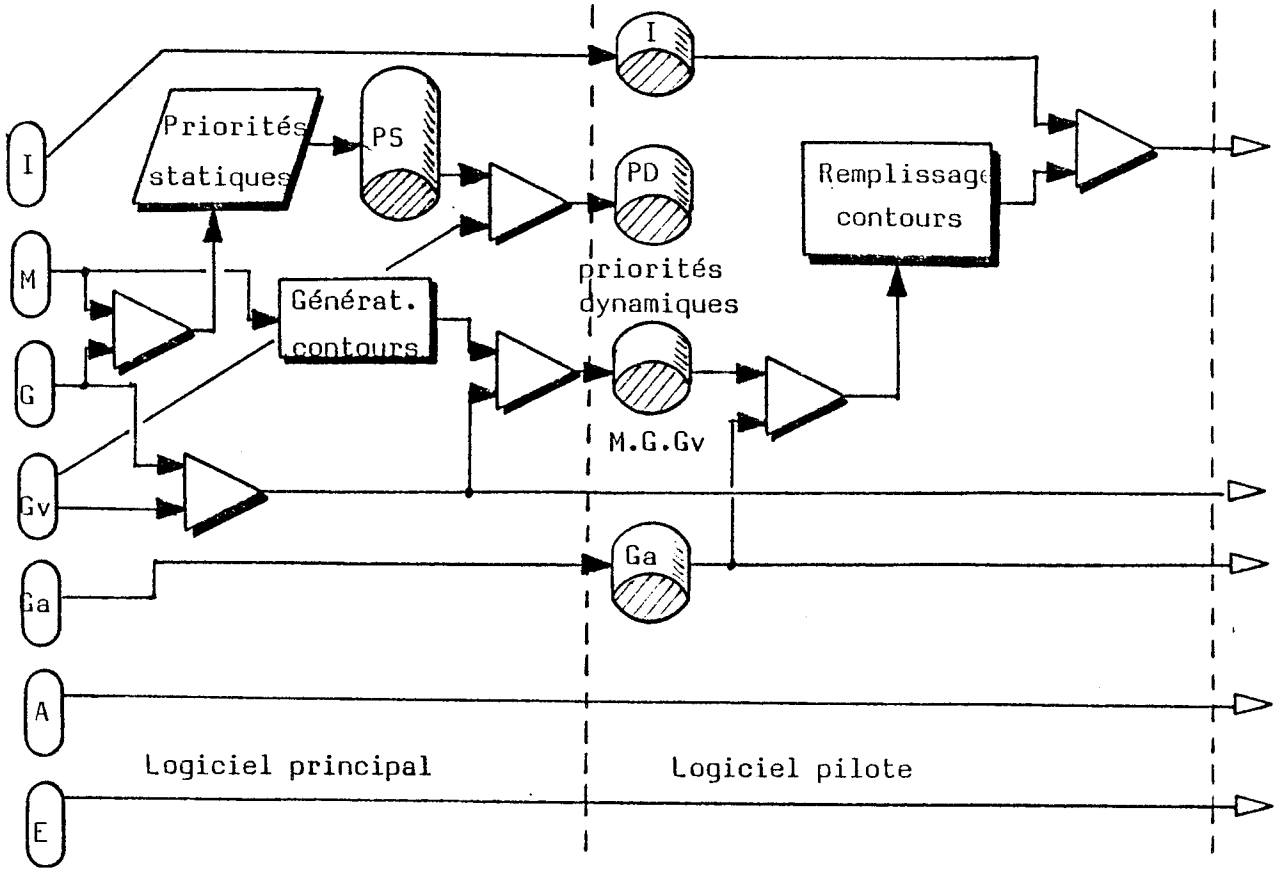


Figure VI.13: Processus général de synthèse

- Les opérateurs élémentaires de description et de visualisation, dont certains seront étudiés au paragraphe suivant.

- Les opérateurs élémentaires du logiciel pilote, c'est-à-dire les commandes:
  * ATTRIBUER-PILOTE,
  * CONSULTER-PILOTE,
  * VISUALISER-PILOTE,
  * DECRIRE-PILOTE.

Ces commandes invoquent directement les processus du logiciel pilote, définis pour des types d'éléments moins évolués que ceux manipulés au niveau principal, et dépourvus de structure (cf 1.2).

Si l'on exclue les fonctions de modélisation des attributs autres que I et S, les primitives proposées à l'utilisateur sont les suivantes.

* STRUCTURE
* IDENTITE
* CONTEXTE
* ATTRIBUER
* CONSULTER
* VISUALISER
* DECRIRE

2.2.2. L'attribution

Le processus d'attribution est en général assez simple puisqu'il utilise directement les opérateurs fournis par l'unité de communication. Il peut être invoqué par la primitive ci-dessous.

ATTRIBUER (ref-multiple, type-attribut, adresse-info, cte/var)

Cette action effectue le parcours de la structure indiquée par ref-multiple (définie par IDENTITE), et provoque l'affectation effective aux feuilles de la structure et le marquage des noeuds ascendants (état 4) (figure 14). Ce marquage sera utilisé par le processus de visualisation.

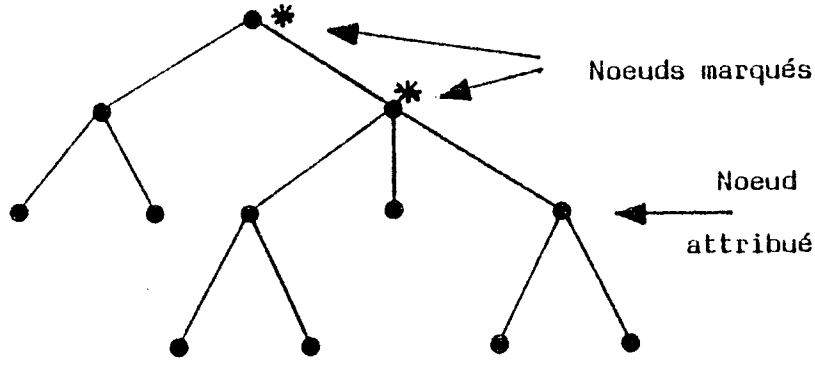


Figure VI.14: Le marquage des noeuds

Cependant quelques cas particuliers doivent-être considérés dans l'exemple de processus choisi.

Les attributs de type "PS", qui représentent les priorités statiques des éléments, sont mémorisés dans le fichier graphique et deviennent invalides dès qu'un attribut géométrique ou morphologique a été modifié. Dans ces deux cas, la primitive ATTRIBUER provoque automatiquement la destruction des priorités statiques, de tous les éléments atteints (cf 3.2).

2.2.3. La consultation

La fonction proposée aux utilisateurs est la suivante:

CONSULTER (ref-simple, type-attribut) : adresse-info

Ce processus fait directement appel à la fonction INFO de l'unité de communication. Aucun traitement complémentaire n'est nécessaire. Il est à noter,cependant que seuls les attributs mémorisés peuvent être consultés, c'est-à-dire M,A,G,E,S,Gv,Ga,PS. La consultation de l'attribut d'identité n'a pas de signification.

2.2.4. La visualisation

VISUALISER (ref-multiple, processus)

Le processus invoqué par la primitive ci-dessus effectue le parcours de l'arbre référencé par ref-multiple, afin de déterminer quels sont les attributs qui ont été modifiés (ou détruits) depuis la dernière visualisation. Seules les données modifiées sont recalculées. Le but final du traitement est de transmettre au calculateur pilote, pour chaque élément, les attributs I,PD,Ml,GG,A, ainsi que les attributs globaux Ga et E (cf. figure 13).

Le processus est représenté par la procédure suivante, appliquée à

chaque noeud marqué de l'arbre.

```
procédure visualiser (ref-simple)
debut
  CONVERSION ( ref-simple -> ref-pilote)
  si EXISTE (ref-simple, "Ga") et MODIF (ref-simple, "Ga") alors
    ATTRIBUER-PILOTE (ref-pilote, "ga", info (ref-simple, "Ga"))
  finsi
  si EXISTE (ref-simple, "E") et MODIF (ref-simple, "E") alors
    ATTRIBUER-PILOTE (ref-pilote, "E", info (ref-simple, "E"))
  finsi
  si EXISTE (ref-simple, "A") et MODIF (ref-simple, "A") alors
    ATTRIBUER-PILOTE (ref-pilote, "A", info (ref-simple, "A"))
  finsi
 si  EXISTE (ref-simple, "PS") alors
    AFFECTER (ref-simple, "PS", CALCUL-PS (ref-simple))
 finsi
 si MODIF (ref-simple, "Gv") ou MODIF (ref-simple, "PS") alors
    ATTRIBUER-PILOTE (ref-pilote, "PD", CALCUL-PD (ref-simple))
 finsi
 si  MODIF  (ref-simple,  "G")  ou  MODIF  (ref-simple,  "M")
    ou MODIF (ref-simple, "Gv") alors
    AFFECTER (ref-simple, "GG", CALCUL-GG (ref-simple))
    ATTRIBUER-PILOTE (ref-pilote, "Ml", CALCUL-Ml (ref-simple))
 finsi
 si MODIF (ref-simple, "G") ou MODIF (ref-simple, "Gv") alors
    ATTRIBUER-PILOTE (ref-pilote, "GG", info (ref-simple, "GG"))
 finsi
 si EXISTE (ref-simple, "GG") alors DETRUIRE (ref-simple, "GG")
fin
```

Les fonctions de calculs utilisées sont les suivantes:

```
fonction CALCUL-PS (ref-simple)
début
  P1 <- info (ref <- simple, "M")
  P2 <- info (ref-simple, "G")
  CALCUL-PS <- priorités-statiques (P1, P2)
fin

fonction CALCUL-PD (ref-simple)
début
  P1 <- info (ref-simple, "PS")
  P2 <- info (ref-simple, "GV")
  CALCUL-PD <- opérateur-3 (P1, P2)
       (* calcul du rang de l'élément cf. 3.2 *)
fin

fonction CALCUL-GG (ref-simple)
début
  P1 <- info (ref-simple, "G")
  P2 <- info (ref-simple, "Gv")
       (* composition des attributs G.Gv *)
  CALCUL-GG <- opérateur-2 (P1, P2)
fin

fonction CALCUL-M1 (ref-simple)
début
  P1 <- info (ref-simple, "M")
  P2 <- info (ref-simple, "Gv")
       (* projection des coordonnées *)
  CALCUL-M1 <- opérateur-4 (P1, P2)
fin
```

2.2.5. La description explicite

Tous les attributs qui peuvent être affectés à un élément, peuvent également être décrits de manière interactive.

L'invocation d'un processus de description s'effectue à l'aide de la

primitive ci-dessous:

```
DECRIRE (ref-multiple, type-attribut, processus)
```

Les différents processus permettent de prévoir plusieurs méthodes différentes (ou des dispositifs différents) pour décrire un même type d'attribut. Si le nom n'est pas mentionné, le choix du processus est laissé au logiciel pilote, qui peut lui-même laisser le choix à l'utilisateur au moment de la description.

Les opérateurs élémentaires utilisés sont:

*	CONSULTER-PILOTE	utilisation des processus de description
*	DECRIRE-PILOTE	du logiciel pilote.
*	VISUALISER	
*	AFFECTER	exploitation de la
*	INFO	structure de données.
*	PARCOURS	transformation de l'ident-multiple en ident-simp

On peut en outre distinguer deux grandes catégories de processus

- la description par construction,

- la description par référence.

Dans le premier cas, l'attribut est entièrement construit par l'opérateur à l'aide des dispositifs de collecte disponibles. Par exemple, la construction d'une face s'effectue en dessinant son contour sur la tablette à numériser. Ce type de processus peut être schématisé par la procédure ci-dessous

```
procédure CONSTRUIRE (ref-simple, type)
début CONVERSION (ref-simple -> ref-pilote)
    DECRIRE-PILOTE (ref-pilote, type-pilote, processus-pilote)
    CONSULTER-PILOTE (ref-pilote, type-pilote, info-pilote)
    CONVERSION (info-pilote -> info)
```

```
|     AFFECTER (ref-simple, type, info, cte)
fin
```

Dans le cas d'une description par référence, l'attribut est celui d'un élément désigné sur l'écran par l'utilisateur. Si l'attribut concerné est l'identité il s'agit d'une simple identification par désignation. S'il s'agit d'un autre attribut, l'identification est suivie d'une recherche dans la structure de données. Le paramètre "ref-multiple" représente l'ensemble des éléments "détectables" par désignation, qui doivent par conséquent être visualisés. La procédure ci-dessous schématise ce genre de processus.

```
procédure REFERENCE (ref-multiple, type)
début
|    VISUALISER (ref-multiple)
|    CONVERSION (ref-multiple -> ref-pilote)
|    DECRIRE-PILOTE (ref-pilote, "I", identification)
|    CONSULTER-PILOTE (ref-pilote, "I", info-retour)
|    CONVERSION (info-retour -> identité)
|    si type = "I" alors
|    PARCOURS
|    début
|    |    AFFECTER (ref-simple, type, INFO (identité, type), cte)
|    fin
fin
```

2.2.6. La description implicite

Nous avons montré dans une étude antérieure (Mar80) l'intérêt présenté par un mode de description implicite, dans lequel la description n'est pas demandée par l'utilisateur mais effectuée systématiquement à chaque fois qu'un attribut est indéfini ou incomplet. La demande de description implicite est engagée lorsqu'une attribution fait référence à une information "vide".

ATTRIBUER (ref-multiple, type-attribut, Ø, cte/var)

Un processus de description implicite sera invoqué lors de la visualisation: la première fois seulement pour une attribution de constantes; à chaque fois pour les attributs variables. Cette facilité présente deux avantages principaux:

- elle dégage l'application de la gestion de la description, qui peut être prise en charge intégralement par le logiciel,

- elle permet d'effectuer la description au moment même de la visualisation, et de bénéficier ainsi des renseignements apportés par les éléments visualisés. L'opérateur peut ainsi désigner des points de contacts ou d'assemblage entre divers éléments, ou simplement situer l'élément décrit par rapport à ceux précédemments affichés.

Nous avons, en outre, étendu le concept de description implicite à l'attribut d'identité. Les différentes entités syntaxiques entrant dans la composition d'une identité peuvent à cet effet, être remplacés par le symbole "?".

Exemple

ident <- IDENTITE ("vélo[?] (roue[2], cadre (?))")

Lors de la définition de cette identité, le système provoque les demandes d'identification nécessaires pour compléter les références inconnues. En vue de faciliter la désignation des éléments le logiciel indique, à chaque instant, le niveau concerné sur la console opérateur.

Dans l'exemple ci-dessus, deux cas peuvent se présenter:

- 1) Si l'opérateur désigne la selle du troisième vélo, il satisfait d'un seul coup à la demande d'identification.

- 2) S'il désigne par exemple l'une des deux roues du troisième vélo, il ne satisfait qu'une partie de la demande, et une nouvelle identification est demandée à partir de l'identité.

"velo[3] (roue[2], cadre (?))"

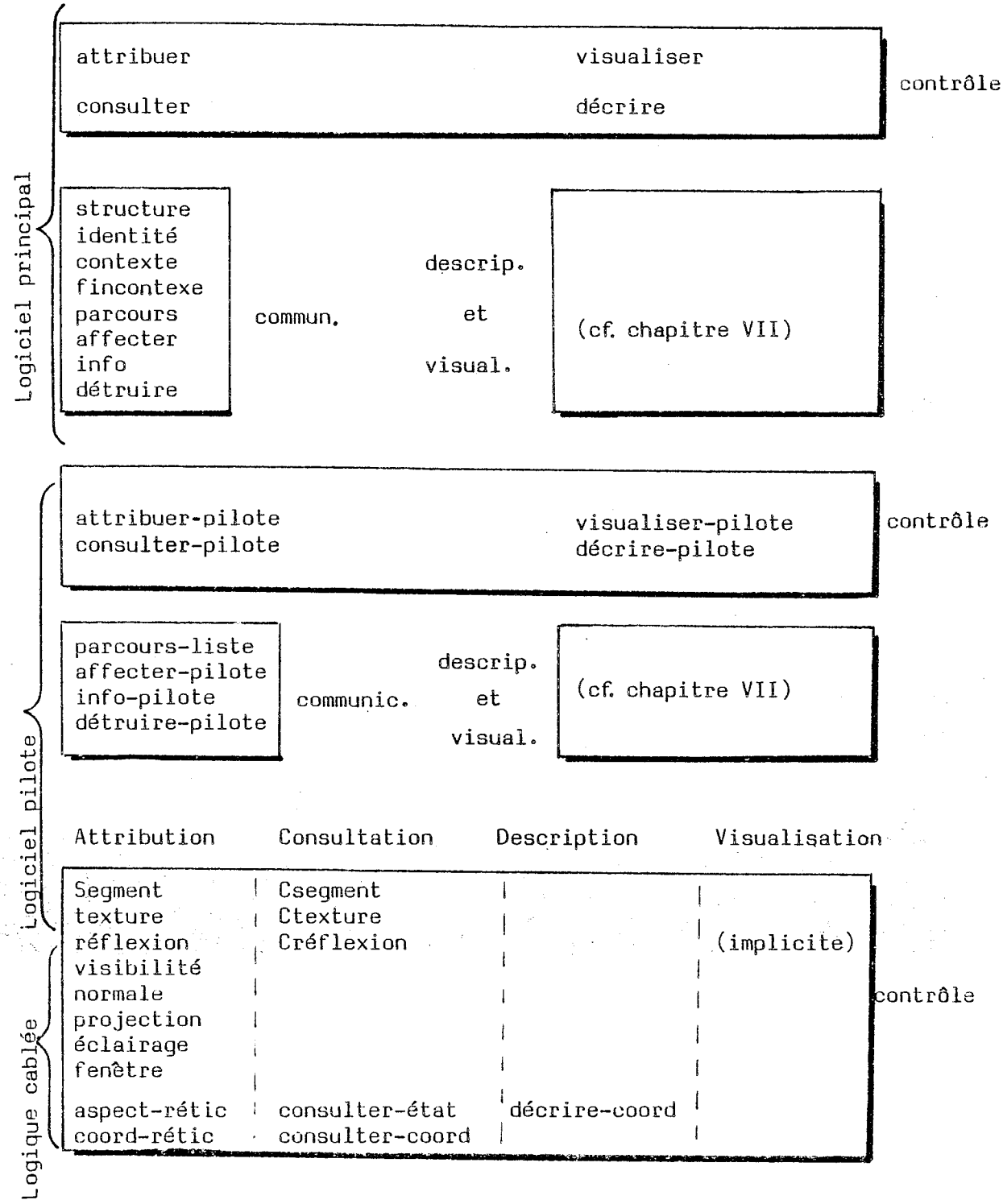
3. Récapitulatif et conclusion

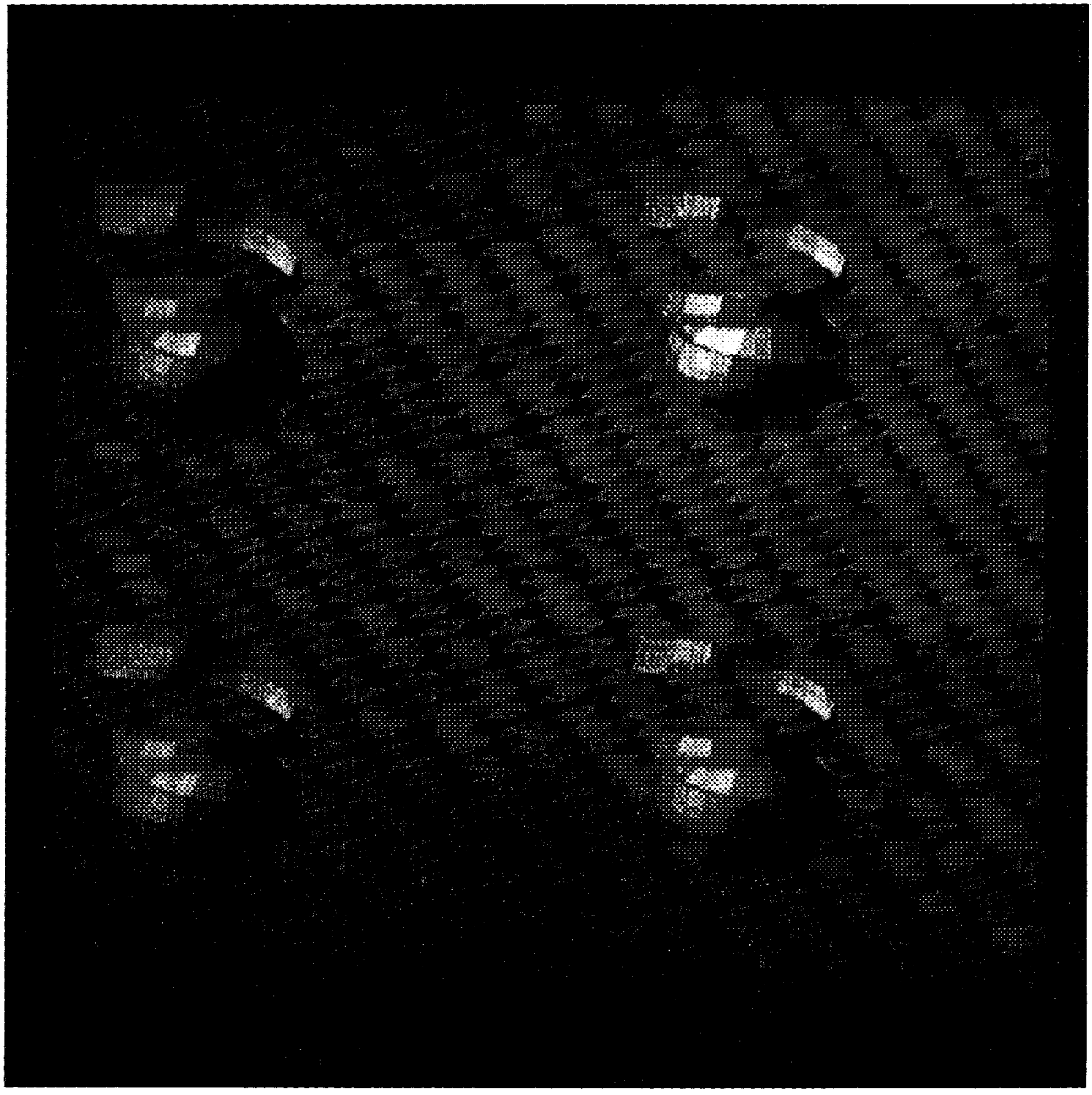
Le tableau récapitulatif des primitives (voir page suivante) met en évidence la similitude existant entre les deux logiciels. Seul le choix d'une structure plus complexe, au niveau du logiciel principal, justifie la présence de primitives supplémentaires dans son unité de communication. Ces primitives, qui permettent la modélisation des attributs d'identité et de structure, sont indispensables si l'on désire dégager totalement l'utilisateur de la connaissance des représentations internes.

Au niveau du logiciel pilote, outre l'absence des primitives de structuration, on peut noter la présence d'actions propres à l'unité de contrôle du synthétiseur cablé. Il s'agit là, comme nous l'avons déjà dit, d'une simple commodité permettant d'accroître la modularité de l'ensemble et de mettre en évidence les échanges entre logiciel et matériel.

En ce qui concerne les opérateurs des unités de visualisation et description, il serait fastidieux et peu intéressant d'en donner la liste complète et la localisation, compte-tenu de l'évolution constante de ces unités et des dépendances étroites existant entre les divers opérateurs. Il nous semble important, au contraire, de présenter l'aspect fonctionnel de ces opérateurs, à travers quelques processus de visualisation et de description choisis parmi les plus originaux ou les plus aptes à faire apprécier les performances et la qualité des résultats obtenus. Le chapitre suivant est entièrement consacré à cette présentation.

Tableau récapitulatif





ORANGES EPLUCHEES

CHAPITRE VII

Aspect fonctionnel

Les opérateurs de description et de visualisation

Après avoir présenté, au chapitre précédent, les unités de contrôle et de communication des deux logiciels développés, nous consacrerons celui-ci aux opérateurs caractéristiques des unités de description et visualisation.

Les pages qui suivent poursuivent en fait un triple objectif:

- illustrer l'intérêt des concepts proposés: structuration arborescente, séparation des informations,
- introduire quelques nouvelles approches en particulier pour le remplissage des taches ou l'élimination des parties cachées,
- présenter, à travers quelques opérateurs de description et quelques utilisations "marginales" du matériel, les résultats obtenus sur ce système.

Nous avons choisi de mettre l'accent sur le réalisme des images obtenues, ainsi que sur le dynamisme des opérateurs proposés. Puisque les photographies illustrant cette thèse suffisent à apprécier l'aspect qualitatif des résultats, nous insisterons plus particulièrement sur le fait que la plupart des opérateurs évoqués s'exécutent en temps réel. Nous évoquerons également, dans ce but, quelques réalisations micro-programmées ou cablées mettant en jeu des processeurs parallèles.

1. Les processus de visualisation

Outre les opérateurs de mémorisation proposés par l'unité de communication, les processus de visualisation nécessitent des opérateurs élémentaires de synthèse et de modélisation qui sont regroupés dans l'unité de visualisation-description.

La diversité de ces opérateurs étant pratiquement illimitée, nous ne considèrerons que ceux qui présentent un intérêt particulier du fait de leur conception ou de leur réalisation.

Nous insisterons notamment sur l'incidence de la structure arborescente et des possibilités offertes par HELIOS, sur les différentes opérations proposées. Il est à noter que ces opérateurs sont localisés soit dans le logiciel principal, soit dans le logiciel pilote comme le montrent les schémas des processus de visualisation (cf. IV.4.3).

1.1. Les transformations géométriques

1.1.1. Les coordonnées homogènes

L'utilisation de coordonnées homogènes pour la représentation des matrices de transformations géométriques n'est pas nouvelle et son utilisation dans le domaine graphique ne l'est pas plus (Rob63). Leurs propriétés intéressantes, et les problèmes qu'elles soulèvent ont suscité de nombreux travaux au cours de ces dernières années (SuH74), (BlN78), (RoA76), et elles sont à l'heure actuelle presque unanimement adoptées.

