N° d'ordre : 1123

UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE

THÈSE

présentée à

L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE

pour obtenir le titre de

DOCTEUR DE 3ème CYCLE

INFORMATIQUE

par

Philippe DURIF

ETUDE D'UNE MACHINE PARALLELE DE SYNTHESE D'IMAGE A DECOUPAGE PAR OBJET

Thèse soutenue le 8 décembre 1983 devant la Commission d'Examen

MEMBRES DU JURY :

Président :	M. DAUCHET
Rapporteur :	V. CORDONNIER
Examinateurs :	M. MERIAUX
	M. MARTINEZ
	M. BALTER

P R O F E S S E U R S CLASSE EXCEPTIONNELLE

M. CONSTANT Eugène	I.E.E.A.
M. FOURET René	Physique
M. GABILLARD Robert	I.E.E.A.
M. MONTREUIL Jean	Biologie
M. PARREAU Michel	Mathématiques
M. TRIDOT Gabriel	Chimie
M. VIVIER Emile	Biologie
M. WERTHEIMER Raymond	Physique

P R O F E S S E U R S 1ère classe

M. BACCHUS Pierre	Mathématiques
M. BEAUFILS Jean-Pierre (dét.)	Chimie
M. BIAYS Pierre	G.A.S.
M. BILLARD Jean (dét.)	Physique
M. BOILLY Bénoni	Biologie
M. BOIS Pierre	Mathématiques
M. BONNELLE Jean-Pierre	Chimie
M. BOUGHON Pierre	Mathématiques
M. BOURIQUET Robert	Biologie
M. BREZINSKI Claude	I.E.E.A.
M. CELET Paul	Sciences de la Terre
M. CHAMLEY Hervé	Biologie
M. COEURE Gérard	Mathématiques
M. CORDONNIER Vincent	I.E.E.A.
M. DEBOURSE Jean-Pierre	S.E.S.
M. DYMENT Arthur	Mathématiques

PROFESSEURS 1ère classe (suite)

M. ESCAIG Bertrand	Physique
M. FAURE Robert	Mathématiques
M. FOCT Jacques	Chimie
M. GRANELLE Jean-Jacques	S.E.S.
M. GRUSON Laurent	Mathématiques
M. GUILLAUME Jean	Biologie
M. HECTOR Joseph	Mathématiques
M. LABLACHE COMBIER Alain	Chimie
M. LACOSTE Louis	Biologie
M. LAVEINE Jean Pierre	Sciences de la Terre
M. LEHMANN Daniel	Mathématiques
Mme LENOBLE Jacqueline	Physique
M. LHOMME Jean	Chimie
M. LOMBARD Jacques	S.E.S.
M. LOUCHEUX Claude	Chimie
M. LUCQUIN Michel	Chimie
M. MIGEON Michel Recteur à Grenoble	E.U.D.I.L.
M. MIGNOT Fulbert (dét.)	Mathématiques
M. PAQUET Jacques	Sciences de la Terre
M. PROUVOST Jean	Sciences de la Terre
M. ROUSSEAU Jean-Paul	Biologie
M. SALMER Georges	I.E.E.A.
M. SEGUIER Guy	I.E.E.A.
M. SIMON Michel	S.E.S.
M. STANKIEWICZ François	S.E.S.
M. TILLIEU Jacques	Physique
M. VIDAL Pierre	I.E.E.A.
M. ZEYTOUNIAN Radyadour	Mathématiques

P R O F E S S E U R S 2ème classe

M. ANTOINE Philippe	Mathématiques (Calais)
M. BART André	Biologie
Mme BATTIAU Yvonne	Géographie
M. BEGUIN Paul	Mathématiques
M. BELLET Jean	Physique
M. BERZIN Robert	Mathématiques
M. BKOUCHE Rudolphe	Mathématiques
M. BODARD Marcel	Biologie
M. BOSQ Denis	Mathématiques
M. BRASSELET Jean-Paul	Mathématiques
M. BRUYELLE Pierre	Géographie
M. CAPURON Alfred	Biologie
M. CARREZ Christian	I.E.E.A.
M. CAYATTE Jean-Louis	S.E.S.
M. CHAPOTON Alain	C.U.E.E.P.
M. COQUERY Jean-Marie	Biologie
Mme CORSIN Paule	Sciences de la Terre
M. CORTOIS Jean	Physique
M. COUTURIER Daniel	Chimie
M. CROSNIER Yves	I.E.E.A.
M. CURGY Jean-Jacques	Biologie
Mle DACHARRY Monique	Géographie
M. DAUCHET Max	I.E.E.A.
M. DEBRABANT Pierre	E.U.D.I.L.
M. DEGAUQUE Pierre	I.E.E.A.
M. DELORME Pierre	Biologie
M. DELORME Robert	S.E.S.
M. DE MASSON D'AUTUME Antoine	S.E.S.
M. DEMUNTER Paul	C.U.E.E.P.

PROFESSEURS 2ème classe (Suite 1)

M. DENEL Jacques	I.E.E.A.
M. DE PARIS Jean-Claude	Mathématiques (Calais)
Mle DESSAUX Odile	Chimie
M. DEVRAINNE Pierre	Chimie
M. DHAINAUT André	Biologie
Mme DHAINAUT Nicole	Biologie
M. DORMARD Serge	S.E.S.
M. DOUKHAN Jean-Claude	E.U.D.I.L.
M. DUBOIS Henri	Physique
M. DUBRULLE Alain	Physique (Calais)
M. DUBUS Jean-Paul	I.E.E.A.
M. FAKIR Sabah	Mathématiques
M. FONTAINE Hubert	Physique
M. FOUQUART Yves	Physique
M. FRONTIER Serge	Biologie
M. GAMBLIN André	G.A.S.
M. GLORIEUX Pierre	Physique
M. GOBLOT Rémi	Mathématiques
M. GOSSELIN Gabriel (dét.)	S.E.S.
M. GOUDMAND Pierre	Chimie
M. GREGORY Pierre	I.P.A.
M. GREMY Jean-Paul	S.E.S.
M. GREVET Patrice	S.E.S.
M. GUILBAULT Pierre	Biologie
M. HENRY Jean-Pierre	E.U.D.I.L.
M. HERMAN Maurice	Physique
M. JACOB Gérard	I.E.E.A.
M. JACOB Pierre	Mathématiques
M. JEAN Raymond	Biologie
M. JOFFRE Patrick	I.P.A.

PROFESSEURS 2ème classe (suite 2)

M. JOURNEL Gérard	E.U.D.I.L.
M. KREMBEL Jean	Biologie
M. LANGRAND Claude	Mathématiques
M. LATTEUX Michel	I.E.E.A.
Mme LECLERCQ Ginette	Chimie
M. LEFEVRE Christian	Sciences de la Terre
Mle LEGRAND Denise	Mathématiques
Mle LEGRAND Solange	Mathématiques (Calais)
Mme LEHMANN Josiane	Mathématiques
M. LEMAIRE Jean	Physique
M. LHENAFF René	Géographie
M. LOCQUENEUX Robert	Physique
M. LOSFELD Joseph	C.U.E.E.P.
M. LOUAGE Francis(dét.)	E.U.D.I.L.
M. MACKE Bruno	Physique
M. MAIZIERES Christian	I.E.E.A.
M. MESSELYN Jean	Physique
M. MESSERLIN Patrick	S.E.S.
M. MONTEL Marc	Physique
Mme MOUNIER Yvonne	Biologie
M. PARSY Fernand	Mathématiques
Mle PAUPARDIN Colette	Biologie
M. PERROT Pierre	Chimie
M. PERTUZON Emile	Biologie
M. PONSOLLE Louis	Chimie
M. PORCHET Maurice	Biologie
M. POVY Lucien	E.U.D.I.L.
M. RACZY Ladislas	I.E.E.A.
M. RAOULT Jean François	Sciences de la Terre
M. RICHARD Alain	Biologie

PROFESSEURS 2ème Classe (suite 3)

M. RIETSCH François	E.U.D.I.L.
M. ROBINET Jean-Claude	E.U.D.I.L.
M. ROGALSKI Marc	Mathématiques
M. ROY Jean-Claude	Biologie
M. SCHAMPS Joël	Physique
Mme SCHWARZBACH Yvette	Mathématiques
M. SLIWA Henri	Chimie
M. SOMME Jean	G.A.S.
Mle SPIK Geneviève	Biologie
M. STAROSWIECKI Marcel	E.U.D.I.L.
M. STERBOUL François	E.U.D.I.L.
M. TAILLIEZ Roger	Institut Agricole
Mme TJOTTA Jacqueline (dét.)	Mathématiques
M. TOULOTTE Jean-Marc	I.E.E.A.
M. TURRELL Georges	Chimie
M. VANDORPE Bernard	E.U.D.I.L.
M. VAST Pierre	Chimie
M. VERBERT André	Biologie
M. VERNET Philippe	Biologie
M. WALLART Francis	Chimie
M. WARTEL Michel	Chimie
M. WATERLOT Michel	Sciences de la Terre
Mme ZINN JUSTIN Nicole	Mathématiques

CHARGES DE COURS

M. ADAM Michel	S.E.S.

CHARGES DE CONFERENCES

M. BAFCOP Joël	I.P.A.
M. DUVEAU Jacques	S.E.S.
M. HOFLACK Jean	I.P.A.
M. LATOUCHE Serge	S.E.S.
M. MALAUSSENA DE PERNO Jean-Louis	S.E.S.
M. NAVARRE Christian	I.P.A.
M. OPIGEZ Philippe	S.E.S.

REMERCIEMENTS

Je voudrais remercier

Monsieur Le Professeur Max DAUCHET qui a bien voulu présider le Jury de cette Thèse.

Monsieur Le Professeur Vincent CORDONNIER qui m'a confié ce sujet et m'a aidé de ses critiques.

Monsieur Michel MERIAUX, chargé de recherche au CNRS, qui n'a pas hésité à consacrer son temps pour me fournir réconfort et soutien scientifique.

Messieurs Francis MARTINEZ, Maître-Assistant à l'Université de Grenoble et BALTER, Ingénieur de Recherche à la BULL, qui m'ont fait l'honneur de participer à ce Jury.

Enfin mes remerciements vont à Madame Bénédicte VANDROEMME, qui a tapé ce document, à Madame DEBOCK et Monsieur GLANC, qui en ont assuré la réalisation matérielle ; ainsi qu'à tous les membres du Laboratoire qui ont créé un climat de travail agréable.

TABLE DES MATIERES

CHAPITRE II CARACTERISATION DES SYSTEMES GRAPHIQUES

CHAPITRE III CALCUL INCREMENTAL

CHAPITRE IV ARCHITECTURE PROPOSEE

CHAPITRE V PROPOSITION DE REALISATION DES PROCESSEURS ELEMENT

INTRODUCTION

Si les premiers temps de l'informatique furent marqués, de la part des informaticiens et des électroniciens, par le souci de tirer le meilleur parti des calculateurs en termes d'efficacité à l'exécution et d'encombrement de la mémoire, il est vite apparu, la complexité des traitements augmentant, nécessaire de favoriser la communication homme-machine. En effet, dans de nombreux domaines, l'ordinateur devient un outil de décision pour l'utilisateur (CAO, EAO, bases de données, intelligence artificielle,...). Par exemple, un message codé sous forme de zéros et de uns est moins lisible que s'il est représenté en caractères alphanumériques.

Ainsi se sont développés, dans le contexte informatique, différents aspects de la communication : communication textuelle, orale, picturale. Le grand intérêt de l'image comme support de communication est qu'elle est appréhendable comme un tout, à la différence d'un texte ou d'une phrase qu'il peut lire ou écouter du début à la fin. Son inconvénient principal est de proposer une information plus qualitative que quantitative (certains systèmes pallient à cette limitation en incorporant la génération de textes à celle d'images).

Ce travail trouve sa place dans le cadre de la synthèse d'images, qui est le processus de communication picturale allant dans le sens machine-homme. Nous ne proposons pas de concepts nouveaux mais une architecture spécialisée réalisant, en temps réel, quelques traitements propres à la synthèse d'images, et répondant à des contraintes exigentes de modularité.

Le chapitre I est une introduction à quelques aspects de la synthèse d'images. Le chapitre II est une étude bibliographique des modèles proposés dans la littérature pour l'analyse des systèmes graphiques. Il présente en outre un essai de classification des architectures graphiques. Le troisième chapitre décrit plusieurs algorithmes de calcul incrémental. L'architecture générale du système proposé fait l'objet du quatrième chapitre, le cinquième en détaille les différentes composantes.

On trouvera en annexe les schémas des mécanismes les plus spécifiques ainsi qu'une étude de performance et de faisabilité du système proposé.

Une bibliographie classée par thèmes puis par ordre alphabétique termine le document.

CHAPITRE I

GENERALITES

I.1. INTRODUCTION

Ce chapitre tente de préciser certaines notions spécifiques à l'informatique graphique. On peut consulter à ce sujet [MORVAN 76], [NEWMAN 79] et [FOLEY 82]. Il ne prétend pas être une étude exhaustive de la synthèse d'images, les références bibliographiques donneront, nous l'espérons, les moyens d'approfondir certains aspects qui ne sont que mentionnés.

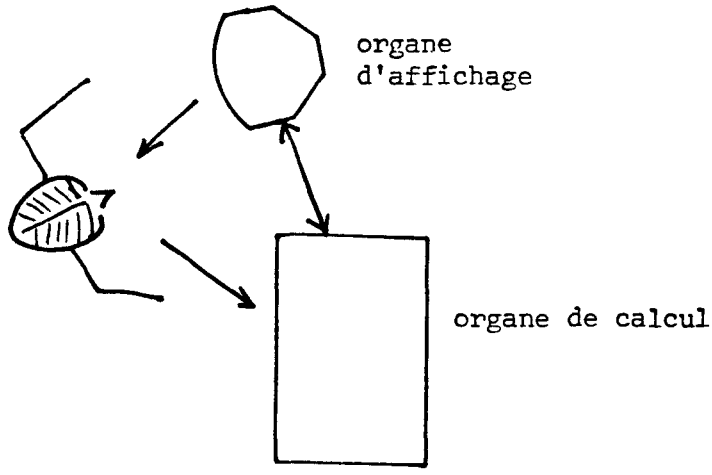
I.2. NOTIONS DE SYSTEME GRAPHIQUE

Dans une première approche nous distinguerons deux parties dans un système graphique :

- L'organe d'affichage qui peut être une table traçante, une imprimante, un écran de balayage de trame ou cavalier. Le tout étant muni d'un minimum de capacité de mémoire et de traitement spécifique au support d'affichage.

- L'organe de calcul qui complète l'organe d'affichage en le dotant de moyens importants de traitements, de communication et de stockage.

Ce système graphique forme alors une boucle ouverte : il lui manque un utilisateur.



Il y aura interaction si l'utilisateur possède au moins un moyen de modifier l'état du système, au niveau de l'organe d'affichage ou à celui de l'organe de calcul.

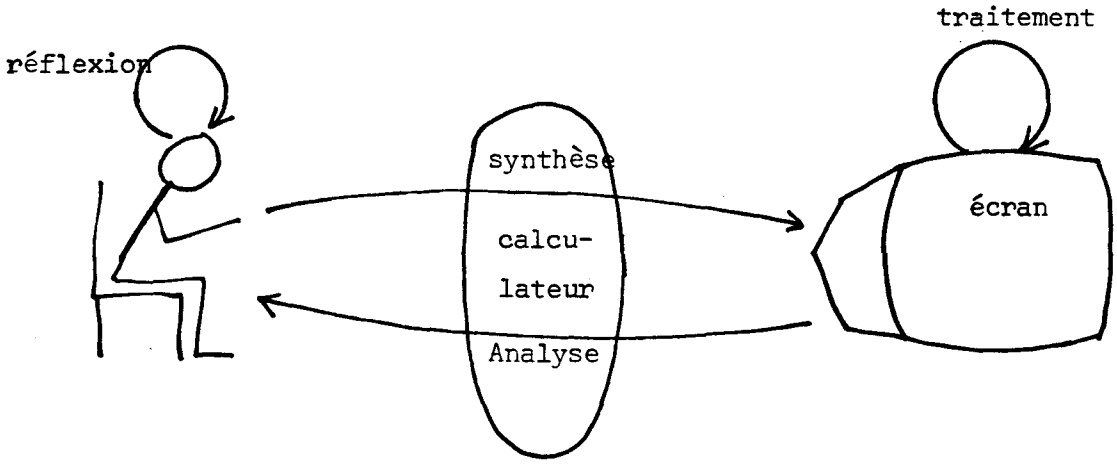
Soulignons qu'il est important que toute interaction subie par un organe puisse se répercuter dans l'autre. Si cette communication est le plus souvent triviale dans le sens organe de calcul organe d'affichage, elle l'est moins dans l'autre sens. Ceci est dû à la faible structuration de l'information image et donc à sa difficulté de manipulation [CEI 82] pp. 143-145, [FOLEY 82] pp. 127-169, pp. 183-215.

I.3. NOTIONS DE SYNTHESE, DE TRAITEMENT, D'ANALYSE

L'interprétation d'un fichier de résultats exprimés sous forme numérique, pour précise qu'elle soit n'en est pas moins longue et fastidieuse, voire impossible. Une des vocations premières de l'informatique graphique a été de faire les premiers pas de ce travail d'interprétation en proposant à l'utilisateur une image qualitative des résultats de calculs. Le tracé de courbes en deux, trois et même quatre dimensions en est un exemple.

Réciproquement l'informatique graphique permet de tirer des renseignements d'ordre numérique d'une image. Par exemple, à partir d'une photographie de satellite prise en infrarouge puis numérisée, l'utilisateur désire faire apparaître en rouge les zones de températures supérieures à trente degrés et en bleu celles qui sont inférieures ; il désire ensuite connaître les surfaces de chacune des zones.

La première approche correspond à la "synthèse" d'une image (dessin de la courbe à partir de couples de coordonnées), la seconde reflète le "traitement" d'image (coloriage des zones) et l'"analyse" de la nouvelle image (évaluation des aires). Il convient de remarquer qu'aucun de ces trois traitements n'ajoute d'information. La figure suivante résume ce qui précède :



Les processus de synthèse et d'analyse sont alors complémentaires.

I.4. NOTIONS D'IMAGE ET DE DESSIN

Une image est considérée comme un ensemble de taches colorées, un dessin est formé d'un ensemble de traits [LUCAS 77]. Un simulateur de conduite produira des images plutôt que des dessins, la représentation graphique de la courbe d'une fonction sera un dessin. Le système graphique décrit dans ce travail est destiné à produire des images.

Cette séparation n'est pas toujours aussi nette, ceci d'autant plus que les logiciels de tracé de dessins et de construction d'images utilisent souvent les mêmes algorithmes de base ; c'est par exemple le cas de l'interpolation linéaire [BRESENHAM 65] :

- Pour le tracé de dessins (sur table traçante ou écran à balayage cavalier).

- Les algorithmes d'éclairement de GOURAUX, de PHONG, ou l'algorithme de surfaces visibles de WARNOCK, qui sont typiques de la synthèse d'images, utilisent massivement l'interpolation linéaire [GOURAUD 71], [PHONG 75], et [WARNOCK 69].

I.5. L'UNIVERS

Les systèmes graphiques possèdent en général plusieurs représentations des données nécessaires à l'affichage. Nous appelons "univers" l'ensemble qui à tout instant contient la totalité des objets affichables.

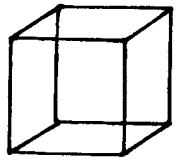
Cet univers est contenu dans la structure de données du programme d'application et aussi dans la scène obtenue par les logiciels de description [LUCAS 77].

Ajoutons que les différentes représentations graphiques se réfèrent à un espace de calcul généralement spécifique à chacune d'entre-elles. Par exemple, l'univers pourra être représenté dans un espace rationnel à n+3 dimensions (3 pour la géométrie, n pour d'autres variables) alors que l'écran lui-même utilise généralement un espace d'entiers à p+2 dimensions (deux dimensions géométriques ; p dimensions décrivant l'état d'un pixel, p=1 pour écran noir et blanc, p=3 pour un écran couleur ou chacune des dimensions correspond à une fondamentale [FOLEY 82] p. 110, [MORVAN 76] pp. 29-31).

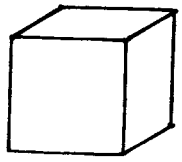
I.6. LA SYNTHESE D'IMAGE

Dans ce paragraphe nous nous intéressons à la synthèse d'images "réalistes" c'est-à-dire au processus qui, à partir d'un univers "réel", fabrique une image destinée à un observateur humain. Cette synthèse doit alors reproduire ou simuler le fonctionnement de l'oeil de l'observateur (perception des couleurs, limitation du champs de vision...) et les mécanismes physiques de transmission de la lumière (propagation rectiligne en milieu homogène, réflexion, diffusion ...), pour autant que les connaissances physiologiques et physiques rendent compte de la réalité.

Il faut souligner que les termes "réel" et "réalisme" ont ici des sens opposés. En effet l'univers réel est supposé contenir des objets décrits "tels qu'ils sont", alors qu'une image réaliste décrit les objets tels qu'on a l'habitude de les percevoir et donc de façon limitée :



Représentation réelle d'un cube, donc non réaliste. L'ambiguité classique "face avant" "face arrière" n'en est pas une puisqu'une telle représentation n'est pas observable par un spectateur humain.