Les principales propriétés des coordonnées homogènes sont:

- la possibilité d'exprimer par une matrice unique toute transformation: rotation, homothétie, translation et même projection (axonométrique ou perspective),
- la possibilité d'exprimer des points très éloignés ou situés à l'infini,
- la possibilité d'exprimer toute combinaison de transformations quelles qu'elles soient, par un simple produit matriciel.

Dans notre cas, trois types de transformations sont possibles selon la classe des attributs concernés (G, Gv ou Ga).

1.1.2. Composition des transformations géométriques

On trouve ici toutes les transformations de base servant à situer et à placer les éléments dans la maquette (classe 'G'). Dans le cas d'une maquette tri-dimensionnelle, ces transformations de base s'expriment à l'aide des matrices 4 * 4 suivantes.

* Rotation d'un angle θ autour de ox

$$Rx = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

* Rotation d'un angle θ autour de oy

$$Ry = \begin{vmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

* Rotation d'un angle θ autour de oz

$$Rz = \begin{vmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

* Translation de (l, m, n)

$$T = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ l & m & n & 1 \end{vmatrix}$$

* Homothétie

$$H = \begin{vmatrix} p & 0 & 0 & 0 \\ 0 & q & 0 & 0 \\ 0 & 0 & r & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Toutes ces transformations peuvent être composées entre elles lors de la description de la maquette, en vue d'obtenir une matrice unique associée à chaque élément.

1.1.3. Structuration des transformations géométriques

L'une des caractéristiques intéressantes de la structure arborescente est la possibilité de composer directement les transformations géométriques au cours du parcours de l'arbre.

Un attribut de la classe 'G' s'applique alors à tous les noeuds de l'arbre dont le noeud courant constitue la racine. Cette facilité permet de réduire l'encombrement mémoire et d'appliquer une combinaison de transformations à tout un ensemble d'éléments, à travers l'attribution d'une seule matrice à un seul noeud.

1.1.4. La prise de vue

Il s'agit ici d'exprimer les paramètres de la prise de vue: point de vue et point de visée (classe 'Gv'). La prise de vue peut être décomposée en trois opérations élémentaires:

* translation de la maquette pour amener l'origine du repère au point de visée (matrice T) ,
* rotations pour faire passer l'axe oz par le point de vue (matrice R),
* projection des coordonnées dans le plan xOy (matrice P).

On peut donc exprimer cette prise de vue par un simple produit matriciel:

$$V = T * R * P$$

P est la matrice de projection ci-dessous:

$$P = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1/d \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

où d représente la distance entre le point de vue et le point visé. On notera que la composante en z des coordonnées est conservée pour les besoins des opérations d'élimination des parties cachées.

La composition des attributs 'Gv' dans la structure arborescente, n'a aucune signification. Seul le dernier attribut 'Gv' rencontré dans le parcours d'arbre est significatif, c'est à dire le plus "bas" dans l'arborescence (figure 1).

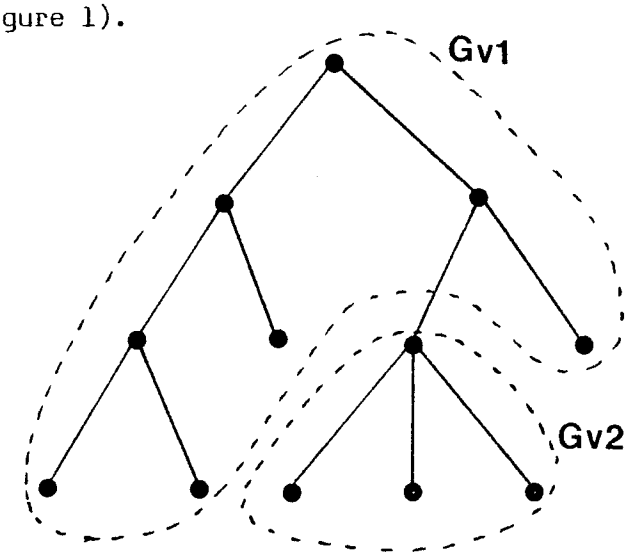


Figure VII.1: Composition des informations 'Gv'

Ceci permet notamment de prendre des vues particulières pour certains éléments afin de les présenter sous un angle différent, sans pour cela modifier la maquette initiale.

1.1.5. L'affichage

Cette opération permet de projeter la vue obtenue (exprimée dans le plan xOy du repère absolu), sur une clôture définie sur l'écran (classe 'Ga'). De même que pour 'Gv', une composition de ces attributs en fonction de la structure arborescente n'a pas de signification. Nous utiliserons les mêmes conventions. Deux opérations distinctes sont incluses dans ce traitement.

* la transformation proprement dite dans le repère de l'écran,
* le découpage aux bords de la clôture.

- 1) La transformation

On peut faire subir de nouvelles transformations à la vue obtenue après application de 'Gv'. Ces transformations bi-dimensionnelles expriment la mise en place de la vue sur l'écran lui-même (rotations, translations, homothéties) comme le montre la figure 2 ci-dessous.

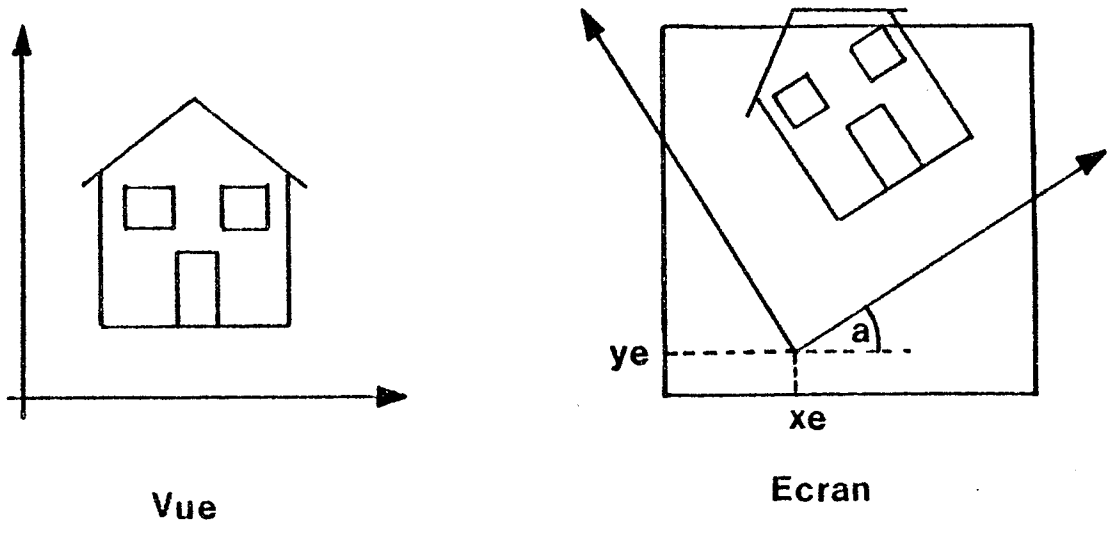


Figure VII.2: Mise en place de la vue sur l'écran

Cette transformation s'exprime en coordonnées homogènes et peut être composée avec les précédentes si l'élimination des parties cachées n'est pas nécessaire.

$$A = \begin{vmatrix} k\cos(a) & -k\sin(a) & 0 \\ k\sin(a) & k\cos(a) & 0 \\ xe & ye & 1 \end{vmatrix}$$

k est le facteur d'homothétie

- 2) Le découpage

Le problème du découpage (clipping) consiste à déterminer quelles sont les parties de la vue, visibles dans la partie rectangulaire de l'écran (clôture) où elle doit être affichée (figure 3). Un algorithme de découpage assez simple, travaillant directement en coordonnées homogènes a été proposé par I.E Sutherland et G.W. Hodgman (SuH74) et permet de traiter aussi bien des faces planes que des faces gauches. Le lecteur intéressé trouvera également dans (Bli78) une étude des problèmes posés par l'utilisation des coordonnées homogènes pour le découpage.

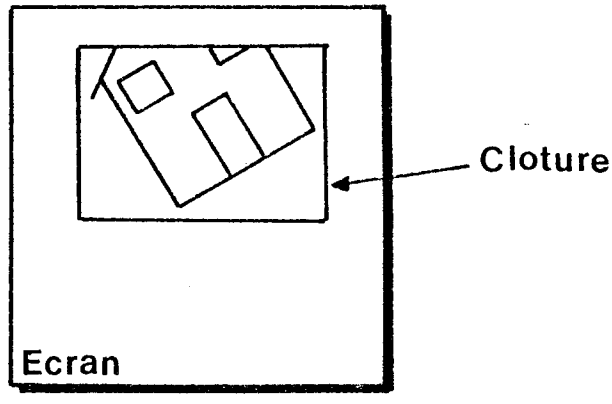


Figure VII.3: Découpage aux bords de la clôture

On notera qu'à travers ces processus, la notion de fenêtre disparaît totalement. Elle correspond en effet à une façon particulière d'exprimer, pour des maquettes bi-dimensionnelles, à la fois les attributs 'Gv' et 'Ga'. Elle peut bien sûr être proposée en tant que primitive destinée à faciliter la modélisation de ces attributs.

1.2. Le remplissage des taches

Nous avons évoqué en III.4.4 les problèmes posés par le remplissage d'une tache polygonale. Cet opérateur est prépondérant au niveau des performances, et doit par conséquent être situé au niveau du logiciel pilote, ou si possible au niveau du matériel. Nous présenterons ci-après deux approches visant à répartir le traitement du remplissage entre plusieurs processeurs afin d'exploiter au maximum le parallélisme de type pipe-line.

1.2.1. Remplissage par décomposition en trapèzes élémentaires

Ce type d'algorithme, présenté en III.4.4.2, nécessite la détermination des régions cohérentes constituant la tache à remplir. Ces régions sont les triangles ou les trapèzes délimités par les sommets du polygone comme le montre la figure 4.

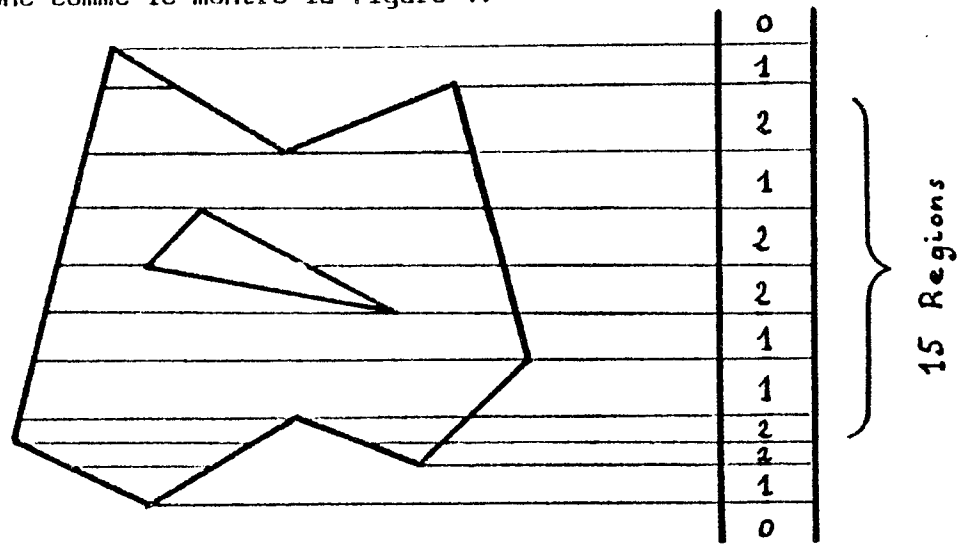


Figure VII.4: Décomposition en régions cohérentes

Le rôle du logiciel pilote se borne à déterminer ces régions élémentaires. Cette détermination débute par le tri des arêtes selon les ymax décroissants. Le tableau ainsi obtenu est ensuite parcouru en tenant à jour une liste de bords gauches et de bords droits. Chaque nouvelle ordonnée rencontrée dans le tableau des arêtes, nécessite une nouvelle étude. Trois cas peuvent se présenter:

- 1) deux arêtes débutent -> une nouvelle région débute ou bien une ancienne se sépare en deux.

- 2) deux arêtes se terminent -> une ancienne région se termine ou deux régions se rejoignent.

- 3) une arête finit et une commence -> une région se termine, une autre commence. La figure 4 montre les 15 régions obtenues par cette approche. Il ne s'agit pas d'une décomposition optimale, et la littérature fournit quelques méthodes permettant d'obtenir la décomposition de la figure 5 qui ne compte que 11 régions (Lee81).

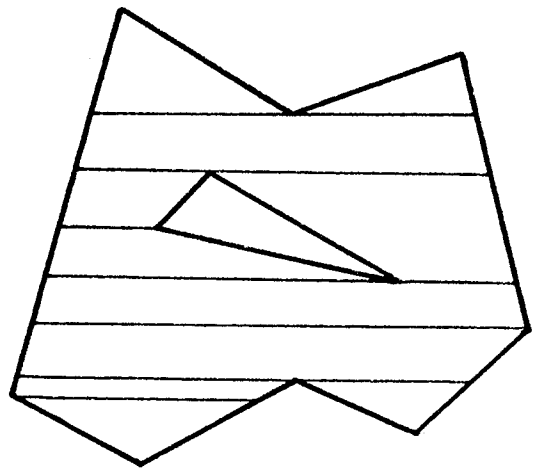


Figure VII.5: Réduction du nombre de régions

Il est clair que le temps passé à cette optimisation n'est pas négligeable et qu'on peut se poser la question de son intérêt dans le cas où le remplissage des régions est confié à un opérateur cablé. Une étude récente (Heg82) montre qu'en pratique cette "optimisation" n'est pas intéressante.

Le remplissage cablé ou micro-programmé d'un trapèze ou d'un triangle ne pose pas de problèmes particuliers. Nous avons choisi une méthode permettant d'utiliser directement les possibilités de compression des

données offertes par le synthétiseur cablé d'HELIOS.

Les régions sont représentées de la façon suivante (figure 6):

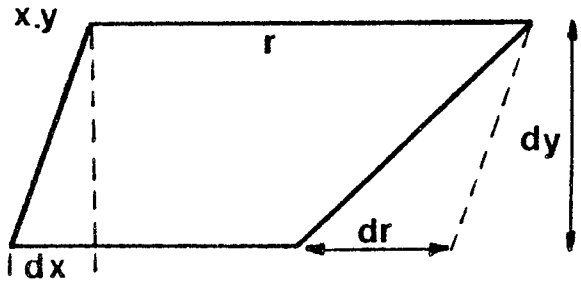


Figure VII.6: Représentation interne d'une région

* (y,dy) représentent l'ordonnée supérieure et la hauteur du trapèze,
* (x,dx) représentent l'abscisse supérieur et la différence d'abscisse de l'arête gauche,
* (r,dr) représentent la longueur de la base supérieure et la différence des bases.

Le traitement d'une région est effectué par un microprocesseur Motorola 6809 placé en série entre le calculateur pilote et le synthétiseur cablé. Ce processeur détermine à chaque ligne i l'abscisse gauche xi et le nombre de points à remplir ri, et initialise les registres "adresse" et "répétition" de la logique cablé (cf V.3.2.4). L'algorithme utilisé effectue une interpolation discrète classique sur x et sur r, ne nécessitant que l'addition et le test d'entiers. Une phase de préparation permet d'effectuer le traitement complet d'une ligne en 40 µs au maximum. L'algorithme d'interpolation sur les abscisses est donné ci-dessous. L'interpolation sur la largeur r de la région s'effectue de la même manière.

```
début (* cas où dx >= 0 *)
|   ix <- 0
|   si dy <> 0 alors
|   tant que dx >= dy faire
|   début
|   |     dx <- dx-dy
```

```
        ix <- ix+1
    fin
    registe-adresse <- (y,x)      (* première ligne *)
    h <- dy div 2
    pour i := y+1 jusqu'à y+dy faire
    début
        h <- h+dx
        x <- x+ix
        si h >= dy alors
        début
            h <- h - dy
            x <- x + 1
        fin
        registre-adresse <- (i,x)      (* lignes suivantes *)
    fin
fin
```

Dans le cas où dx est négatif il suffit de changer les "+1" en "-1".

Trois opérations s'effectuent en parallèle dans cette architecture pipe-line.

- 1) Le logiciel pilote détermine les régions et les communique au fur et à mesure au micro-processeur de remplissage.

- 2) Le micro-processeur détermine les paramètres de chaque ligne pour une région.

- 3) Le synthétiseur cablé effectue le remplissage d'une ligne à raison d'un point toutes les 125 ns. La figure 7 représente cette organisation.

Seul, le tri des arêtes doit impérativement être effectué avant tout le traitement sans aucune possibilité de parallélisme. Dans le cas d'une présentation dynamique, ce tri doit être effectué à chaque nouvelle vue pour chaque face, et pénalise les performances de l'ensemble. Les modifications

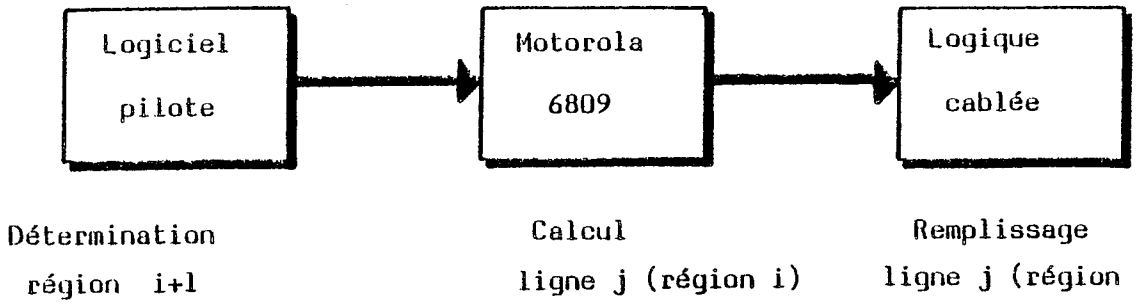


Figure VII.7: Pipe-line du remplissage

minimes d'une vue à l'autre suggèrent l'emploi d'un tri par remontée des bulles sur les résultats de la vue précédente.

1.2.2. Remplissage par inversions successives

Nous avons pour notre part proposé et mis en oeuvre une méthode permettant de supprimer le tri des arêtes et la détermination des régions (Mar79b). Cet algorithme travaille arête par arête de la manière suivante:

- numérisation et inscription de l'arête dans la mémoire de trame,
- inversion ("ou" exclusif) de la zone mémoire située à droite de cette arête, dans le rectangle englobant la face. La figure 8 présente cette méthode appliquée à un triangle.

Les singularités du contour sont traitées facilement en n'effectuant pas la phase d'inversion pour la dernière ligne de chaque arête.

Les deux principaux reproches que l'on puisse faire à cet algorithme concernent:

* la nécessité de disposer d'une mémoire booléenne de travail,
* le grand nombre de points traités inutilement.

Il faut en fait nuancer ces deux arguments. En premier lieu, il est possible de travailler directement dans la mémoire de trame du synthétiseur

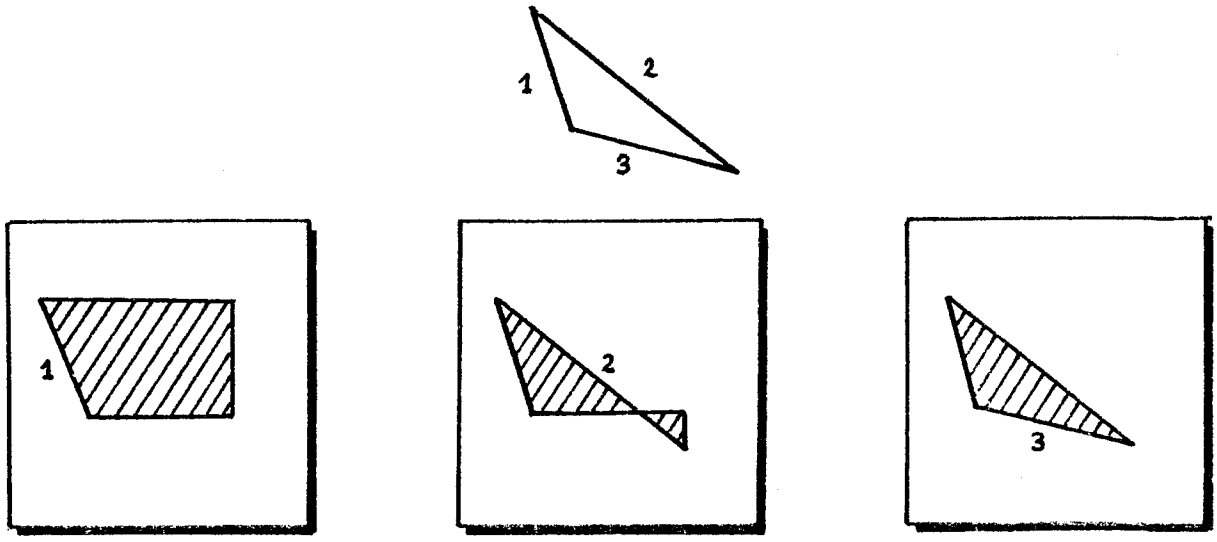


Figure VII.8: Remplissage par inversions successives

cablé, dans le cas où les faces ne se chevauchent pas (ce qui est généralement le cas après une élimination de parties cachées à travers un algorithme du style "Atherton et Weiler" (AtW77)). L'inversion s'effectue dans ce cas en opérant pour chaque point un "ou" exclusif entre le numéro précédemment inscrit dans la mémoire de trame et le numéro de la face courante.

Les changements visuels provoqués par chaque arête sur les faces précédemment affichées sont assez rapides pour ne pas être réellement gênants.

En ce qui concerne les points inutilement traités, on peut arguer qu'une réalisation cablée, pour laquelle l'opération d'inversion est quasi-instantanée, fait disparaître cet inconvénient. Dans le cas d'HELIOS l'inversion d'un point peut-être effectuée en 125 ns ce qui assure qu'une ligne complète soit traitée au rythme vidéo en 64 μs.

En conservant le même algotithme d'interpolation discrète sur le Motorola 6809, aucun traitement n'est plus nécessaire au niveau du logiciel pilote (si ce n'est la transmission des arêtes).

1.2.3. Comparaison des deux méthodes

L'évaluation théorique de ces deux méthodes semble extrèmement complexe à formuler, compte-tenu du grand nombre de facteurs rentrant en ligne de compte.

* morphologie de la tache,
* longueur du périmètre, de la hauteur, etc.,
* vitesse des processeurs utilisés,
* possibilité de parallélisme.

Nous baserons notre étude sur quelques comparaisons globales dans chacun des trois processeurs.

Au niveau du calculateur pilote

La seconde méthode ne nécessite que la transmission des arêtes, alors que la première requiert le tri de ces arêtes, la détermination des régions et leur transmission. On peut en outre, évaluer le nombre d'arêtes par rapport au nombre de régions nécessaires.

En règle générale si aucune arête n'est horizontale, un contour polygonal formé de n arêtes détermine n-1 régions, comme on peut le vérifier sur la figure 9 ci-dessous.

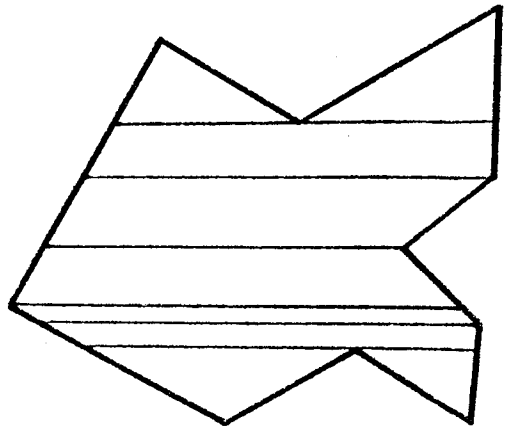


Figure VII.9: Relation entre les arêtes et les régions

S'il s'agit d'une tache "trouée", l'adjonction d'un trou de p arêtes crée p+1 régions (figure 10).

n = 10 p = 3

n+p = 13 arêtes

(n-1)+(p+1)=13 régions

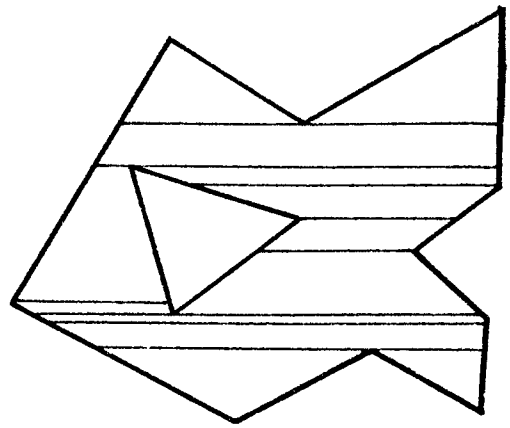


Figure VII.10: Cas des taches trouées

Nous admettrons donc, en moyenne, que le nombre de régions est comparable au nombre d'arêtes ce qui entraîne un très net avantage en faveur de la méthode des inversions successives, puisque seule la transmission des arêtes est effectuée.

Au niveau du processeur d'interpolation

Il est clair que le nombre d'interpolations nécessaires dans les deux cas est égal à la somme des hauteurs des arêtes du contour, d'où un temps de traitement global identique pour les deux méthodes. Cependant, au niveau d'une ligne, le traitement d'une région nécessite deux interpolations alors que celui d'une arête n'en requiert qu'un ce qui donne un temps réduit de moitié en faveur de la méthode des inversions. Etant donné que dans une architecture de type pipe-line, les performances de l'ensemble sont conditionnées par celles de l'étape la plus lente, il y a donc avantage, ici encore, en faveur de cette seconde méthode.

Au niveau du remplissage ou de l'inversion d'une ligne

Cette dernière phase, au contraire, joue au détriment de la technique des inversions successives pour deux raisons essentielles.

* le nombre de lignes à remplir dans la méthode des trapèzes est égal à la demi-somme des hauteurs des arêtes, alors que le nombre de lignes à inverser est égal à la somme de ces hauteurs.
* le nombre de points à inverser est toujours supérieur au nombre de points à remplir puisqu'il concerne tous les points jusqu'à la frontière droite du rectangle englobant.

En conclusion, il apparaît que la première méthode est limitée par la vitesse du logiciel, puis du processeur d'interpolation, alors que la seconde dépend presque exclusivement de la vitesse du matériel. Les essais que nous avons effectués ont confirmé cette hypothèse en donnant en moyenne sur un grand nombre de taches différentes, un avantage dans un rapport proche de 2 en faveur de la technique des inversions successives.

1.2.4. Le remplissage des faces gauches

Le terminal HELIOS peut être configuré pour permettre la visualisation de surfaces gauches (cf. VI.1.4.2). Dans ce type d'utilisation, les plans d'identification sont remplis, non pas avec le numéro de la face, mais avec la normale en chaque point. Il est clair que le calcul de cette normale par logiciel est coûteux et interdit l'utilisation directe des possibilités de compactage offertes par le matériel.

La technique utilisée permet d'effectuer une interpolation linéaire de la normale, (exprimée en coordonnées sphériques) sur une face polygonale dont la normale en chacun des sommets est connue. Le calcul s'insère dans la méthode de remplissage par décomposition en régions élémentaires, présentée ci-dessus.

Trois interpolations distinctes sont nécessaires:

- La première est effectuée par le logiciel pilote lors de la détermination des régions pour obtenir la normale en chacun des sommets du trapèze ou du triangle.

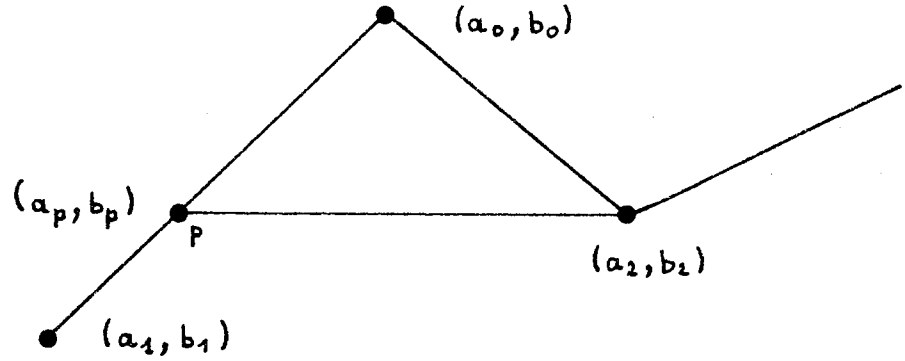
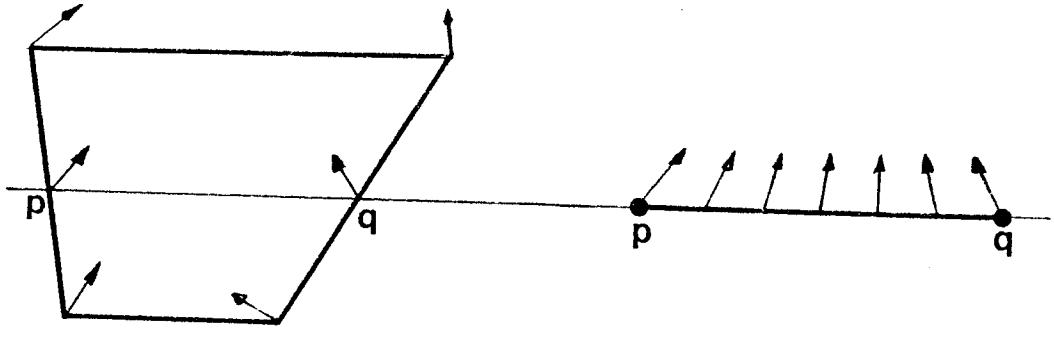


Figure VII.11: Remplissage des faces gauches

Dans l'exemple de la figure 11, la normale (ap,bp) au point p s'obtient par:

```
ap = (1-e) * a0 + e * a1
bp = (1-e) * b0 + e * b1
avec e = (y2-y0) / (y1-y0)
```

- La seconde interpolation, confiée au micro-processeur de remplissage, est, en tout point, analogue à celle présentée au paragraphe précédent et détermine la normale sur chaque ligne entre l'arête gauche et l'arête droite (figure 12a).



Première interpolation Deuxième interpolation

Figure VII.12: Interpolations de la normale

- La troisième interpolation s'effectue horizontalement sur la ligne courante. Bien qu'une réalisation cablée soit envisageable, nous avons, dans une première approche, effectué cette opération sur le micro-processeur de remplissage (figure 12b).

Malgré des performances moins bonnes que dans le cas des faces planes, cette technique assure des résultats visuels de bonne qualité, comme en témoigne la photographie de la page.

1.3. L'élimination des parties cachées

1.3.1. Une nouvelle approche

Une étude critique de différents algorithmes publiés, effectuée par Philippe Boulle (Bou80), nous permet de disposer à l'heure actuelle de quatre algorithmes d'élimination de surfaces cachées directement inspirés des travaux de:

* J.E. Warnock (War69),
* G.S. Watkins (Wat70),
* M.E. Newell, R.G. Newell, T.L. Sancha (NNS72),
* Atherton, Weiler (AtW77).

Ces algorithmes ont été améliorés pour atteindre une meilleure efficacité, et permettre la production de dessins au trait. Nous avons vu que le choix d'un algorithme peut conduire à modifier considérablement le processus de visualisation (cf. III.5). Ainsi dans le cas du terminal HELIOS, l'algorithme de Watkins, conduit à effectuer le "remplissage" des faces au cours de l'élimination elle-même, ce qui ne permet pas de profiter pleinement des facilités offertes par le logiciel pilote, le processeur de remplissage et le matériel.

Nous avons choisi de développer un algorithme directement adapté au processus utilisé, combinant:

- le découpage des faces proposé par Newell, Newell et Sancha (NNS72),

- la séparation entre priorités statiques et dynamiques, préconisée par Schumacker (SBG69).

- une technique d'ordonnancement des faces, nouvelle et originale, permettant d'accélérer le traitement, en tenant compte dans une certaine mesure de la structure et de la "complexité" de la maquette.

Les paragraphes qui suivent présentent la méthode proposée. Nous terminons en situant cette nouvelle proposition par rapport à l'algorithme initial de Schumacker, et aux diverses améliorations qui lui ont été apportées au cours des dernières années.

1.3.2. Les coefficients statiques

Cette opération prend normalement place au moment de la construction de la maquette. Les résultats obtenus sont valables pour toute la durée d'existence de celle-ci. Le but de ce prétraitement est d'associer à chacune des faces de la maquette un ensemble de coefficients qui permettront, lorsque le point de vue sera connu, de minimiser les calculs nécessaires pour l'ordonnancement des faces. Celles-ci sont ensuite affichées dans l'ordre obtenu, de la plus lointaine à la plus proche, comme le suggère Newell.

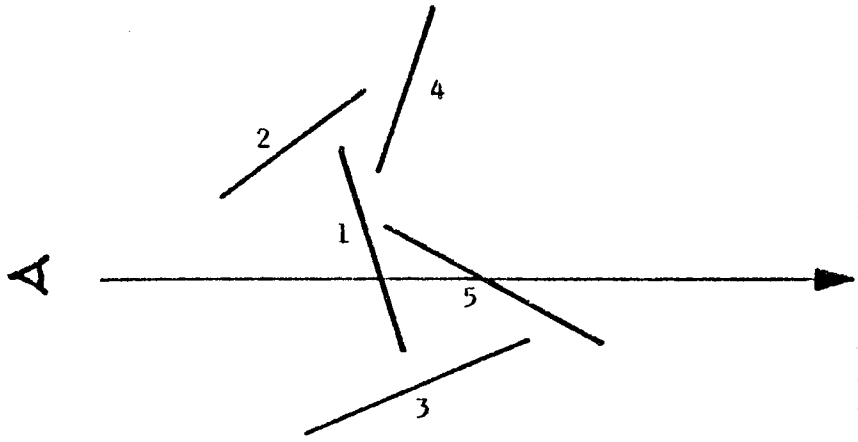
Afin de faciliter l'exposé, nous présenterons la méthode sur un exemple bi-dimensionnel. En outre, nous considèrerons dans un premier temps qu'aucune face de la maquette n'en coupe une autre. Ce cas sera étudié au paragraphe suivant.

Le but final recherché par cette méthode est le classement de toutes les faces de la maquette en fonction d'une direction de visée $\vec{V}$ issue du point de vue. Si les faces ne se coupent pas, le classement de deux faces Fi et Fj est toujours possible à travers la relation d'ordre définie ci dessous:

Définition

Nous dirons qu'une face Fi est devant une face Fj (noté Fi < Fj) si tous les points de Fi sont situés dans le demi-espace de Fj "négatif" par rapport à la direction V, ou si tous les points de Fj sont dans le demi-espace "positif" de Fi. Nous dirons également dans ce cas que Fj est derrière Fi.

Dans l'exemple de la figure 13, toutes les faces peuvent être classées deux à deux mais il reste cependant à les ordonner globalement de façon à les afficher de la plus lointaine à la plus proche. A l'opposé de la méthode de tri proposée par Newell et al mettant en jeu une gestion délicate des faces à travers un algorithme assez complexe, la méthode que nous proposons ci-après permet d'effectuer cet ordonnancement d'une manière plus systématique et débouche sur la possibilité très importante de pré-calculer, une fois pour toutes, l'ensemble des relations statiques de la maquette.



Une matrice possible

	1	2	3	4	5
1		<	>	>	>
2	>		>	>	>
3	<	<		<	>
4	<	<	>		>
5	<	<	<	<	

Figure VII.13: Exemple de maquette et relations associées

Si l'on associe à chaque face Fj de la figure 13 un coefficient statique S(j), relatif à la direction V, tel que:

$$S(j) = \text{nb de faces derriere } Fj - \text{nb de faces devant } Fj$$

on obtient:

$$S(1) = 2$$
$$S(2) = 4$$
$$S(3) = -2$$
$$S(4) = 0$$
$$S(5) = -4$$

on en déduit par conséquent l'ordre ci-dessous:

$$F5 < F3 < F4 < F1 < F2$$

Dans le cas général, pour n faces, l'ensemble des coefficients statiques est:

$$C = (n-1,\ n-3,\ n-5,\ n-7,\ \ldots,\ 7-n,\ 5-n,\ 3-n,\ 1-n)$$

On peut d'ores et déjà remarquer que:

- 1 $\sum_j S(j) = 0$
- 2 Il existe une bijection entre C et E = (0, 1, ..., n-1) définie par:

$$\forall\ Ej \in E,\ Ej = 1/2(n-1 + S(j))$$

bijection qui permet d'exprimer l'ordre des faces.

En effet, lorsque toutes les faces ont été classées, celle dont le coefficient statique est le plus faible est située derrière toutes les autres. C'est donc celle qui doit être affichée en premier. La suivante est celle qui ne comporte qu'une seule face derrière elle (la précédente) et ainsi de suite. La dernière face à afficher comporte le coefficient statique le plus élevé (n-1); elle est située devant toutes les autres.

Deux inconvénients importants pénalisent pourtant cette approche:

- 1) Si le point de vue change, il est nécessaire de recalculer entièrement toutes les relations entre les faces ainsi que les nouveaux coefficients statiques.

- 2) Il est possible que l'on ait à la fois Fi < Fj et Fj < Fi (faces F4 et F5 de la figure 13). Le choix entre ces deux résultats découle alors de la transitivité de la relation d'ordre. Dans l'exemple ci-dessus on a:

$$F4 < F3 \text{ et } F3 < F5 \Longrightarrow F4 < F5$$

Ces deux inconvénients peuvent être totalement supprimés si l'on utilise une nouvelle relation d'ordre "moins fine", définie comme suit:

Définition

Dans une direction donnée U, nous dirons qu'une face Fi est devant une face Fj (noté Fi < Fj) si:

$$\| \quad umax(Fi) <= umin(Fj)$$

où umax et umin représentent respectivement les coordonnées maximales et minimales de chaque face par rapport à la direction $\vec{U}$.

Cette définition implique qu'il existe un plan perpendiculaire à $\vec{U}$, séparant Fi et Fj. On notera que si umax(Fi) = umin(Fj), le plan séparateur passe nécessairement par ces deux points qui peuvent être confondus si les faces sont en contact (figure 14).

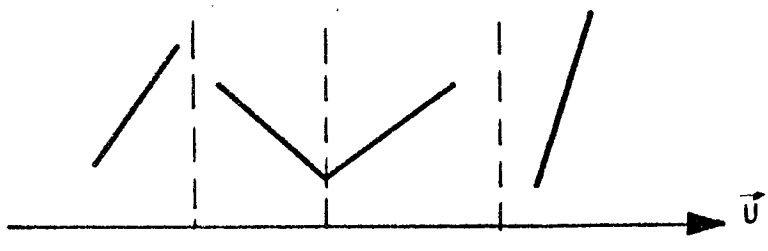


Figure VII.14: Les plans séparateurs

On peut vérifier que les deux inconvénients précédents disparaissent.

- 1) Le classement reste valable pour toutes les directions de visée $\vec{V}$ telles que $\vec{U}.\vec{V} >= 0$ et dans le cas où $\vec{U}.\vec{V} < 0$ il suffit d'inverser l'ordre obtenu.

- 2) Les ambiguités de classement ne sont plus possibles puisqu'on ne peut avoir simultanément Fi<Fj et Fj<Fi.

En revanche, cette relation est insuffisante et de nombreux couples de faces ne peuvent être classés. Ainsi dans le cas de la figure 15, les faces F5 et F2 ne peuvent être classées l'une par rapport à l'autre dans la direction $\vec{U1}$.

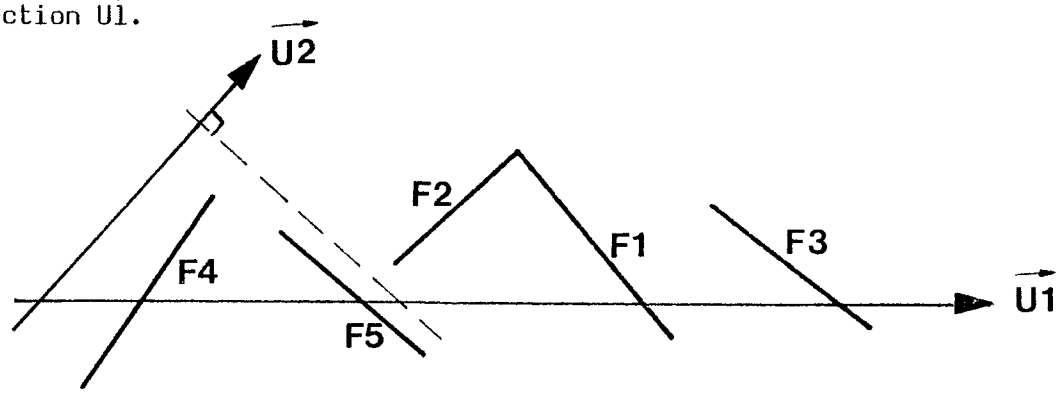


Figure VII.15: Utilisation de deux directions d'étude

Le calcul des coefficients statiques S(1,j) relatifs à U1 donne:

S(1,1) = -2
S(1,2) = 1 (on ne peut rien dire pour F5)
S(1,3) = -4
S(1,4) = 4
S(1,5) = -1 (on ne peut rien dire pour F2)

La définition du coefficient statique doit alors être modifiée comme suit:

S(k,j) = nb de faces <u>classables</u> situées <u>derrière</u> Fj selon Uk

- nb de faces <u>classables</u> situées <u>devant</u> Fj selon Uk

La propriété 1, évoquée plus haut, reste valable tandis que la 2 n'est plus vérifiée. Afin de résoudre les relations entre F5 et F2, on considère une nouvelle direction d'étude $\vec{U2}$, représentée sur la figure 16, on peut écrire:

$$F5 < F2$$

Si l'on ne considère que ces deux faces (les faces F1, F3 et F4 étant entièrement classées selon $\vec{U1}$), les coefficients statiques selon $\vec{U2}$ sont:

$$S(2,1) = 0$$
$$S(2,2) = -1$$
$$S(2,3) = 0$$
$$S(2,4) = 0$$
$$S(2,5) = 1$$

En regroupant les coefficients relatifs aux deux directions, on obtient une matrice de coefficients statiques comportant autant de lignes que de faces et autant de colonnes que de directions d'étude. Dans le cas présent:

$$S = \begin{vmatrix} -2 & 0 \\ 1 & -1 \\ -4 & 0 \\ 4 & 0 \\ -1 & 1 \end{vmatrix}$$

On remarque que si l'on effectue la somme ou la différence des colonnes de cette matrice, les propriétés 1 et 2 sont à nouveau vérifiées sur le vecteur résultant, qui permettra donc d'ordonner l'ensemble des faces.

Cas général

Nous avons vu que dans le cas où les n faces d'une maquette peuvent être classées selon une direction unique $\overrightarrow{U1}$ (cas d'école), les coefficients statiques dans cette direction sont:

$$(n-1,\ n-3,\ n-5,\ \ldots,\ 5-n,\ 3-n,\ 1-n)$$

Si quelques faces ne peuvent être classées mutuellement (par exemple de la (p+1)ème à la (q)ème dans l'ordre issu de $\overrightarrow{U1}$), on obtient pour celles-ci des cofficients égaux, puisque le nombre de faces classables situées devant et derrière est identique pour chacune d'elles. On a:

$(n-1,\ n-3,\ \ldots,\ n-2p+1,$	coefficients pour les p premières faces
$n-(p+q),\ \ldots,\ n-(p+q),$	coefficients égaux pour les faces p+1 à q
$n-2q-1,\ \ldots,\ 3-n,\ 1-n)$	coefficients pour les n-q dernières faces

S'il existe une seconde direction d'étude permettant de classer les q-p faces non classées selon $\overrightarrow{U1}$, les coefficients statiques selon $\overrightarrow{U2}$ sont:

$$(\ (q-p)-1,\ (q-p)-3,\ \ldots,\ 3-(q-p),\ 1-(q-p)\ )$$

pour ces faces, et nuls (par définition) pour les autres qui ne participent pas au classement.

Si l'on ajoute le coefficient unique de toutes ces faces selon $\overrightarrow{U1}$ (c'est-à-dire n-(p+q)), on obtient:

$$(n-2p-1,\ n-2p-3,\ n-2p-5,\ \ldots,\ n-2q+5,\ n-2q+3,\ n-2q+1\ )$$

ce qui restitue pour l'ensemble des n faces la séquence initiale:

$$(\ n-1,\ n-3,\ n-5,\ \ldots,\ 5-n,\ 3-n,\ 1-n\ )$$

On peut ainsi énoncer deux autres propriétés fondamentales de la matrice des coefficients statiques:

- 3] La somme ou la différence des colonnes de la matrice est un vecteur de n composantes comprenant toutes les valeurs de l'ensemble:
  (n-1,n-3, n-5, ..., 5-n, 3-n, 1-n)

- 4] Si l'on inverse le signe d'une des colonnes, la propriété 3] reste vraie du fait de la symétrie des valeurs de chaque colonne par rapport à zéro. Nous examinerons au paragraphe 3.3.5, la façon dont la matrice des coefficients statiques peut être obtenue. On peut noter, d'ores et déjà, que l'obtention automatique d'une matrice "quelconque" ne pose aucun problème, mais qu'il est beaucoup plus délicat d'obtenir une matrice optimale comportant le minimum de directions d'étude.

1.3.3. L'ordonnancement des faces

Après avoir obtenu la matrice des coefficients statiques, il reste à déterminer pour chaque point de vue, l'ordre dans lequel les faces doivent être affichées. Cette méthode permet d'obtenir directement, par un calcul simple, le rang de chacune des faces, en fonction du point de vue.

Ce calcul s'opère en deux temps:

- expression des coefficients directeurs (uj) associés aux directions d'étude utilisées: les vecteurs $\vec{Uj}$,

- multiplication par la matrice des priorités statiques.

Soit $\vec{V}$ le vecteur issu du point de vue dans la direction de l'origine du repère. On obtient les coefficients directeurs uj à l'aide des expressions ci-dessous.

$$uj = -1 \text{ si } \vec{Uj}.\vec{V} >= 0$$
$$uj = 1 \text{ si } \vec{Uj}.\vec{V} < 0$$

Cette expression a pour but d'exprimer si la direction de visée est opposée ou non à chacune des directions d'étude. Dans le cas où $\vec{Uj}.\vec{V} = 0$ (directions perpendiculaires), le résultat est indifférent.

Le rang Ri de la face Fi s'obtient alors directement à l'aide de l'expression suivante:

$$Ri = 1/2 \left(n-1 + \sum_{j=1}^{p} uj \cdot S(i,j) \right) \qquad (1)$$

Si l'on fait la synthèse des propriétés [2], [3], [4] exposées ci-dessus on peut vérifier que les valeurs Ri définissent bien une bijection de l'ensemble des faces dans l'intervalle $(0, \ldots, n-1) \subset N$. L'ordre des faces est ainsi déterminé sans qu'aucun tri ne soit nécessaire.

Du point de vue pratique l'expression (1) est intéressante, sous plusieurs aspects:

- tous les termes sont des entiers ce qui rend la réalisation aisée,
- elle ne nécessite que des additions ou des soustractions selon le signe de uj ,
- la division par 2 se traduit par un simple décalage, le terme à diviser étant toujours un entier pair,
- lorsque le point de vue change, les nouveaux rangs peuvent être obtenus directement à partir des anciens avec un minimum de calcul.

Si le nouveau point de vue diffère du précédent pour une direction d'étude $\vec{Uk}$, le nouveau coefficient directeur u'k dans cette direction est:

$$u'k = -uk$$

On peut déduire immédiatement, pour chaque face Fi, le nouveau rang R'i en fonction du précédent Ri. Il vient:

$$R'i = Ri + u'k.S(i,k) \qquad (2)$$

Il suffit donc d'ajouter S(i,k) si le nouveau coefficient directeur est positif ou de retrancher S(i,k) s'il est négatif.

Par extension si le nouveau point de vue diffère du précédent pour un ensemble $\underline{K}$ de directions d'étude (celles pour lesquelles les coefficients directeurs changent de signe), on obtient le nouveau rang R'i, à l'aide de l'expression ci-dessous:

$$R'i = Ri + \sum_{j \in K} uj.S(i,j) \qquad (3)$$

On peut noter en outre la symétrie qui existe dans le cas où tous les coefficients directeurs uj diffèrent (pt de vue opposé par rapport à l'origine), on obtient:

$$R'i = (n-1) - Ri \qquad (4)$$

opération qui peut être évitée en parcourant la liste des faces à l'envers.

En utilisant judicieusement ces propriétés, le passage d'une situation à une autre ne nécessite que (p div 2) additions par face, dans le cas le plus défavorable.

Exemple:

Nous illustrerons cette méthode par l'exemple simple représenté sur la figure 16, ci-dessous.

En utilisant les trois directions d'études $\vec{X}$, $\vec{Y}$, $\vec{Z}$ constituées par les axes du repère, une matrice possible des coefficients statiques est donnée ci-dessous:

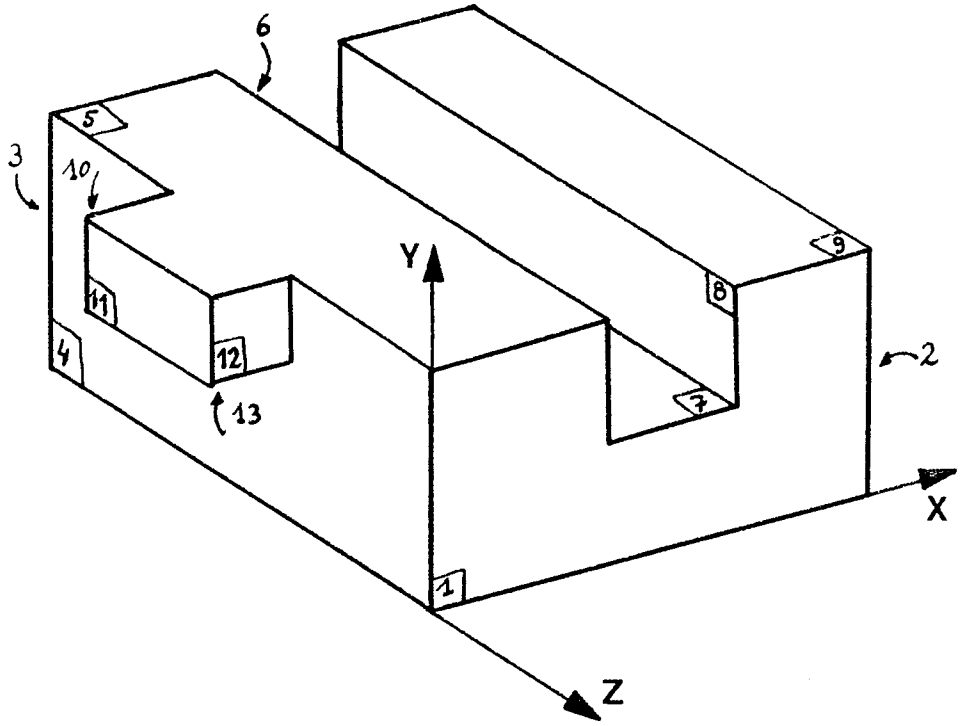


Figure VII.16: Exemple d'application: solide à 13 faces

$$S = \begin{vmatrix} 0 & 0 & -12 \\ -11 & 0 & 1 \\ 1 & 0 & 11 \\ 1 & 1 & 0 \\ 2 & -8 & 0 \\ -1 & 1 & 0 \\ -3 & 1 & 0 \\ -5 & 1 & 0 \\ -8 & 0 & 0 \\ 6 & 1 & 3 \\ 7 & 1 & 0 \\ 6 & 1 & -3 \\ 5 & 1 & 0 \end{vmatrix}$$

$\vec{X}$ $\vec{Y}$ $\vec{Z}$

On peut d'ores et déjà réorganiser les faces en calculant leur rang pour une situation donnée, par exemple lorsque les trois coefficients directeurs sont négatifs:

$$x = y = z = -1$$

On obtient la liste suivante

numéro de face	3	10	11	13	12	4	6	7	8	5	9	2	1
coefficient	-12	-10	-8	-6	-4	-2	0	2	4	6	8	10	12
rang	0	1	2	3	4	5	6	7	8	9	10	11	12
	sens d'affichage des faces ->												

Les numéros des faces sont insérés directement dans un tableau au fur et à mesure du calcul du rang.