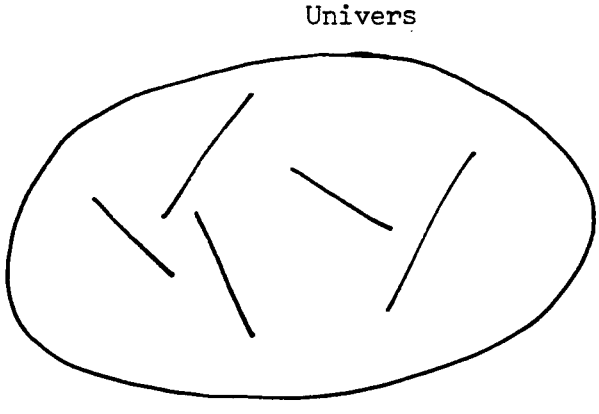
Représentation réaliste, donc non réelle. Ici il y a effectivement ambiguité : s'agit-il encore d'un cube ? En effet les faces arrières ne sont pas visibles.

Dans le cadre de la synthèse d'images réalistes la géométrie de l'univers sera souvent décrite en trois dimensions. Nous n'illustrons que les traitements que la synthèse doit prendre en compte du fait de la propagation rectiligne de la lumière et des limites du champs de vision de l'observateur.

1.6.1. Processus de construction de l'image dont les causes sont extérieures à l'observateur

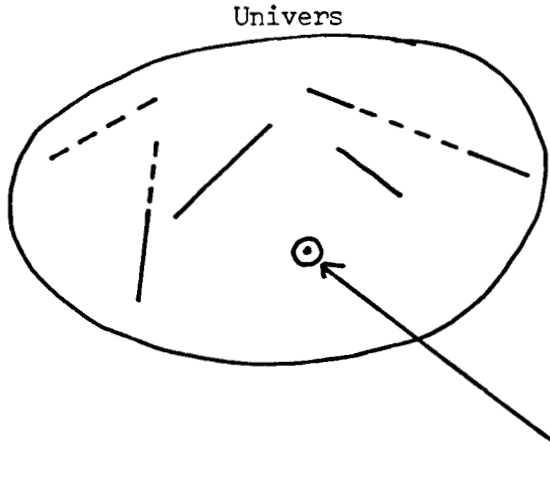
Pour des raisons de clarté et de limitation du support, certains schémas sont transposés en deux dimensions.

Nous supposons que l'univers est fermé, les objets le constituant sont dans son intérieur :



Les traits sont des objets

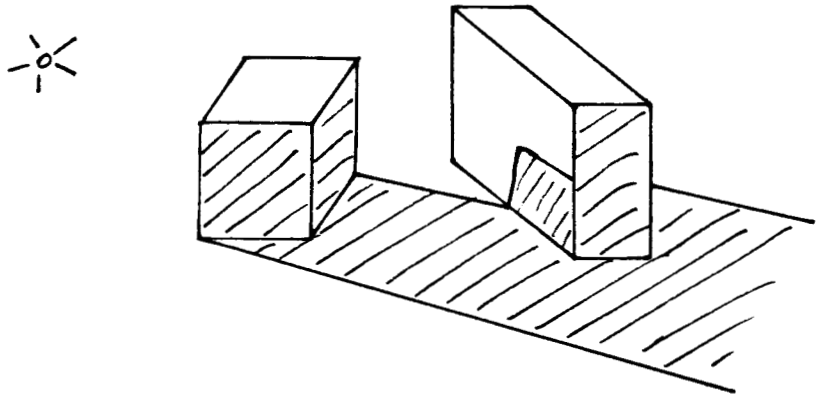
1er Problème : Etant donné un point de vue, il apparaît clairement que certains objets seront cachés par d'autres.



Les parties hachurées sont invisibles à l'observateur.

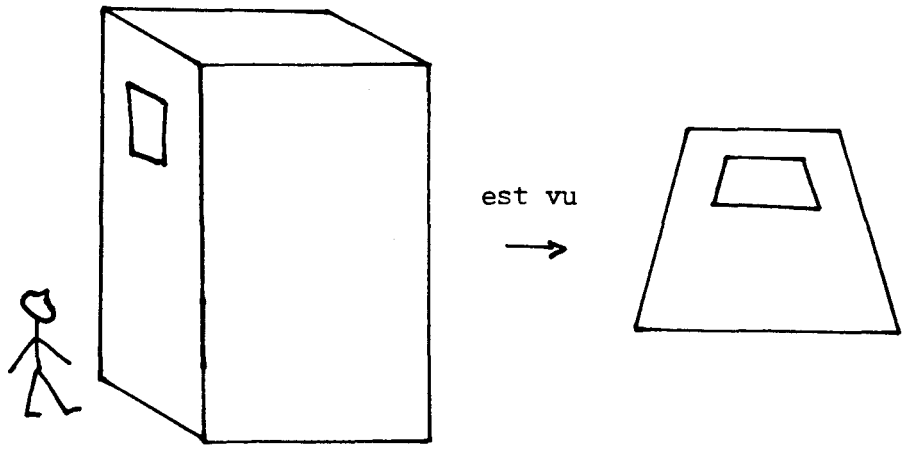
Ce sont les algorithmes de lignes cachées (ou de surfaces visibles) qui traitent ce problème en construisant un nouvel univers ne contenant plus que les "bouts" d'objets réellement visibles.

2ème Problème : Si l'univers est éclairé par une source lumineuse, il se crée alors des ombres portées.



Les algorithmes qui traitent les ombres portées peuvent être simplement des algorithmes de surfaces visibles adaptés où le point d'observation est remplacé par la source lumineuse. Ils produisent à partir de l'univers initial un nouvel univers où l'aspect des objets prend cette fois en compte les parties exposées ou non à la lumière.

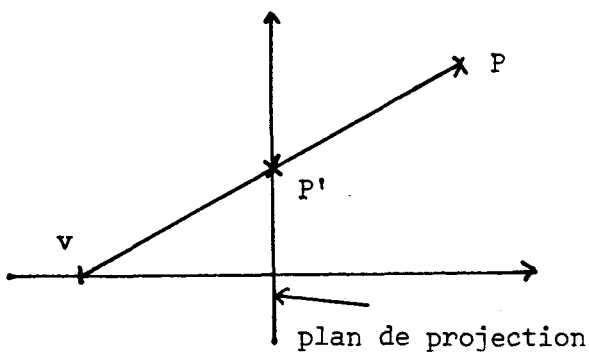
3ème Problème : Il s'agit de l'effet de perspective (on trouvera dans [FOLEY 82] un tour d'horizon très complet de tous les "aspects" des projections en particulier les projections perspectives p. 268).

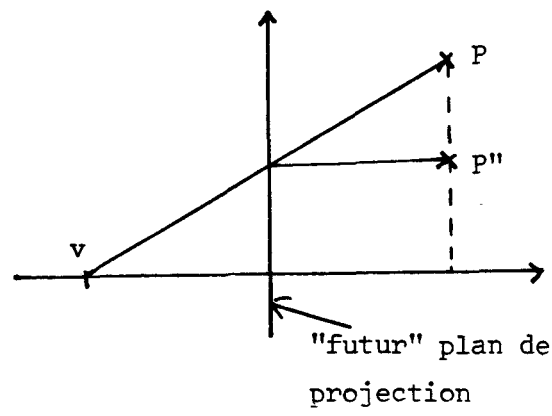


Ce sont les transformations géométriques perspectives qui traitent ce phénomène dans les systèmes de synthèse d'images. Elles modifient la forme des objets. Nous distinguerons deux sortes de transformations perspectives :

- La projection perspective (PP) sur un plan, elle suppose que les deux traitements précédents aient déjà été effectués.

- La transformation perspective (TP) qui reste dans les trois dimensions en conservant les distances relatives des objets par rapport à l'observateur. Cette dernière caractéristique permet alors d'effectuer le traitement de surfaces visibles postérieurement à TP.



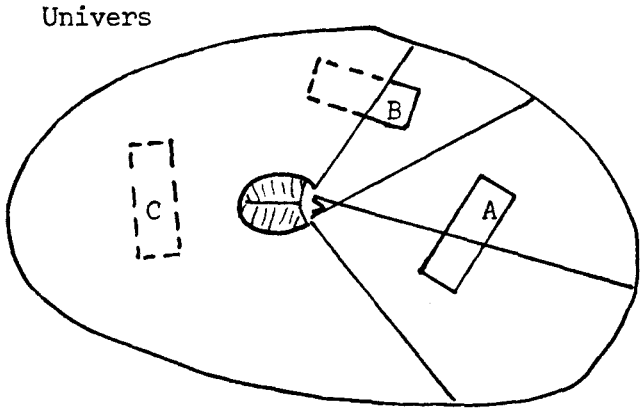


$P' = PP_v(P)$ $\qquad\qquad$ $P'' = TP(P)$

dans les deux figures v est le point de vue.

I.6.2. Processus de construction de l'image dont la cause est interne à l'observateur

L'angle d'ouverture du champs de vision d'un observateur est d'environ 140 degrés. Tout objet en dehors de ce champs de vision est alors invisible :



C'est l'opération de "clôture" qui, dans un système de synthèse, traite cet effet. Elle consiste à réduire l'univers à l'ensemble des objets contenus dans le champs de vision en les découpant éventuellement (objet B).

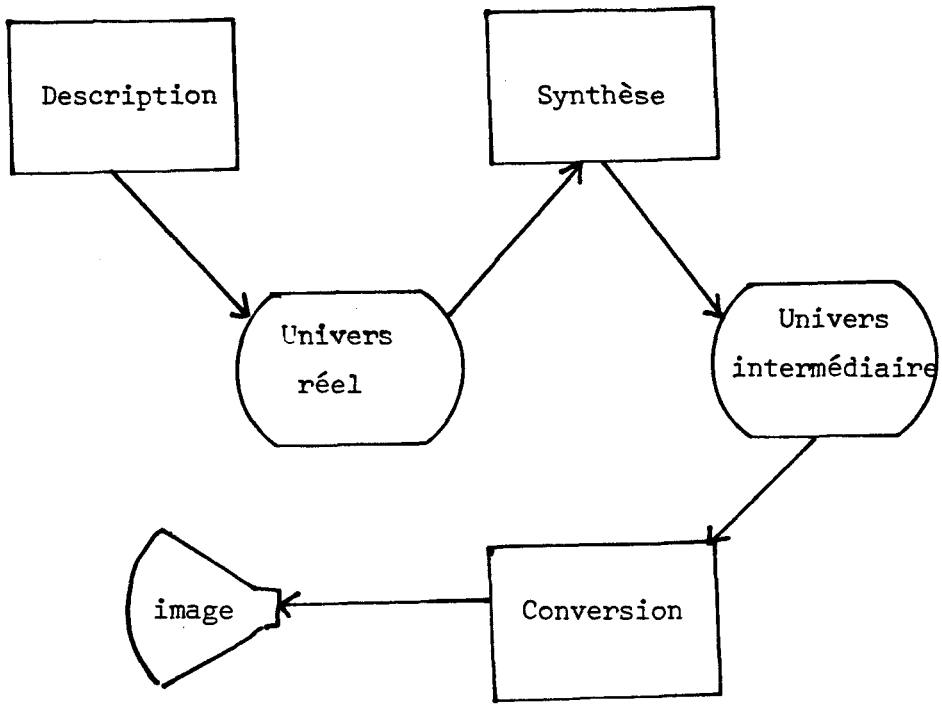
I.6.3. Les autres traitements de la synthèse

Les autres phénomènes dont la synthèse réaliste peut rendre compte sont les effets de réflexion, de réfraction, d'absorption, de diffusion de l'information lumineuse par les différents objets. Ces effets peuvent modifier profondément l'aspect des objets et leur prise en compte représente une des parties les plus complexes de la synthèse d'images.

I.6.4. Description et conversion

Les logiciels de "description" permettent de "construire" l'univers réel ou scène.

La conversion ou "mise en écran" permet d'obtenir l'image finale à partir de l'univers modelé par les traitements de synthèse :



I.6.5. Conclusion

Il est bien difficile de donner a priori un ordre chronologique aux différents traitements de synthèse, nous ne le ferons donc pas.

Le travail effectué par la conversion est pour sa part assez mal défini, la plus simple pouvant être une conversion numérique-analogique, d'autres, plus complexes, peuvent réaliser le "remplissage de taches" et même résoudre le traitement des faces visibles (cf. chapitre IV et V).

Pour conclure ajoutons que la synthèse d'images, par le réalisme auquel elle tend, nous éloigne de la réalité pour mieux tirer parti de nos capacités d'interprétation.

Sur la synthèse d'images et ses algorithmes on peut consulter [CEI 82], [NEWMAN 79], [BOULLE 80],...

I.7. NOTION DE TEMPS REEL

La notion de temps réel est difficile à cerner. Un système graphique peut fonctionner en temps réel par rapport à l'utilisateur (simulateur de vol), de la même manière, l'organe d'affichage peut travailler en temps réel par rapport à

l'organe de calcul (une imprimante peut difficilement être considérée comme un organe d'affichage temps réel).

Nous définissons alors un critère de décision comme un délai maximum d'exécution. Par exemple dans le cas d'un système graphique dont l'organe d'affichage nécessite θ secondes pour l'affichage d'une image, nous parlerons de temps réel par rapport à l'utilisateur si la commande qu'il lance est exécutée au bout d'un temps inférieur à θ. Pour un écran à balayage de trame θ pourrait être le temps d'inscription d'une trame sur l'écran. Cependant ce critère n'est plus valable quand θ varie avec la complexité du dessin (écran à balayage cavalier), θ peut alors être une constante conventionnelle ou être dictée par le support d'affichage (θ doit être assez petit pour éviter le clignotement d'un écran à balayage cavalier).

En outre, les commandes proposées par un système graphique ne sont pas toutes exécutables en temps réel, il convient donc d'énumérer celles qui le sont effectivement.

Dans la suite la notion de temps réel utilisera un θ égal au temps de rafraîchissement d'un écran à balayage de trame.

I.8. L'INTERACTIVITE

Si l'utilisateur d'un système graphique a la possibilité de modifier l'état du système alors il y a interactivité. En fait la raison d'être de l'utilisateur est justement définie par la capacité d'interaction du système. L'organe de calcul étant un ordinateur à vocation générale relié à un terminal graphique (organe d'affichage), nous distinguons deux types d'interaction :

- L'interaction locale dans laquelle la commande de l'utilisateur n'est prise en compte que par le terminal et ne modifie l'état du système que de façon ponctuelle (i.e. rendre une tâche de l'image clignotante en réponse à un nommage depuis l'écran).

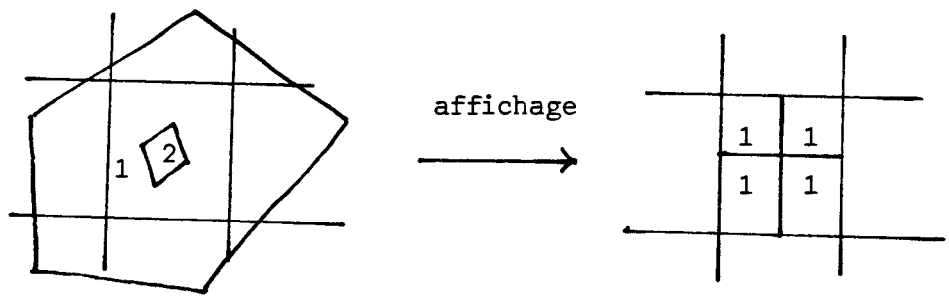
- L'interaction globale qui modifie l'état du système plus profondément en s'adressant directement au calculateur principal.

LUCAS et KILGOUR proposent un modèle plus général de l'interaction [LUCAS 77], [KILGOUR 81]. Nous parlerons d'autonomie du terminal par rapport à l'organe de calcul pour qualifier sa capacité à prendre en compte des interactions locales.

I.9. LE PROBLEME DE L'ALIASSAGE

Le terme d'aliassage regroupe les phénomènes parasites apparaissant dans la construction d'une image discrète. Trois causes président à de tels phénomènes :

- Les calculs d'image sont effectués avec une plus grande précision que celle offerte par le support. De "petites" taches peuvent alors ne pas apparaître dans l'image.



espace de calcul"continu" espace discret

Ce sont les algorithmes de préparation à la visualisation qui provoquent ce genre d'erreur [LUCAS 77].

- Les calculs d'affichage sont effectués à la même précision que celle proposée par le support, un effet classique de "marches d'escalier" en résulte. Ce sont cette fois les logiciels "élémentaires" qui sont en cause [LUCAS 77], [GRAVE 80] annexe.

- Tout support d'affichage propose deux dimensions de définition : une définition géométrique qui définit la taille d'un pixel, et une définition de l'information apportée par un pixel (nombre de niveaux de gris, de couleurs). Ce deuxième aspect peut favoriser une mauvaise interprétation, par exemple sur un support en noir et blanc (deux niveaux de gris) une image sera un ensemble de taches noires ou blanches.

Les solutions à ces problèmes d'aliassage consistent généralement à :

- Etre prudent lors du passage d'une représentation "continue" à une représentation discrète.

- Effectuer a posteriori un filtrage de l'image pour atténuer les discontinuités.

- Augmenter le plus possible la qualité de définition du support d'affichage lui-même.

- Le dernier problème est résolu en remplaçant un pixel par un groupe de pixels, qui apportent plus d'information [CEA 82] p. 41.

On pourra voir à ce sujet [WHITTED 83], [CROW 77], [GUPTA 81].

I.10. CONCLUSION

Le nombre et la complexité des traitements qu'implique la synthèse d'images réalistes, se traduit le plus souvent par des temps de calcul très élevés. Par exemple pour créer une image de définition 1000 sur 1000 pixels (une image de qualité cinématographique représente au minimum 4000 * 4000 points) il est nécessaire de calculer un million de pixels simplement au niveau de la conversion. En supposant que n opérations élémentaires de une microseconde sont nécessaires à l'évaluation d'un point, il faudra alors n secondes pour la simple mise en écran de l'image.

Il apparait alors nécessaire de structurer les traitements et de reproduire ces structures sous forme d'architectures parallèles afin d'accélérer le processus de synthèse.

Ces différentes structurations et leurs applications matérielles font l'objet du chapitre suivant.

CARACTERISATION DES SYSTEMES GRAPHIQUES

Dans un essai de classification des multiples systèmes graphiques existant, plusieurs modèles ont été introduits. Nous en présentons trois, la description en couches de LUCAS, la représentation pipe-line de CARLBOM et le modèle fonctionnel de MARTINEZ. Une classification est ensuite développée, qui a été succintement introduite par WEINBERG dans [WEINBERG 81], et qui s'intéresse aux architectures parallèles. Enfin celle-ci est illustrée à travers plusieurs machines graphiques.

II.1. MODELES DE SYSTEMES GRAPHIQUES

II.1.1. Le modèle en couches [LUCAS 77]

Tout système graphique est découpé en quatre couches fonctionnelles :

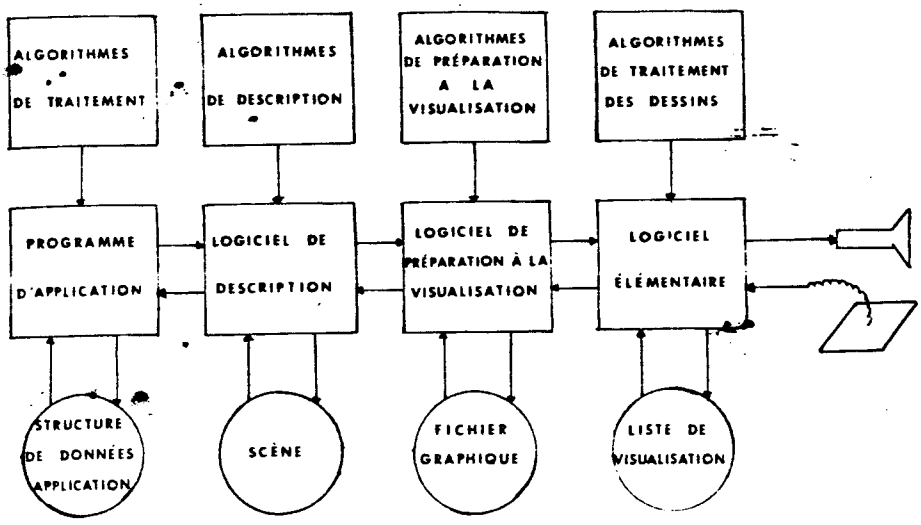
- La première couche est le "logiciel d'application". Typique des besoins de l'utilisateurs elle manipule une structure de données qui n'est pas spécifiquement graphique (par exemple un logiciel de CAO de circuits qui peut gérer à la fois des paramètres électriques et graphiques) [GARDAN 83].

- La deuxième couche extrait de la structure précédente des informations graphiques pour construire la "scène" (ou univers graphique). Cette partie est appelée "logiciels de description".

- La troisième couche extrait de la scène la sous-scène destinée à être affichée, elle peut aussi lui appliquer certains traitements tels que clôture, résolution des faces cachées, effet de perspective. LUCAS la nomme "logiciel de préparation à la visualisation".

- La quatrième couche ou "logiciels élémentaires" transcode la représentation en deux dimensions de la sous-scène en un ensemble d'informations affichables (par exemple la commande des mouvements d'un crayon (faisceau d'électrons) sur une table traçante (sur un écran cathodique à balayage cavalier)). Ce sont ces logiciels qui seuls dépendent des caractéristiques matérielles . terminal d'affichage.