Soit à déterminer la nouvelle liste pour le point de vue
P1 = (-3,-4,12)
Le vecteur $\vec{V}$ est égal à (3,4,-12) et on obtient les coefficients directeurs: x=-1, y=-1, z=1.
Seule la composante en z change, ce qui entraîne des modifications pour les seules faces 1,2,3,10,12 pour lesquelles les coefficients de la matrice ne sont pas nuls. Seules ces faces sont traitées et permutent dans le tableau. En appliquant l'expression (3), le nouveau rang de ces faces s'obtient en additionnant au rang précédent (puisque le coefficient directeur est positif), les coefficients statiques relatifs à $\vec{Z}$. On obtient:
rang de la face 1 : 12 - 12 = 0
rang de la face 2 : 11 + 1 = 12
rang de la face 3 : 0 + 11 = 11
rang de la face 10: 1 + 3 = 4
rang de la face 12: 4 - 3 = 1

numéro de face	1	12	11	13	10	4	6	7	8	5	9	3	2
rang	0	1	2	3	4	5	6	7	8	9	10	11	12
	sens d'affichage -->												

Pour le point de vue P2 = (7,-8,-6) on a les coefficients directeurs: x=1, y=-1, z=-1.

Au lieu d'inverser x et z, par rapport à la situation précédente, on choisit d'inverser y pour passer dans la situation :
x = -1, y = 1, z = 1
Comme précédemment le rang s'obtient en additionnant les coefficients statiques relatifs à $\vec{Y}$. Puis on passe dans la situation x=1, y=-1, z=-1 en inversant simplement l'ordre de parcours de la liste des faces.

numéro de face	1	5	12	11	13	10	4	6	7	8	9	3	2
rang	0	1	2	3	4	5	6	7	8	9	10	11	12
	sens d'affichage <--												

La figure 17 illustre quelques vues de cet objet obtenues à l'aide de cette méthode.

1.3.4. Exploitation de la structure arborescente

L'une des principales caractéristiques de cette méthode est sa faculté d'adaptation, à la "complexité" de la maquette, caractérisée ici, par le nombre de directions d'étude nécessaires. Cette adaptation est encore accrue par l'exploitation de la structure arborescente de la maquette.

S'il est possible de trouver des plans séparateurs entre les différents objets polyédriques de la maquette, on peut leur appliquer globalement le même traitement que pour les faces.

Ainsi, pour les trois objets A, B, C de la figure 18, composés respectivement de q, r et s faces, on obtient dans la direction d'étude $\vec{U1}$ les coefficients statiques ci-dessous:

$\forall$ Fj $\in$ A, S(1,j) = r+s
$\forall$ Fj $\in$ B, S(1,j) = s-q
$\forall$ Fj $\in$ C, S(1,j) = -q-r

Si l'on considère une face Fi appartenant à l'objet B, son rang s'écrit comme précédemment:

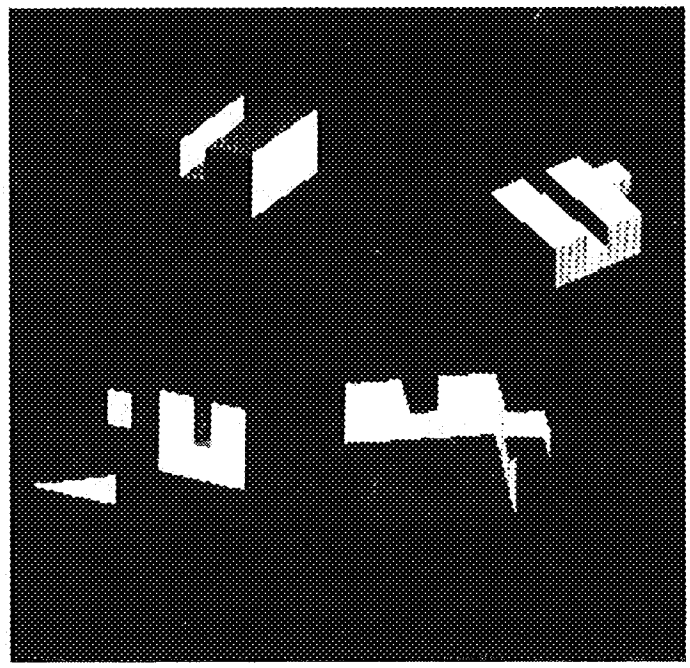


Figure VII.17: Quelques vues du solide de la figure 16

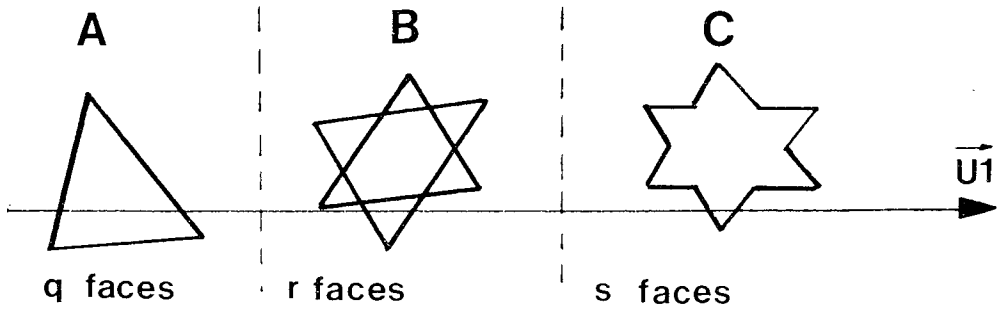


Figure VII.18: Classement de trois objets dans une direction

$$\| \qquad Ri = 1/2 \ (q+r+s-1 + \sum_{j=1}^{p} uj.S(i,j) \ )$$

puisque q+r+s = n.

L'expression ci-dessus peut être transformée de façon à faire apparaître

deux termes

$$Ri = 1/2\ (q-1 + \sum_{j=1}^{p} uj.S(i,j)) + 1/2\ (q+s + ul.S(1,j)$$

On reconnaît dans le premier terme, le rang de la face Fi à l'intérieur de l'objet B. Le second terme (constant puisque $\forall$ Fj $\in$ B , S(1,j) = s-q) exprime le rang de l'objet B dans la maquette.

On remarque que ce rang est égal à:

q si ul = -1

s si ul = 1

ce qui correspond au rang de la première face de l'objet B dans ces deux cas (figure 18). Ce raisonnement peut être étendu à un nombre quelconque de directions d'étude pour le classement global des objets entre eux.

Une solution particulièrement intéressante consiste alors à calculer le rang de chaque face de manière récursive lors du parcours de la structure arborescente. Il suffit pour celà, lors de la construction de l'arbre, d'affecter à chaque noeud i, les informations géométriques statiques ci-dessous:

- les vecteurs des directions d'études nécessaires pour évaluer les descendants du noeud i considéré;
- la matrice des coefficients statiques pour l'objet ou la face correspondant au noeud i;
- le rang initial, par exemple dans la situation où tous les coefficients directeurs sont égaux à -1.

Si l'on note $\hat{i}$ le noeud ascendant de i, le nouveau rang R'i correspondant à un changement de point de vue, s'écrit:

$$R'i = R'\hat{i} + Ri + \sum_{j \in K} uj.S(i,j)$$

Le rang de la racine est considéré comme étant nul.

Exemple

Considérons l'exemple représenté par la figure 19 ci-après, dans lequel les objets A et B sont tous deux étudiés par rapport aux deux directions $\vec{X}$ et $\vec{Y}$ et sont séparés par une troisième direction $\vec{Z}$.

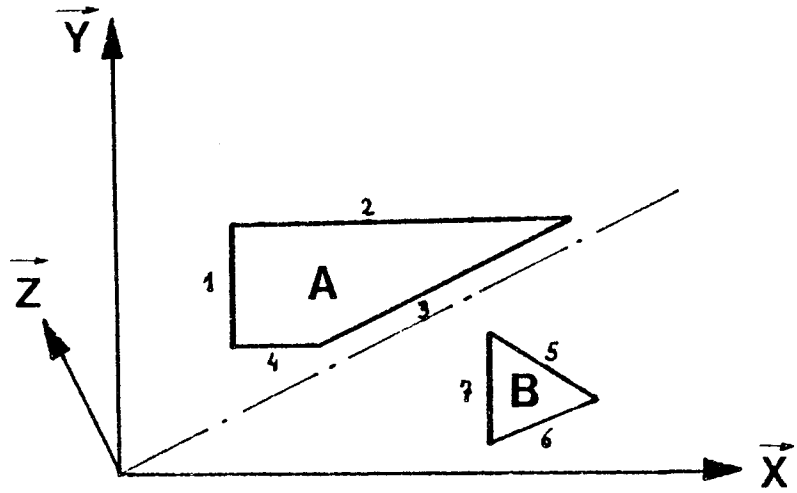


Figure VII.19: Exemple de deux objets

Les matrices de coefficients statiques associées respectivement à A et B sont:

$$Sa = \begin{vmatrix} 3 & 0 \\ -1 & -2 \\ -1 & 0 \\ -1 & 2 \end{vmatrix} \qquad Sb = \begin{vmatrix} -1 & -1 \\ -1 & -1 \\ 2 & 0 \end{vmatrix}$$

$\vec{X}$ $\vec{Y}$ $\qquad$ $\vec{X}$ $\vec{Y}$

La matrice permettant de classer globalement A et B est:

$$S = \begin{vmatrix} -3 \\ \\ 4 \end{vmatrix} \quad \begin{matrix} A \\ \\ B \end{matrix}$$

$\vec{Z}$

L'arbre peut alors être initialisé comme l'indique la figure 20 ci-après.

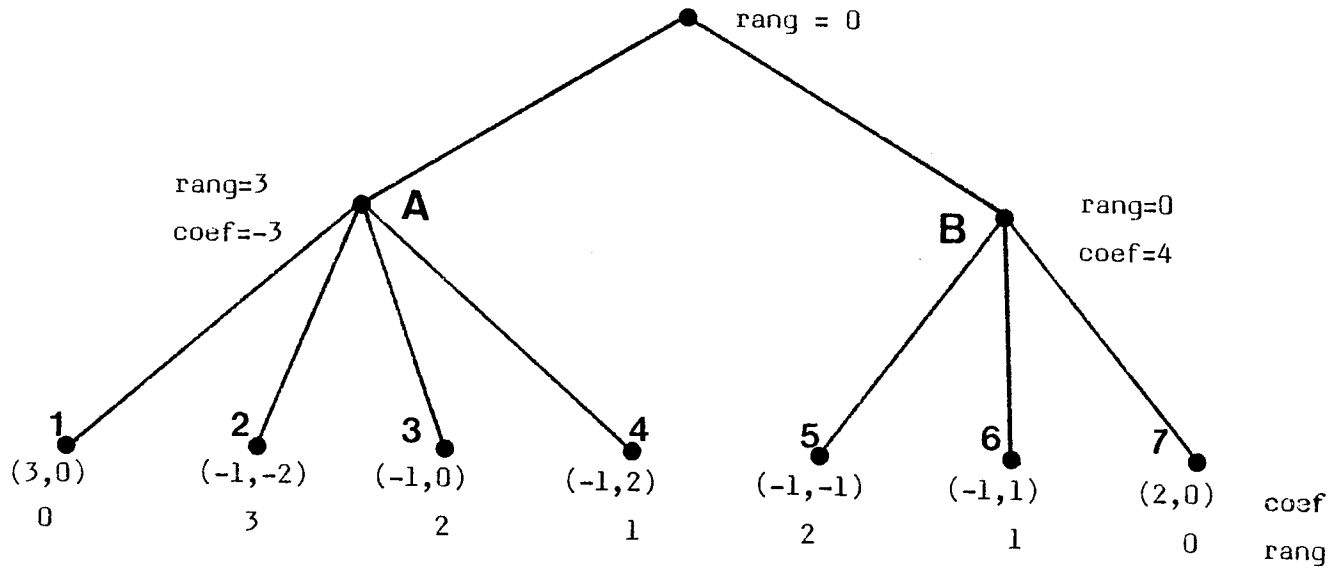


Figure VII.20: Initialisation de l'arbre

Outre le gain de temps apporté par cette technique récursive, on peut noter le gain de place qui en découle puisque 16 coefficients statiques seulement sont mémorisés, là où 21 auraient été nécessaires pour caractériser 7 faces selon 3 directions d'étude.

1.3.5. Détermination des directions d'étude et découpage des faces

Nous avons supposé dans la présentation ci-dessus qu'il existait toujours un plan séparateur entre deux faces quelconques et que ce plan pouvait toujours être déterminé (ou sa normale qui constitue la direction d'étude). Nous examinerons ici les problèmes posés par cette détermination.

L'existence d'un plan séparateur entre deux faces Fi et Fj est assurée si le plan support de l'une des deux faces ne coupe pas l'autre face. Ce plan peut dans ce cas faire office de plan séparateur (voir figure 21.a).

Dans le cas contraire (figure 21.b), chaque plan support coupe l'autre

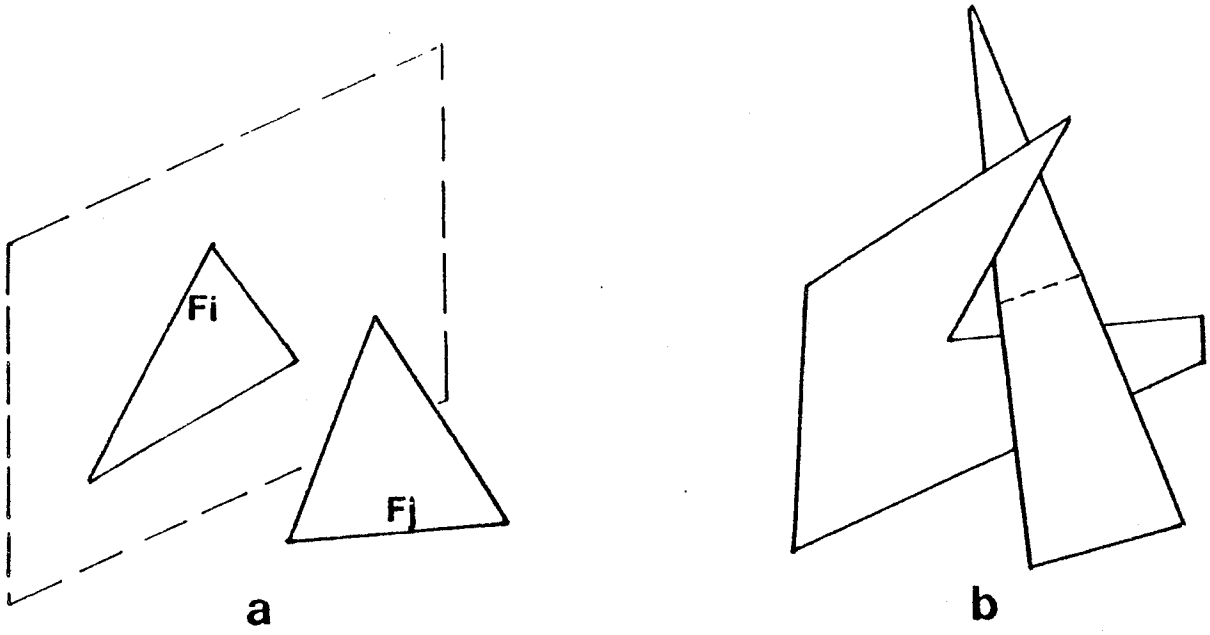


Figure VII.21: Découpage des faces

face et il est nécessaire de découper l'une d'elles selon le plan de l'autre en utilisant, par exemple, la technique proposée par Newell et al. (NNS72).

Lorsque les faces ont été découpées, une fois pour toutes dans la maquette, l'étape suivante consiste à déterminer les directions d'étude nécessaires. La détermination automatique de ces directions est un problème délicat puisque le nombre de directions conditionne grandement les performances de la méthode (occupation mémoire et temps de calcul).

Il va de soi que dans certaines maquettes, des directions d'étude apparaissent nettement privilégiées. Ainsi dans l'exemple de la figure 16 les directions $\vec{X}$, $\vec{Y}$, $\vec{Z}$ sont évidemment les plus appropriées. Dans des cas aussi simples, somme toutes relativement fréquents, les directions d'étude peuvent être détectées en partitionnant l'ensemble des faces en fonction de leur normale. Les faces sont ensuites examinées deux à deux selon la direction normale au plus grand nombre de faces. En cas d'échec (pas de plan séparateur perpendiculaire à cette direction), on essaie successivement les autres directions dans l'ordre décroissant des occurences dans la maquette.

Cette méthode, qui est celle que nous avons utilisée pour les exemples présentés dans cette thèse, permet de limiter, dans la plupart des cas, le nombre de directions mais ne met pas à profit les propriétés géométriques de

la maquette. Nous avons, pour notre part, laissé à l'opérateur la possibilité d'indiquer manuellement des directions qu'il juge intéressantes pour exploiter les symétries, la structure ou la convexité des objets présents. Il semble qu'il y ait là un axe important de recherche, relatif à la détermination automatique et optimale des directions d'étude.

Afin de clarifier les idées nous énoncerons, sans les démontrer, quelques constatations qui pourraient servir de point de départ à des études plus formelles et plus exhaustives, visant à diminuer considérablement, voire même à supprimer, la contribution manuelle.

- 1) Tout objet engendré par révolution selon une trajectoire convexe autour d'un axe, peut être caractérisé par trois directions seulement, dont l'une est l'axe de révolution et les deux autres sont situées dans un plan perpendiculaire à la première. Ce cas englobe naturellement les polyèdres convexes usuels: parallélépipèdes, pyramides, cylindres et cônes approchés par des faces planes, etc.

- 2) Les directions optimales peuvent être totalement indépendantes de l'orientation des faces, comme l'illustre la figure 22 dans laquelle l'objet représenté peut parfaitement être étudié selon $\vec{U1}$ et $\vec{U2}$.

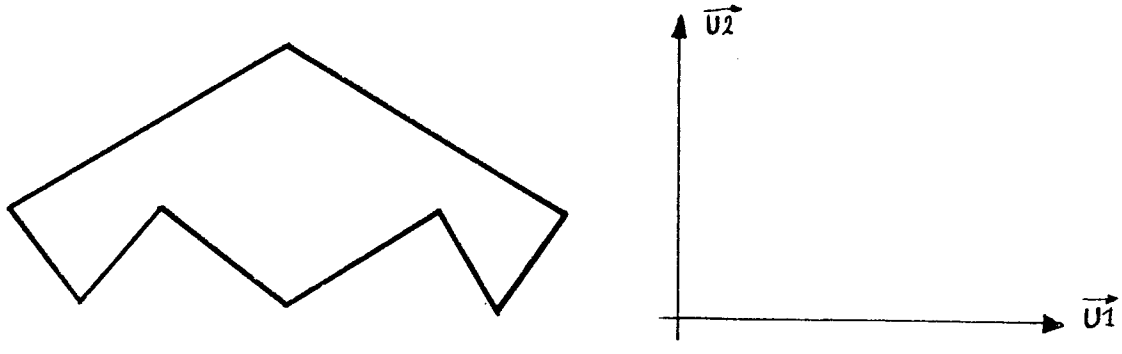


Figure VII.22: Indépendance des directions d'étude

- 3) La tolérance de quelques "erreurs" de classement peut diminuer le nombre de directions nécessaires sans aucun effet néfaste sur les résultats. La figure 23 reprend l'exemple précédent légèrement modifié pour lequel les directions $\vec{U1}$ et $\vec{U2}$ donnent encore de bons résultats bien

que les couples de faces (1,4) et (1,5) ne puissent être classés.

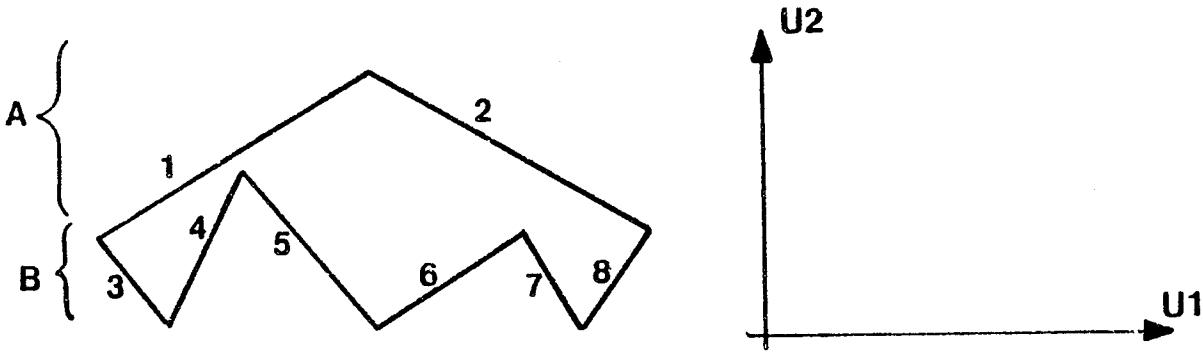


Figure VII.23: Tolérance des 'erreurs' de classement

Il suffit pour celà de considérer que l'objet est composé de deux sous-objets:

* le premier (A) comprenant les faces 1 et 2, classés selon $\overrightarrow{U1}$;
* le second (B) comprenant les faces 3,4,5,6,7 et 8, classées selon $\overrightarrow{U1}$

Les deux objets sont alors classés globalement selon $\overrightarrow{U2}$ (B < A), ce qui est inexact selon la définition mais n'altère pas les résultats.

Ces quelques remarques permettent d'apprécier les difficultés relatives à l'obtention entièrement automatique des directions d'étude et l'intérêt qu'il y a à profiter des connaissances de l'opérateur humain.

En règle générale, il est possible et même souhaitable de déterminer les directions d'étude au fur et à mesure des opérations de description et de construction et de les composer directement en fonction de la structure de la maquette. On peut ainsi prendre en compte immédiatement des informations relatives à la situation des objets ou à leurs symétries, avant qu'elles ne disparaissent au cours d'opérations effectuant l'union, l'intersection, ou la fusion de plusieurs objets.

1.3.6. Etude critique

Nous tenterons dans ce paragraphe de situer cette approche par rapport aux travaux de Schumacker (SBG69), et de Fuchs (FKN80), et d'indiquer les limitations et les possibilités qu'elle implique.

En premier lieu, les notions de priorités statiques et dynamiques introduites par Schumacker sont assez éloignées de celles présentées ci-dessus, sous plusieurs aspects.

Dans le cas de Schumacker:

* Les faces sont nécessairement convexes (sans trou)
* Les objets sont des solides totalement fermés et opaques, ce qui revient à dire que les faces arrières ne sont jamais visibles.

Moyennant ces restrictions, Schumacker calcule les priorités statiques pour chaque face à l'intérieur de chaque objet, indépendamment du point de vue et de toute direction d'étude. Le calcul des priorités dynamiques entre les divers objets est effectué en sélectant parmi tous les ordres possibles, préalablement calculés, celui qui correspond au point de vue donné. Si p plans séparateurs sont nécessaires pour distinguer les objets, la sélection s'effectue en parcourant un arbre contenant les 2^p listes ordonnées d'objets possibles. L'ensemble des faces peut alors être trié, puis communiqué au matériel.

Alors que, dans le système proposé par Schumacker, le calcul de l'arbre, est presque entièrement manuel, Fuchs (FKN80) propose, une technique permettant d'obtenir cet arbre automatiquement, et de l'utiliser pour calculer récursivement les priorités dynamiques. Néanmoins, trois inconvénients accompagnent cette approche.

- Elle conduit à un découpage excessif des objets, par rapport aux plans des faces; opération longue et coûteuse,

- Elle ne s'applique qu'aux faces convexes,

- Le calcul des priorités dynamiques n'exploite pas la cohérence existant entre deux images successives, car l'arbre doit être re-parcouru tout entier, à chaque fois, même si un seul plan a été traversé par le point de vue.

Face à ces deux algorithmes, la méthode que nous proposons, apporte des améliorations appréciables.

- les faces peuvent être quelconques et même trouées,

- les objets peuvent être transparents ou privés de certaines faces,

- aucun tri n'est nécessaire, le rang étant obtenu directement par un calcul simple,

- la structure même de la maquette est directement exploitée (et non un arbre fictif représentant des régions de l'espace),

- la cohérence d'une image à l'autre est exploitée, puisque seules les composantes modifiées sont recalculées.

On peut noter, en outre, qu'il est possible de choisir un compromis entre les directions précalculées et le calcul effectué en fonction du point de vue. On peut, par exemple, pré-déterminer les coefficients statiques pour trois directions données, même si toutes les faces ne peuvent être classées, puis en fonction du point de vue (direction d'étude $\vec{V}$), completer ce classement pour obtenir une quatrième colonne de coefficients statiques permettant le calcul final du rang selon la méthode proposée. Cette solution est particulièrement intéressante lorsque la scène se compose d'objets fixes (précalculés) et d'objets mobiles (calculés à chaque fois).

L'un des principaux reproches que l'on puisse faire à ce processus, est sa dépendance envers le nombre de directions d'étude. Ce qui revient en d'autres termes à poser la question suivante: quel est le rapport entre le nombre de directions et la complexité de la scène et des objets qui la composent?

1.3.7. Réalisation cablée

Pour terminer cette présentation nous évoquerons les possibilités de réalisation cablée ou micro-programmée de cet algorithme. Il est clair que si l'on ne tient pas compte de la structure arborescente, le cablage du calcul des priorités dynamiques ne présente aucune difficulté. La figure 24 ci-après représente le schéma de principe, d'une réalisation en cours d'étude à l'heure actuelle, et qui devrait à terme être intégrée au terminal HELIOS.