Lucas propose le schéma suivant :



Nous n'avons présenté que l'aspect synthèse, travail progressif depuis le programme d'application jusqu'à l'affichage. Cependant lorsqu'il s'agit d'interaction la communication se fait dans l'autre "sens", c'est ce qui est indiqué par les flèches qui reviennent jusqu'au programme d'application.

En particulier la reconnaissance d'un objet graphique désigné par la tache qu'il produit sur l'écran est un problème difficile à résoudre du fait que les représentations intermédiaires ne rendent pas forcément compte de la structure du nom des objets. Dans le cas de systèmes graphiques permettant une modification "continue" de l'image, KILGOUR propose un mécanisme général de prise en compte de l'interaction dans chaque couche [KILGOUR 81].

II.1.2. Le modèle pipe-line [CARLBOM 80 et 83]

Ce modèle est très semblable à celui de LUCAS. Cependant il utilise les notions de processus et de processeurs plutôt que de logiciels [FOLEY 82]. En effet son but est avant tout de permettre a priori d'évaluer les performances d'un système graphique (en particulier lorsqu'il s'agit de ressources partagées par les processus i.e. bus de communication) et par là de comparer différentes architectures.

Vis-à-vis de ces deux modèles en couches les auteurs s'accordent à dire que dans la réalité ces séparations ne sont pas toujours aussi nettes. On peut donner comme exemple l'algorithme de traitement des faces visibles de WARNOCK qui permet d'obtenir un dessin directement affichable [WARNOCK 69].

Le modèle de MARTINEZ rend compte des interférences entre couches et décrit plus finement les particularités des systèmes graphiques.

II.1.3. Le modèle fonctionnel [MARTINEZ 82]

Ce modèle propose tout d'abord trois outils qui vont permettre par composition de modéliser les systèmes graphiques. Ce sont :

- Les classes d'informations d'une image.
- Les opérateurs élémentaires de synthèse.
- La notion d'élément.

II.1.3.1. Les classes d'informations d'une image

Une image contient six classes d'informations qui sont appelées attribut :

- L'identification représente le nom d'un objet dans l'univers.
- La morphologie décrit la forme d'un objet.
- L'aspect contient les informations de couleur, transparence, texture... attachées à un objet.
- La géométrie définit l'emplacement d'un objet dans l'univers.
- L'éclairage indique quelles sont les sources lumineuses ainsi que la lumière ambiante de l'univers.
- La structure qui peut indiquer les liaisons logiques entre les différents objets de l'univers.

Chacune des six classes est affinée par l'utilisation de types correspondant à des critères de distinction de nature structurelle, sémantique ou syntaxique.

Ces attributs, décrits de façon indépendante les uns des autres, vont devoir faire l'objet d'un traitement de synthèse pour qu'une image soit produite.

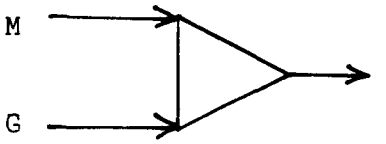
II.1.3.2. Les opérateurs élémentaires de synthèse

Si on les dénombre par leur fonctionnalité, il peuvent être a priori en nombre illimité. En effet dans une "chaine de synthèse" c'est la manière

dont sont agencés chacun des opérateurs et les attributs sur lesquels ils travaillent qui indique le domaine de traitement de chacun d'eux.

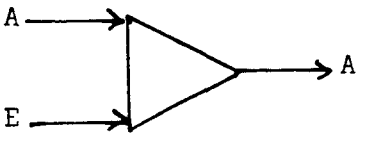
Quelques exemples simples d'opérateurs sont les suivants d'après MARTINEZ) :

La synthèse M.G. Cette opération a pour résultat une nouvelle morphologie dont le type peut être différent de celui de l'attribut initial.



peut correspondre à une translati

Citons la synthèse A.E. qui modifie l'aspect A (en particulier la couleur) d'un objet en fonction de l'éclairage E.



Nous devons maintenant distinguer parmi les attributs ceux qui qualifient un objet et ceux qui sont globaux.

II.1.3.3. La notion d'élément

Remarquons d'abord que les attributs Eclairage et Structure qualifient l'univers de façon globale. En revanche les quatre autres représentent la "personnalité" de chaque objet. Le terme "élément" désigne les différents états par lesquels passe un objet au cours de la synthèse.

II.1.3.4. Conclusion

L'identification d'un objet est d'autant plus simple à réaliser que la pénétration de l'attribut Identification est importante dans le processus de synthèse.

Cette méthode de description permet de visualiser les caractéristiques propres à un système graphique sans avoir à distinguer logiciel et matériel.

II.2. CLASSIFICATION DES SYSTEMES GRAPHIQUES : TYPES DE PARALLELISME

La complexité des traitements effectués par un système de synthèse d'images pousse leurs concepteurs à trouver des moyens de réduire les temps de calcul. Trois voies de recherche se distinguent :

- Amélioration de l'efficacité des algorithmes. Par exemple l'algorithme de BRESENHAM pour le tracé de segments de droites est mieux adapté aux tables traçantes que la résolution de l'équation d'une droite. [BRESENHAM 65].

- Spécialisation du système graphique à un domaine limité d'applications. Un exemple limite est celui des jeux vidéo.

- Réarrangement des machines par l'utilisation du parallélisme puisqu'il existe des limites aux progrès technologiques.

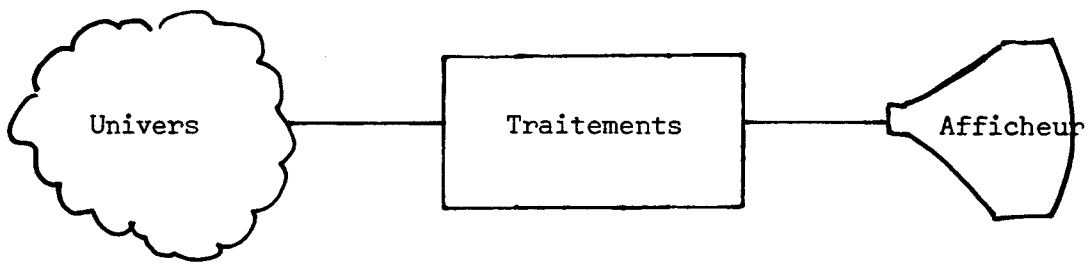
C'est cette dernière voie que nous nous proposons d'explorer, tout en gardant à l'esprit que ces trois voies sont en étroite relation.

Un système graphique peut être représenté grossièrement par trois éléments :

- L'univers ou structure de données strictement graphique obtenue par le logiciel de description.

- L'ensemble des traitements qui, appliqués à l'univers, fournissent les informations nécessaires à l'affichage du dessin ou de l'image.

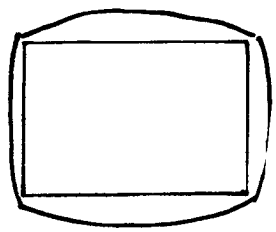
- L'"afficheur" qui gère le support de dessin.

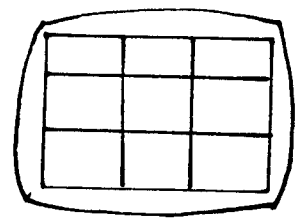


Le parallélisme d'une machine est alors le fruit du "découpage" d'un au moins de ces éléments. Les paragraphes qui suivent précisent ces notions de découpage.

II.2.1. Découpage géographique

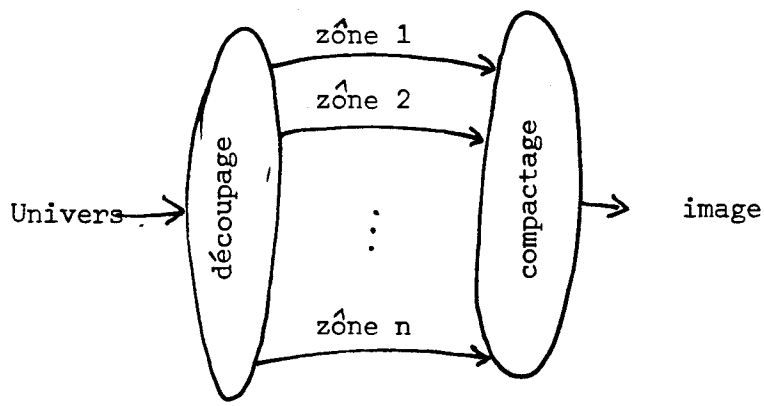
Le support d'affichage est découpé en zônes souvent toutes de même taille et de même géométrie. Dans le cas d'un écran cathodique à balayage de trame on aurait :





Si l'écran est divisé en neuf pavés. Le système graphique assure alors l'élaboration simultanée de chacune des zônes.

Si l'idée de réduire un grand domaine de calcul (tout l'écran) à plusieurs sous-domaines paraît séduisante, un problème apparaît cependant qui est celui de la manière dont un traitement est attribué à une zône. C'est donc l'opération de découpage de l'univers qui est critique.

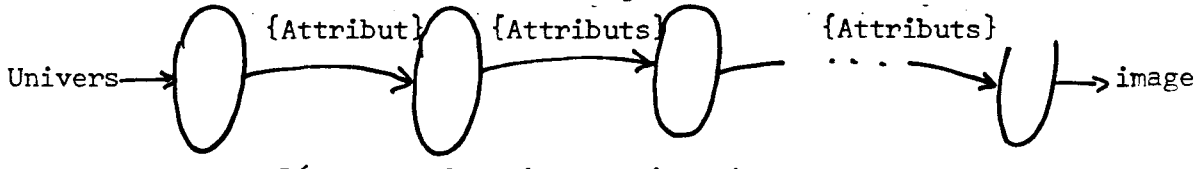


Le "compactage" effectué lors de l'affichage devient non trivial si des traitements d'anti-aliassage sont désirés. Notons que le découpage est d'autant plus complexe qu'il est effectué près de l'univers. Enfin ce découpage est plus souvent appliqué à l'analyse qu'à la synthèse d'images.

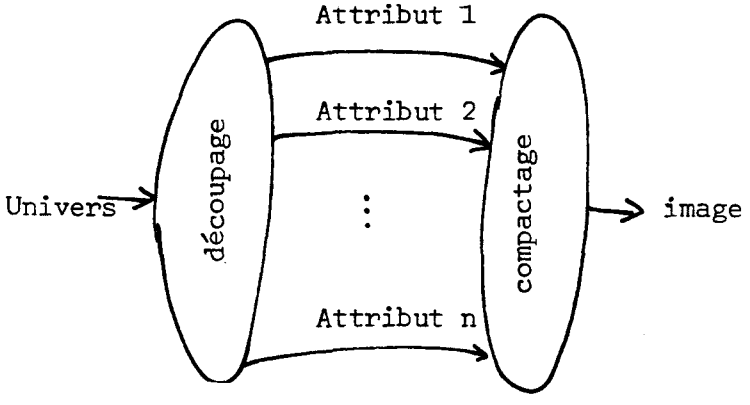
Les réseaux cellulaires sont un exemple extrême de ce type de découpage.

II.2.2. Découpage fonctionnel ou classe deux

Le chapitre précédent présente les principaux traitements de la synthèse d'images. Le découpage fonctionnel consiste à effectuer ceux-ci en parallèle. On peut distinguer un parallélisme "pipe-line" (par rapport aux traitements) et un parallélisme par attribut au sens de MARTINEZ. Les deux schémas suivant découlent de cette remarque.



Découpage fonctionnel pipe-line



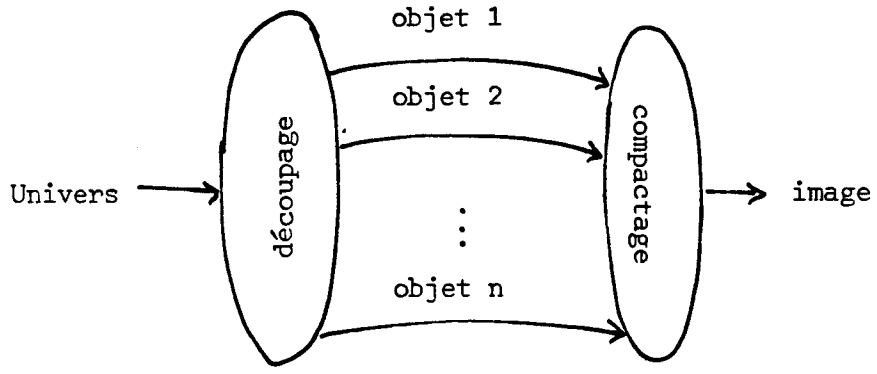
Découpage par attribut

Plutôt que le découpage et le compactage la difficulté semble être ici de savoir quels sont les traitements à effectuer sur les différents attributs.

II.2.3. Découpage élémentaire ou classe trois

C'est cette fois l'univers lui-même qui est découpé en objets élémentaires. Une structuration hiérarchique de la scène facilite un tel découpage. Le parallélisme consiste donc à traiter ces objets simultanément.

Les problèmes apparaissent lors du compactage final. La figure illustre ce type de découpage :



La description de différentes architectures graphiques, présentée dans le paragraphe suivant, montre que les trois classes présentées ne sont pas étanches entre elles.

II.3. APPLICATION A QUELQUES SYSTEMES GRAPHIQUES

Dans la suite chacune des trois catégories d'architectures est illustrée.

II.3.1. Machine à parallélisme géographique

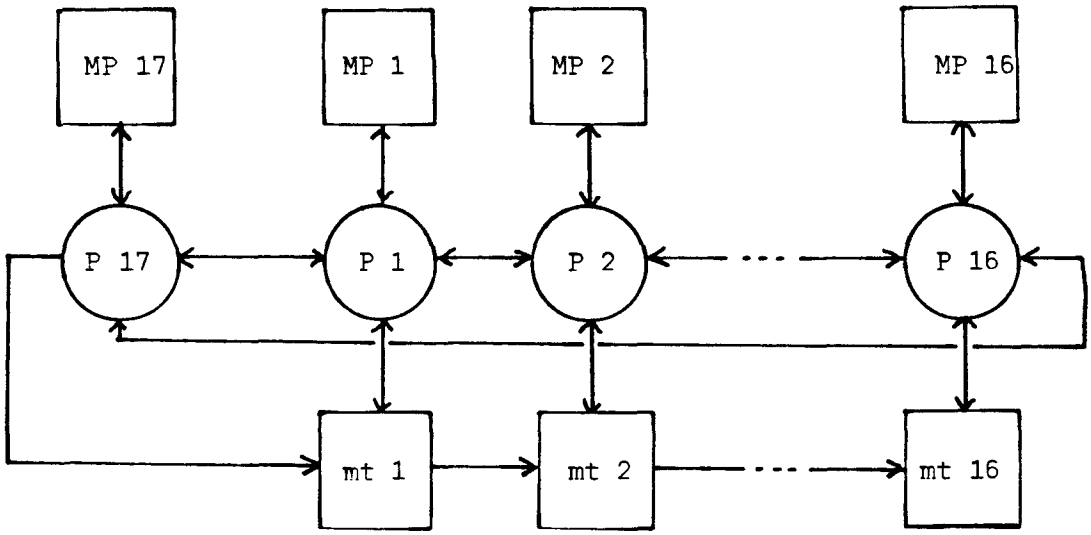
On peut inclure dans cette classe les réseaux cellulaires et les terminaux multiprocesseurs où chaque processeur se voit chargé du contrôle d'une partie de l'écran.

II.3.1.1. MAP (Multiprocesseur Associatif Parallèle) [CORDONNIER 81]

Cette machine, réalisée dans le Laboratoire d'Informatique de Lille [REDJIMI 82], comporte seize processeurs, chacun d'entre eux étant chargé des traitements relatifs à un seizième de l'image à afficher. C'est en fait la mémoire de trame qui est divisée en seize régions formant une partition de l'écran.

En outre les processeurs de traitement n'ayant pas la possibilité d'adresser eux-mêmes la mémoire de trame, c'est un dix-septième processeur qui gère le calcul des adresses et le contrôle des traitements.

La figure suivante résume l'architecture de MAP :



MPi = Mémoire de Programme du Processeur i

Pi = Processeur i

mti = Partie de la Mémoire de Trame calculée par Pi

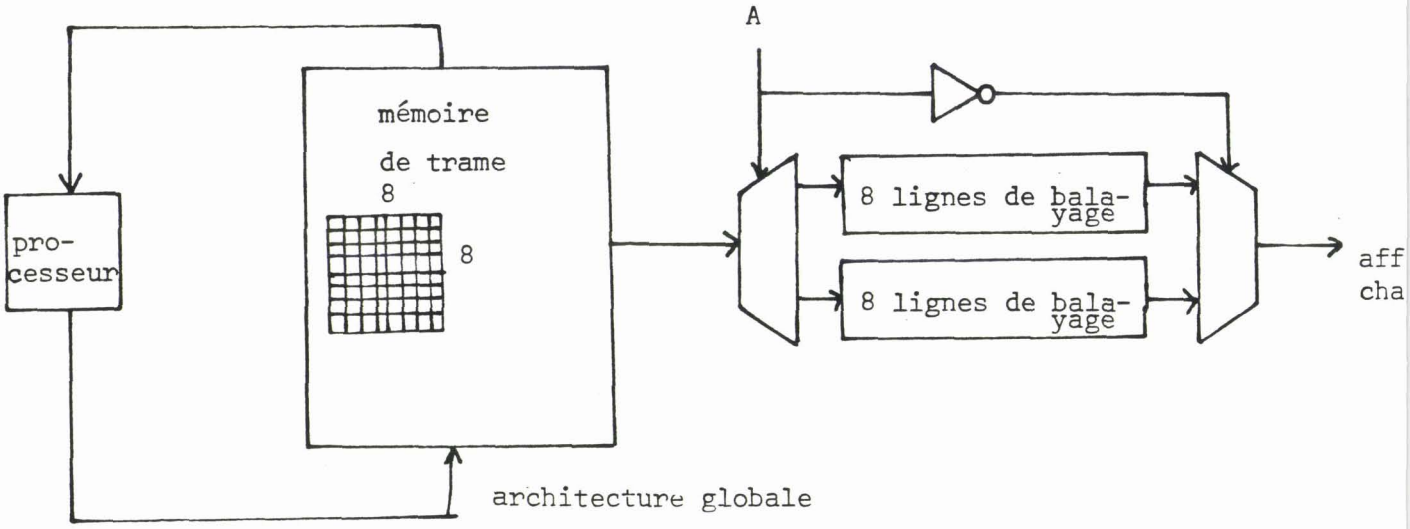
Ce terminal s'est vu doté de logiciels élémentaires, selon la terminologie de LUCAS, classiques au traitement (manipulation de la mémoire de trame) et à la synthèse d'images (tracé de segments de droites, de cercles) [PELERIN 82], [JABRI 82]. Cependant l'étude d'un système complet incorporant un processeur hôte n'a pas encore été effectuée.

Un découpage élémentaire éventuel serait en contradiction avec le découpage géographique. Un découpage fonctionnel au niveau du terminal lui-même semble illusoire compte tenu de l'aspect séquentiel de l'exécution de chacun des processeurs (ce sont des micro-processeurs). MAP appartient donc exclusivement à la première classe.

Ajoutons que si la qualité d'image est a priori assurée, l'extension d'un tel système qui pourrait correspondre à une augmentation des dimensions de l'affichage semble être un problème difficile à résoudre. De plus l'identification d'un objet désigné depuis l'écran n'est pas favorisée par cette architecture.

II.3.1.2. La machine 8 sur 8 [GUPTA 81]

L'idée de base est ici d'accélérer l'accès en mémoire de trame. Ce terminal permet la manipulation simultanée d'un pavé de 8 * 8 pixels en mémoire de trame. Le coin bas-gauche de celui-ci pouvant être positionné en tout point de la mémoire.



architecture globale

Le goulot d'étranglement, classiquement dû à la lenteur des accès mémoire, est ici reporté sur le processeur qui nécessite plus de huit cycles pour former un pavé 8 * 8 (un bit par pixel).

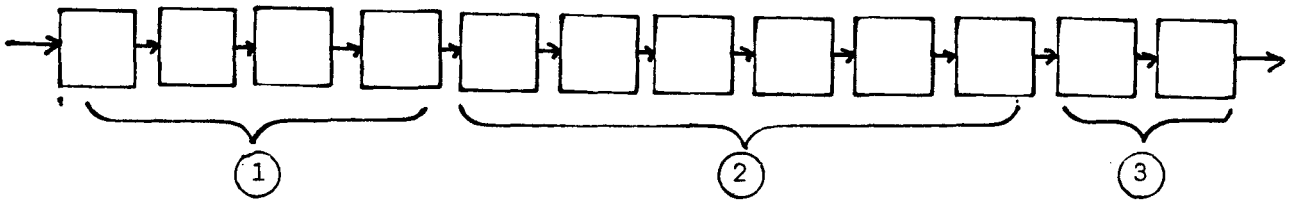
Ce système propose des logiciels de description, de préparation à la visualisation ainsi que des logiciels élémentaires [SPROULL 82], [SPROULL 83]. Au niveau matériel, des opérations de manipulation directe de la mémoire d'image ("RasterOp") sont aussi fournies.

Enfin le classement de cette machine dans la première catégorie est relativement arbitraire du fait que le découpage n'apparaît que lors de l'écriture/lecture en mémoire de trame. Les difficultés de désignation et d'extension ont leur contre-partie dans la qualité de l'image qui peut être bonne pourvu que la mémoire soit de taille suffisante.