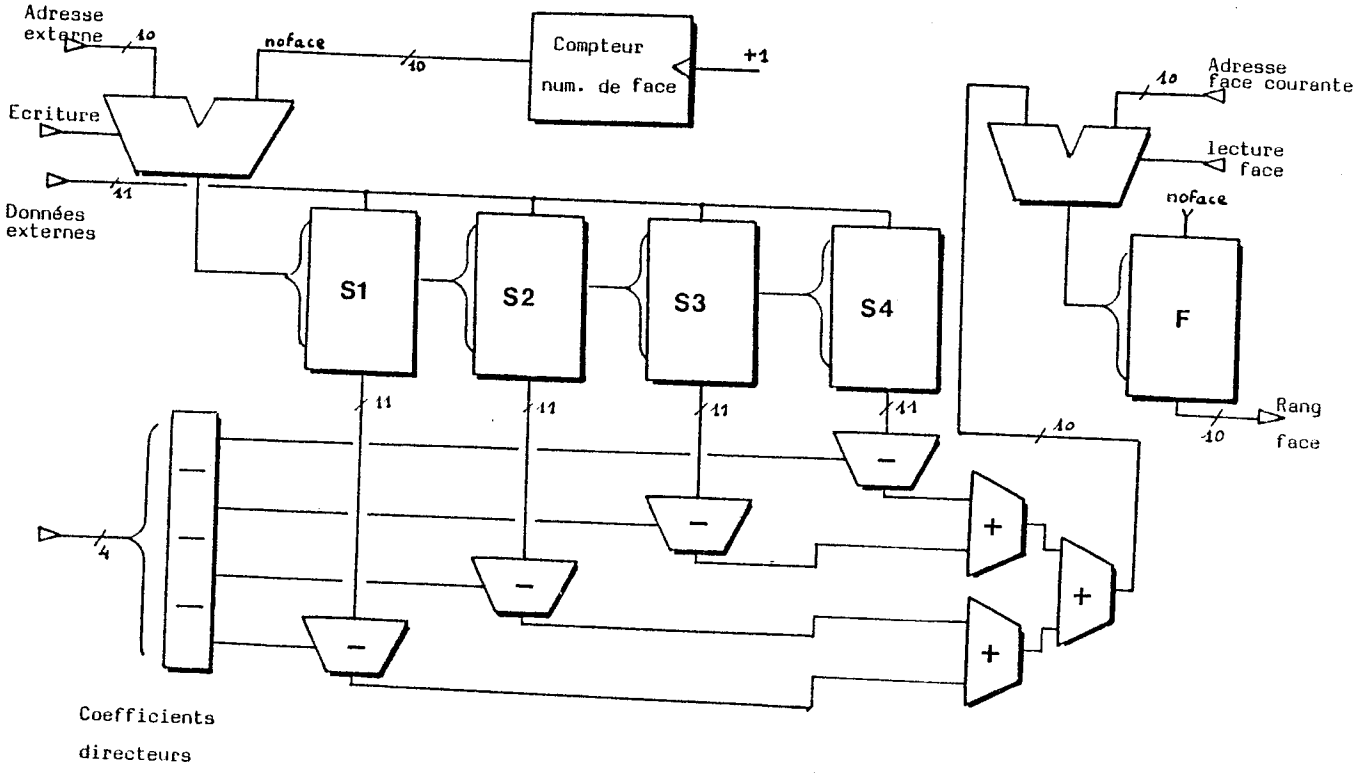


Figure VII.24: Réalisation cablée du calcul du rang

Les mémoires S1, S2, S3, S4, contiennent les colonnes de la matrice des priorités statiques, limitée à 4 directions. Le calcul s'effectue directement à partir de l'expression (1) présentée en 1.3.3. La mémoire F contient en fin de calcul, la liste des faces ordonnées. Elle est ensuite exploitée par le processeur chargé du remplissage des faces (cf. 3.2).

L'ordonnancement complet des 1024 faces ne demande que 500 micro-secondes laissant ainsi le maximum de temps pour la transformation des coordonnées et le remplissage des faces. Cet algorithme est actuellement simulé sur le calculateur pilote associé au terminal et réalise l'ordonnancement de 1024 faces en 120 milli-secondes environ.

2. Les opérateurs de description

2.1. Description d'aspect: les textures

Les textures mémorisées au sein de la logique cablée du synthétiseur HELIOS permettent de personnaliser le terminal en fonction de l'application envisagée. En vue de faciliter cette configuration (cf. VI.1.4.2), le logiciel pilote offre à l'utilisateur un certain nombre d'opérations exécutables en mode "local" et permettant de décrire et d'archiver les textures simulant les divers matériaux nécessaires. Ce système, bien qu'imcomplet, permet de décrire aisément quelques textures particulières, de les archiver sur les disques souples du poste de travail et de gérer la banque intégrée au synthétiseur cablé. Il est à noter, en outre, que tous les opérateurs proposés agissent en temps réel sur l'image visualisée, et permettent ainsi une interaction efficace et puissante. Toutes les opérations présentées ci-après peuvent être exécutées sans altérer le contenu des plans d'identification.

2.1.1. Notion de micro et macro-textures

La grande majorité des matériaux réels résulte de l'assemblage ou de la réunion de plusieurs matériaux de base différents par leur nature, leur couleur, leur brillance, etc. Ainsi dans un mur de briques distingue-t-on les briques elles mêmes qui peuvent différer légèrement par leur couleur, et les joints de ciment qui les lient entre-elles. De la même façon le bois peut être considéré comme étant formé de couches hétérogènes différant par leur couleur dominante et leur brillance.

L'hétérogénéité de la plupart des matériaux nous a orientés vers une

méthode permettant de prendre en compte cette propriété en considérant séparément:

- la micro-texture relative à l'aspect de chaque composant élémentaire (ciment, brique, pierre, etc.)

- la macro-texture relative à l'assemblage de plusieurs matériaux de base.

La séparation entre micro et macro-texture n'est pas toujours aussi nette qu'il y paraît, puisque la plupart des matériaux sont, à leur tour, décomposables en grains (ciment), ou en fils (tissus), de taille et de couleurs différentes, ce qui montre clairement qu'une macro-texture devient micro-texture au niveau supérieur de composition. On peut ainsi, en partant de quelques couleurs de base représentant les matériaux originels, obtenir des textures complexes en appliquant, par étapes successives, les macro-textures permettant d'assembler ces divers éléments constitutifs.

La macro-texture, par définition, devient donc indépendante de la couleur des matériaux mais exprime en quelque sorte son "identité" en chaque point.

La définition d'une texture quelconque peut ainsi être obtenue à travers un processus hiérarchique symbolisé sous la forme d'un arbre (figure 25) dont les feuilles représentent les couleurs de base et les noeuds les opérateurs de macro-textures appliqués. Dans le cas de la figure 25, l'obtention de la texture T5 a nécessité 6 couleurs de base et 5 masques de macro-texture.

La solution retenue pour la descriptiobn d'une texture met en jeu deux textures quelconques T0 et T1 précédemment décrites (par cette même méthode ou par tout autre procédé), ainsi qu'un masque représentant une macro-texture choisie par l'opérateur dans une bibliothèque présentée sur l'écran. Le masque est en fait une texture carrée dont chaque point contient un coefficient compris entre 0 et 1.

- La valeur 0 exprime qu'au point (i,j) de la texture décrite on affecte la

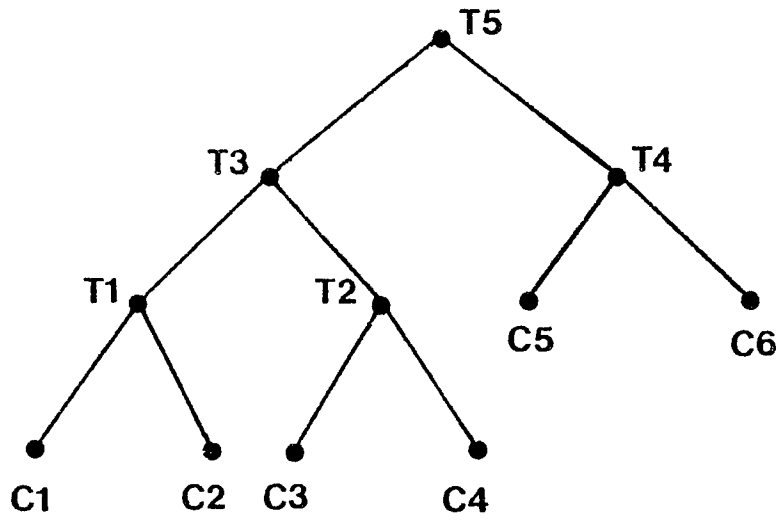


Figure VII.25: Construction hiérarchique des textures

couleur du point (i,j) de la texture T0.

- La valeur 1 exprime que la couleur affectée est celle du point (i,j) de la texture T1.

- Une valeur intermédiaire exprime une couleur obtenue par interpolation entre ces deux couleurs extrêmes. Cette possibilité permet d'effectuer "l'anti-aliassage" des textures en utilisant au niveau des masques, des valeurs intermédiaires proportionnelles à la surface couverte par chacune des deux textures dans chacun des pixels. Cette facilité autorise notamment la description de motifs plus fins et plus détaillés. L'interpolation entre deux couleurs, qui est l'opération de base de ce système, est effectuée en R.V.B. de la manière suivante:

$$\begin{aligned} R &= R0 * (1-k) + R1 * k \\ V &= V0 * (1-k) + V1 * k \\ B &= B0 * (1-k) + B1 * k \end{aligned}$$

où k représente le coefficient d'interpolation issu du masque.

Une étude précédente (Bag81) nous a montré que les différences entre deux interpolations linéaires effectuées, l'une dans l'espace (R,V,B) l'autre dans l'espace (T,L,S), sont fort peu appréciables à l'oeil et qu'il est inutile d'effectuer cette transformation relativement coûteuse. Les

expressions de passage d'un espace dans l'autre (cf. 4.1.1.) montrent en effet que la différence ne porte que sur la teinte puisque la luminance et la saturation varient linéairement avec (R,V,B).

Un deuxième type de masque peut être utilisé pour engendrer des textures de base. Le numéro contenu dans chaque pixel du masque exprime alors le numéro de la texture à utiliser au point considéré, ce qui autorise un choix parmi 16 textures de base mais ne permet plus l'anti-aliassage.

Les principaux avantages apportés par la technique des masques concernent:

- la rapidité et la facilité de description d'une texture qui s'effectue sans calcul, en désignant sur l'écran les textures de base et les masques présentés en noir et blanc.

- la souplesse d'utilisation, puisqu'un même masque permet d'engendrer une grande variété de textures différentes.

- la réduction substantielle de l'espace disque nécessaire à l'archivage d'une texture puisqu'elle peut s'exprimer par simple réference à un masque et à d'autres textures de base. On notera que 128 octets sont nécessaires pour archiver un masque de 16 * 16 pixels avec 16 niveaux de coefficients, soit trois fois moins que pour une texture de même taille.

La description d'une texture nécessite donc deux opérations distinctes, qui seront détaillées dans les paragraphes qui suivent:

* la description des couleurs de base,
* la description des masques de macro-texture.

2.1.2. Description d'une couleur

Le but de ce processus est de permettre la description interactive d'une couleur, sans connaissance du codage nécessaire à sa modélisation. Le procédé le plus simple consiste à présenter sur l'écran du terminal une palette de couleurs, parmi laquelle l'utilisateur désigne la nuance qu'il

désire, à l'aide du réticule. Cependant la réalisation de cette opération se heurte à deux difficultés.

- 1) HELIOS autorise 4096 nuances qui doivent donc être toutes présentées simultanément sur l'écran, ce qui implique un effacement de l'image affichée, et un temps d'affichage important.

- 2) La présentation "harmonieuse" de toutes les nuances sur l'écran n'est pas possible compte-tenu de la nature tri-dimensionnelle des espaces de couleurs. La palette présente donc nécessairement des discontinuités comme le montre la partie supérieure de la figure 26.

Nous avons dans une étude précédente (Bag81), proposé un processus de description, permettant d'engendrer toute la gamme des couleurs à partir d'une palette réduite et de quelques opérateurs appliqués sur les couleurs sélectées.

La solution adoptée travaille dans l'espace TLS (teinte, luminance, saturation), en proposant tout d'abord une palette permettant de choisir la teinte et la saturation, puis des opérateurs permettant de fixer la luminance.

Ces opérateurs "plus clair", "plus foncé", sont facilement apréhendés, même par un non initié, et sont effectués en temps réel à l'aide des touches [+] et [-] de la console opérateur.

Le passage de l'espace RVB à l'espace TLS, et inversement, s'effectue à l'aide des expressions ci-dessous.

$$T = \arcsin\left(\sqrt{3}/2 * (V-R)/S\right)$$
$$L = (R + V + B) / 3$$
$$S = \sqrt{R^2 + V^2 + B^2 - B*V - B*R - V*R}$$

$$R = L - 1/3 * S * \cos(T) - 1/\sqrt{3} * S * \sin(T)$$
$$V = L + 2/3 * S * \cos(T)$$
$$B = L - 1/3 * S * \cos(T) + 1/\sqrt{3} * S * \sin(T)$$

Figure VII.26: Palettes RVB (en haut) et TLS (en bas)

La présentation d'une palette de couleurs à luminance constante ne va pas sans poser des problèmes, dûs au fait que le 'volume' des couleurs synthétisables sur le terminal est un cube dont la diagonale passant par l'origine représente l'axe de la luminance L (figure 27).

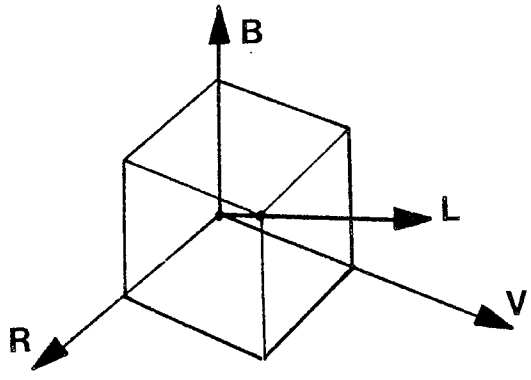


Figure VII.27: Axe de luminance

Si l'on coupe ce cube par des plans perpendiculaires à l'axe de luminance on obtient les sections représentées ci-après sur la figure 28.

Il est clair que le choix d'un point dans l'une de ces sections quelle qu'elle soit, ne permet pas d'obtenir l'ensemble des couleurs possibles par simple translation sur l'axe de luminance. Nous avons choisi de présenter une palette non plane comportant des couleurs dont la luminance varie entre 1/3 et 2/3. Cette palette représentée sur la partie inférieure de la figure 26, est la juxtaposition des lignes de couleurs indiquées sur la figure 29.

La palette est enregistrée systématiquement dans la banque des textures lors de l'initialisation du poste de travail sous forme d'une texture 64 * 64, et peut ainsi être présentée sur l'écran sans qu'aucune modification de l'image affichée ne soit nécessaire. La présentation s'effectue en décalant, par translation, les plans d'identification de façon à faire apparaître le plan de fond auquel la texture "palette" est attribuée.

Lorqu'un point a été désigné avec le réticule, la couleur concernée apparaît sur l'écran, et l'opérateur peut "ajuster" celle-ci en modifiant

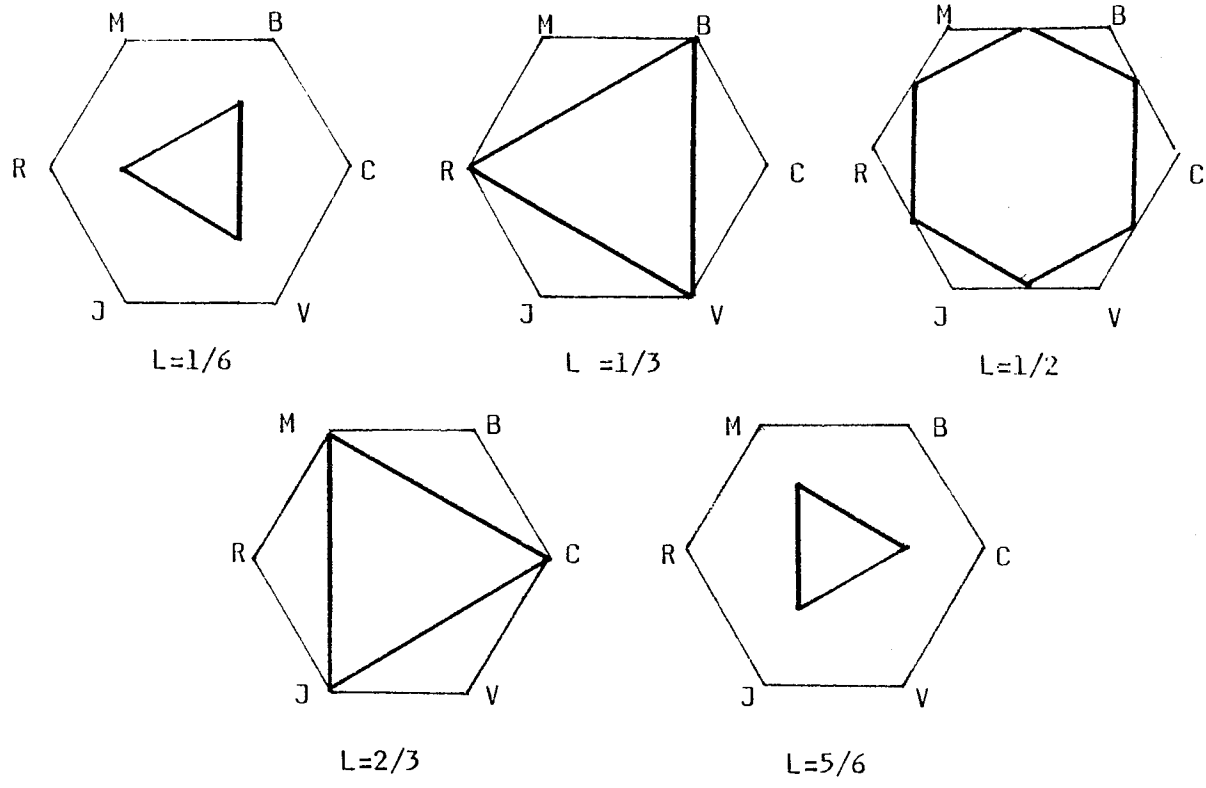


Figure VII.28: Coupes a luminance constante

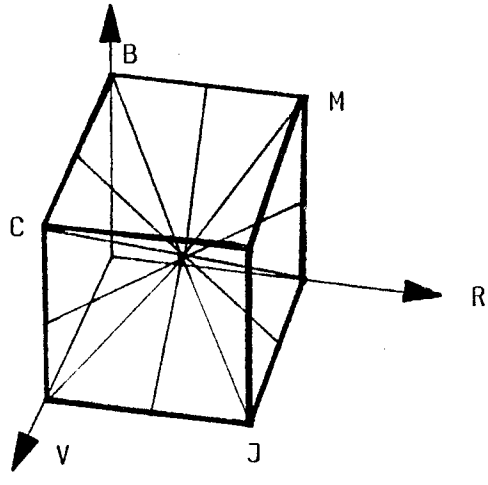


Figure VII.29: Lignes de couleurs de la palette TLS

dans un sens ou dans un autre les paramètres (T,L,S). Cette modification s'effectue en temps réel en utilisant les touches [+] et [-] du pupitre d'interaction. Le choix du paramètre concerné est effectué par l'entremise d'un menu présenté sur la console-opérateur.

En plus des opérateurs ci-dessus, des actions permettant de modifier globalement les trois paramètres sont égalememt proposées. Il s'agit des opérateurs "plus rouge", plus vert", "plus bleu", "plus jaune", "plus cyan", "plus magenta" qui opèrent par interpolation linéaire depuis la couleur initiale vers la nuance concernée. Ces opérateurs permettent d'agir à la manière des peintres qui rajoutent des "touches" de couleur jusqu'à l'obtention de la nuance désirée.

2.1.3. Description des masques de macro-texture

La principale difficulté posée par la description des masques de macro-texture est la recherche de "motifs répétitifs" susceptibles d'être inscrits dans un carré et de donner, après pavage, l'impression recherchée sans faire apparaître de discontinuités ou de macro-textures indésirables. La complexité du problème, la grande diversité des matériaux et la nature plus psychique que mathématique des phénomènes visuels engendrés, nous ont conduits à abandonner l'idée d'une technique générale de génération, pour nous orienter vers des solutions ponctuelles adaptées à quelques cas "types".

Outre la possibilité donnée à l'utilisateur de décrire ses masques point par point le système actuel propose quelques opérateurs permettant de créer ou de transformer des textures.

2.1.3.1. Les textures pseudo-aléatoires

Ce type de texture très fréquemment utilisé pour donner un aspect moins synthétique à divers matériaux ou éléments naturels (herbe, feuillage, roches, graviers, etc.) peut être obtenu à partir de masques aléatoires obtenus selon l'un des deux procédés suivants:

- Interpolation aléatoire: Le système remplit le masque avec des valeurs comprises entre 0 et 1, choisies de manière aléatoire ou pseudo aléatoire. Le résultat final est une texture dont les couleurs sont obtenues par interpolation entre deux couleurs extrêmes. Cette technique

est particulièrement bien adaptée à la simulation de surfaces rugueuses, bosselées ou granulées, (graviers, ciments, tissus, etc). (voir textures 2, 3, 5, 6, 13, 16 de la figure 30).

- Distribution aléatoire: Le système distribue de manière aléatoire quelques "1" à l'intérieur d'un masque initialement rempli avec des "0". La distribution de ces points respecte la densité demandée par l'opérateur. Ce type de masque permet par exemple de disséminer quelques fleurs dans les prés ou des étoiles dans le ciel, ou encore des "imperfections naturelles" au sein d'un matériau quelconque. (cf. textures 6,8 de la figure 30). Il est à noter que si la texture Tl est elle-même une texture aléatoire, les fleurs ou les étoiles distribuées peuvent être de couleur et de brillance différentes.

2.1.3.2. Les textures à motifs géométriques

Il s'agit beaucoup plus ici de conception assistée de textures que de génération automatique. Le système se borne à offrir à l'utilisateur les outils lui permettant de dessiner aisément ses masques, à l'aide de la tablette à numériser. Les principales facilités offertes sont:

* Le remplissage automatique de rectangles exprimés à partir de leurs coins inférieur gauche et supérieur droit.
* La numérisation et le remplissage de cercles exprimés par leur centre et leur rayon.
* Le remplissage de polygones quelconques dont le contour est dessiné sur la tablette.
* Le tracé de vecteurs d'épaisseur paramètrable.
* Le tracé de veines sinusoidales.

Toutes ces générations peuvent s'effectuer avec ou sans anti-aliassage selon le type de masque utilisé.

Une grande variété de textures peut ainsi être obtenue à partir des masques directement dessinés par l'opérateur. Les outils proposés permettent notamment la création de textures simulant les pierres, les briques, les tuiles, les carreaux, les fenêtres, ainsi que quelques essences de bois

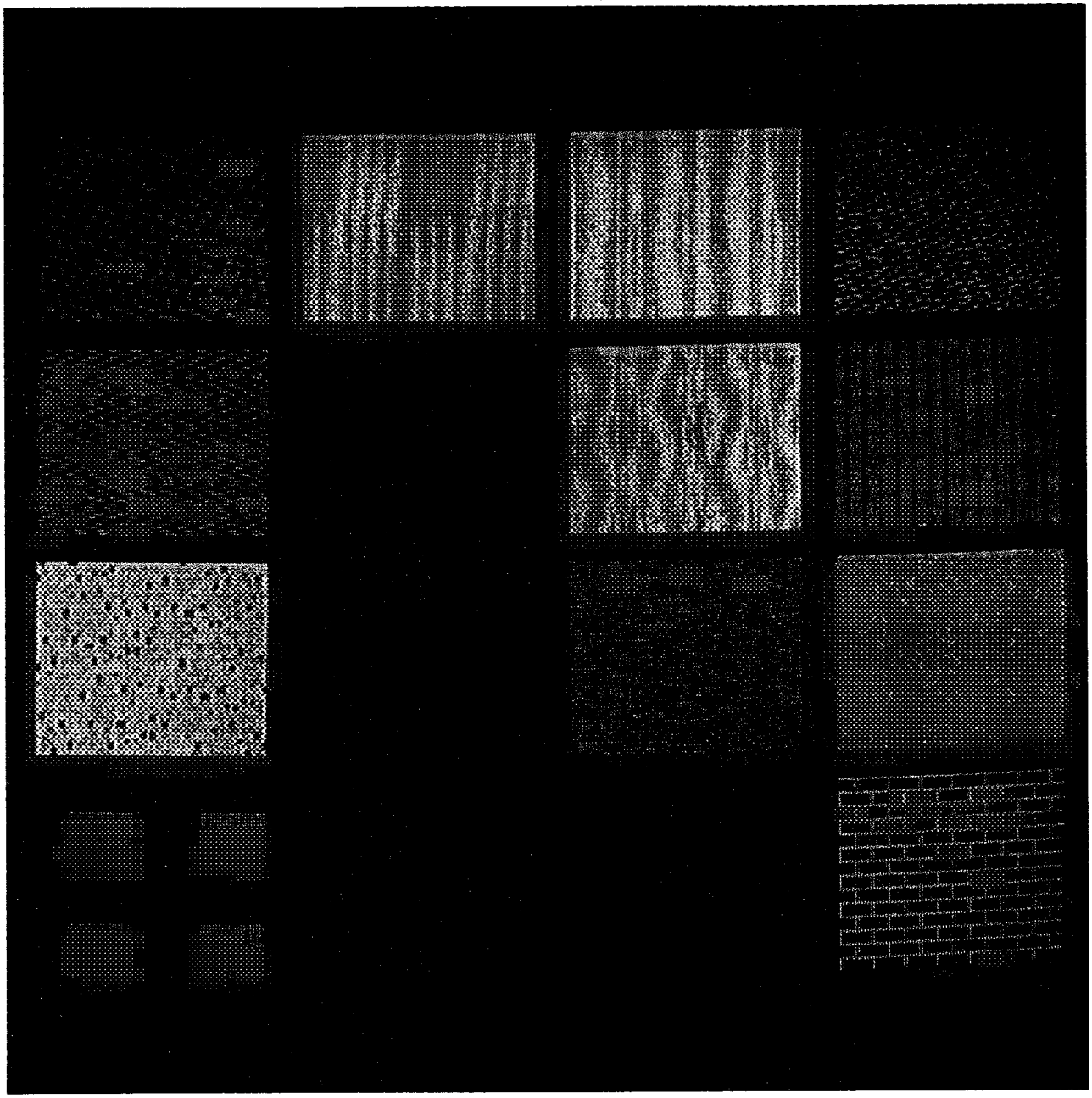


Figure VII.30: Palette de textures

obtenues à l'aide d'une fonction permettant d'engendrer des "veines" dont la

taille et la sinuosité peuvent être paramétrées (cf textures 1, 4, 9, 10, 11, 12, 14, 15 de la figure 30).