II.3.2. Machines à parallélisme fonctionnel

II.3.2.1. Le "geometrie engine" [CLARK 82]

Cette machine, constituée de douze circuits tous identiques, permet des calculs géométriques en virgule flottante. Les circuits sont arrangés en pipe-lin



Les trois groupes de pipe-line sont :

- Le groupe 1 se charge des transformations géométriques en deux ou trois dimensions, obtenues par multiplication matricielle en coordonnées homogènes [NEWMAN 79].

- Le deuxième effectue le découpage de la scène suivant une fenêtre (deux dimensions) ou une pyramide tronquée (trois dimensions).

- Le troisième assure les opérations de projections perspectives ou orthogonales ainsi que la mise en écran (ou mise à l'échelle). Des vues stéréoscopiques peuvent être obtenues.

Ces opérations sont toutes de nature géométrique. Le parallélisme ici est avant tout basé sur le principe de découpage fonctionnel. L'opération de compactage est triviale puisqu'elle correspond à la sortie du pipe-line.

Le système géométrique étant inséré dans un système graphique très succintement décrit par l'auteur nous ne pouvons que préciser qu'il n'utilise pas de découpage géographique. La qualité de l'image contrebalance la difficulté d'une utilisation en temps réel.

II.3.2.2. Le terminal Helios [FERREIRA 81], [MARTINEZ 82]

Comme dans le modèle de MARTINEZ les objets manipulés sont décrits par des attributs. Une des caractéristiques d'HELIOS est de traiter séparément des classes d'attributs. Ces classes se fondent progressivement les unes dans les autres au cours de la synthèse pour former finalement l'information affichable.

Le terminal reçoit des informations présynthétisées ou non qui sont :

- I.M.G.Gv : la mémoire de trame (ensemble de plans d'identification) contient la projection sur le plan de vision des faces, chaque point d'une face contenant son identificateur. Un prétraitement des surfaces visibles a été réalisé dont le résultat est matérialisé par les différentes priorités des plans d'identification.

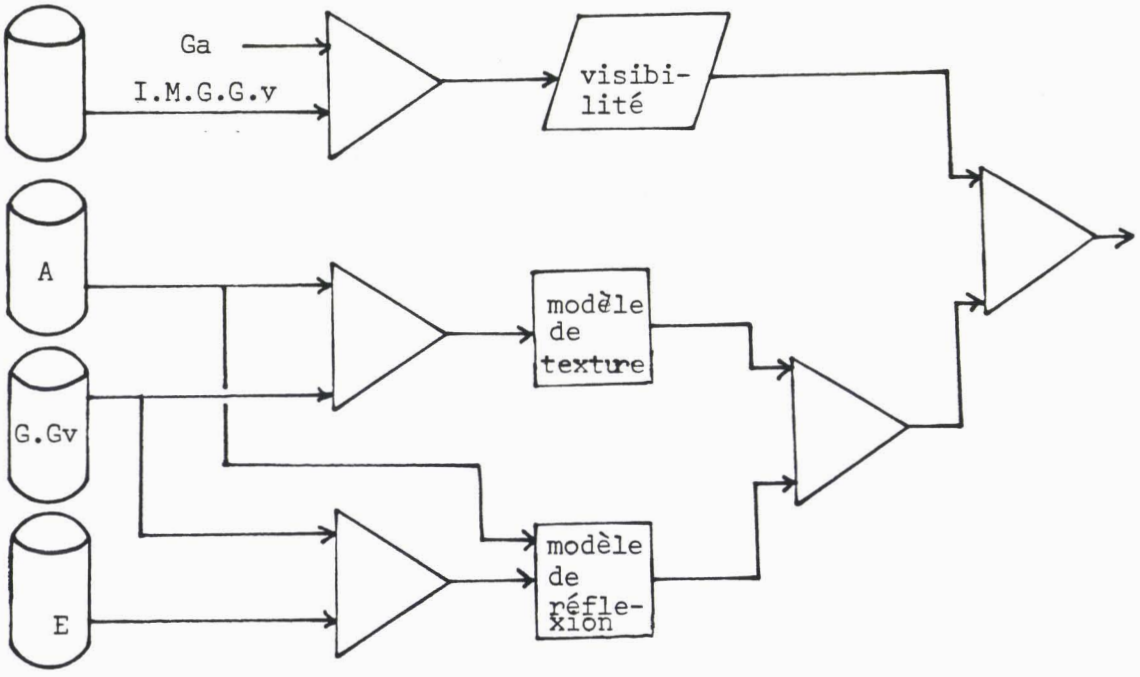
- A : l'aspect des éléments contenus dans la mémoire de trame.

- G.Gv : projection du repère de la face dans le plan de vue.

- E : éclairage.

- Ga : fenêtre à afficher.

En utilisant le formalisme de MARTINEZ l'architecture du terminal est la suivante :



où les cylindres représentent des tampons de mémorisation.

Le découpage fonctionnel paraît évident. Les deux autres modes de découpage n'apparaissent pas.

L'identification d'un objet est rendue possible par les plans d'identification, d'autre part, la synthèse de certains attributs effectuée en temps réel implique une limitation de la qualité de l'image en ce qui concerne sa définition et les effets de perspective des textures.

II.3.3. Machines à parallélisme élémentaire

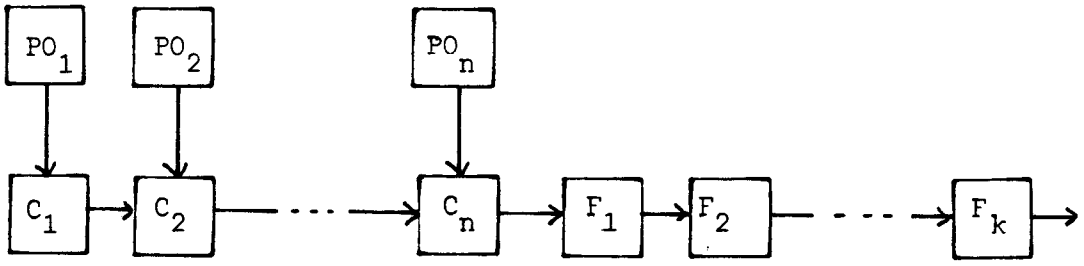
II.3.3.1. La machine de WEINBERG

Le système graphique proposé est composé de trois parties fonctionnelles :

- Un ensemble de processeurs-objet qui traitent chacun un trapèze (avec deux bords horizontaux).

- Un ensemble de comparateurs montés en pipe-line dont le but est de construire la liste ordonnée en YXZ de l'image à afficher.

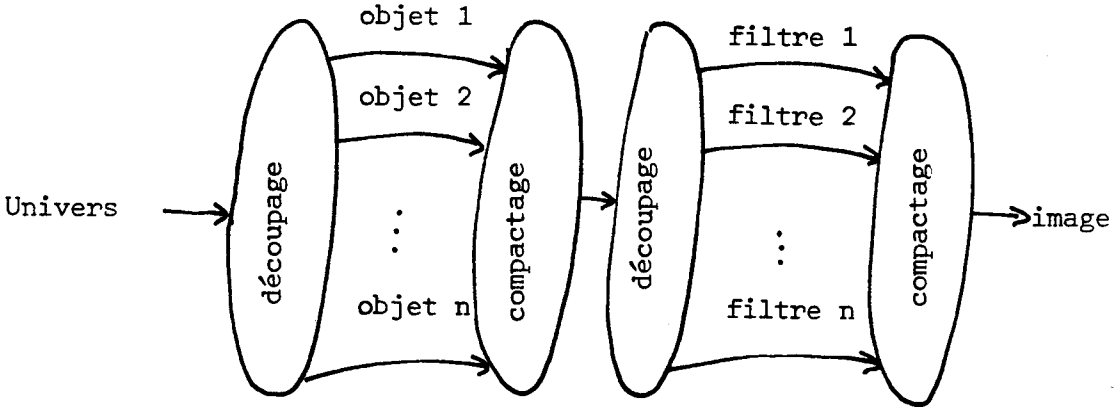
- Un ensemble de filtres qui à partir d'une sous-liste ordonnée en Z va calculer la couleur d'un pixel de façon à répondre au problème d'aliassage.



En fait chacun des filtres travaille sur une ligne de balayage d'une sous-grille de k lignes sur k colonnes par pixel réel.

Si le découpage élémentaire est ici évident, il existe aussi un découpage géographique : chacun des filtres étant assigné de façon statique à une sous-ligne de balayage, on peut dire que l'écran est découpé en k sous-écrans au niveau du filtrage.

Une autre représentation de cette architecture pourrait être :



Le premier compactage correspond à la sortie du pipe-line de comparateurs, le second est la sortie du pipe-line de filtres.

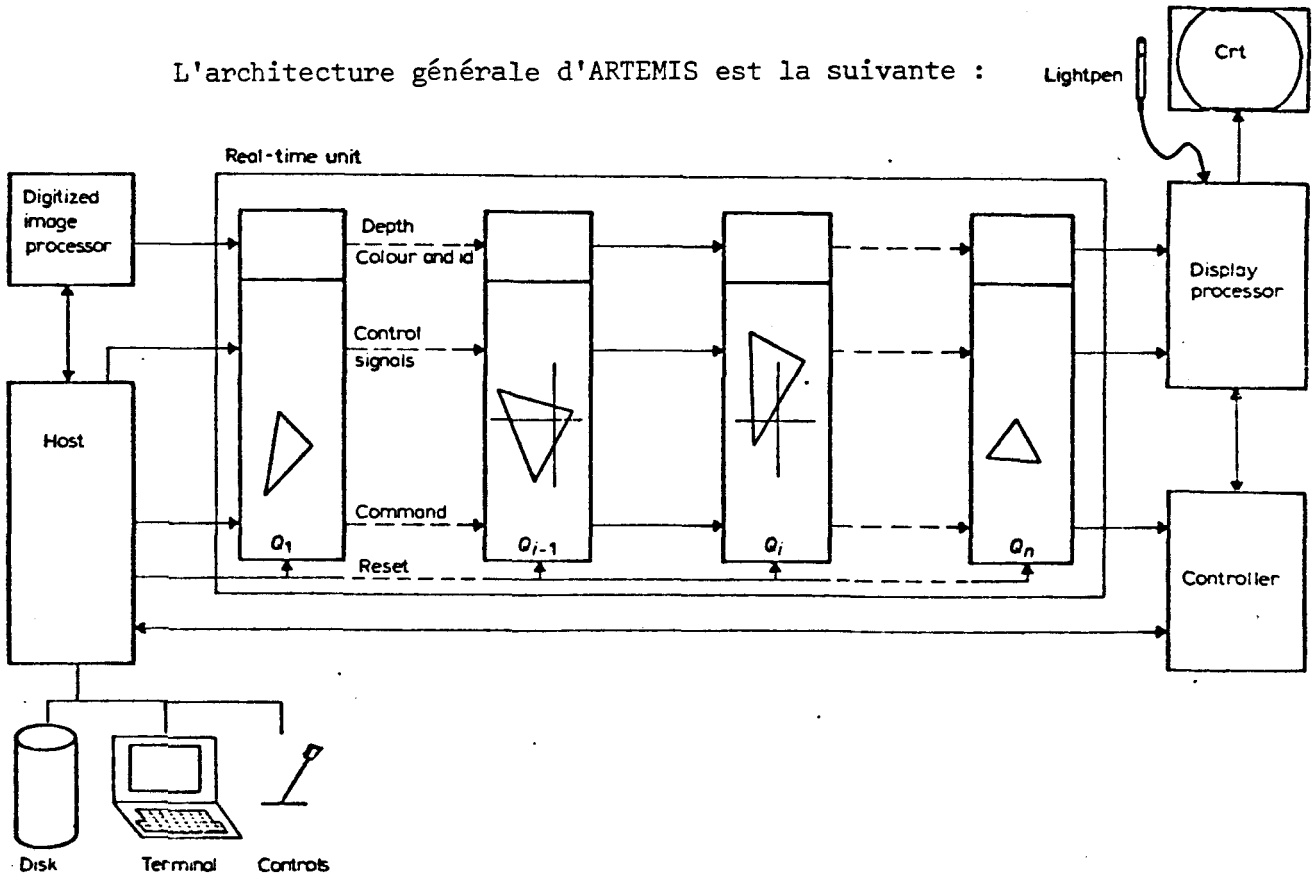
Le volume d'informations à transférer entre chaque étage du pipe-line et l'importance des traitements (anti-aliassage) interdit le temps réel tout en permettant une bonne qualité de l'image. L'ajout de nouveaux processeurs-objet ne semble pas poser de problème. En revanche le caractère différé du traitement rend ardue l'identification d'un objet.

Notons dès maintenant que dans ce genre d'architectures, le problème des surfaces visibles est résolu en dernier ressort (avant le filtrage) sans sanctionner la rapidité des traitements ; ceci au prix d'un grand nombre de circuits.

II.3.3.2. ARTEMIS [GRUIA-CATALIN 81]

Comme précédemment l'univers est décomposé en objets. Chaque objet est constitué d'un ensemble de facettes triangulaires. La machine traite tous ces triangles en parallèle et temps réel. Elle comprend un grand nombre de "processeurs-triangle" sous forme d'un pipe-line. La résolution des surfaces visibles passe encore par un pipe-line de comparateurs.

L'architecture générale d'ARTEMIS est la suivante :



Comme dans l'architecture précédente de pipe-line de comparaison-sélection permet de réduire le temps de sélection entre n triangles au temps de sélection entre deux triangles (s'il y a n processeurs-triangle).

Le processeur d'affichage du fond est en fait une mémoire d'image contenant le couple (couleur, profondeur) pour chaque pixel.

Chacun des processeurs-triangle est chargé de convertir un triangle au rythme du balayage de l'écran. Ceux-ci acceptent des commandes de gestion (création, suppression... du triangle), ou des demandes de transformations géométriques (translation, rotation...) issues du processeur-hôte.

Toutes les communications (commandes, contrôle, données à afficher...) empruntent le pipe-line.

Les modifications possibles en temps réel portent sur les attributs de géométrie, de transparence et sur la distance point de vue-écran.

Les processeurs-triangle comprennent deux parties ; la première chargée de la gestion et des transformations d'attributs, la seconde de la conversion proprement dite. Cette dernière nécessite 43 registres, 7 additionneurs et 23 comparateurs et ne travaille que sur l'aspect géométrique d'un triangle (sa profondeur) la couleur étant constante.

Enfin l'identification est immédiate puisque le pipe-line de sélection convoie à la fois la profondeur, la couleur et l'identificateur de l'objet prioritaire.

ARTEMIS appartient exclusivement à la troisième classe et n'a pas donné lieu, à notre connaissance, à une réalisation matérielle.

II.3.4. Récapitulation

Il convient de lire le tableau suivant avec prudence. En effet les machines décrites ne sont pas au même état d'avancement, d'autre part les documents disponibles ne relataient pas toujours les mêmes aspects. L'échantillon proposé n'est pas exhaustif.

La suite de ce travail décrit une architecture à découpage élémentaire. Certaines des propriétés des machines de la même classe s'y retrouvent. Le principal inconvénient de cette catégorie est le mauvais taux d'utilisation des processeurs-objet (PO) :

- Si un PO travaille, il n'élabore qu'une très petite partie de l'image finale.

- Si les PO contiennent l'univers (c'est le cas d'ARTEMIS) seul un sous-ensemble de ces processeurs participe à l'élaboration de l'image.

Ajoutons que le contexte technologique est une cause de la non-réalisation des machines de la troisième catégorie.

Le chapitre trois propose une étude de plusieurs algorithmes de calcul couramment utilisés par les logiciels élémentaires. Ces algorithmes servent ensuite à la description de la machine.

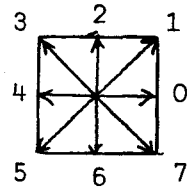
	Machine	Temps réel	Interactivité Désignation	Qualité Image	Extensibilité Modularité	Mode dégradé	Logiciels	Câblée ?	Taux d'utilisation
Découpage géographique	MAP	décalage d'image 1 segment de droite 1 cercle	difficile	oui	difficile	non	élémentaire manipulation de la mémoire	oui	bon
	8 × 8	? non	difficile	oui	difficile	non	préparation à la visu. élémentaire, manipulation de la mémoire		
Découpage fonctionnel	Σ géométrique	non	/	oui	2D → 3D	non	élémentaire semi câblé	oui	bon
	HELIOS	position/éclairage, couleur, texture, invisibilité/transparence	oui	moyen	plans d'identification nombre d'objets	non	tri Z et aspect câblé		
Découpage élémentaire	Weinberg	non	difficile	oui	nombre et type des objets	oui	tri en XYZ câblé	non	mauvais
	ARTEMIS	point de vue, couleur, transformation géométrique, tansparence	oui	moyen			élémentaire matériel		

CHAPITRE III

CALCUL INCREMENTAL

PRELININAIRES

Les systèmes graphiques utilisent lors de l'ultime étape de synthèse des logiciels élémentaires qui ont pour vocation de transcoder une structure de données en un ensemble d'informations directement affichables. Par exemple un programme de passage du code de FREEMAN à un dessin au trait est un logiciel élémentaire [FREEMAN 61]. Le code de FREEMAN permet de coder un contour à l'aide de huit directions élémentaires et de l'origine du contour :



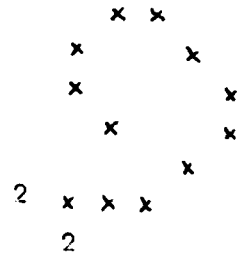
sont les huit directions alors le logiciel de transcodage permettra d'obtenir le dessin suivant à partir de la liste.

Exemple de transcodage :

(2, 2)

0	0	1	1	2	3	3	4	5	6	7

Logiciel de transcodage

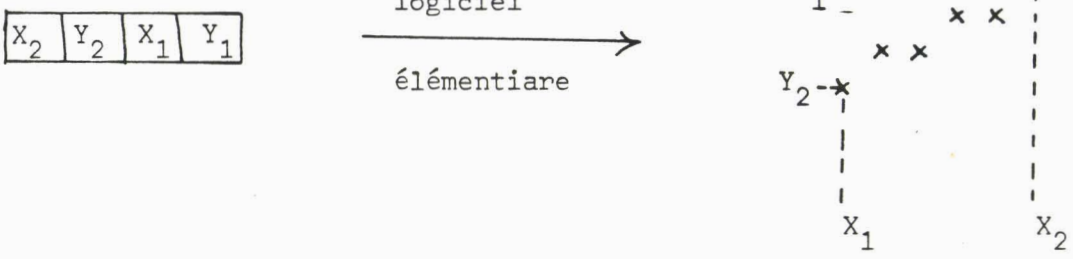


Les inconvénients d'un tel code sont la place occupée par la liste et le fait que sa construction nécessite le calcul de tous les points du dessin. Précisons que les logiciels élémentaires travaillent dans un espace discret qui correspond généralement à celui défini par le support d'affichage.

D'autres transcodeurs plus évolués existent qui sont par exemple les interpolateurs linéaires. Toutes les tables traçantes, à notre connaissance, et beaucoup de terminaux à écran cathodique utilisent ces interpolateurs [CARREZ 77]. Ils traduisent deux couples de coordonnées (points extrêmes du segment de droite) en un ensemble de points approximant au mieux le segment.

(X1, Y1) (X2, Y2)

Logiciel élémentaire



Interpolation linéaire (cas numérique).

Les méthodes d'interpolation sont soit analogiques (écrans cathodiques à balayage cavalier) soit numériques [FOLEY 82], [CEA 82]. Les méthodes numériques sont souvent incrémentales c'est-à-dire que tout point tracé est défini par rapport au précédent en fonction de certains critères ou indicateurs de qualité.

Les algorithmes incrémentaux ont généralement les caractéristiques suivantes :

- Utilisation de variables entières ou en virgule fixe.
- La multiplication et la division sont prohibées.
- La prise en compte du passé du tracé permet de simplifier le calcul d'un nouveau point.

Ce chapitre propose ou présente divers algorithmes incrémentaux (tracé de segment de droite, de cercles, de polygones du second degré). Ils fonctionnent dans des espaces à deux dimensions et la suite décrit de quelle manière on peut les agencer pour permettre un calcul dans trois dimensions (ou plus). Les dimensions supplémentaires servent à décrire l'aspect. Enfin la méthode d'éclairement proposée simule la réalité sans la reproduire ce qui simplifie les calculs.

III.1. QUELQUES ALGORITHMES D'INTERPOLATION LINEAIRE

De nombreux algorithmes d'interpolation linéaire ont été proposés par différents auteurs. Cependant pour fixer les idées nous travaillons à partir de l'interpolateur de BRESENHAM, la plupart des autres étant des variantes de celui-ci plus ou moins adaptées à telle ou telle particularité.

Nous proposons trois algorithmes dérivant de la solution de BRESENHAM après une présentation des caractéristiques permettant de les analyser. Enfin une étude comparative de performance conclue ce paragraphe.

III.1.1. Notations et éléments d'analyse

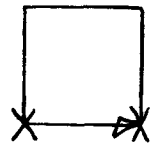
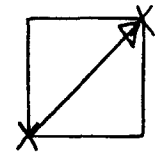
Les segments de droite sont décrits par leurs extrémités (X1, Y1) et (X2, Y2) avec :

DX = X2 - X1
DY = Y2 - Y1
(X, Y) est la coordonnée courante du point à tracer.

Cette présentation se limite au cas ou le segment de droite appartient au premier octant (DX> = DY> = 0).

Les algorithmes incrémentaux possèdent tous les trois particularités suivantes :

- Possibilité de n "mouvements" (notés Mi) différents à chaque pas du calcul. Ce nombre dépend de la géométrie du support de dessin (en général un maillage carré) et des caractéristiques de l'interpolateur, Le nombre de mouvements est le plus souvent de deux (dans le premier octant) :

mouvement 1  mouvement 2 

- Au plus n "indicateurs de qualité" (notés IQi) mesurant la qualité des mouvements. La manière dont ils sont réévalués à chaque itération est un facteur de personnalisation de l'algorithme.

- Au plus n critères de choix qui combinés aux indicateurs de qualité permettent de choisir un mouvement. Ils influent par leur précision sur la qualité du tracé et sur son efficacité [YASUHITO 79].

III.1.2. Algorithme A1 [BRESENHAM 65]

III.1.2.1. Les mouvements

Deux mouvements sont possibles :

- mouvement 1 : horizontal, il est équivalent à

```
M1 : Afficher (X, Y) ;
     X=X+1
```

- mouvement 2 : diagonal, il est équivalent à

```
M2 : Afficher (X, Y) ;
     X=X+1 ; Y=Y+1
```

III.1.2.2. L'indicateur de qualité

La pente du segment de droite est DY/DX, alors chaque fois que X est incrémenté (coordonnée entière) Y doit être augmenté de DY/DX. Cette quantité étant inférieure à 1 et Y ne pouvant être augmenté que de 0 ou de 1, l'indicateur de qualité IQ est augmenté à chaque itération de :

- DY/DX quand Y progresse de 0,
- DY/DX - 1 quand Y progresse de 1.

Remarque : LUCAS modifie cet algorithme en rendant les valeurs entières :

$0 \leq DY/DX \leq 1$ devient $0 \leq DY \leq DX$

$-1 \leq DY/DX-1 \leq 0$ devient $-DX \leq DY-DX \leq 0$

III.1.2.3. Critère de choix

A chaque itération IQ est comparé à DX et l'algorithme de choix est le suivant :

```
si IQ <= DX alors M1 ; IQ := IQ+DY
           sinon M2 ; IQ := IQ-DX+DY
```

où IQ est initialisé à DX/2 de façon à "centrer" le tracé.

Cependant une comparaison à 0 étant plus efficace qu'une comparaison à DX, IQ subit initialement une translation de -DX.

III.1.2.4. Algorithme

```
début
  M1 := horizontal ;
  M2 := vertical ;
  A := DY ;
  B := DX-DY ;
  IQ := -DX/2 ;
  FIN := DX
  pour I = i à FIN faire
    si IQ <= 0 alors M1 ; IQ := IQ+A
               sinon M2 ; IQ := IQ-B
    fsi
  fait
fin.
```

III.1.2.5. Propriété du tracé

Tout segment de droite engendré par l'algorithme A1 est de la forme suivante :

1 - Il n'existe pas de mouvements 1 consécutifs si B <= A,

2 - Il n'existe pas de mouvements 2 consécutifs si A <= B.

La démonstration de cette propriété est faite par récurrence sur le nombre d'itérations de la boucle de tracé, elle ne s'intéresse qu'au cas 1, le cas 2 pouvant être vérifié par dualité.

On a :

$A = DY,\ B = -DY + DX$

d'où

$$DX/2 < A \leq DY \qquad (3)$$
$$-DX/2 < -B \leq 0 \qquad (4)$$

La condition initiale est la suivante :

$$IQ_1 = -DX/2$$

Le premier mouvement est donc M1 et :

$$IQ_2 = IQ_1 + A = DY - DX/2 > 0$$

Le deuxième mouvement est M2, le premier mouvement M1 est donc unique.

A la ième itération on suppose que IQ_{i-1} vérifie la relation

$$0 < IQ_{i-1} \leq DX - DY$$

ce qui implique l'exécution d'un mouvement M2. On montre qu'on peut toujours arriver dans cette situation car tant que IQ est strictement positif chaque itération le diminue de la quantité DX - DY qui est strictement positive (si DX = DY la propriété est vérifiée).

Nous avons alors :

$$IQ_i = IQ_{i-1} - B = IQ_{i-1} + DY - DX$$

d'où

$$DY - DX < IQ_i \leq 0 \qquad \text{et} \qquad M1$$

puis

$$IQ_{i+1} = IQ_i + A = IQ_i + DY$$

d'où

$$2*DY - DX < IQ_{i+1} \leq DY \qquad \text{et} \qquad M2$$

car

$$0 < 2*DY - DX$$

Le cas 1 est démontré, le cas 2 se démontre de la même façon.

III.1.3. Algorithme A2

Cet algorithme qui utilise le principe développé par LOCEFF est ici présenté de façon différente [LOCEFF 80].

III.1.3.1. Les mouvements

La quantité DX/DY indique la variation en X correspondant à une variation de 1 en Y. Posons n = ENT(DX/DY), alors :

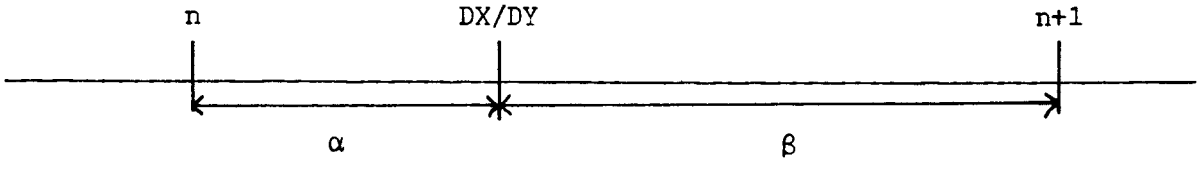
M1' = (n-1)*M1, M2
M2' = n*M1, M2

On se sert de la propriété 1.2.5. et du fait que :

$$1 \leq n \leq DX/DY < N+1$$

III.1.3.2. L'indicateur de qualité

On peut représenter l'encadrement de DX/DY de la manière suivante :



avec $\alpha = DX/DY - n$ et $\beta = n + 1 - DX/DY$

alors a chaque M1' une erreur de α est commise et à chaque M2' une erreur de $-\beta$ est commise.

α et β sont rendus entiers par multiplication par DY :

$$A' = \alpha * DY = DX - n*DY$$
$$B' = \beta * DY \; ; \; (n+1)*DY - DX$$

avec $0 \leq A' < DY$ et $0 < B' \leq DY$

L'indicateur de qualité IQ cumule alors les erreurs faites à chaque mouvement :

$$M1' : IQ = IQ + A'$$
$$M2' : IQ = IQ - B'$$

III.1.3.3. Critère de choix

A chaque itération IQ est comparé à DY, s'il est inférieur à DY alors un mouvement M1' est choisi, sinon M2' est choisi.

Pour les mêmes raisons qu'en 1.2.3. IQ est translaté de -DY, il est donc initialisé à la valeur -DY/2.

III.1.3.4. Algorithme

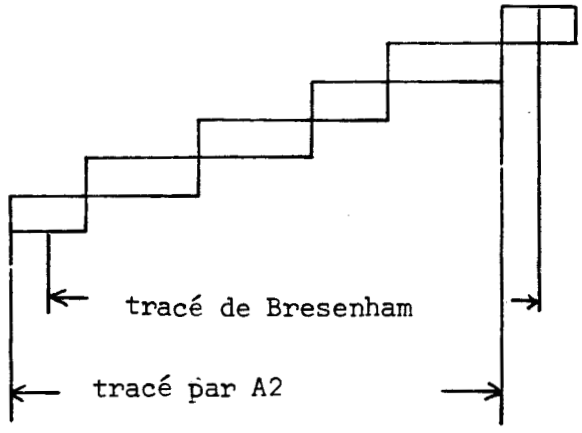
Lorsque DX/DY est une quantité entière seul le mouvement M1' est utilisé, d'autre part l'erreur effectuée à chaque itération est nulle en effet A' = Dy*n - DX = 0. Le cas DY = 0 est un cas particulier de cette remarque que nous excluons de l'algorithme par souci de clarté :

si DY = 0 alors DY = 1 ;

L'algorithme s'écrit :

```
début
    N = ENT(DX/DY)
    A' = DX - N*DY
    B' = (N+1)*DY - DX
    IQ = - DY/2
    FIN = DY
    pour I = 1 a FIN faire
        si IQ    0 alors M1' ; IQ = IQ + A'
                 sinon M2' ; IQ = IQ - B'
                 fsi
        fait
fin
```

Remarques : - Le centrage du tracé sur la droite idéale nécessiterait deux tracés particuliers avant et après la boucle principale. Cette modification permettrait d'obtenir le même ensemble de points qu'avec l'algorithme A1.



- La plupart du temps l'interpolateur ajoute des points en fin de tracé, un dispositif matériel permettrait d'éviter cet inconvénient sans pénaliser les performances.

III.1.3.5. Propriété

La propriété 1.2.5. se retrouve ici en remplaçant (DX, DY) par (B', A') :

1- Il n'existe pas de M1' consécutifs si A' ≥ B'

2- Il n'existe pas de M2' consécutifs si B' ≥ A'

La démonstration de cette propriété est bien entendu la même que précédemment.

III.1.4. Algorithme A3

Cet algorithme se déduit des précédents en utilisant la propriété 1.3.5.

III.1.4.1. Les mouvements

Le rapport α/β indique le nombre de M1' exécuté pour un M2' exécuté. Nous ne nous intéressons qu'au cas $\alpha \leq \beta$, le cas $\beta < \alpha$ étant symétrique.

On a :

$$p < \alpha/\beta < p+1 \qquad p \in N$$

Deux mouvements sont alors possibles :

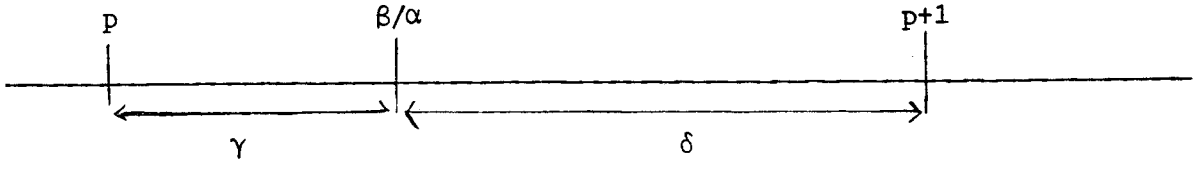
M1" : p*M1', M2'

M2" : (p+1)*M1', M2'

Dans le cas non étudié ou $\alpha > \beta$ et $p \leq \alpha/\beta < p+1$ il faudrait intervertir M1' et M2' dans les expressions de M1" et M2" précédentes.

III.1.4.2. Indicateur de qualité

L'encadrement de β/α est le suivant :



Chaque M1" induit une erreur de γ,

chaque M2" induit une erreur de $-\delta$.

Posons : $A'' = \gamma * (\alpha * DY)$

$B'' = \delta * (\alpha * DY)$

avec $0 <= A'' < DX - n*DY$

$0 < B'' <= DX - n*DY$ et $A'', B'' \in N$

L'indicateur de qualité cumule les erreurs comme suit :

M1" : IQ = IQ + A"

M2" : IQ = IQ - B"

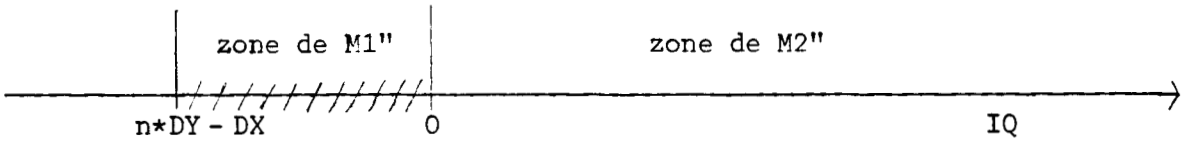
III.1.4.3. Critère de choix

La comparaison de IQ a 0 permet de choisir :

M1" si IQ ≤ 0

M2" si IQ > 0

En initialisant IQ à -(DX - n*DY)/2 on obtient les zones de mouvement suivantes :



III.1.4.4. Algorithme

Les deux cas particuliers suivants sont exclus de l'algorithme :

1- DY = 0 on a ici une ligne horizontale, on force

DY à 1 et p à 1,

2- A' = 0 Le rapport DX/DY est un nombre entier et n devient égal à DY, ce qui revient à forcer A' à 1.

L'algorithme s'écrit :

```
début
    si DY = 0 alors DY = 1;
    N = ENT(DX/DY);
    A' = DX - N*DY;
    B' = (N+1)*DY - DX

    si A' = 0 alors A' = 1
    P = ENT(B'/A')
    A" = B' - P*A'
    B" = (P+1)*A' - B'
    IQ = -(DX - N*DY)/2
    FIN = ENT(DY/(P+1)) + 1
    pour I = 1 à FIN faire
         si IQ <= 0 alors M1" ; IQ = IQ + A"
                    sinon M2" ; IQ = IQ - B"
                    fsi
         fait
    fin
```

Les remarques du paragraphe 1.3.4. sont encore valables ici, cependant une initialisation qui permettrait d'obtenir le même ensemble de points que l'algorithme A1 paraît difficilement réalisable.

III.1.4.5. Propriété

De la même façon qu'en 1.2.5. la propriété suivante est vérifiée :

1- Il n'existe pas de M1" consécutifs si A" ≥ B"
2- Il n'existe pas de M2" consécutifs si B" ≥ A"

III.1.5. Performance

La performance d'un algorithme Ai sera définie par le rapport du nombre d'opérations effectuées par Ai sur le nombre d'opérations effectuées par Ai pour tracer le même segment de droite :

$$P(Ai) = NO(A1) / NO(Ai)$$

III.1.5.1. Conventions

Du nombre d'opérations sont exclues d'une part les initialisations qui peuvent être réalisées en différé, d'autre part ce qui est en rapport avec l'affichage, certaines machines permettant en effet d'afficher simultanément plusieurs points [GUPTA 81]. De plus la gestion des coordonnées (X, Y) peut être effectuée en simultanéité avec le calcul de mouvement.

Le nombre d'opérations (NO) ne comprend plus alors que les actions effectuées sur l'indicateur de qualité.

Pour les algorithmes étudiés nous avons :

NO(A1) = DX * ("?","+","=")
NO(A2) = DY * ("?","+","=")
NO(A3) = (DY / (1 + max(ent(β/α), ent(α/β)))) * ("?","+","=")

où = DX/DY - ENT(DX/DY)
= 1 - α

et "op" représente le temps d'exécution de l'opération op :

III.1.5.2. Calcul

Nous avons :

- P(A2) = NO(A1) / NO(A2) = DX/DY (7)
- P(A3) = NO(A1) / NO(A3) = (1+p) * (DX/DY) (8)

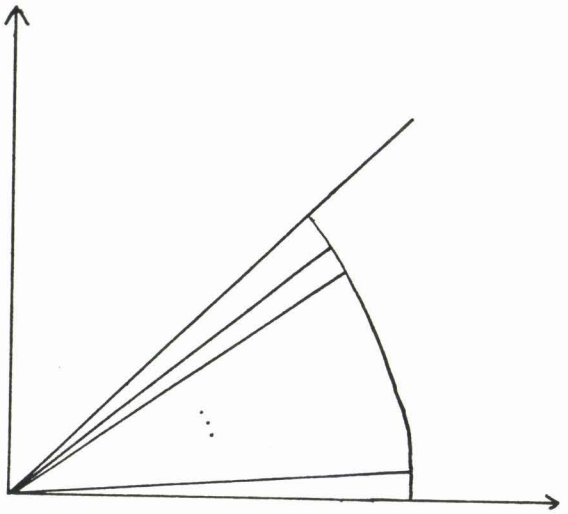
où p = max(ent(β/α), ent(α/β)).

Le calcul des performances P(A2) et P(A3) est effectué sur tous les segments de droite de même longueur contenus dans le premier octant (dans les autres octants les performances sont symétriques).

Soit N la longueur choisie pour les segments alors tout segment sur lequel est effectué le calcul de performance vérifie :

$$X^2 + Y^2 = N^2 \text{ d'où } Y = (N^2 - X^2)^{1/2}$$

où (X, Y) est la coordonnée du point à atteindre, le point de départ étant (0, 0) :



En réécrivant (7) et (8) :

$$P(A2) = DX / (N^2 - DX^2)^{1/2}$$

$$P(A3) = P(A2)(1 + p(DX,(N^2 - DX^2)^{1/2}))$$

La figure 1 présente les valeurs de P(A2) et P(A3) lorsqu'on balaye circulairement le premier octant avec N = 250.

III.1.5.3. Moyenne des performances

La moyenne des performances pour un algorithme Ai est décrite par :

$$M(Ai) = \left[\sum_{k=1}^{n} P(Ai(Dk)) \right] / n$$

où Dk est le kième segment tracé et n est le nombre total de segments.

Lors des calculs il apparaît que M(Ai) est une fonction du paramètre N longueur des segments.

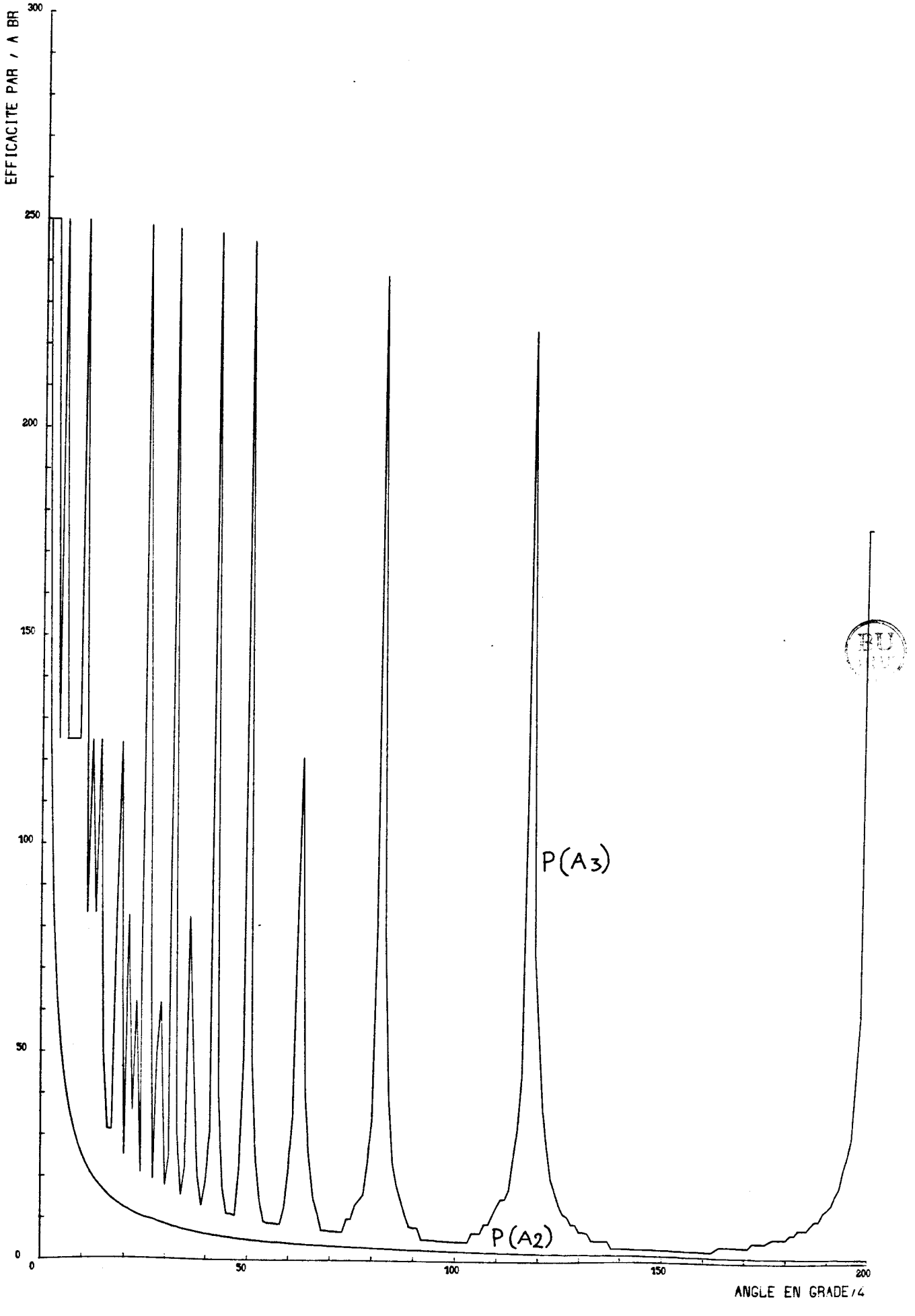

EFFICACITE PAR / A BR
300
250
200
150
100
50
0
0
50
100
150
200
ANGLE EN GRADE/4
P(A3)
P(A2)

Figure 1

N	M(A2)	M(A3)
10	3.7	8.2
50	5.6	21.1
100	6.9	28.8
150	7.5	28.7
200	8.2	37.7
250	8.7	39.4

Ajoutons en guise de conclusion que la fidélité des segments de droite tracés par A2 et A3 par rapport au segment mathématique est la même que celle de l'algorithme A1 (à une translation en X et aux points exédentaires près).

III.1.6. Domaine de variation des indicateurs de qualité

Dans l'optique de réalisations concrêtes il est important de connaître l'encombrement en nombre de bits des différentes variables d'un algorithme de calcul.

III.1.6.1. Algorithme A1

Supposons que X et Y utilisent N bits de codage, il reste à déterminer l'encombrement des variables A, B, IQ, FIN, DX et DY.

Par permutation des points d'extrémité on peut dire que DY sera toujours positif.