2.1.3.3. Les opérateurs de transformation des textures

Basé sur le même principe d'interpolation que les opérateurs ci-dessus, le système propose un opérateur permettant de créer un masque uniforme, c'est-à-dire dont tous les pixels contiennent la même valeur. L'intérêt de tels masques est grand puisqu'il permet d'obtenir une texture dont tous les points sont obtenus par interpolation (à coefficient égal) entre deux textures paramètres T0 et T1. Le coefficient d'interpolation est communiqué par l'utilisateur et permet notamment de paramétrer la transparence d'un matériau ou les conditions de visibilité (brumes, fumées, etc.).

La transparence

La texture T0 représente le matériau transparent, il s'agit le plus souvent d'une texture unie. Le coefficient d'interpolation, communiqué par l'opérateur exprime la transmittance de T0.

* "0" signifie que T0 est opaque donc T1 est totalement invisible.
* "1" signifie que T0 est totalement transparent et donc invisible.
* Une valeur intermédiaire exprime une transmittance partielle.

Les brumes

La texture T0 représente la brume. Il s'agit le plus souvent d'une texture unie de couleur dé-saturée (gris ou gris-bleu).

* "0" signifie que la brume est opaque donc seule visible.
* "1" signifie l'absence totale de brume.
* Une valeur intermédiaire permet de doser "l'épaisseur" de la brume.

Outre ces opérateurs basés sur l'interpolation, il est également possible de transformer directement une texture en lui appliquant globalement les mêmes opérateurs que pour les couleurs (cf.2.1.2) permettant

de modifier, la teinte, la luminance, la saturation, ajouter ou retrancher du rouge, du bleu, du vert.

Les photographies de la figure 33 montrent des bouteilles transparentes simulées par transformation de textures.

2.1.4. La gestion de la banque des textures

Outre l'identification et l'archivage des textures sur les unités de disques souples, le logiciel pilote gère la banque des textures intégrée au synthétiseur cablé HELIOS.

La gestion de cette banque, requiert une attention toute particulière afin d'utiliser au mieux l'espace disponible. Une mauvaise gestion, peut en effet limiter dans des proportions très importantes le nombre de textures enregistrées. Dans l'exemple ci-dessous (figure 31), la situation (1) ne permet plus d'enregistrer de texture 256*256, alors qu'une meilleure organisation (2) permet d'en inclure encore deux (5 et 6).

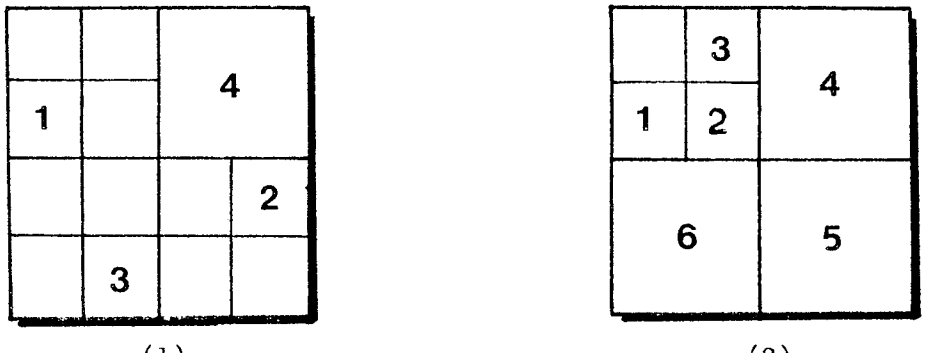


Figure VII.31: Gestion de la banque des textures

La banque des textures peut être représentée par deux tableaux logiques:

libre (0..31,0..31)
complet (0..31,0..31)

exprimant pour chaque carré élémentaire (16*16), s'il est libre et si le carré le plus grand, dont il constitue le coin supérieur gauche, est complet ou non.

Si la taille est égalememnt exprimée en carrés élémentaires: (16*16)-> 1; (512*512)-> 32; l'allocation du pointeur peut être réalisée à l'aide de la fonction récursive ci-dessous.

```
fonction place (i, j, t) : logique
début
    place <- faux
    si  complet (i,j) alors
      si t= taille alors
        si libre (i,j) alors
          place <- vrai
          complet (i,j) <- vrai
          libre (i,j) <- faux
          pointeur-texture <- (i,j)
        finsi
      sinon
        place <- vrai
        libre (i,j) <- faux
        t <- t div2
        si  place (i,j,t) alors
           si  place (i+t,j,t) alors
              si  place (i,j+t,t) alors
                 si  place (i+t,j+t,t) alors place <- faux finsi
      finsi
    finsi
fin
```

Cet algorithme cherche tout d'abord une place libre dans le coin supérieur gauche, ce qui garantit une occupation maximale de l'espace. La figure 32 ci-après illustre le résultat des appels suivants:

1- texture 256 * 256
2- texture 128 * 128
3- texture 128 * 128
4- texture 64 * 64
7- texture 128 * 128
8- texture 128 * 128
9- texture 64 * 64
10- texture 64 * 64

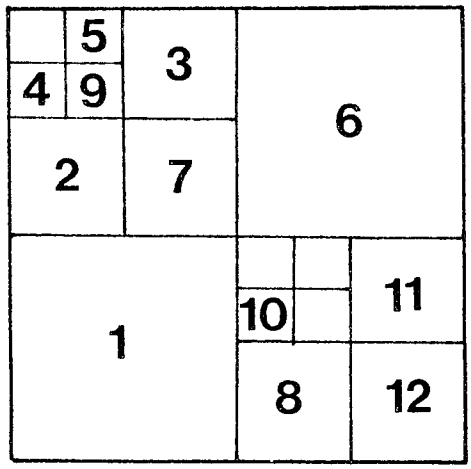


Figure VII.32: Exemple d'organisation de la banque des textures

5- texture 64 * 64
6- texture 256 * 256
11- texture 128 * 128
12- texture 128 * 128

2.1.5. Le cas des surfaces gauches

L'impossibilité d'appliquer directement une texture sur une surface gauche au niveau du terminal HELIOS, nous a conduits à envisager quelques outils permettant de pallier cette lacune.

Etant donné que les plans d'identification contiennent dans ce cas, non pas un numéro de face, mais la normale en chaque point (cf. VI.1.4.2), il devient possible de décrire des fonctions permettant de faire correspondre, à chaque normale théorique (considérée comme un numéro de face):

* une normale réelle,
* une texture,
* un modèle de réflexion.

2.1.5.1. Perturbation de la normale

Cet opérateur inspiré de la méthode proposée par Blinn (Bli78), permet de simuler une perturbation du relief de la surface. Il est à noter que, contrairement au cas de Blinn, cette perturbation n'est pas indépendante de la taille de l'objet, du fait de sa définition en coordonnées sphériques.

L'opérateur décrit la perturbation au moyen de deux fonctions Da et Db relatives chacune à l'un des deux angles a et b qui expriment la direction normale. Le processus consiste à affecter à chaque face (normale théorique), une normale réelle, opération qui prend place entre deux trames et autorise ainsi une certaine animation du relief (déformations, simulations de rotations, etc.).

2.1.5.2. Affectation d'une texture ou d'un modèle de réflexion

On peut, de la même manière que ci-dessus, associer à chaque normale fictive une texture et un modèle de reflexion propres. Cette association s'effectue à travers une fonction à deux variables (les angles a et b) prenant ses valeurs dans l'ensemble des modèles de texture ou de réflexion disponibles. Un exemple est présenté sur les sphères de la photographie de la page 156 où deux textures sont affectées en alternance, en fonction des angles a et b.

Il est à noter que toutes ces opérations nécessitent au plus 1024 affectations (une pour chaque numéro de face), traitement qui peut dans tous les cas être executé entre deux trames, ce qui permet d'effectuer une certaine animation en temps réel. La rotation d'une sphère autour d'un axe vertical peut, par exemple, être simulée en effectuant une permutation circulaire des textures affectées aux faces (c'est-à-dire aux normales théoriques) selon l'angle a.

2.2. Description d'éclairage

La description des paramètres d'éclairage peut, comme dans le cas de l'aspect, s'effectuer entièrement au niveau du logiciel pilote, sans détruire l'image affichée et avec visualisation immédiate des effets provoqués par les modifications. Le calcul de l'éclairage étant effectué au sein même du matériel, tout changement d'un paramêtre quelconque est répercuté en temps réel sur l'image.

2.2.1. La source lumineuse

Les paramètres liés à la source lumineuse sont:

* la direction,
* la couleur,
* l'intensité de la lumière ambiante.

2.2.1.1. Direction des sources principale et secondaire

Le déplacement de la source lumineuse s'effectue en temps réel de deux manières distinctes. L'utilisateur peut indiquer une direction directement au clavier à l'aide des coordonnées sphériques (angles a et b), ou encore déplacer la source en utilisant les touches  de la console opérateur. Cette dernière possibilité facilite notamment la recherche d'une direction optimale d'éclairage permettant d'apprécier au mieux l'image visualisée. Compte-tenu du fait que le système est également chargé du calcul de la direction médiane entre la source lumineuse et l'observateur, il est possible, en modifiant la valeur réelle de cette médiane, de simuler une seconde source lumineuse (source secondaire) dont les effets sont limités à la réflexion spéculaire. Cet artifice permet par exemple de rendre les reflets produits par une fenêtre dans une pièce déjà éclairée par ailleurs, ou d'augmenter le réalisme par l'adjonction de reflets.

2.2.2. La couleur

Bien que, dans la plupart des cas, la source lumineuse soit blanche et de luminance maximale, il peut être intéressant pour certaines applications de faire varier ce paramètre, notamment pour simuler un éclairage naturel, (vision crépusculaire, coucher ou lever de soleil) ou encore un éclairage artificiel (lampes incandescentes, tubes à gaz, utilisation de filtres colorés, etc). Nous retrouverons ici l'ensemble des outils proposés au paragraphe précédent, et permettant d'agir sur la couleur donnée en augmentant ou en diminuant les paramètres R, V, B ou T, L, S.

2.2.2.1. La lumière ambiante

La lumière ambiante simule la proportion de lumière dispersée dans l'atmosphère et illuminant la scène en intensité égale dans toutes les directions. Cette proportion peut être modifiée en temps réel en utilisant les touches [+] et [-] de la console opérateur. Ce paramètre, difficilement quantifiable, peut être ainsi approché par tatonnement jusqu'à une valeur optimale permettant d'obtenir à la fois un contraste acceptable et une bonne perception des détails.

Une phase future de ce système devrait contenir quelques outils permettant de faire varier simultanément les trois paramètres attachés à la source lumineuse. Ainsi si l'on examine le cas caractéristique du coucher de soleil, il apparaît que la couleur et la lumière ambiante sont intimenent liées à la "hauteur" du soleil sur l'horizon (Max 81).

Un tel opérateur recevrait comme seul paramètre la donnée du plan de rotation du soleil, et permettrait de déplacer ce dernier dans ce plan en calculant automatiquement la translation de la couleur vers le rouge, l'augmentation de lumière ambiante et la diminution de l'intensité directe, dues aux positions basses du soleil. Le passage progressif du jour à la nuit peut également être envisagé.

Les photographies de la figure 33 présentent une même scène soumise à

des conditions d'éclairages différentes (déplacement de la source vers la gauche et diminution d'intensité).

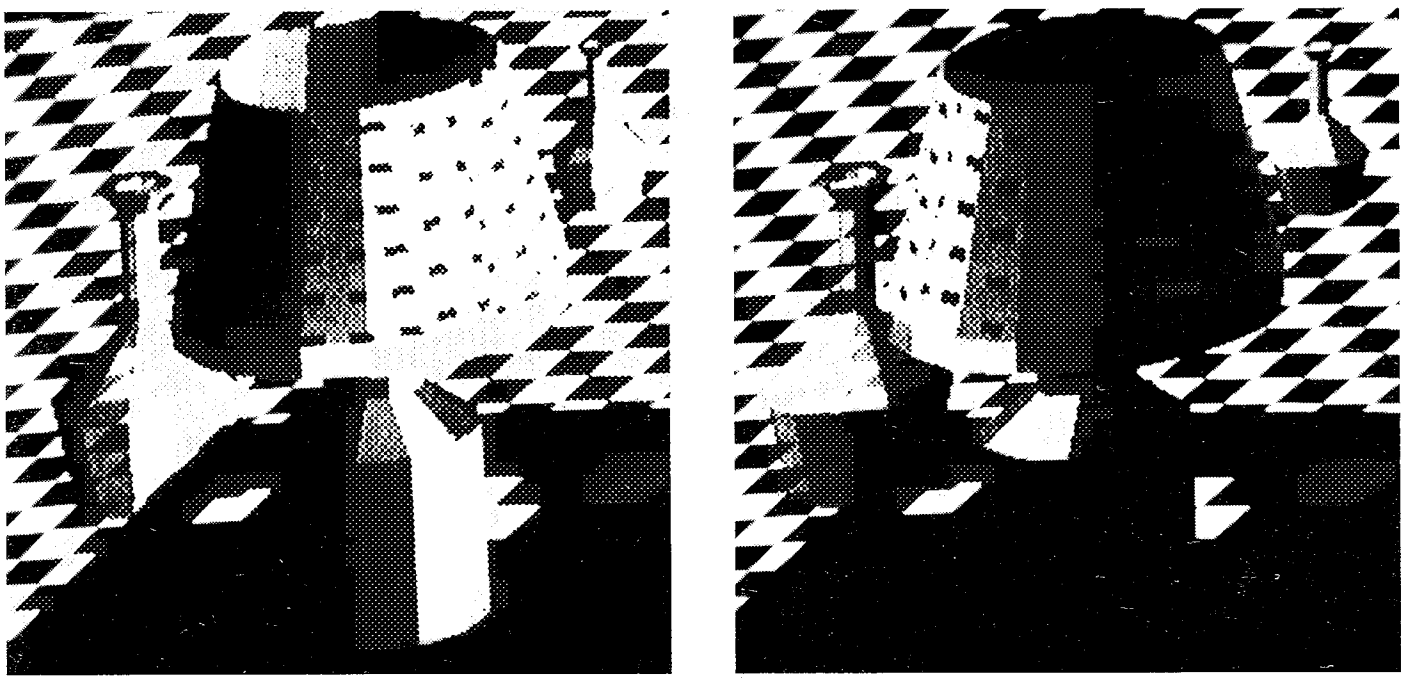


Figure VII.33: Modification des conditions d'éclairage

2.2.3. La banque des modèles de réflexion

Le calcul cablé de l'éclairage nécessite l'utilisation de modèles de réflexion diffuse et spéculaire enregistrés en RAM dans le terminal HELIOS (cf V.2.3.3)

A chaque classe de matériaux on peut ainsi associer un couple de modèles D(l) et S(m) exprimant respectivement le coefficient de reflexion diffuse en fonction de l'angle l (angle entre la normale et la source) et le coefficient de réflexion spéculaire en fonction de m (angle entre la normale et la direction médiane entre l'observateur et la source).

2.2.3.1. Les modèles standards

Les valeurs des coefficients sont exprimés entre 0 et 1, et peuvent être approchées en supposant que la surface des matériaux réels est composée d'une grande quantité de micro-facettes refletant chacune parfaitement la lumière (ToS67). Il est clair que, pour la plupart, ces facettes sont orientées selon la normale théorique à la face et que leur nombre diminue au fur et à mesure que l'on s'éloigne de cette direction. Diverses fonctions de distribution de ces facettes ont été testées par Blinn (Bli77) qui, à très peu près, se ramènent à une distribution gaussienne représentée par l'expression ci-dessous.

$$G = c.e^{-(\frac{x}{k})^2}$$

La figure 34 montre les courbes obtenues pour différentes valeurs de k lorsque c=1.

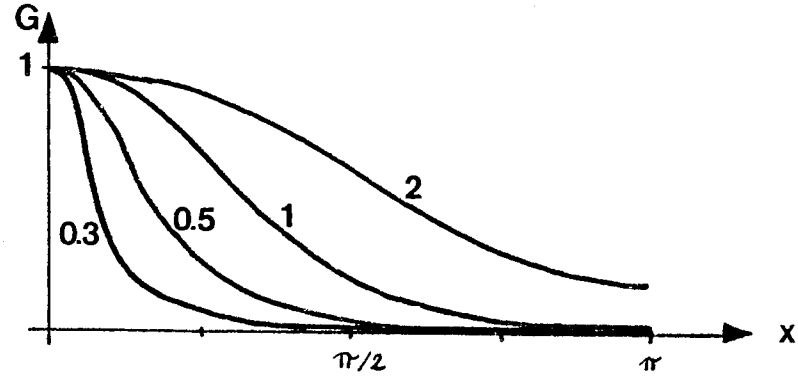


Figure VII.34: Profil des fonctions de distribution

Les valeurs élevées de k correspondent aux surfaces rugeuses alors que les valeurs faibles représentent les surfaces lisses. On notera d'ores et déjà que si les valeurs supérieures à 1 ne correspondent pas à une réalité physique, elles permettent néanmoins de différencier le niveau d'éclairement des faces qui ne sont pas éclairées directement (angle compris entre π et $\pi/2$).

Afin de faciliter la description des modèles de réflexion par l'utilisateur le sytème se charge de déterminer les deux modèles de réflexion diffuse et spéculaire, à partir de la seule donnée du type de matériau à

simuler.

$$D(l) = Cd \cdot e^{-(\frac{x}{kd})^2}$$

$$S(m) = Cs \cdot e^{-(\frac{x}{ks})^2}$$

Ce calcul passe par la détermination des quatre paramètres Cd, kd, Cs et ks, qui dépendent des propriétés physiques de la matière. On peut notamment considérer que la réflexion diffuse dépend des couches profondes du matériau alors que la réflexion spéculaire dépend uniquement de la couche superficielle qui peut être radicalement différente (par exemple vernie ou oxidée). On peut ainsi donner une signification approximative de ces quatre coefficients:

Cd : coefficient de réflexion maximal du matériau,

kd : indice de rugosité du matériau,

Cs : coefficient de réflexion maximal de la couche superficielle,

ks : indice de rugosité de la couche superficielle.

Les valeurs numériques de ces coefficients peuvent être trouvées dans un recueil regroupant des centaines de matériaux divers (Pur70), ou encore être déterminé de manière empirique par des essais successifs. La photographie de la figure 35 montre les effets de différents modèles appliqués sur des sphères.

2.2.3.2. Les modèles particuliers: la transparence

Nous abordons ici les modèles de réflexion particuliers ne correspondant pas à une réalité physique mais permettant de simuler des phénomènes tels que la transparence.

Nous avons vu au paragraphe précédent (cf 2.1.3) qu'il était possible de simuler la transparence d'un objet en composant sa texture avec celle de l'objet situé derrière lui. Il faut cependant compléter cet artifice par une adaptation des modèles de réflexion.

Si l'on considère la situation représentée sur la figure 36, on peut

Figure VII.35: Les effets des modèles d'éclairage

constater qu'en un point donné deux normales sont présentes: celle de l'objet X transparent, et celle de l'objet Y vu par transparence. Etant donné qu'une seule direction normale peut être associée à chaque face, il est nécessaire d'utiliser des modèles spécifiques permettant de compenser dans une certaine mesure l'erreur commise. La normale qui est affectée à la face F2 (intersection de X et de Y) est la normale de la face transparente X. Ce choix est nécessaire pour rendre les reflets importants (réflexion spéculaire) qui caractérisent généralement les matériaux transparents tels que le verre. En ce qui concerne la réflexion diffuse les problèmes sont beaucoup plus complexes puisqu'il faudrait considérer simultanément:

* 1* la dépendance de la réflexion de X par rapport à l'angle Lx
* 2* la dépendance de la réflexion de Y par rapport à l'angle Ly
* 3* la dépendance de la réflexion de Y par rapport à l'angle Vx

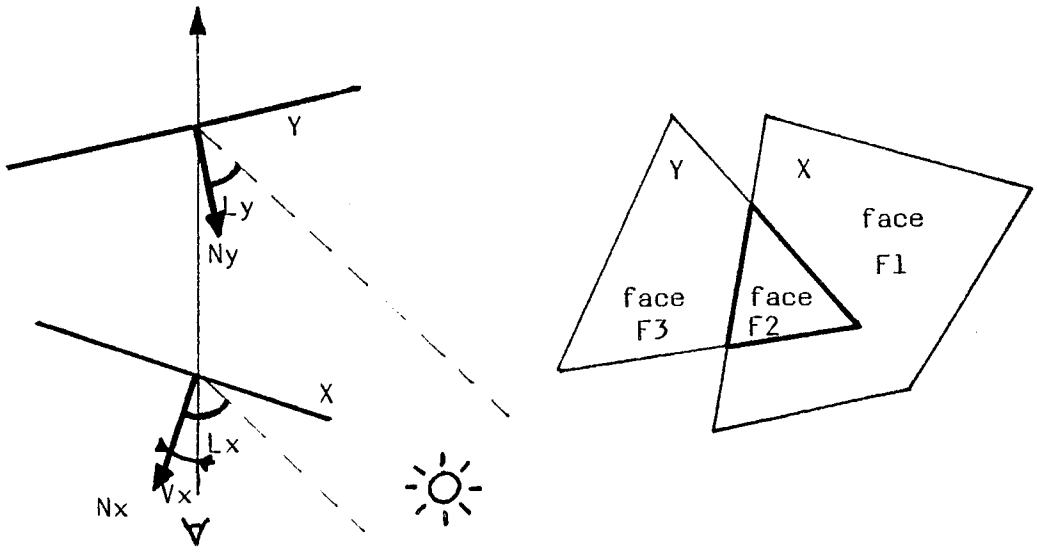


Figure VII.36: Problème de la transparence

Ce dernier facteur découle en effet de la diminution de la transmittance en fonction de l'épaisseur du matériau, épaisseur plus importante lorsque la face X est traversée obliquement (cf. figure 37) abstraction faite ou pas du phénomène de réfraction .

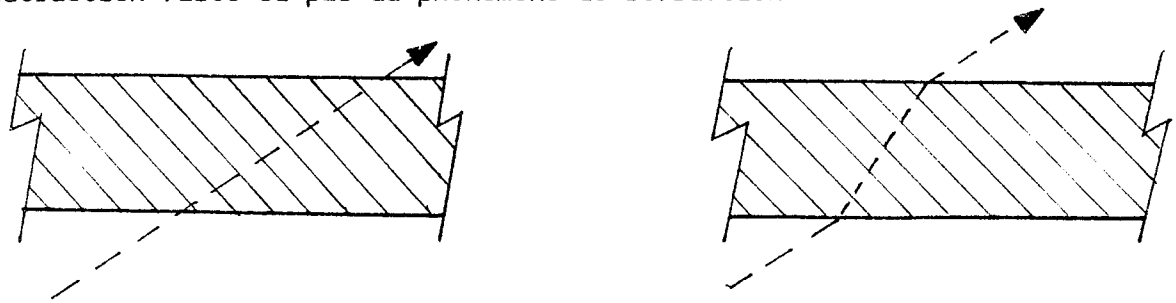


Figure VII.37: Diminution de la transmittance

Il est clair que seul le premier terme peut être approché à l'aide du modèle de réflexion diffuse puisque l'angle dont il dépend est bien celui qui est considéré par le calcul de l'éclairage. Cependant dans le cas d'un matériau transparent cette réflexion diffuse est extrêmement faible et le réalisme dépend presque exclusivement de l'importance des reflets. Le modèle utilisé est celui représenté sur la figure 38.

On peut remarquer le facteur d'atténuation paramètrable $\underline{t}$ et la forme

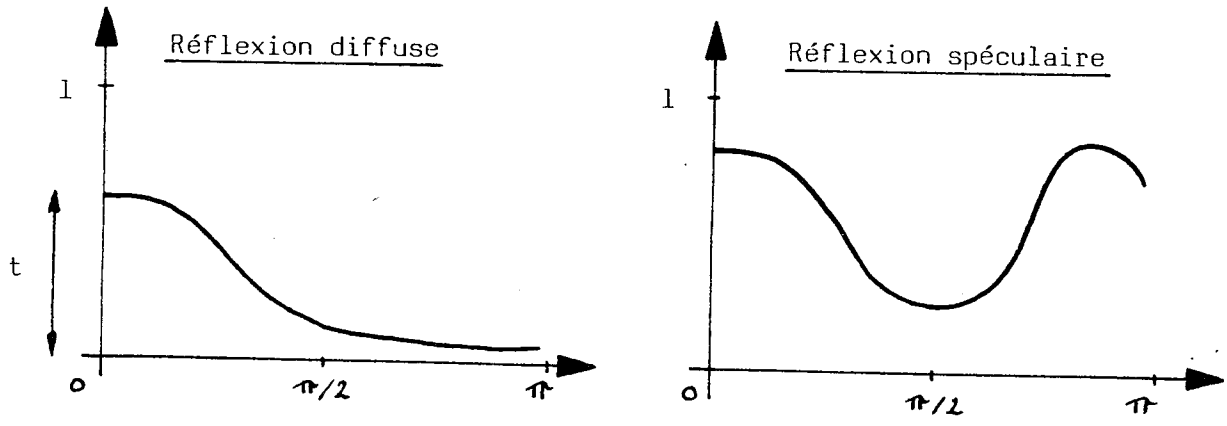


Figure VII.38: Modèles simulant la transparence

du modèle de réflexion spéculaire qui permet d'engendrer des "faux reflets" dans plusieurs directions, comme c'est généralement le cas pour les objets réels.La photographie de la figure 39 illuste les effets acceptables obtenus avec ce modèle.