DX utilise N bits
DY utilise N bits
FIN utilise N bits
A utilise N bits
B peut être encadré de la façon suivante :

$$\min(DX - DY) \leq B \leq \max(DX - DY)$$

$$0 \leq B \leq 2^N - 1$$

B nécessite donc N bits.

Une des conséquences de la démonstration de 1.2.5. est de montrer que

$$-B + 1 \leq IQ \leq A$$

d'où

$$-2^N + 1 \leq IQ \leq 2^N$$

IQ utilise alors N+2 bits de codage.

III.1.6.2. Algorithme A2

Comme précédemment nous avons les domaines suivants :

$$0 \leq X < 2^N$$
$$0 \leq Y < 2^N$$
$$0 \leq DX < 2^N$$
$$0 \leq DY < 2^N$$
$$1 \leq N < 2^N$$
$$0 \leq A' < 2^N$$
$$0 \leq B' < 2^N$$
$$-B' + 1 \leq IQ \leq A'$$
$$-2^N + 2 \leq IQ \leq 2^N - 1$$

IQ nécessite donc N+1 bits.

III.1.6.3. Algorithme A3

$$1 \leq P \leq 2^N - 1$$
$$0 \leq A'' \leq 2^N - 2$$
$$0 \leq B'' \leq 2^N - 2$$
$$-B + 1 \leq IQ \leq A''$$
$$-2^N + 3 \leq IQ \leq 2^N - 3$$

IQ nécessite donc N + 1 bits.

Remarque : Ces résultats sont valables dans le premier octant, en outre sans tenir compte de l'initialisation des mouvements, les variables de contrôle IQ, A, B, A', B', A" et B" possèdent le même encombrement dans les autres octants.

III.1.7. Conclusion

La démarche suivie pour passer de l'algorithme A1 à A2, puis de A2 à A3 pourrait être à nouveau utilisée. Cependant la complexité des initialisations paraît incompatible avec la simplicité attendue de tels algorithmes. En revanche ce principe de divisions successives pourrait servir à coder de façon efficace des segments de droite, d'autant que trois divisions seraient suffisantes dans beaucoup de cas pour décrire le segment.

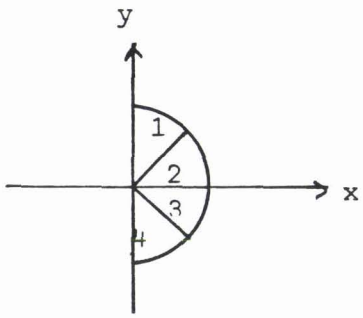
A partir de l'algorithme A2 des problèmes de convergence en fin de segment apparaissent (si aucune précaution n'est prise ces algorithmes auront en effet tendence à afficher des points qui "appartiennent" à la droite mais pas au segment). Un dispositif parallèle au calcul, de sélection des points permettrait de résoudre ce problème sans pénaliser les performances. On peut aussi considérer ces algorithmes comme des traceurs de droites et non de segments, la difficulté de l'arrêt disparaît.

Remarquons que certaines transformations géométriques comme la translation et le changement d'échelle (de même valeur sur tous les axes) n'affectent pas les constantes et n'obligent donc pas à les recalculer.

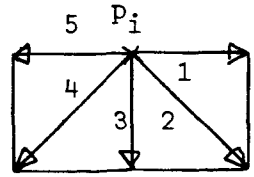
Enfin la résolution de l'aliassage pourrait être effectuée une fois pour toutes pour chacun des deux mouvements.

III.2. INTERPOLATION INCREMENTALE SEMI-CIRCULAIRE

L'algorithme présenté se déduit de celui de BRESENHAM [BRESENHAM 77]. Il calcule le demi-cercle Est de rayon R et de centre l'origine.



Le calcul commence au point (0, R) et se termine en (0, -R). Les mouvements utilisés sont les suivants :



A chacun des mouvements Mk est associé un indicateur de qualité IQk qui permet de choisir le prochain point le plus près du cercle. Donnons une fonction croissante de l'éloignement d'un point (X, Y) par rapport au cercle :

$$D = X^2 + Y^2 - R^2$$

Alors si le point Pi a été calculé comme étant près du cercle le prochain point P_{i+1} sera obtenu par le mouvement Mk tel que :

$$IQk = \inf(|D1|, |D2|, |D3|, |D4|, |D5|) \qquad k \in [1, 5].$$

Pi ayant pour coordonnées (X, Y). Les évaluations des IQi sont les suivantes :

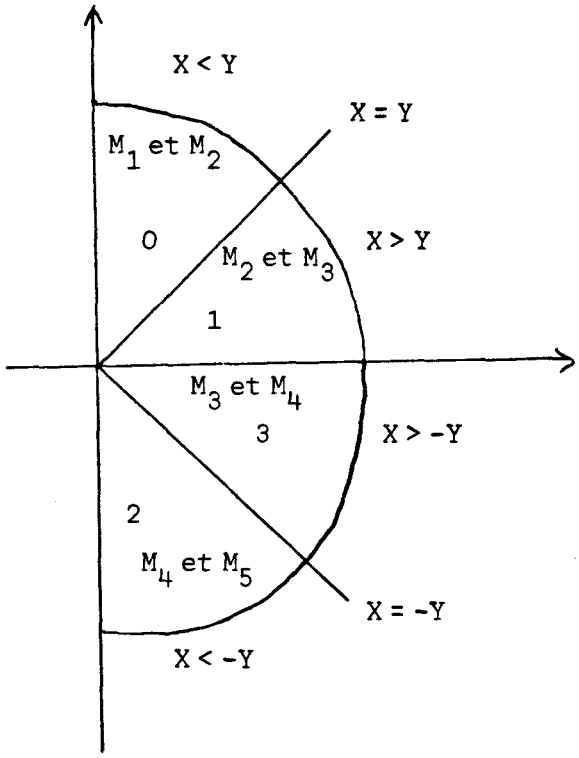
$$M1 : IQ1 = (X+1)^2 + Y^2 - R^2 = Di + 2^X + 1$$
$$M2 : IQ2 = (X+1)^2 + (Y-1)^2 - R^2 = Di + 2^X - 2^Y + 2$$
$$M3 : IQ3 = X^2 + (Y-1)^2 - R^2 = Di - 2^Y + 1$$
$$M4 : IQ4 = (X-1)^2 + (Y-1)^2 - R^2 = Di - 2^X - 2^Y + 2$$
$$M5 : IQ5 = (X-1)^2 + Y^2 - R^2 = Di - 2^X + 1$$

où Di est la valeur de D pour Pi.

L'évaluation des IQk n'utilise donc que des additions, soustractions et multiplications par deux.

Minimisation

Dans chacun des octants du demi-espace seuls deux mouvements sont nécessaires et donc deux indicateurs de qualité. Il est nécessaire pour profiter de cette réduction de nommer chacun des octants comme cela est fait dans la figure suivante :



Une autre remarque est que le terme IQ5 (resp. IQ4) se déduit de IQ1 (resp. IQ2) par simple changement de signe du facteur 2*X.

En réécrivant les évaluations des indicateurs de qualité il vient :

```
IQ1 = Di + 2*X + 1
IQ2 = IQ1 - 2*Y + 1,     IQ2 = IQ3 + 2*X + 1
IQ3 = Di - 2*Y + 1
IQ4 = IQ5 - 2*Y + 1,     IQ4 = IQ3 - 2*X + 1
IQ5 = Di - 2*X + 1
```

Encadrement des indicateurs de qualité

L'erreur exacte commise à chaque point est la suivante :

$$-n \leq (X^2 + Y^2)^{1/2} - R \leq n \text{ avec } n = 1/((2)^{(1/2)})$$

Une majoration de l'encadrement pourrait alors être :

$$-4*R \leq IQi \leq 5*R$$

III.2.1. Algorithme

```
début
    NX, NY = 0
    ZONE = 0
    D = 0
    M = 1
    pour I = 1 à 2*R + 1 faire
         DCRY = faux
         tant que non DCRY faire
              cas ZONE mod 2
         1 : début
                   A = D + M*2*NX + 1
                   B = A - 2*NY + 1
                   si |B| ≤ |A| alors DCRY = vrai
                                      NY = NY - 1
                                      D = B
                              sinon D = A
                              fsi
                   NX = NX + M
                   fin
         2 : début
                   DCRY = vrai
                   A = B - 2*NY + 1
                   B = A + M*2*NX + 1
                   si |B| ≤ |A| alors NX = NX + M
                                      D = B
                              sinon D = A
                              fsi
```

```
            NY = NY - 1
            fin
        fin cas
        cas ZONE
    1 :     si NY ≤ NX alors ZONE = 1 fsi
    2 :     si DY = 0 alors ZONE = 3
                            M = -1 fsi
    3 :     si NX = 0 alors DCRY = vrai fsi
    4 :     si -NY ≥ NX alors ZONE = 2 fsi
      fin cas
    fait
fin
```

III.2.2. Conclusion

Citons en conclusion l'algorithme de C. CARREZ semblable au précédent mais qui évite dans certains cas l'évaluation d'un des termes A ou B. Cette amélioration ne nous paraît cependant pas nécessaire dans cette étude du fait de la séquentialité impliquée [CARREZ 77]. En effet le but poursuivi est plus de pouvoir paralléliser les opérations effectuées que de diminuer leur nombre en augmentant la séquentialité.

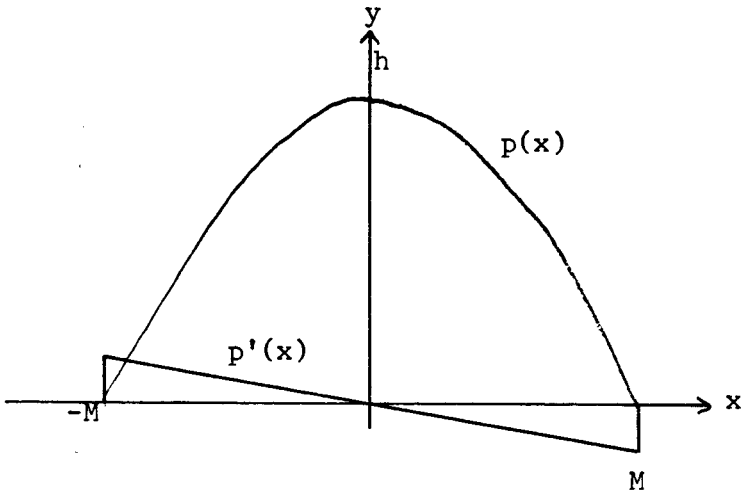
III.3. CALCUL INCREMENTAL DE COURBES SYMETRIQUES DU SECOND DEGRE

L'algorithme présenté n'est pas de portée générale. Les fonctions P(x) à calculer vérifient en effet les contraintes suivantes :

- P(x) est de la forme : $P(x) = -(h/M^2)*x^2 + h$, $(M, h) \in N^2$

Le calcul porte sur l'intervalle [-M, M]

P(x) a donc l'aspect suivant :



avec P'(-M) = -P'(M) = 2*(h/M)

P'(0) = 0

P(x) est calculé avec la méthode des trapèzes, P'(x) indiquant la variation de P(x) à chaque pas. P'(x) est calculé directement et de façon incrémentale.

L'algorithme s'écrit alors :

```
P1 : début
        P = 0
        P' = 2*(H/M)
        I = 0
        pour j = 1 à M faire
             P' = P' - 2*(H/M^2)
             tant que I < P' faire
                  I = I + 1
                  P = P + 1
                  fait
             I = I - P'
             fait
        /* P' est nul */
        pour J = 1 à M faire
             P' = P' + 2*(H/M^2)
             tant que I < P' faire
```

```
                I = I + 1
                P = P - 1
                fait
            I = I - P'
            fait
        fin
```

Dans cet algorithme seule la variable P peut être considérée entière. Pour rendre I et P entier il faut multiplier les constantes par M^2 et l'algorithme devient :

```
P2 : début
        P = 0
        P' = 2*H*M
        I = 0
        pour j = 1 à M faire
            P' = P' - 2*H
            tant que I < P' faire
                I = I + M^2
                P = P + 1
                fait
            I = I - P'
            fait
        pour j = 1 à M faire
            P' = P' + 2*H
            tant que I < P' faire
                I = I + M^2
                P = P - 1
                fait
            I = I - P'
            fait
        fin
```

Les intervalles de variation de I et P' sont alors très importants. Dans le cas particulier où M = H ces intervalles peuvent être réduits :

```
P3 : début
        P = 0
        P' = 2*M
        I = 0
        pour J = 1 à M faire
             P' = P' - 2
             tant que I < P' faire
                  I = I + M
                  P = P + 1
                  fait
             I = I - P'
             fait
        pour J = 1 à M faire
             P' = P' + 2
             tant que I < P' faire
                  I = I + M
                  P = P - 1
                  fait
             I = I - P'
             fait
        fin
```

Encombrement des variables

P2 : $0 \leq P' \leq 2*H*M$

$0 \leq I \leq 2*H*M - 1 + M^2$

P3 : $0 \leq P' \leq 2*M$

$0 \leq I \leq 3*M - 1$

Dans les trois algorithmes la variable I permet de connaître la valeur d'incrément de P en chaque X. Autrement dit I discrétise les variations de P'. L'erreur sur I est comprise dans [0, 1] (resp. [0, M^2], [0, M]) dans P1 (resp. P2, P3), l'erreur sur P est alors comprise dans [0, 1] pour les trois algorithmes.

On peut centrer cette erreur en O en initialisant I à 1/2 (resp. M^2, M/2) dans P1 (resp. P2, P3).

Remarquons en outre que le nombre d'itérations des boucles internes ("tant que") est limité par :

$$f(2*H/M)$$

avec $f(x) = x$ si $x \in N$

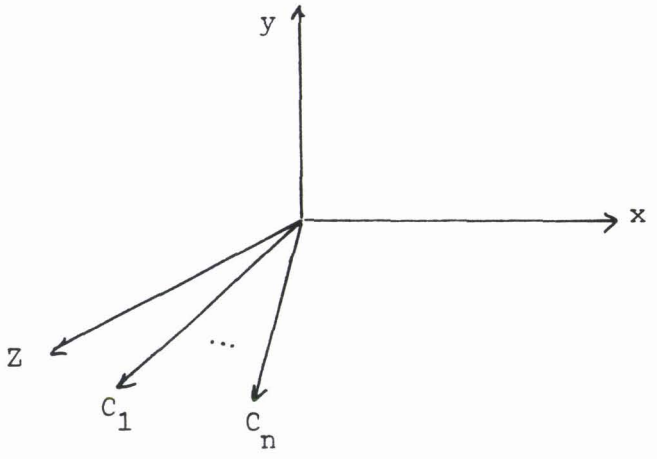
$f(x) = ent(x) + 1$ sinon

Le nombre d'itérations de P3 est donc au maximum de deux. Dans la suite P3 est utilisé comme approximation d'un demi-cercle.

Le terme d'"interpolateur" sera utilisé par abus de langage pour désigner les différents algorithmes incrémentaux qui viennent d'être présentés.

III.4. APPLICATIONS DES INTERPOLATEURS

Ce paragraphe s'inspire de la technique d'éclairement introduite par GOURAUD [GROURAUD 71]. Les calculs géométriques se font dans l'espace écran ramené à trois dimensions (ajout de la dimension "profondeur"). Les calculs d'aspect se font dans l'espace écran ramené à n+2 dimensions (les n dimensions supplémentaires pouvant être "intensité", "transparence",...). Plus généralement l'espace de calcul sera :



où : - (X O Y) est le plan de l'écran,
- Z est la profondeur,
- {C1,..., Cn} représente l'aspect.

III.4.1. Interpolation linéaire en trois dimensions

Si le segment de droite à interpoler est défini par ses deux extrémités (X1, Y1, Z1) et (X2, Y2, Z2) et que l'interpolation se fait par rapport à Y, il suffit à priori d'interpoler simultanément en X et en Z en utilisant un des algorithmes présentés en III.1. Cependant pour le segment entier, le nombre de points calculés en X peut être différent de celui pour Z. Il est alors nécessaire de pouvoir associer à un ou plusieurs points en X un point en Z.

L'algorithme A2 est particulièrement bien adapté à ce problème de correspondance puisqu'il ne produit qu'un et un seul mouvement pour une variation en Y (i.e. il produit autant de mouvements en X qu'en Z). La difficulté est maintenant de superposer deux mouvements :

si la longueur du mouvement en X est de p et
la longueur du mouvement en Z est de n
on obtient dans le plan XOY :

Y + 1			
Y	$Z+\frac{n}{p}$	$Z+\frac{2n}{p}$	Z+n
Y - 1	X+1	X+2	X+p

La valeur n/p n'étant à priori pas entière il est nécessaire d'utiliser un nouvel interpolateur linéaire dans le plan y=Y entre les points (X+1, Z+n/p) et (X+p, Z+n). Un algorithme de type A1 demandant peu d'initialisations pourra être utilisé à moins qu'un temps constant de calcul par point ne soit requis.

Généralisation : Dans un espace à N dimensions ($N \geq 4$), une interpolation linéaire peut se faire pour N-2 variables de la même manière que pour Z. En effet ces calculs sont effectués dans le but d'une projection sur le plan XOY.

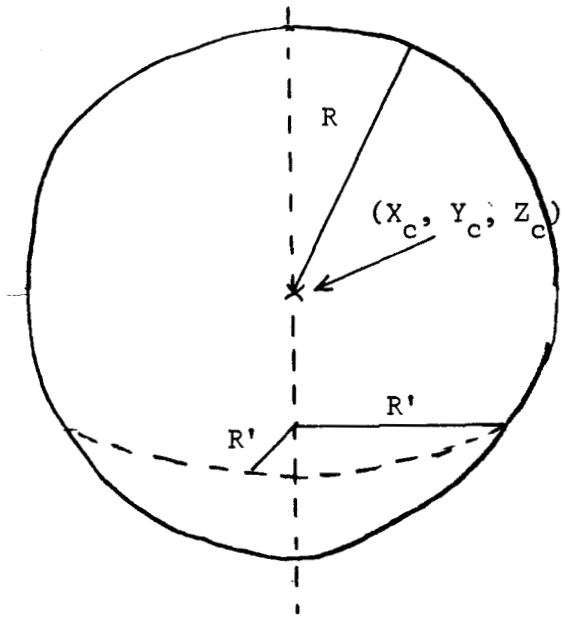
III.4.2. Construction de sphère

La méthode présentée permet le calcul de la demi-sphère visible dans l'espace écran ramené à trois dimensions. Elle suppose l'utilisation d'un algorithme de surfaces visibles du type tampon de profondeur [CATMULL 75].

Le calcul des points d'une sphère de centre (Xc, Yc, Zc) peut être décomposé en deux étapes :

- Calcul des points du périmètre de la sphère. Celle-ci étant destinée à être visualisée sur un support plan, le périmètre est le bord apparant de la sphère.

- Connaissant deux points de bord de la sphère il est possible de calculer le demi-cercle visible de rayon R'.



Deux interpolateurs semi-circulaires suffisent à la construction de la sphère. Cependant pour des raisons d'efficacité on préfèrera utiliser l'algorithme P3 pour calculer la profondeur.

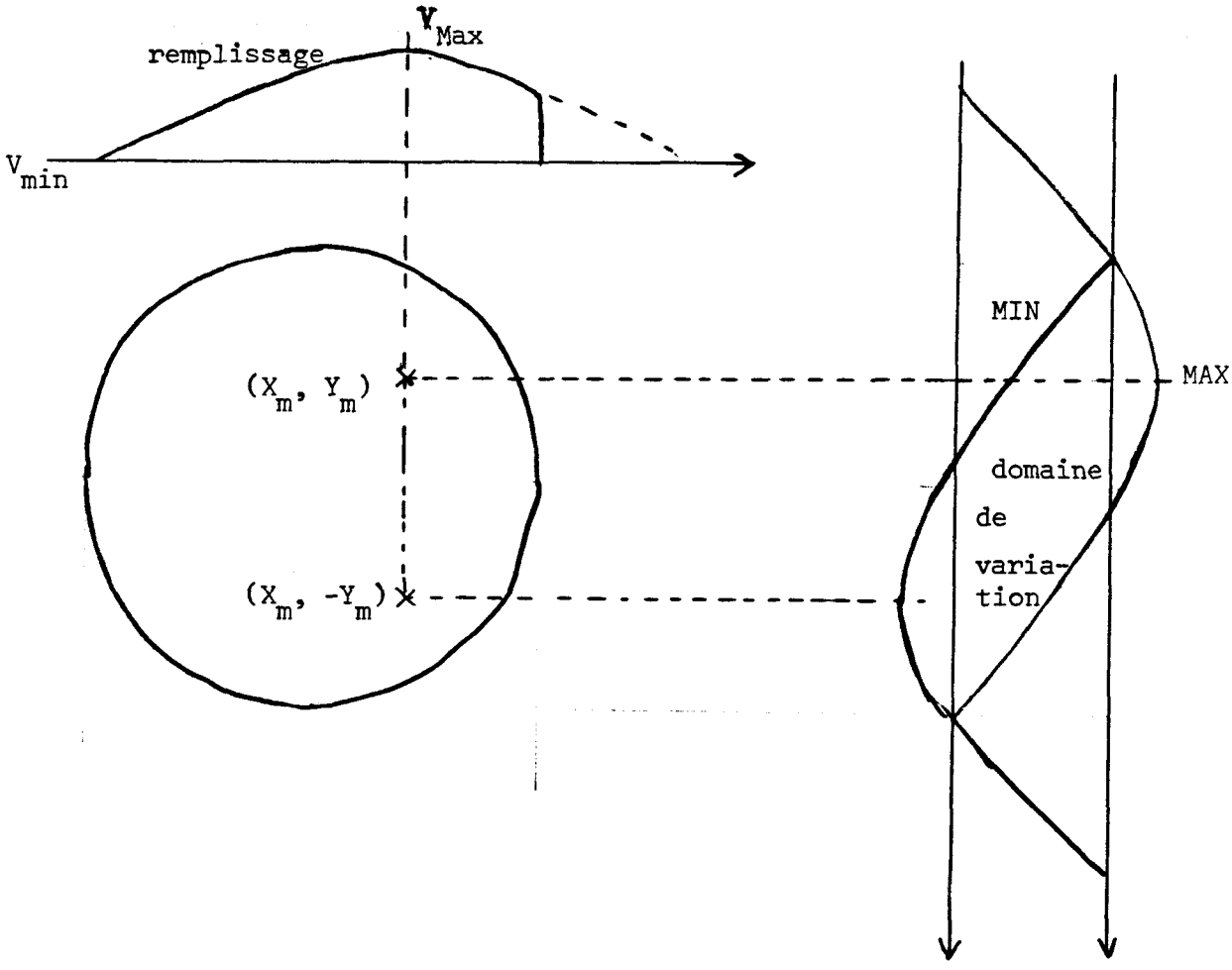
III.4.3. Simulation d'éclairage d'une sphère

Comme précédemment les calculs sont effectués ligne à ligne dans l'espace écran. Le principe du modèle d'éclairage est le même que pour la sphère, il se décompose en deux parties :

- Evaluation du domaine de variation de l'intensité pour chaque ligne traversant le plan de projection de la sphère. Ce domaine est contenu entre deux valeurs extrêmes de l'intensité.

- Pour chaque domaine, calcul point à point de l'intensité effective.

Les deux étapes utilisent un polynome du second degré étudié en III.3.



où (Xm, Ym) représente le point d'intensité maximum par rapport au centre de la sphère.

La courbe "Min" du domaine de variation possède les mêmes paramètres que la courbe "Max" si ce n'est que Ym est remplacé par
- Ym et qu'elle est rendue négative.

III.5. CONCLUSION

Nous avons montré qu'à partir de tracés de figures simples il est possible d'obtenir des objets plus complexes. Ces différentes méthodes de calcul vont servir dans la description de l'architecture du terminal dans le chapitre suivant.

CHAPITRE IV

DESCRIPTION FONCTIONNELLE

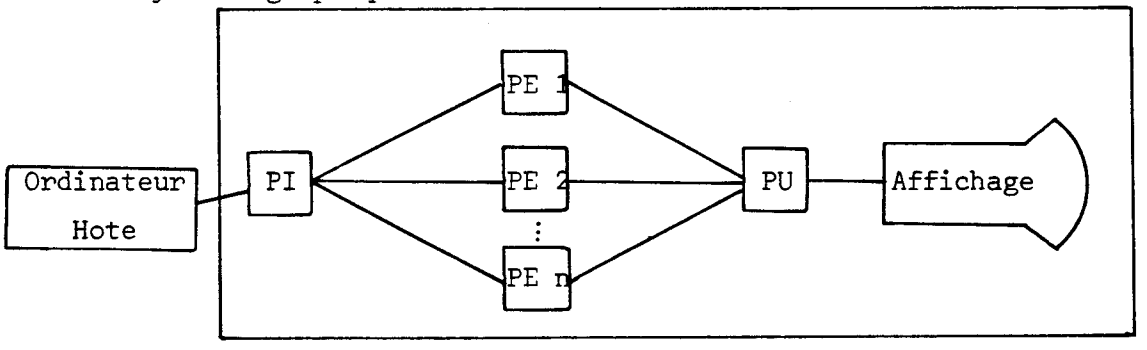
Le système graphique présenté tend a réaliser trois buts :

- Suppression de la mémoire d'image.
- Manipulation d'objets en temps réel.
- Modularité et extensibilité de l'architecture.

Nous avons fait le choix initial d'affecter un "processeur-élément" à chaque élément. Tout élément est alors converti en temps réel par un tel processeur.

L'ensemble des processeurs-élément (PE) est relié au monde extérieur par un processeur interface (PI) qui assure le dialogue ordinateur-hôte, {PE} et par un processeur d'unification (PU) qui sélectionne la ou les informations à afficher. Ce dernier résout les problèmes de surfaces visibles et de conflit d'accès au bus. Enfin un mécanisme de traduction de l'information à afficher en une couleur est intégré à l'affichage est un écran cathodique couleur à balayage de trame.

Système graphique



Cette architecture doit être extensible du point de vue du nombre de PE et modulaire du point de vue de leurs possibilités de traitement. Les PE travaillent au même rythme que l'écran, l'identification entre un point de l'écran et un élément traité par un PE sera immédiate.

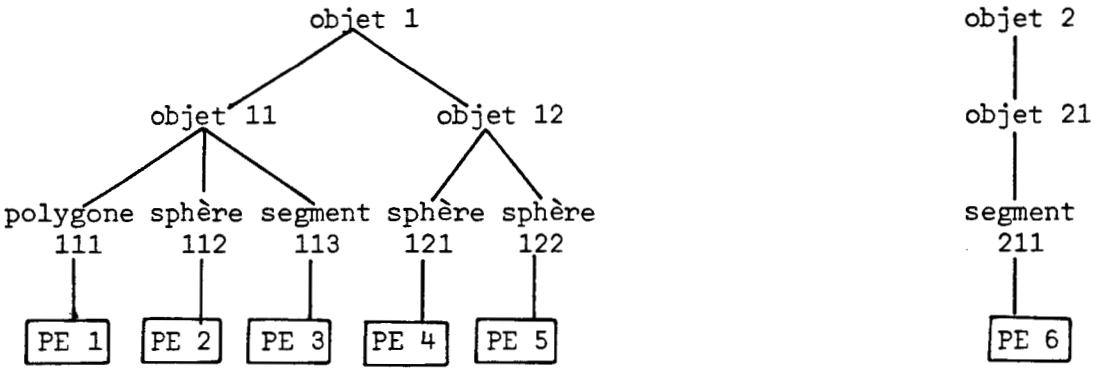
Le premier paragraphe propose une description logique et fonctionnelle du processeur graphique. Les suivants s'intéressent au problème de synchronisation et de communication entre le PI et l'ensemble des PE.

L'étude des PE, du PU et des mécanismes de traduction est reportée au Chapitre V.

IV.1. PRINCIPE ET ARCHITECTURE GENERALE

IV.1.1. Structuration des objets

La structuration adoptée dans le système graphique pour la représentation des objets est arborescente. Un objet peut être composé de sous objets étant eux-mêmes des objets. Les primitives ou feuilles de l'arbre sont les éléments (segments, polygones, sphères) directement calculables par les processeurs-élément. Les contraintes apportées à cette représentation par une implémentation matérielle sont de figer le nombre de niveaux de l'arborescence et d'obliger les feuilles a être toutes au niveau le plus bas, en outre le nombre de fils d'un noeud est limité [BRADIER 81] :



Exemple de structuration figée a trois niveaux et d'affectation des éléments au PE.

Un élément est désigné par son chemin d'accès depuis la racine de l'arborescence. Dans l'exemple précédent un chemin d'accès à un élément comporte trois champs :

(nom d'objets, nom de sous-objet, non d'élément).

Le résultat d'une identification peut concerner un objet complet (i.e. la racine) ou non.

D'autre part cette structuration permet une gestion globale (i.e. par objet) ou locale (i.e. par élément) de la scène.

Tous les éléments sont typés, chaque type correspond à une figure géométrique générique du type. Les figures sont décrites en trois dimensions et sont les suivantes :

- Segment de droite.
- Facette polygonale
- Sphère (traitée comme une surface gauche non fermée).

Les attributs qualifiant un élément sont : son identificateur, sa géométrie (où il se trouve dans l'espace de calcul), sa morphologie (composée de son type et des paramètres associés), son aspect (lié au type de l'élément) au sens de MARTINEZ [MARTINEZ 82].

IV.1.2. Les processeurs-élément

Comme les éléments les processeurs-élément sont typés. Si un PE peut convertir plusieurs types d'éléments, son type sera celui de l'élément effectivement traité ; s'il ne peut traiter qu'un seul type d'élément c'est celui-ci qu'il possède de façon statique. La deuxième possibilité est adoptée par la suite comme hypothèse simplificatrice.

La principale fonction d'un PE est de convertir en temps réel la morphologie et l'aspect d'un élément. L'expression "temps réel" signifie que les résultats des calculs doivent être obtenus au même rythme que l'affichage sur l'écran cathodique. La conversion est effectuée ligne par ligne et colonne par colonne, ce qui justifie l'utilisation d'algorithmes incrémentaux pour les calculs. L'ensemble, ou un sous-ensemble, des PE contient a un instant donné toutes les informations nécessaires à l'élaboration de l'image, on peut alors assimiler celles-ci à une liste de visualisation selon la terminologie de MORVAN [MORVAN 76].

Les autres fonctions d'un PE sont du domaine de l'interaction et de l'affichage qu'on assimile respectivement a un dialogue amont (avec le PI) et à un dialogue aval (avec l'affichage).

Le premier est assuré par le "gérant" :

- Gestion de la sous-liste du PE,
- Prise en compte des demandes d'identification.

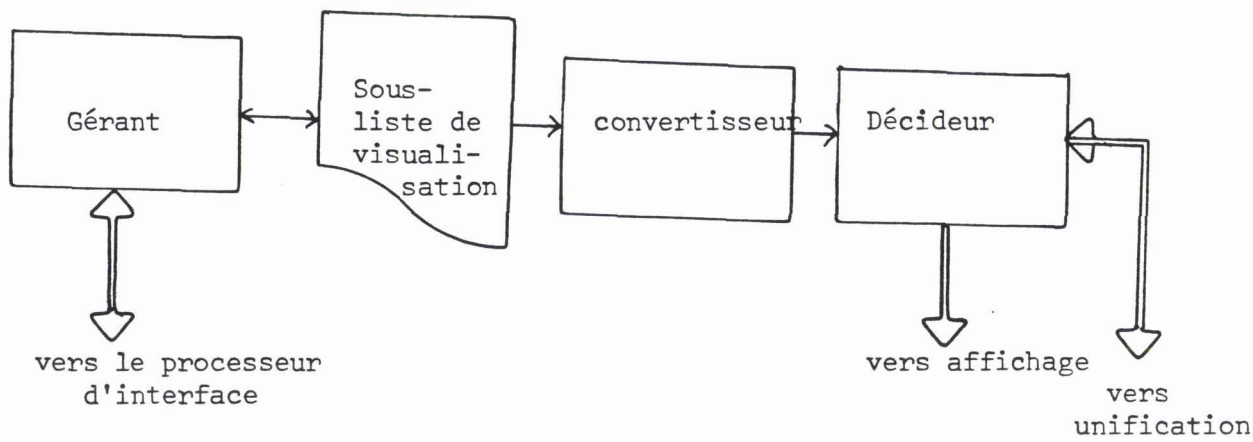
Le second est assuré par le "décideur" qui détermine la visibilité de l'élément du PE.

La sous-liste de visualisation contient l'ensemble des attributs définissant l'élément traité. Ceux-ci ont déjà subi une présynthèse dans le processeur-hôte :

- La morphologie et la géométrie ont été clôturées, modifiées par la perspective...
- Les paramètres de l'attribut d'aspect dépendent du type de l'élément de l'éclairement, de la géométrie.

Notons que si le gérant permet de lire ou d'écrire dans la sous-liste de visualisation d'un PE, il n'a en revanche pas été prévu que celui-ci puisse appliquer des transformations (translation, rotation,...) aux valeurs des attributs, comme c'est le cas dans ARTEMIS [GRUIA 81].

La figure suivante résume l'architecture d'un processeur-élément :



IV.1.3. Le processeur d'interface

Le rôle du processeur d'interface est triple :

- Gérer la communication extérieure, i.e. avec le processeur-hôte.
- Gérer l'assignation des éléments aux PE.
- Assurer une certaine autonomie du terminal par rapport au calculateur principal.

IV.1.3.1. Gestion de la communication externe

Celle-ci doit permettre avant tout de masquer au terminal le support de transmission ordinateur-terminal, selon les principes de FERREIRA [FERREIRA 81 p. 123]. Le PI n'étant pas détaillé par la suite, nous n'irons pas plus avant dans la description de ces communications.

IV.1.3.2. Gestion des processeurs-élément

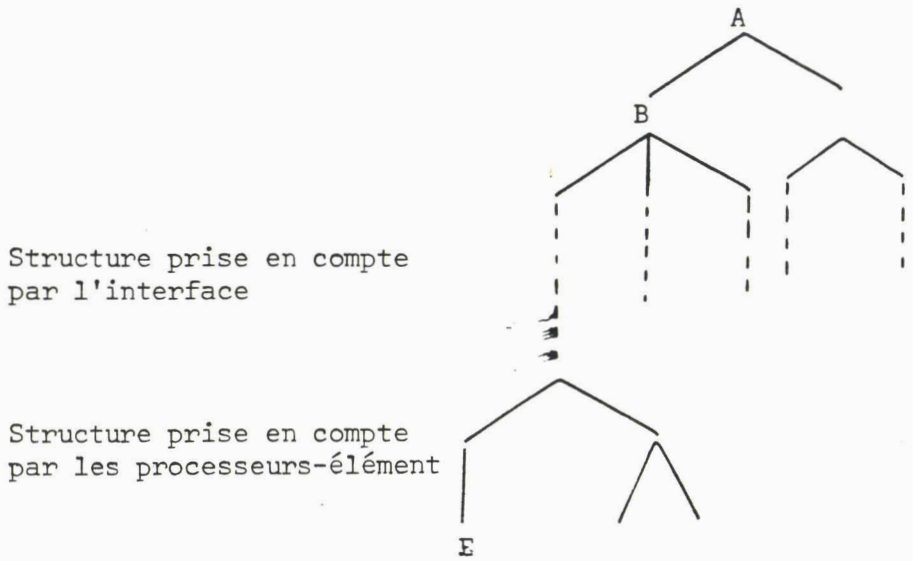
Cette partie fait apparaître les problèmes d'accès au PE. L'ensemble des PE se comporte comme une mémoire associative, nous distinguerons deux modes d'adressage :

- Adressage par nom qui sélectionne le ou les processeurs-élément traitant un objet.
- Adressage par type qui sélectionne un PE inoccupé (i.e. auquel aucun élément n'a encore été affecté).

Cette gestion doit être suffisamment souple pour tenir compte des caractéristiques des différents types (par exemple le nombre de paramètres).

IV.1.3.3. Autonomie du système

A un instant donné le terminal possède un sous-ensemble de l'univers en terme d'objets. Les processeurs-élément possèdent les derniers niveaux d'une structure arborescente, pour rendre cette représentation cohérente nous proposons de doter le PI des niveaux supérieurs de l'arbre :



La désignation de E permet de trouver le sous-objet B

Il s'agit en fait de déplacer une partie de l'"intelligence" du processeur hôte vers le système graphique considéré comme un terminal. Ce travail ne traitant pas de la conception d'un PI nous ne détaillerons pas cet aspect.

IV.1.4. Le processeur d'unificatior

Ce processeur effectue la synthèse des sorties des PE pour un pixel donné. Il résout le problème des surfaces visibles en sélectionnant l'élément visible en un point donné. Il arbitre en outre les conflits en cas d'intersection de plusieurs éléments.

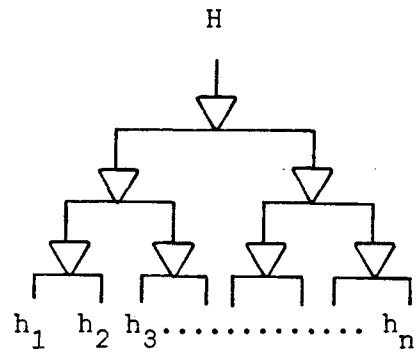
IV.1.5. Mécanisme de traduction et de composition de couleur

Dernière étape avant l'affichage, ces mécanismes composent deux couleurs par effet de transparence et produisent les trois paramètres permettant l'affichage, R, V et B qui sont les fondamentales utilisées par les écrans cathodiques couleurs actuels.

IV.2. SYNCHRONISATION

Les PE peuvent être des milliers, d'autre part ceux-ci doivent travailler de façon synchrone les uns par rapport au autres. Il se pose alors le problème de la distribution de la même horloge à tous les PE.

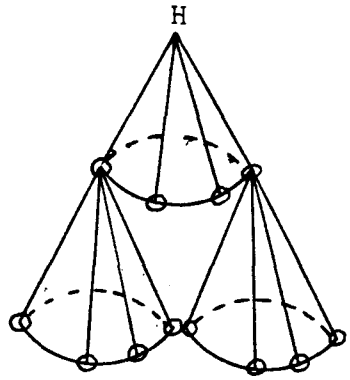
La distribution d'un signal unique a des milliers d'entrées suppose un arbre d'amplification à plusieurs niveaux :



Les différences de retard occasionnées par cet arbre provoquent des déphasages entre les signaux résultants, d'où une désynchronisation des processeurs commandés par ces horloges.

Nous proposons deux solutions partielles à ce problème :

- La période de H est suffisamment "grande" et les différences de phase occasionnées par l'arbre sont assez petites pour être négligées. On peut les minimiser avec un arbre conique ou tout les fils sont de même longueur :



- La fréquence de H est multiple de celle désirée pour les horloges hi. Un arbre d'amplification produit les horloges hi', il suffit alors de diviser les fréquences des hi' pour obtenir les hi.

En initialisant correctement les diviseurs de fréquence il est donc possible de s'assurer que le déphasage est inférieur à une période de H. La difficulté restant en suspens est d'initialiser "correctement" les diviseurs.

En définitive on retiendra la première solution avec laquelle on peut espérer alimenter jusqu'à 4000 processeurs pour un déphasage de l'ordre de dix nanosecondes.

IV.3. LE PROCESSEUR D'INTERFACE

Les communications externes permettant la liaison du PI au processeur hôte ne sont pas étudiées. Seules sont décrites les communications internes (PI-PE) ainsi que deux approches de réalisation du PI.

IV.3.1. Communications internes

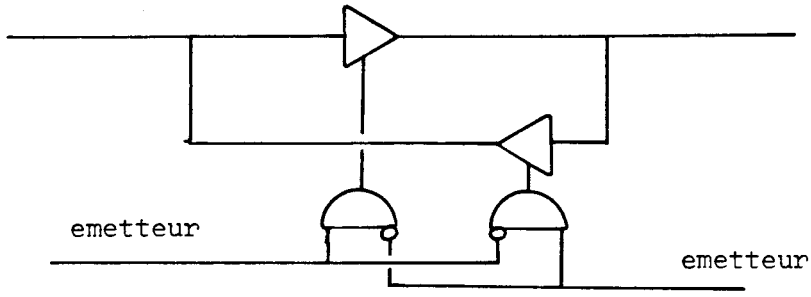
Ces communications relient le PI aux PEs. Deux types d'accès sont distingués, l'accès par nom et l'accès par type. Une troisième sorte d'accès apparaît lors de la désignation d'un point de l'écran.

IV.3.1.1. Accès par nom

Un nom (chemin d'accès) d'objet est associé de façon unique à un objet. Le PI connait le nom de l'objet recherché et le transmet à tous les PEs. Il peut alors être nécessaire que le(s) PE(s) sélectionnés renvoient un accusé de sélection. Cette méthode de sélection est associative puisque se sont les entités adressées elles-mêmes qui se sélectionnent.

De plus le protocole de sélection peut prévoir la sélection à n'importe quel niveau de l'arborescence figée en ajoutant un caractère de fin de nom.

Un bus bidirectionnel distribué à tous les PEs permet la réalisation de cet accès. L'amplification sur un tel bus peut se faire de la façon suivante :

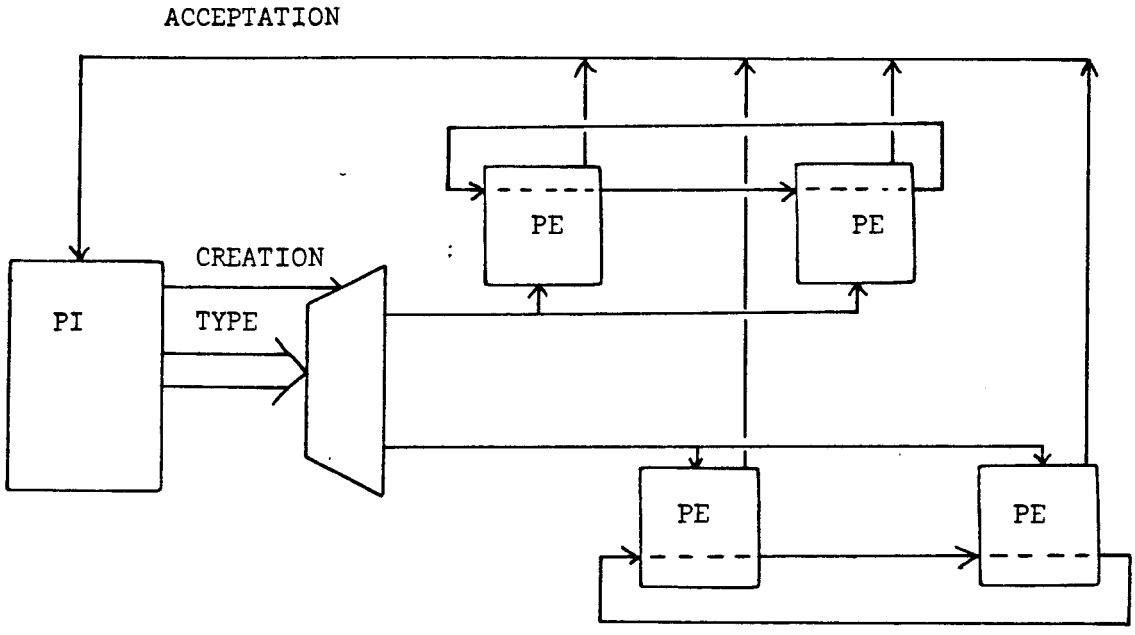


IV.3.1.2. Accès par type

Lors de la création d'un élément de type donné, il s'agit de sélectionner un PE de même type non déjà occupé. La difficulté est de ne sélectionner qu'un processeur-élément. La solution retenue utilise un jeton de vacance par type de PE [DELATTRE 84].