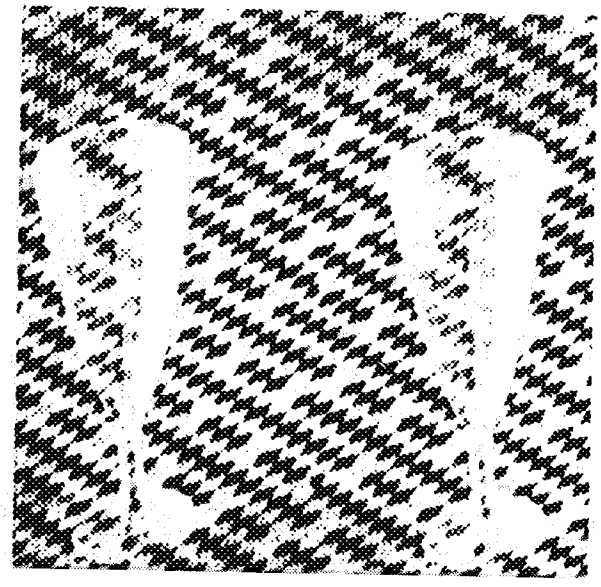
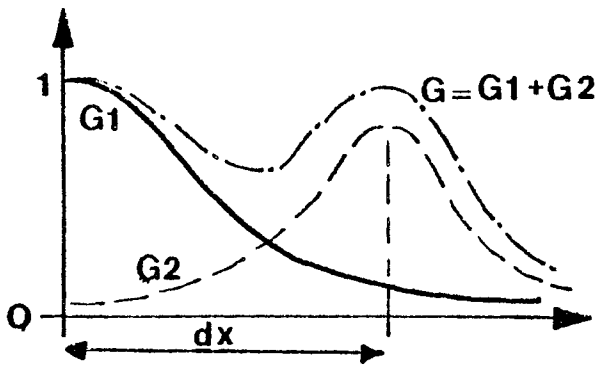


Figure VII.39: Objets transparents

2.2.3.3. Les effets spéciaux

Multiplicité des sources lumineuses

L'un des effets spéciaux les plus faciles à réaliser est celui simulant la présence de plusieurs sources lumineuses. Cet artifice permet notamment d'augmenter le nombre de reflets, comme dans le cas de la transparence, et de diminuer le contraste des objets présentés afin d'approcher les conditions réelles existant dans une pièce où plusieurs fenêtres et sources lumineuses sont généralement présentes. Les modèles décrits sont de la forme présentée sur la figure 40 dans laquelle la différence d'angle dx est donnée par l'utilisateur. Le modèle G1 + G2 doit ensuite être normalisé pour prendre ses valeurs entre 0 et 1. La photographie de la page 156 illustre cet effet de double source lumineuse. On notera l'effet de croissant lumineux obtenu sur les sphères.



$$G1 = C1\, e^{-\left(\frac{x}{k1}\right)^2}$$

$$G2 = C2\, e^{-\left(\frac{x}{k2}\right)^2}$$

Figure VII.40: Multiplicité des sources lumineuses

Il est également possible de simuler une symétrie de la source par rapport à l'origine pour certains objets afin de donner l'illusion d'une source située au centre de la scène en provenance d'une lampe par exemple.

Inhibition des effets de l'éclairage

Il s'agit là d'une possibilité indispensable dans les cas suivants:

- Le terminal est utilisé pour la présentation d'images non réalistes ou en provenance directe de caméras numériques.

- Il s'agit d'images réalistes mais certains objets sont soumis à une autre source lumineuse indépendante, par exemple le paysage visible à travers la fenêtre d'une pièce possédant son éclairage propre. Cette inhibition est réalisée à l'aide des modèles présentés sur la figure 41.

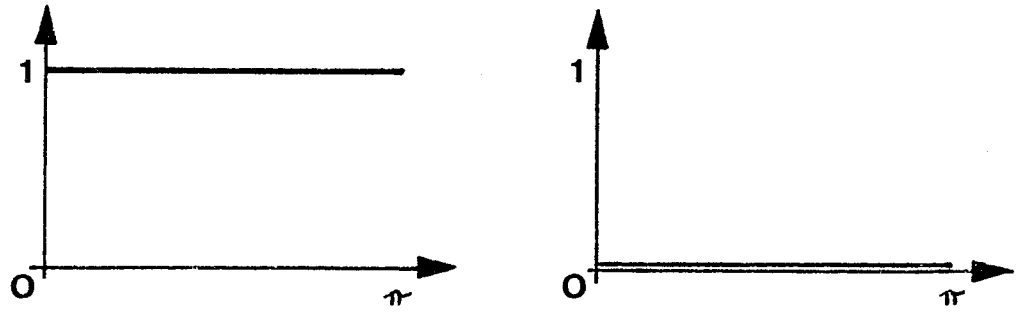


Figure VII.41: Inhibition de l'éclairage

2.3. Conclusion

Bien qu'il s'agisse d'une première version très limitée, les opérateurs de description présentés ci-dessus nous ont permis, à travers leurs qualités et aussi leurs insuffisances, de mieux apprécier les limites de l'assistance que l'homme peut attendre de l'ordinateur dans ce domaine délicat de la description des informations visuelles. Cette difficulté de description semble en outre diriger les recherches vers l'utilisation de textures obtenues par photographie de matériaux réels. Cette technique, séduisante à première vue, soulève cependant un problème important.

Si l'on désire utiliser le pavage automatique offert par le matériel, il est indispensable d'obtenir un modèle texture carré dont les bords (gauche et droit; inférieur et supérieur) coincident parfaitement afin de ne pas engendrer de discontinuités gênantes. Etant donné que cette condition n'a aucune raison d'être satisfaite sur l'image initiale il est nécessaire

d'effectuer un traitement propre à atténuer ces effets indésirables. Deux méthodes peuvent être utilisées.

- La première proposée par Dungan et al. (DSS78) consiste simplement à appliquer sur la texture un filtre "passe-bas" permettant d'atténuer les transitions. Il est clair que la définition de la texture est amoindrie et que cette technique ne donne des résultats acceptables que dans le cas de textures aléatoires ou pseudo-aléatoires (herbe, feuillage, etc)

- Une seconde méthode consiste à appliquer deux opérateurs de symétrie à l'image originale de manière à obtenir une texture quatre fois plus grande. Ce procédé illustré sur la figure 42 permet de supprimer les discontinuités mais peut engendrer des motifs fort différents de ceux présentés par la photographie originale. Néanmoins, cette méthode convient parfaitement pour la saisie directe de la texture de quelques essences de bois. En guise de conclusion, il apparaît très nettement qu'aucune méthode de description, quelle qu'elle soit, ne puisse satisfaire tous les cas possibles et que les études à venir vont tenter de classifier et de caractériser les différents types de matériaux afin de déterminer la technique la mieux adaptée à leur description synthétique.

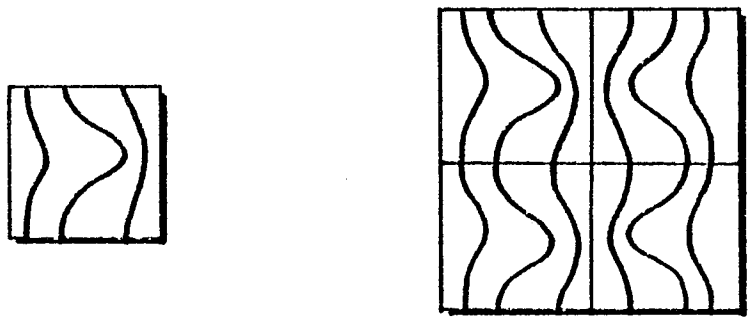


Figure VII.42: Symétries appliquées à une texture originale

3. Les possiblilités d'animation

Outre les possibilités d'animation de couleurs offertes par les opérateurs décrits ci-dessus, les performances de la logique cablée d'HELIOS permettent d'envisager une animation réelle, c'est à dire agissant sur les attributs M et G de la scène. Nous n'évoquerons pas ici les problèmes posés par la description de l'animation ni les outils permettant de rendre cette tâche moins fastidieuse. Le lecteur intéressé trouvera en (Mar77), (Mar77b), (Mar78) l'état de notre expérience personnelle dans ce domaine.

3.1. Les techniques de production d'animation

Si l'on suppose que l'on sait décrire et visualiser chacune des vues de la séquence à animer, la question qui reste à résoudre est celle du choix de la technique propre à restituer cette animation. Ce choix dépend essentiellement des facteurs suivants:

* le support du film final,
* la qualité désirée,
* le degré d' interactivité nécessaire,
* les possibilités offertes par le terminal disponible. Cette dernière contrainte, généralement prédominante, conduit au choix représenté sur le tableau ci-après:

console / calcul	Modification d'une image <= 40 ms	Modification d'une image > 40ms
Calcul d'une image <=40 ms	Animation directe temps réel	Animation différée vue par vue
Calcul d'une image >40 ms	Animation différée temps réel	Animation différée vue par vue

3.1.1. L'animation différée vue par vue

Nous entendons par "animation différée vue par vue" le procédé qui consiste à perdre le temps nécessaire au calcul de chaque vue et à sa visualisation sur une console, puis à enregistrer celle-ci au moyen d'une caméra (si le support désiré est le film) ou d'un vidéo-disque (si le support désiré est la vidéo). Malgré les gros avantages que procure cette dernière solution (possibilité de ralentis, d'arrêts sur image ou de retours arrière), le coût actuel de ce type d'appareil (300.000F minimum) limite encore considérablement sa rentabilité.

L'utilisation d'une caméra (en général 16 mm) soulève, d'un autre côté trois problèmes techniques importants:

- Il est nécessaire de déclencher automatiquement la caméra à partir du calculateur lorsqu'une vue a été affichée, ce qui impose l'utilisation d'une caméra offrant cette possibilité.

- Afin d'assurer une exposition homogène de chacune des vues, le calculateur doit afficher un nombre identique de trames entières pour chaque vue. L'obturateur de la caméra doit, bien entendu, rester ouvert pendant ce temps là.

- La sensibilité chromatique de la pellicule utilisée doit être connue de manière à compenser, sur l'image elle-même, les distorsions de couleurs qui se produiront lors du développment.

Nous citerons les résultats obtenus à l'Institut National de l'Audiovisuel en utilisant une variante de cette technique (Cou80). Afin d'augmenter la définition de l'image et de faciliter la prise en compte de la sensibilité chromatique de la pellicule, le tube couleur est remplacé par un tube noir et blanc haute définition ou les trois primaires (rouge, vert, bleu) sont présentés successivement à l'objectif de la caméra à travers des filtres colorés.

Dans notre cas la prise de vue directe sur l'écran couleur peut être améliorée grâce aux possibilités de synchronisation offertes par HELIOS (cf. V.3.2.4). Il est en effet possible de limiter l'affichage à un nombre quelconque de trames en utilisant la translation cablée des plans pour faire apparaître, puis disparaître, l'image en synchronisme avec les "retours de trames". Cette technique assure une exposition homogène de chacune des vues du film.

Il est clair que cette approche de l'animation différée ne convient qu'aux applications à faible degré d'interactivité, puisqu'il est nécessaire d'attendre le développement du film pour apprécier les résultats.

3.1.2. L'animation différée précalculée

Si l'on dispose d'une console de visualisation capable de mémoriser une image dans sa mémoire de trame en 40 ms, on peut envisager une animation temps réel. Les vues constituant le film sont précalculées au rythme imposé par les calculs et sont ensuite mémorisées sur une unité de disque rapide. Cette unité doit être capable, dans un deuxième temps, d'alimenter la console à une vitesse suffisamment rapide pour assurer l'animation continue. Nous avons présenté au cours du chapitre III (cf. III.3.4.1) des systèmes basés sur ce principe et utilisant la compression des informations pour accélérer le transfert. Il va de soi que le taux de compression est généralement inversement proportionnel au réalisme de l'image pour un terminal à mémoire d'image classique.

Dans le cas du synthétiseur cablé d'HELIOS le taux de compression est totalement indépendant du réalisme, puisque la mémoire de trame ne contient que l'identification des faces affichées. Il faut cependant noter que cette animation temps réel précalculée n'a pu être envisagée compte-tenu de la lenteur des unités de disques souples et du port parallèle du calculateur pilote.

3.1.3. L'animation temps réel

L'animation directe en temps réel nécessite des moyens de calculs considérables qui ne se justifient réellement que dans le cas très particulier de la simulation de conduite pour laquelle l'interaction doit être puissante et immédiate.

On peut cependant, à l'aide de quelques systèmes spécifiques tel que celui réalisé au CELAR (Ler80), ou le terminal HELIOS présenté au chapitre V, envisager une certaine animation temps réel limitée à quelques attributs spécifiques (G,E,A). Le paragraphe 2 du présent chapitre nous a permis de présenter les possibilités d'animation temps réel au niveau de l'aspect et de l'éclairage et nous consacrerons la suite de cet exposé aux autres possibilités d'animation offertes.

3.2. L'animation temps réel sur HELIOS

3.2.1. Les translations en X et Y

L'animation la plus simple que l'on puisse envisager est la translation puisqu'elle est offerte directement par le synthétiseur cablé. Bien qu'il s'agisse d'une translation applicable à la totalité d'un plan d'identification, il est possible d'animer un objet particulier s'il est seul sur le plan et si la ou les faces ne faissant pas partie de l'objet sont rendues invisibles. La figure 43 montre une configuration permettant d'animer deux objets en translation sur un plan de fond immobile.

La translation des plans P0 et P1 est effectuée en temps réel à l'aide des touches [↑] [↓] [→] [←] de la console opérateur, ou peut également être asservie au déplacement du crayon de la tablette à numériser.

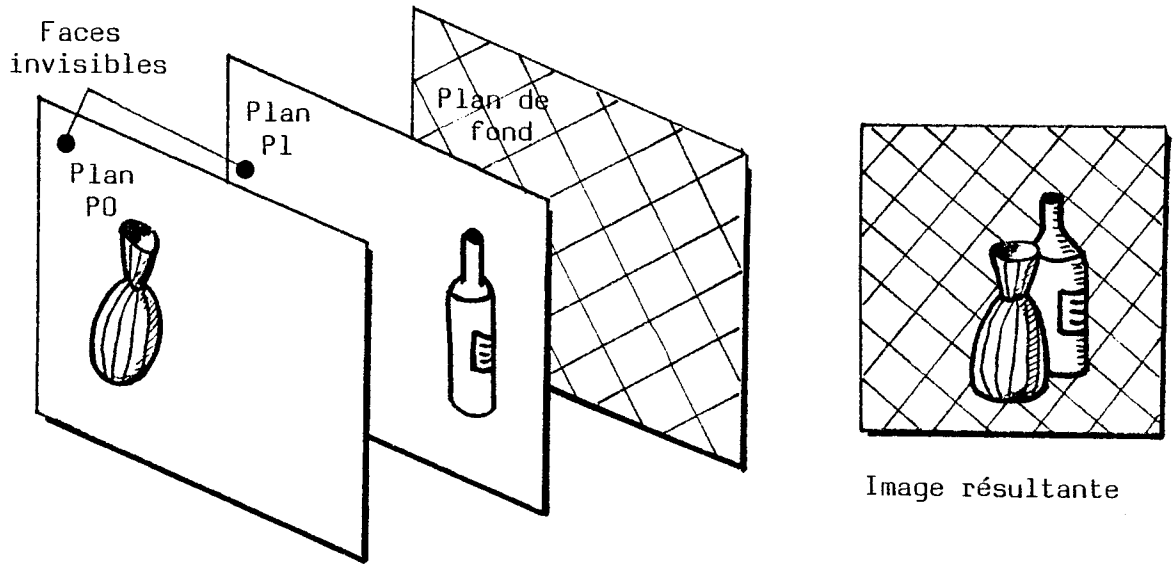


Figure VII.43: Animation en translation

3.2.2. Les translations en Z

Si l'on fait abstraction des variations morphologiques dues à la perpective, que nous évoquerons au paragraphe suivant, le problème qui se pose est celui de l'occultation mutuelle des objets. En effet si l'objet U situé sur le plan P0 translate derrière l'objet V mémorisé sur le plan P1, il est nécessaire que V cache U ce qui est impossible compte-tenu des priorités fixes des plans d'identification. L'artifice utilisé pour simuler cette translation en profondeur est illustré par la figure 44 ci-dessous.

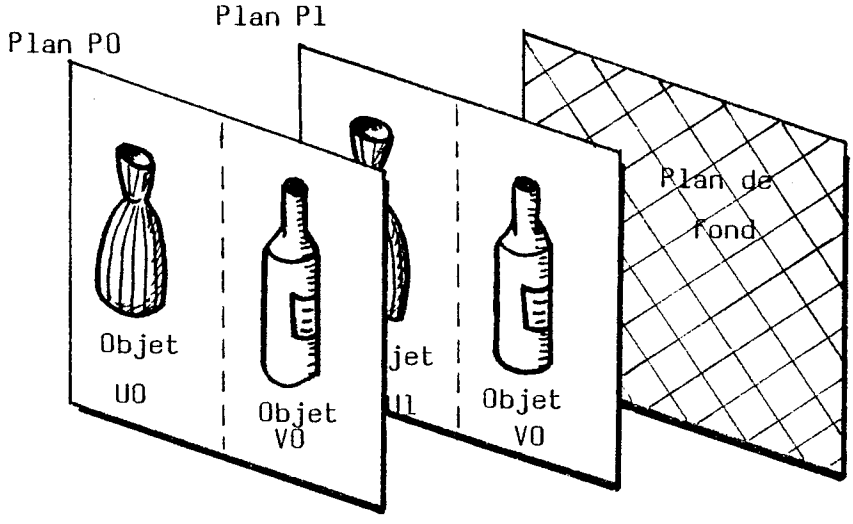


Figure VII.44: Translation en profondeur

CONCLUSION

Nous voudrions, pour présenter nos conclusions, rappeler les motivations qui sont à l'origine de notre étude et les évolutions successives qui nous ont conduits progressivement jusqu'aux résultats exposés dans cette thèse.

Depuis sa création, l'équipe de Communication Graphique de l'IMAG, s'est toujours intéressée beaucoup plus à l'aspect conceptuel des problèmes graphiques qu'à l'aspect technique. On peut évoquer deux motivations principales à cette orientation: en premier lieu la vocation plutôt "pédagogique" des personnes concernées mais aussi l'insuffisance (voire même l'absence) de matériel graphique approprié, pour l'élaboration et le test des techniques de pointe.

L'étude présente a bénéficié ainsi, dès son départ, d'une expérience solide et d'une décantation déjà très avancée des concepts et des notions qui sont à la base de la communication graphique interactive. La mutation technologique considérable de ces dix dernières années, due principalement à l'accroissement d'intégration est venue infléchir et diversifier les activités de l'équipe sous plusieurs aspects.

En premier lieu, la transition de l'ère du "graphique" vers l'ère de la "synthèse d'images réalistes" a provoqué une remise en cause importante des notions de base et conduit les concepteurs de logiciels à séparer nettement ces deux types d'applications en leur associant des systèmes totalement distincts.

En second lieu, l'apparition de consoles de visualisation de plus en plus "sophistiquées" a occasionné une autre remise en cause, celle de la séparation entre logiciel et matériel. L'utilisation intensive des

micro-processeurs permet en effet d'intégrer à l'intérieur de la console quelques fonctions qui, jusqu'alors, étaient à la charge du système. L'anarchie et la diversité de ces intégrations place également les concepteurs de système devant un dilemme: ignorer les possibilités offertes au détriment des performances ou les exploiter au détriment de l'indépendance et de la normalisation. Ces constatations, qui sont à l'origine des concepts proposés, nous ont également poussés à abolir le clivage, désormais artificiel, entre logiciel et matériel et à expérimenter nos idées dans ces deux domaines. Nous pensons avoir démontré à travers la conception et la réalisation des prototypes CLOVIS et HELIOS qu'il est possible de concevoir un système général réparti, permettant à la fois le dessin au trait et la synthèse d'images réalistes, et exploitant à fond toutes les spécificités cablées ou micro-programmées offertes par le matériel.

Outre l'aspect conceptuel, nous devrons également porter à l'actif de cette étude, les pas importants qui ont été faits vers la connaissance et la maîtrise des techniques de base de la visualisation, en particulier en ce qui concerne le remplissage des taches et l'élimination des parties cachées, de même que vers la détermination des modèles d'aspect de réflexion ou d'éclairage susceptibles d'assurer le réalisme des images synthétisées.

On peut regretter cependant, l'insuffisance des moyens matériels et humains qui nous a contraints à réaliser nos prototypes en version réduite, et à écarter provisoirement quelques axes intéressants. L'une des principales "lacunes" concerne la description et la modélisation des informations morphologiques et géométriques, pour lesquelles nous ne disposons actuellement que de quelques outils limités à des objets particuliers: polyèdres engendrés par révolution autour d'un axe, génération de terrains à partir de coupes transversales (AQT79), union de polyèdres convexes (DeT82).

La conception d'opérateurs interactifs de description et de modélisation des informations morphologiques et géométriques, apparaît donc comme un prolongement logique des travaux actuels. Il est à noter, en outre

que la structuration arborescente de la base de données et les avantages qu'elle procure, met à jour de nouvaux axes de recherche dans ce domaine. Ainsi, "l'historique" de la construction d'un objet (sous-objets constitutifs, opérateurs de placement, etc.) peut être conservé dans la structure, et exploité judicieusement lors de la visualisation, par exemple pour le calcul des coefficients statiques. Nous pensons qu'il s'agit là d'une direction prometteuse ayant des retombées importantes au niveau de la puissance et de la rapidité de l'interaction homme-machine.

En ce qui concerne les développements directs des outils actuels, nous citerons quelques uns des points qui sont d'ores et déjà à l'étude ou envisagés à court terme.

- L'industrialisation du synthétiseur HELIOS constitue notre objectif principal dans le domaine du matériel. Nous avons, à cette intention, engagé une étude visant à améliorer et à optimiser les techniques de réalisation, des opérateurs cablés. Dans le cadre d'un contrat financé par l'Agence de l'Informatique, ce travail aboutira à terme, à la conception d'un second prototype dont la fiabilité et le coût seront compatibles avec les normes industrielles.

- Parallèlement à cette activité, le domaine du logiciel offre également des perpectives de développement intéressantes, notamment vers l'extension du nombre et du niveau des éléments acceptés et des processus de visualisation et de description proposés. Au niveau du logiciel pilote, cet accroissement des possibilités devrait se traduire par une extention des outils interactifs et des opérateurs de description des informations visuelles: modèles de texture et de réflexion.

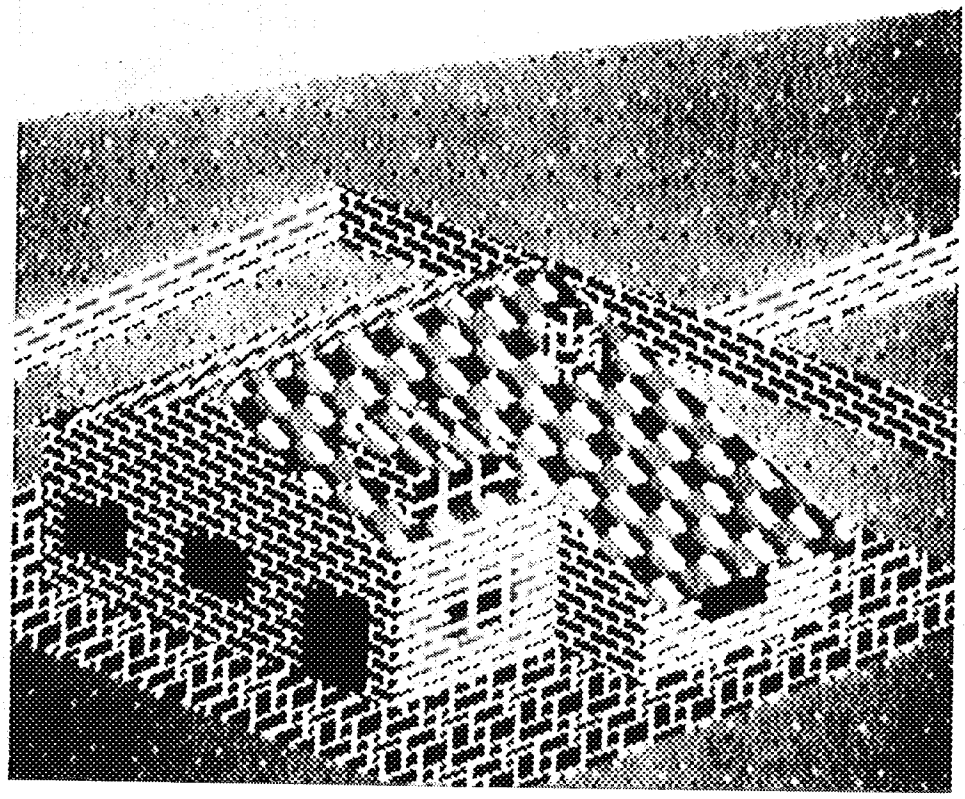
D'autres retombées directes de nos travaux nous paraissent devoir susciter, à plus long terme, de nouveaux thèmes de recherche. C'est le cas notamment de la détermination automatique des processus de visualisation. On peut en effet envisager la production automatique ou assistée de ces processus à partir de règles exprimant les caractéristiques des éléments considérés et les possibilités des diverses couches de synthétiseurs disponibles. Cette facilité permettrait alors d'obtenir des systèmes

réellement extensibles et adaptables aux exigences du plus grand nombre d'applications.

Une seconde voie d'investigation concerne l'obtention automatique des coefficients statiques utilisés dans notre approche de l'élimination des parties cachées. Ce problème évoqué brièvement au chapitre précédent, mérite un intérêt tout particulier compte-tenu de son influence directe sur l'espace nécessaire à l'archivage des données et sur les temps de calcul.

S'il est permis d'espérer que ces quelques pages montrent l'ampleur du chemin restant à parcourir, il n'en est pas moins sûr que les magnifiques photographies, publiées dans la littérature spécialisée, laissent à penser que la plupart des problèmes sont actuellement résolus. Ce paradoxe n'est qu'apparent et marque en fait la différence qui existe entre une image "luxueuse" composée à l'aide de millions de points nécessitant chacun un traitement long et coûteux, et une image "utile" offrant à l'utilisateur un réalisme moins spectaculaire au bénéfice d'une interactivité performante et d'un coût réduit.

Nous pensons que nos travaux vont directement dans cette dernière direction et nous espérons qu'il contribueront à prouver que l'image synthétique n'est pas l'apanage de quelques informaticiens privilégiés mais bien un outil puissant et efficace destiné à satisfaire et à susciter un nombre toujours croissant d'applications.



BIBLIOGRAPHIE

AgW81 B.D. Agkland, N.H. Weste
The Edge Flag Algorithm - A Fill Method for Raster Scan Display
IEEE Transactions on Computers, Vol C-30, No 1, 1-81

App67 A. Appel
The Notion of Quantitative Invisibility and The Machine Rendering of Solids
Proc. of ACM 2nd National Conference, 67

App68 A. Appel
Some Techniques for Shading Machine Rendering of Solids
proc. of the Spring Joint Computer Confrrence, Vol 32, 68

AQT80 J.O. Amberg, B. de Queylar, J.P. Tramoni
CIPAYE: Création interactive de paysages élémentaires
Projet d'élèves ingénieurs, Juin 80

AtW77 P. Atherton, K. Weiler
Hidden Surface Removal Using Polygon Area Sorting
Computer Graphics SIGGRAPH-ACM, vol 11, No 2, 77

AWG78 P. Atherton, K. Weiler, D. Greenberg
Polygon Shadow Generation
Computer Graphics, SIGGRAPH-ACM, vol 12, No 3, 78

Bag81 M. Bagherzadeh
Etude et réalisation d'un logiciel de description pour la synthèse d'image
Projet de D.E.A., Grenoble, Juin 81

BaH81 M. Bagherzadeh, M. Haxaire
Conception d'un logiciel pilote pour le prototype HELIOS
Projet d'élèves ingénieurs, Juin 81

Bas81 D. H. Bass
Using the Video Lookup Table for Reflectivity Calculations
Computer Graphics and Image Processing, No 17, 81

Bli77 J.F. Blinn
Models of Light Reflection for Computer Synthetized Pictures
Computer Graphics SIGGRAPH-ACM, vol 11, No 2, 77

Bli78 J.F Blinn
Simulation of Wrinkled Surfaces
Computer Graphics, SIGGRAPH-ACM, vol 12, No 3, 78

BlN76 J.F. Blinn, M.E. Newell
Texture and Reflection in Computer Generated Images
CACM vol. 19, No 10, oct 76

BlN78 J.F. Blinn, M.E. Newell
Clipping Using Homogeneous Coordinates
Computer Graphics, SIGGRAPH-ACM, vol 12, No 3, 78

Boi79 S. Boinodiris
Parallel Interactive Processing in Shaded Color Graphics
Ph. D. Thesis, North California State University, 5-79

Bou80 Ph. Boulle
Etude et réalisation d'algorithmes pour la visualisation de scènes composées de facettes planes.
Thèse de Docteur-Ingénieur, Grenoble, 9-80

Bra74 I.C. Braid
The Synthesis of Solids Bounded by Many Faces
CACM vol. 18, No 4, 4-74

Bre65 J.E Bresenham
Algorithm for Computer Control of a Digital Plotter
IBM System Journal, vol.4, No 1, 65

Bre77 J.E. Bresenham
An Incremental Algorithm for Digital Display of Circular Arcs
CACM vol 20, No 2, 2-77

Bre82 J.E. Bresenham
Incremental Line Compaction
The Computer Journal, vol 25, No 1, 82

BrF79 K.E. Brassel, R. Fegeas
An Algorithm for Shading of Regions on Vector Display Devices
Computer Graphics, SIGGRAPH-ACM, vol 13, No 3, 79

CaS80 E. Catmull, A.R. Smith
3-d Transformations of Images in Scanline Order
Computer Graphics, SIGGRAPH-ACM, vol. 14, No 3, 8-80

Cat74 E.E. Catmull
A Subdivision Algorithm for Computer Display of Curved Surfaces
ph. D. Thesis, University of Utah, 12-74

Cat78 E. Catmull
A Hidden Surface Algorithm with Anti-aliasing
Computer Graphics, SIGGRAPH-ACM, vol 12, No 3, 78

Cla76 J.H. Clark
Hierarchical Geometric Models for Visible Surface Algorithms
CACM vol 19, No 10, 10-76

Coh76 E. Cohen
A Method for Plotting Curves Defined by Implicit Equations
Computer Graphics, SIGGRAPH-ACM,76

Coo67 S.A. Coons
Surfaces for Computer Aided Design or Space-forms
MIT MAC TR-41, 6-67

Cot81 R.L. Cook, K.E. Torrance
A Reflectance Model for Computer Graphics
Computer Graphics, SIGGRAPH-ACM, vol. 15, No 3, 8-81

Cou80 F. Coupigny
Fabrication de dessins animés à l'aide du système PSYCHE-ANIM 2
Congrès AFCET-ITI, Nancy, 11-80

Cro77a F.C. Crow
Shadows Algorithms for Computer Graphics
Computer Graphics SIGGRAPH-ACM, vol 11, No 2, 77

Cro77 F.C. Crow
The Aliasing Problem in Computer-generated Shaded Images
CACM, vol 20, No 11, 11-77

CrP75 F.C. Crow, B.T. Phong
Improved Rendition of Polygonal Models of Curved Surfaces
2 nd USA-Japan Computer Conference, 75

Def82 M. Deprecq, M. Tavouka
Créations et Manipulations de Volumes
Projet d'élèves ingénieurs, 6-82

Dor79 M. Doros
Algorithms for Generation of Discrete Circles, Rings and Disks
Computer Graphics and Image Processing 10, 79

DSS78 W. Dungan, A. Strenger, G. Sutty
Texture Tile Considerations for Raster Graphics
Computer Graphics, SIGGRAPH-ACM, vol 12, No 3, 78

Ear77 R.A. Earnshaw
Line-Tracking for Incremental and Raster Devices
Computer Graphics SIGGRAPH-ACM, vol 11, No 2, 77

EEK79 R. Eckert, G. Enderle, K. Kansy, F.J. Prester
GKS79: Proposal of a Standard for a Graphic Kernel System
Congres EUROGRAPHICS, Bologna, 9-79

EEK80 J. Encarnacao, G. Enderle, K. Kansy et al.
The Workstation Concept of GKS and the Resulting Conceptual Differences to the GSPC Core System
Computer Graphics, SIGGRAPH-ACM, vol. 14, No 3, 8-80

Eng75 J. M. England et al.
A Display-Optimized Processor
Proc. 2 nd Annual Symposium on Computer Architecture, 1-75

Eve52 R.R. Everett
The Whirlwind I Computer
Joint AIEEE-IRE, Electr. Digit. Computer, 52

Fer81 F. Nunes Ferreira
Conception et réalisation d'un système interactif pour la synthèse d'images réalistes: HELIOS
Thèse de Docteur-Ingénieur, Grenoble, 9-81

FKN80 H. Fuchs, Z.M. Kedem, B.F. Naylor
On Visible Surface Generation by a Priori Tree Structure
Computer Graphics, SIGGRAPH-ACM, vol. 14, No 3, 8-80

FLC80 E. Feibush, M. Levoy, R.L. Cook
Synthetic Texturing Using Digital Filters
Computer Graphics, SIGGRAPH-ACM, vol. 14, No 3, 8-80

Fol76 J.D. Foley
A Tutorial on Satellite Graphics Systems
Computer, vol 9, No 8, 8-76

Ful78 K.S. Fu, S.Y. Lu
Computer Generation of Textures Using a Syntactic Approach
Computer Graphics, SIGGRAPH-ACM, vol 12, No 3, 78

GaM69 R. Galimberti, U. Montanari
An Algorithm for Hidden Line Elimination
CACM, vol 12, No 4, 4-69

Gou71 H. Gouraud
Computer Display of Curved Surfaces
Ph. D. Thesis, University of Utah, 6-71

Gra80 M. Grave Etude d'un noyau de système de synthèse d'images. Application à la visualisation de scènes tridimensionnelles
Thèse de Docteur-Ingénieur, Lille, 12-80

Gr378 D. Greenberg, G.H. Joblove
Color Spaces for Computer Graphics
Computer Graphics, SIGGRAPH-ACM, vol 12, No 3, 78

GSP77 Status Report of the Graphics Standard Planning Commitee
Computer Graphics SIGGRAPH-ACM, vol 11, No 2, 77

GSP79 Status Report of the Graphics Standard Planning Commitee
Computer Graphics, SIGGRAPH-ACM, vol 13, No 3, 79

GSS81 S. Gupta, R. Sproull, I.E. Sutherland
A VLSI Architecture for Updating Raster-Scan Displays
Computer Graphics, SIGGRAPH-ACM, vol. 15, No 3, 8-81

Heg82 G. Hégron
Techniques de remplissage de taches pour une surface à pointillage
RR No 6, Informatique, Université de Nantes, 10-82

Hvi79 H. Hvistendahl
IGL: A Structured Langage for Interactive Graphics
Congres EUROGRAPHICS, Bologna, 9-79

Job79 G.H. Joblove
Color Spaces and Computer Graphics
Master's thesis, Cornell University, N.Y., 1-79

Ken77 J.R. Kender
Instabilities in Color Transformations
Pattern recognition and image processing, 6-77

Led79 A. Ledru
La structuration dans les logiciels graphiques interactifs. Etude et réalisation d'un prototype
Projet de D.E.A., Grenoble, 9-79

Lee81 D.T. Lee
Shading of Regions on Vector Display Devices
Computer Graphics, SIGGRAPH-ACM, vol. 15, No 3, 8-81

Ler80 P. Leray
Réalisation d'un système de génération synthétique d'images en temps réel: GSI
Congrès AFCET-TTI, Nancy, 11-80

Lev80 J. Z. Levin
QUADRIL: A Computer Language for the Description of Quadric-Surface Bodies
Computer Graphics, SIGGRAPH-ACM, vol. 14, No 3, 8-80

Lie78 H. Lieberman
How to Color in a Coloring Book
Computer Graphics, SIGGRAPH-ACM, vol 12, No 3, 78

LLM76 A. Leduc-Leballeur, M. Lucas, F. Martinez
GRIGRI: Logiciel de base pour l'utilisation des consoles de visualisation du CICG
Note technique No 49, 12-76

LLM78 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
revue RAIRO Informatique, vol.12, No 2, 78

Luc77 M. Lucas
Contribution à l'étude des techniques de communication graphique avec un ordinateur. Eléments de base des logiciels graphiques interactifs
Thèse d'état, Grenoble, 12-77

Luc77b M. Lucas
Technologie des consoles de visualisation: présent et perspectives
L'onde électrique, vol. 57, No 12, 12-77

Maß79 P.C. Maxwell, P.W. Baker
The Generation of Polygons Representing Circles, Ellipses and Hyperbolas
Computer Graphics and Image Processing 10, 79

Mah72 R. Mahl
Visible Surface Algorithm for Quadric Patches
IEEE Trans. Computers Vol c-21, No 1, 1-72

MaF81 F. Martinez, F. Ferreira
HELIOS: Terminal interactif pour la synthèse d'images réalistes
Congrès AFCET-ITI, Gif sur Yvette, 11-81

MaF82 F. Martinez, F. Ferreira
HELIOS: Terminal vidéo interactif pour la synthèse d'images rèalistes
Le Nouvel Automatisme, NO 30, 5-82

Mar77a F. Martinez
Etude des problèmes de conception et de réalisation d'animation: Le système SAFRAN
Thèse de 3 eme cycle, Grenoble, 5-77

Mar77b F. Martinez
Techniques de passage d'un dessin à un autre par déformations successives. Application à un système d'animation
Rapport de recherche No 65, 1-77

Mar78 F. Martinez
Les concepts liés à la description de l'animation
Congrès AFCET-TTI, Gif sur Yvette, 11-78

Mar79a F. Martinez
La synthèse d'image: revue bibliographique
Convention LMT-MICADO, Grenoble, 1-79

Mar79b F. Martinez
An Approach to the Modelling and Display of Lanscapes
Congres EUROGRAPHICS, Bologna, 9-79

Mar80 F. Martinez
CLOVIS: Complexe Logiciel pour la Visualisation Interactive Structurée
Congrès AFCET-TTI, Nancy, 11-80

Mar81 F. Martinez
Utilisation de textures pour la synthèse d'images réalistes
Journées INA: Les nouvelles images, Arc et Senans, 6-81

Mas81 C. Marson
Terminaux: la couleur en plus
01 Informatique, No 147, 2-81

Mat78 P. Matherat
A Chip for Low-Cost Raster-Scan Graphic Display
Computer Graphics, SIGGRAPH-ACM, vol 12, No 3, 78

Max81 N.L. Max
Vectorized Procedural Models for Natural Terrains: Waves and Island in the Sunset
Computer Graphics, SIGGRAPH-ACM, vol. 15, No 3, 8-81

MeG80 G.W. Meyer, D.P. Greenberg
Perceptual Color Spaces for Computer Graphics
Computer Graphics, SIGGRAPH-ACM, vol. 14, No 3, 8-80

Mer79 M. Meriaux
Etude et réalisation d'un terminal graphique couleur tridimensionnel fonctionnant par taches
Thèse de Docteur-Ingénieur, Lille, 1-79

Mer81 M. Meriaux
Architectures adaptées à la synthèse d'images
Congrès AFCET-TTI, Gif sur Yvette, 11-81

MoL76 P. Morvan, M. Lucas
Images et ordinateur - Introduction à l'infographie interactive
ed. LAROUSSE, Série Informatique, 9-76

Mun39 A.H. Munsell
A Color Notation
8th ed., Munsell Color Co., Baltimore, 39

Mye75 A.J. Myers
An Efficient Visible Surface Program
Tech. Report No 74-00768, 7-75

NaN74 T. Nishita, E. Nakamae
An Algorithm for Half-toned Representation of 3-D Objets
Inf. Process. Japan, vol 14, 74

NeS79 W. Newman, R.F. Sproull
Principles of Interactive Computer Graphics
2nd ed., Mc Graw-Hill, New York, 79

NNS72 M.E. Newell, R.G. Newell, T.L. Sancha
A New Approach to the Shaded Picture Problem
Proc. of ACM National Conference, Vol 1, Boston, 8-72

Pav78 T. Pavlidis
Filling in Raster Graphics
Computer Graphics, SIGGRAPH-ACM, vol. 15, No 3, 8-81

Pay82 J.L. Payan
Etude et réalisation d'un sytème interactif de description et d'archivage de textures
Projet de D.E.A., Grenoble, 9-82

Pho75 B.T. Phong
Illumination for Computer Generated Pictures
CACM, vol 18, No 6, 6-75

PGC82 D. Perny, M. Gangnet, Ph. Coueignoux
Perspective Mapping of Planar Textures
Computer Graphics, vol.16, No 1, 5-82

PIG82 M.L. Pitteway, A.J. Green
Bresenham's Algorithm with Run Line Coding
The Computer Journal, vol.25, No 1, 1-82

Por78 T.K. Porter
Spherical Shading
Computer Graphics, SIGGRAPH-ACM, vol 12, No 3, 78

Pur70 Purdue University
Thermophysical Properties of Matter
vol 7-8-9, 70

RoA76 D.F. Rogers, J.A. Adams
Mathematical Elements for Computer Graphics
Mc Graw-Hill, New York, 76

Rob63 L.G. Roberts
Machine Perception of Three-Dimensionnal Solids
TR 315, MIT Lincoln Laboratory, 5-63

RuW80 S.M. Rubin, J.T. Whitted
A 3-Dimensional Representation for Fast Rendering of Computer Scenes
Computer Graphics, SIGGRAPH-ACM, vol. 14, No 3, 8-80

Sar82 M.T. Sarrasin
Analyse et implémentation du logiciel pilote d'un synthétiseur d'images
Mémoire d'ingénieur CNAM, Grenoble, 10-82

SBG69 R.A. Schumacker, B. Brand, M. Gilliland, W. Sharp
Study for Applying Computer Generated Images to Visual Simulation
TR 69-14, U.S. Air Force Laboratory, 9-69

Sha67 A.C. Shaw
A Proposal Language for the Formal Description of Pictures
TR GSG-28, Standford Lin. Accel. Center, 67

Sha80 U. Shani
Filling Regions in Binary Raster Images: A Graph-Theoretic Approach
Computer Graphics, SIGGRAPH-ACM, vol. 14, No 3, 8-80

SKK79 Y. Suenaga, T. Kamae, T. Kobayashi
A High-Speed Algorithm for the Generation of Straight Lines and Circular Arcs
IEEE Trans. on Computers, vol C-28, NO 10, 10-79

Smi78 A.R. Smith
Color Gamut Transform Pair
Computer Graphics, SIGGRAPH-ACM, vol 12, No 3, 78

Smi79 A.R. Smith
Tint Fill
Computer Graphics, SIGGRAPH-ACM, vol 13, No 3, 79

SpS68 R.F. Sproull, I.E. Sutherland
A Clipping Divider
AFIPS FJCC Conf. Proc., vol 33, 68

SSS74 I.E. Sutherland, R.F. Sproull, R.A. Schumacker
A Characterization of Ten Hidden-Surface Algorithms
ACM Computing Surveys, vol 6, No 1, 3-74

Sta78 J. Staudhammer
On Display of Space Filling Atomic Models in Real Time
Computer Graphics, SIGGRAPH-ACM, vol 12, No 3, 78

StT79 J. Staudhammer, J. Tartar
Shaded Graphics Hardware
Congres EUROGRAPHICS, Bologna, 9-79

SuH74 I.E. Sutherland, G.W. Hodgman
Reentrant Polygon Clipping
CACM, vol.17, No 1, 1-74

Sut63 I.E. Sutherland
SKETCHPAD: A Man Machine Graphical Communication System
AFIPS SJCC Conf. Proc., vol. 23, 63

Sut74 I.E. Sutherland
Three Dimensionnal Data Input by Tablet
Proc. IEEE, vol 62, No 4, 4-74

Tar81 M. Tartes et al.
Etude et réalisation d'un logiciel pour la visualisation interactive structurée
Projet d'élèves ingénieurs, 6-81

The72 F. Theron
Sur le programme EUCLID: création, manipulation et visualisation de formes tridimensionnelles dans un langage géométrique
Thèse de 3ème cycle, Paris, 72

ToS67 K.E. Torrance, E.M. Sparrow
Theory for Off-Specular Reflection from Roughened Surfaces
J. Optical Soc. of America, vol 57, No 9, 9-67

Tru81 J.R. Truckenbrod
Effective Use of Color in Computer Graphics
Computer Graphics, SIGGRAPH-ACM, vol. 15, No 3, 8-81

VCV77 J. Van Der Bos, L.C. Caruthers, A. Van Dam
GPGS: A Device Indépendant General Purpose Graphic System for Stand Alone and Satellite Graphics
Computer Graphics SIGGRAPH-ACM, vol 11, No 2, 77

WaK79 J. Warner, N. Kiefhaber
The DIGRAF Implementation of the Proposed GSPC Standard
Congres EUROGRAPHICS, Bologna, 9-79

War69 J. E. Warnock
A Hidden Surface Algorithm for Computer Generated Half-Tone Pictures
TR No 4-15, University of Utah, 6-69

Wat70 G. S. Watkins
A Real-Time Visible Surface Algorithm
TR UTEC CSc-7o-101, University of Utah, 6-70

Wei80 C. F. Weiman
Continuous Anti-aliased Rotation and Zoom of Raster Images
Computer Graphics, SIGGRAPH-ACM, vol. 14, No 3, 8-80

Wei81 R. Weinberg
Parallel Processing Image Synthesis and Anti-Aliasing
Computer Graphics, SIGGRAPH-ACM, vol. 15, No 3, 8-81

Whi80 J.T. Whitted
An Improved Illumination Model for Shaded Display
CACM, vol. 23, No. 6, 6-80

dernière page de la thèse

AUTORISATION DE SOUTENANCE

VU les dispositions de l'article 5 de l'arrêté du 16 avril 1974,

VU les rapports de présentation de Messieurs

. V. CORDONNIER, Professeur

. M. LUCAS, Professeur

. J. MERMET, Maître de recherche

Monsieur MARTINEZ Francis

est autorisé à présenter une thèse en soutenance pour l'obtention du grade de DOCTEUR D'ETAT ES SCIENCES.

Fait à Grenoble, le 21 octobre 1982

Le Président de l'U.S.M.G

Le Président

Le Président de l'I.N.P.-G.

D. BLOCH
Président
de l'Institut National Polytechnique
de Grenoble

Lorsque U est situé devant V seules les faces appartenant aux objets U0 et V1 sont visibles alors que dans le cas contraire seules les faces des objets V0 et U1 sont rendues visibles. Dans chaque cas la translation en X est ajustée (+ou- 256) pour laisser chaque objet à sa position primitive. Toutes ces opérations prennent place pendant le temps imparti au retour de trame.

3.2.3. Autres transformations

Bien que l'architecture d'HELIOS ne permette pas les modifications en temps réel des attributs morphologiques et géométriques des objets, il est possible d'obtenir un certain degré d'animation en étendant la technique présentée ci-dessus. On peut, en effet, si l'objet à animer est de taille relativement réduite, mémoriser sur un plan d'identification quelques attitudes clés de cet objet sur sa trajectoire. La figure 45 illustre le cas d'un objet représenté à l'aide de 16 attitudes enregistrées sur des carrés de 128 * 128 pixels. Etant donné que le nombre maximum de faces est 1024, chaque altitude de l'objet ne peut nécessiter plus de 1024/16 = 64 faces. Le choix d'une attitude s'effectue à l'aide de deux opérations:

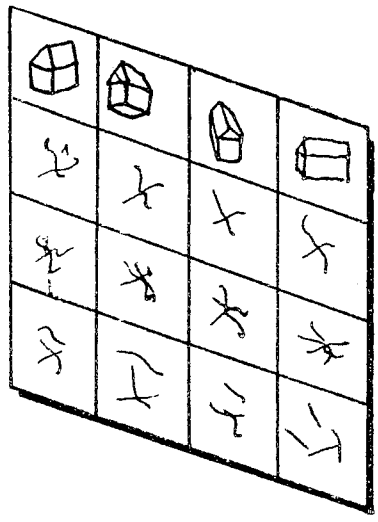


Figure VII.45: Pré-enregistrement d'attitudes-clés

* translation de plan pour situer l'attitude choisie à sa bonne place,
* invisibilité de l'attitude précédente et visibilité de l'attitude choisie.

Il est à noter que trois techniques présentées ci-dessus peuvent être utilisées simultanément pour simuler une animation tridimensionnelle.

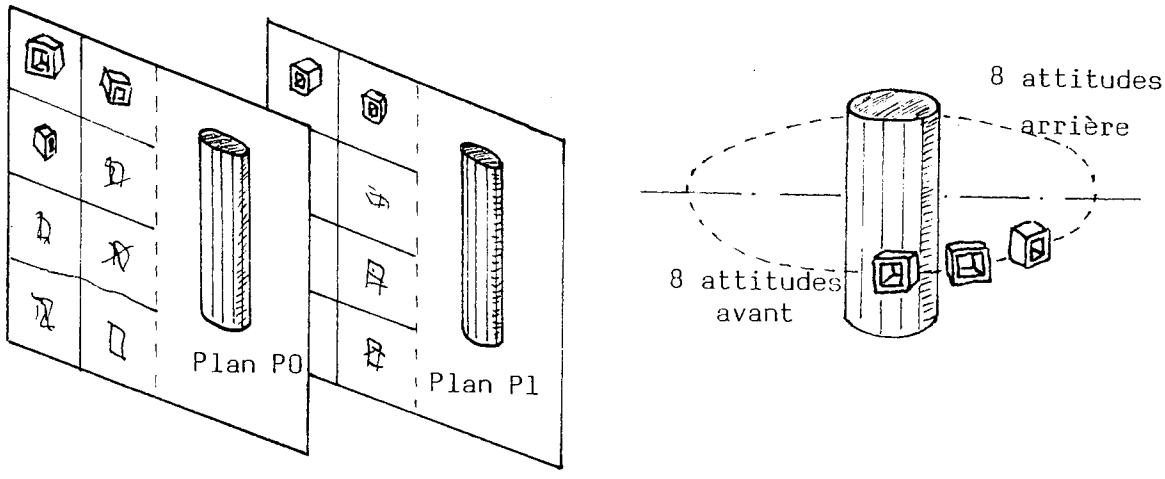


Figure VII.46: Animation tri-dimensionnelle

Dans l'exemple de la figure 46, l'animation à reproduire est celle de la rotation d'un cube perforé autour d'un cylindre mobile en translation. Pendant la demi-période durant laquelle le cube est situé en avant, s'effectuent simultanément:

* la translation du cylindre visible sur P1 seulement,
* la visibilité de l'attitude du cube sur P0 correspondant à l'angle de rotation,
* la translation de P0 pour placer cette attitude sur la trajectoire.

Wil78 L. williams
Casting Curved Shadows on Curved Surfaces
Computer Graphics, SIGGRAPH-ACM, vol 12, No 3, 78

Woo71 P.A. Woodsford
The Design and Implementation of the GINO 3D Graphics Software Package
Software-Practice and Experience, vol. 1, No 4, 10-71

Wri73 T.J. wright
A Two-Space Solution to the Hidden Line Problem for Plotting Functions of Two Variables
IEEE Trans. Computers, vol.c-22, No 1, 1-73

Zuc75 S. W. Zucker
Toward a Model of Texture
Computer Graphics and Image Processing, No 5, 6-75