Les PEs sont regroupés par type et dans chaque groupe circule un "jeton de vacance" qui est retenu par le premier PE inoccupé. Lors d'un accès par type, le groupe du bon type est sélectionné et le PE de ce groupe qui possède le jeton signale au PI qu'un processeur est sélectionné.

Remarque : C'est le PI qui doit connaître l'état d'occupation des PEs d'un groupe, en particulier si tous les PEs de ce groupe sont occupés.



La figure présente un schéma de principe de ce mécanisme dans le cas de deux types et de deux PE par types.

IV.3.1.3. Les commandes

Toute commande envoyée aux processeurs-élément comporte deux phases qui sont la sélection d'un sous-ensemble de PEs puis l'envoi de la commande proprement dite. Nous nous limitons aux commandes suivantes :

- CREATION (nom d'élément, type, liste d'attributs).
- MISE-A-JOUR (nom d'objet, type, liste des nouveaux attributs).
- LECTURE (nom d'élément, liste d'attributs concernés).
- DESTRUCTION (nom d'objet).

La signification de ces commandes est détaillée au paragraphe V.1.

IV.3.1.4. Cas de l'identification

L'utilisateur pointant une tache de l'écran, l'identification consiste à reconnaître l'objet ainsi désigné. L'architecture proposée, en associant à tout élément un processeur, permet de résoudre aisément cette identification.

Nous distinguerons deux types d'identification, dans le premier le programme d'application attend une demande d'identification, il suffit de consulter les PE pour obtenir le nom de l'élément désigné [FOLEY 82]. Le second type correspond à une interruption de la part de l'utilisateur, dans ce cas le programme d'application n'"attend" pas de demande d'identification. La difficulté est alors d'assurer qu'une commande en cours d'émission par le PI et d'exécution par un PE n'est pas perturbée par l'interruption.

Une première solution consiste à interrompre simultanément les PEs et le PI. Les gérants abandonnent la commande éventuellement en cours pour envoyer l'identificateur de leur élément vers le PI. Le PI doit être muni des mécanismes nécessaires lui permettant de prendre en compte la demande puis de reprendre au début la commande qui était éventuellement en cours. Cependant cette solution reporte le problème de cohérence au niveau du processeur hôte.

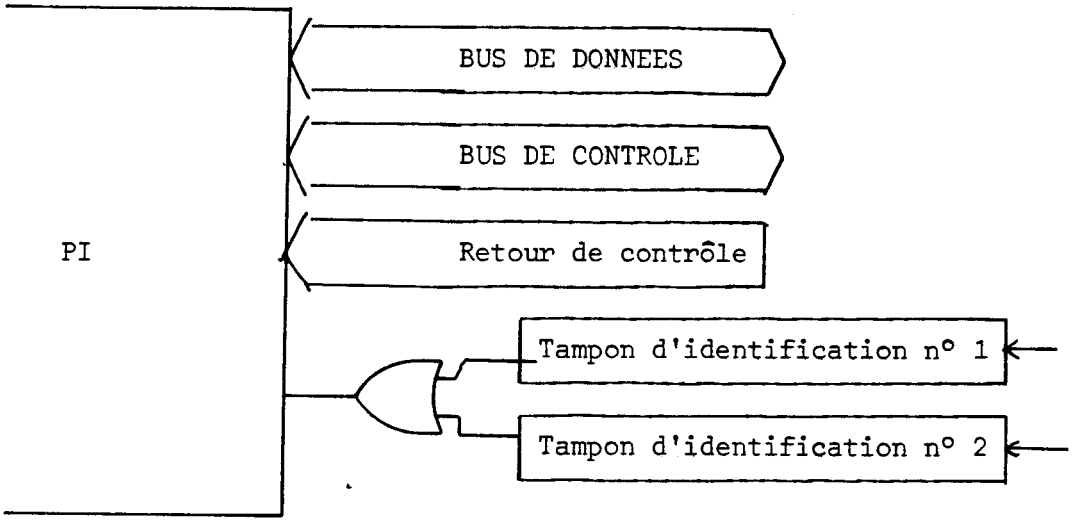
Une autre solution est de rendre les processus de traitement et d'identification indépendants l'un de l'autre et ce au niveau des PEs. Ils peuvent se dérouler en parallèle et le PI ne prend l'identification en compte qu'en dehors de toute commande.

C'est cette dernière solution qui sera retenue.

IV.3.2. Proposition de réalisation du processeur d'interface

Suivant la complexité des fonctions qu'on désire assigner au PI on pourra opter pour un simple automate de communication ou un système à microprocesseur. Celui-ci offrirait une relative autonomie du système graphique considéré comme terminal.

Coté PI on propose la représentation suivante des bus de communication interne :



Les tampons d'identification contiennent les identificateurs des objets reconnus lors d'une demande d'identification. Le premier contient l'identificateur d'un objet transparent éventuel, placé devant l'objet opaque dont l'identificateur se trouve dans le deuxième tampon. Dans la suite, nous faisons l'hypothèse qu'il y a au plus une face transparente "devant" un objet opaque.

IV.4. LES PROCESSEURS-ELEMENT

L'intérêt du découpage élémentaire est de permettre aux processus de haut niveau (logiciel d'application...) de traiter les éléments de façon globale, par exemple au lieu de manipuler les points de contour d'un polygone il suffit de manipuler le nom du polygone. En d'autres termes plus un PE prend à son compte les traitements à appliquer à un élément, plus le programme d'application pourra ignorer la représentation fine d'un élément, et donc plus efficace sera son action.

Nous proposons plusieurs représentations d'un élément en allant de la plus synthétique à la plus analytique :

- "cercle-1".
- cercle (coordonnées du centre, rayon),
- {(X, Y) / (X-Xc)**2 + (Y-Yc)**2 = R**2}

En outre la représentation arborescente proposée par Corinne PARENT permet la manipulation d'objets [PARENT 81]. Il ne reste au programme d'application qu'à formuler des transformations à appliquer sur des noms sans s'occuper de leur réalisation.

Après une présentation de ce que serait un PE idéal par rapport au critère précédent, nous tenterons de décrire une implémentation volontairement limitée de ces idées.

IV.4.1. Approche idéale

Ce paragraphe se compose de deux parties distinctes :

- La première traite de la synthèse, traitement faisant passer de la représentation intrinsèque d'un élément à sa représentation dynamique.

- La seconde s'intéresse à la conversion d'un élément qui à partir de sa représentation dynamique produit sa représentation "image".

IV.4.1.1. Processeur-élément et synthèse

La représentation intrinsèque (RI) d'un élément est exprimée par les valeurs de ses attributs définies de façon indépendante d'un environnement (i.e. les autres éléments, les conditions d'éclairage,...). On peut inclure dans cette représentation la géométrie d'un élément à condition qu'elle se réfère à un repère absolu connu de tous les objets.

Dans la représentation dynamique (RD) les valeurs des attributs sont dépendantes (au moins en partie) de l'environnement.

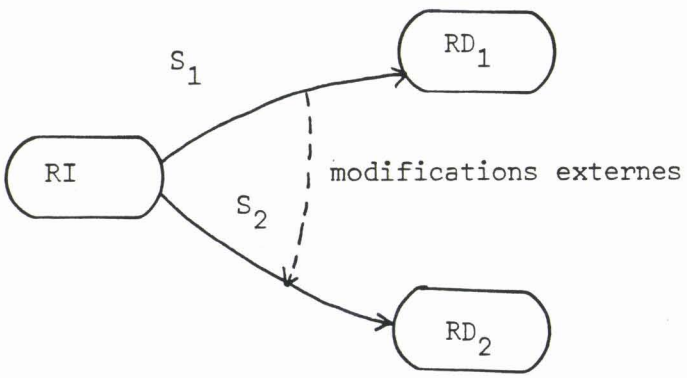
La synthèse modifie donc les attributs d'un élément en les exposant aux conditions d'observation. Les traitements effectués peuvent être :

- Le calcul de répartition de la lumière sur un élément qui produit la couleur apparente (fonction des sources lumineuses).

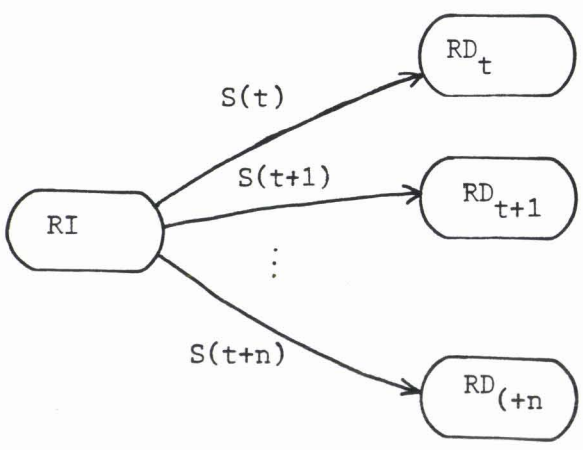
- Le calcul de perspective qui modifie la morphologie d'un élément en fonction du point de vue.

- La morphologie peut aussi être modifiée par le champs de vision utilisé (il peut y avoir découpage voire disparition de l'élément).

Il n'a été question pour l'instant que de synthèse "statique", si la synthèse opérée par un PE n'est pas modifiée de l'extérieur celle-ci donnera toujours la même représentation dynamique de la représentation intrinsèque d'un élément.

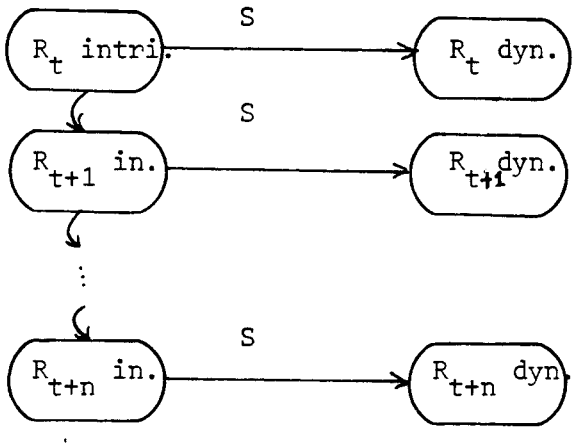


Cependant la synthèse est elle-même conditionnée par les valeurs de certains attributs (point de vue, intensité de l'éclairage,...). Ceux-ci définissent l'état de l'environnement commun à tous les éléments. Si ces paramètres sont des fonctions du temps évaluées par les PE alors l'opération de synthèse devient dynamique :



Par exemple le programme d'application ou l'utilisateur demande aux PE de faire varier l'intensité de l'éclairage d'une quantité Di pendant le temps Dt de façon linéaire. Les PE calculeront alors pour chaque image la variation de l'intensité di.

Par application du même principe certains paramètres de la représentation intrinsèque d'un élément peuvent être fonction du temps et donner ainsi une plus grande autonomie aux PE :



Par exemple un élément doit être déplacé de son emplacement actuel à un endroit donné suivant une trajectoire rectiligne à vitesse constante en un temps Dt.

Remarque : Il semble plus approprié d'affecter la modification en fonction du temps des attributs de synthèse à un processus global à tous les objets.

Ces idées de développement sont largement inspirées des principes développés par MARTINEZ [MARTINEZ 82]. L'architecture d'ARTEMIS propose dans ses processeurs-objet une synthèse statique partielle (modification de la distance de l'observateur à l'écran) ainsi que la possibilité de modifier certains attributs intrinsèques [GRUIA 81].

IV.4.1.2. Processeur-élément et conversion

La description image d'un élément est représentée par l'ensemble des points ou pixels du support d'affichage qui seront éventuellement affectés par l'élément (éventuellement car l'élément peut être caché par un autre élément).

Les algorithmes d'interpolation linéaire sont une des pierres de base des convertisseurs. On peut citer les exemples suivants où ils sont utilisés de façon intensive :

- Les différents systèmes de tracé de vecteurs (dessin) utilisant des écrans à balayage cavalier, ou des tables traçantes.

- Les algorithmes d'éclairement de surface de GOURAUD et de PHONG [FOLEY 82], [GOURAUD 71], [PHONG 75].

Soulignons que l'aspect tridimentionnel des éléments et donc de l'univers doit être conservé jusqu'à la fin de la conversion. C'est ce qui permettra de n'afficher que les parties visibles des éléments. Le traitement correspondant n'est pas du ressort du PE qui ne connaît que son élément.

IV.4.2. Approche proposée

La synthèse opérée par les PE étant complète, c'est-à-dire qu'aucun traitement de synthèse n'est effectué avant les PEs, ceux-ci contiennent la totalité de l'univers ({RI}). La structure de données est donc unique aux représentations dynamiques près. On peut reprocher à cela le faible taux de travail des PEs, et le fait que peu d'entre eux participeront à l'élaboration de l'image.

La proposition d'architecture du chapitre suivant limite le traitement des PEs à l'opération de conversion. La synthèse est alors à la charge des processeurs amonts. L'avantage de cette contrainte est que tous les PEs occupés par un élément participeront à l'élaboration de l'image. En revanche toute transformation d'un élément doit être effectuée par le processeur hôte ou le PI.

CHAPITRE V

PROPOSITION DE REALISATION DES PROCESSEURS-ELEMENT

Les principales limitations apportées aux PEs dans ce chapitre sont les suivantes :

- Les PEs ne possèdent que la représentation dynamique de l'élément qu'ils traitent. Ils n'effectuent donc aucune synthèse et se limitent à la conversion de leur élément.

- Tous les PEs possédant un élément participent à l'élaboration de l'image.

Les trois composantes d'un PE ainsi que le mécanisme de traduction/composition de couleurs sont présentés successivement dans les paragraphes suivants.

V.1. LE GERANT

Le gérant doit connaître l'état d'une communication éventuellement en cours avec le PI ainsi que l'état du PE qu'il administre (état d'occupation du PE, état de visibilité de l'élément traité, un indicateur de possession du jeton de vacance ...).

Ses fonctions sont les suivantes :

- Se sélectionner s'il se reconnaît lors d'un adressage associatif.

- Exécuter, s'il est sélectionné, les commandes émises par le processeur d'interface.

- Lors d'une demande d'identification il doit éventuellement envoyer le nom de son élément vers le tampon d'identification approprié (opaque ou transparent).

V.1.1. Sélection par accès associatif

IV.1.1.1. Accès par nom

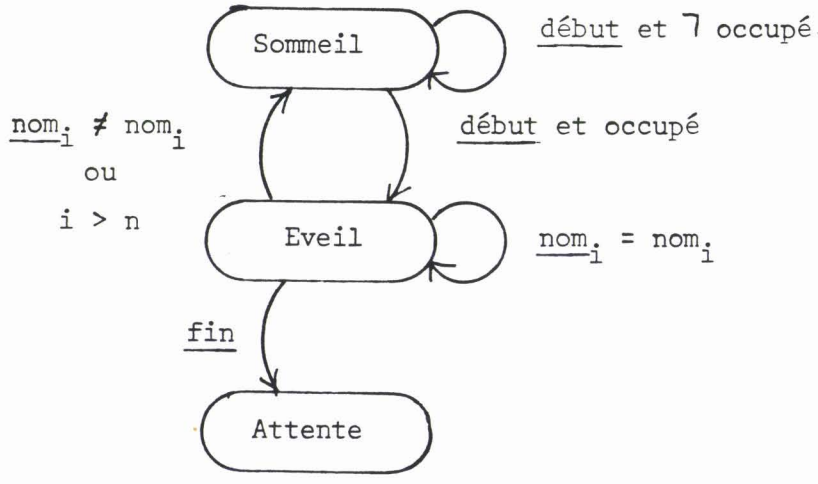
Le PI possède le nom de l'objet à atteindre. Un signal "fin" marque la fin du nom permettant ainsi d'atteindre n'importe quel niveau dans l'arborescence.

Le signal "début" réveille tous les gérants occupés.

Le nom d'un objet est décomposé en n champs.

Un gérant éveillé compare le $i^{ème}$ champs reçu avec le $i^{ème}$ champs du nom de son propre élément, s'ils sont égaux il reste éveillé sinon il se rendort.

Un gérant éveillé recevant "fin" attend l'arrivée d'une commande.



Les variables entre guillemées sont reçues par le gérant depuis le PI, celles qui ne le sont pas sont internes au gérant. "nomi" indique le $i^{ème}$ champs d'un nom.

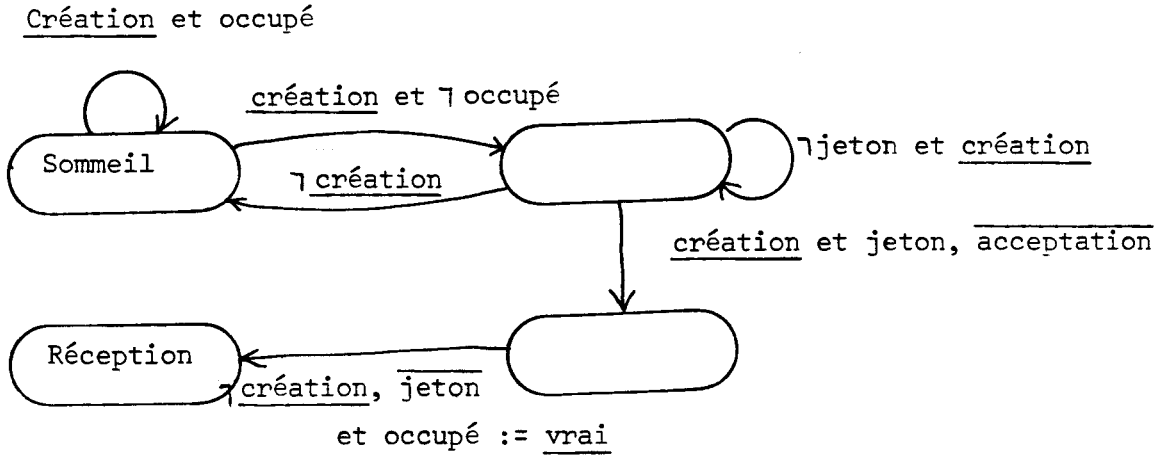
V.1.1.2. Accès par type

Un PE peut être dans trois états par rapport à un accès par type :

- Il est occupé par un élément, il est alors transparent aux demandes de création.

- Il est libre mais ne possède pas de jeton de vacance , il doit alors attendre qu'un jeton arrive pour se sélectionner ou que la demande de création cesse.

- Il est libre et possède le jeton, il se sélectionne en envoyant /acceptation/ au PI puis se positionne occupé et envoie le jeton à son voisin lorsque la demande de création cesse.



V.1.2. Exécution des commandes

Différentes commandes de gestion des représentations dynamiques peuvent être envoyées aux PEs.

CREATION (nom d'élément, type, liste d'attributs).

Cette commande construit un nouvel élément, le PI dispose alors du nom, du type et des attributs de l'élément. Afin d'éviter la duplication d'un élément ou d'un nom, ce qui causerait des conflits lors d'accès ultérieurs, le mécanisme suivant semble nécessaire :

Côté processeur interface :

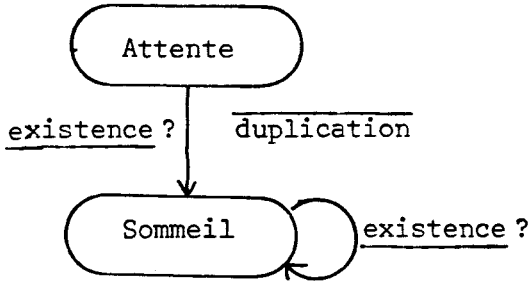
- Essai de sélection par le nom de l'élément.

- Une question "existence ?" est alors posée.

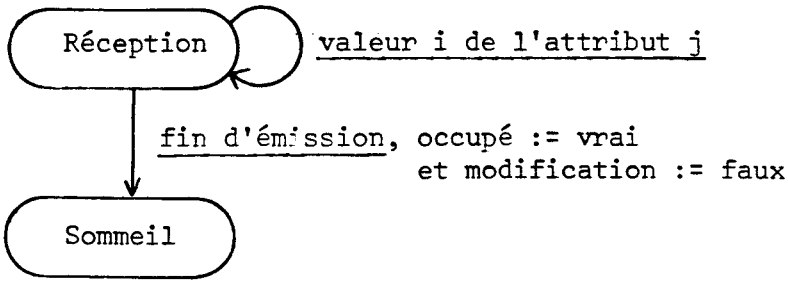
- Si un PE répond à cette question il y a duplication du nom.

- Sinon la création peut avoir lieu, elle continue par une sélection par type et la transmission des données.

Côté gérant il y a alors deux schémas de réaction :



Détection de l'erreur, la connection de celle-ci relevant de l'interface ou du processeur hôte.



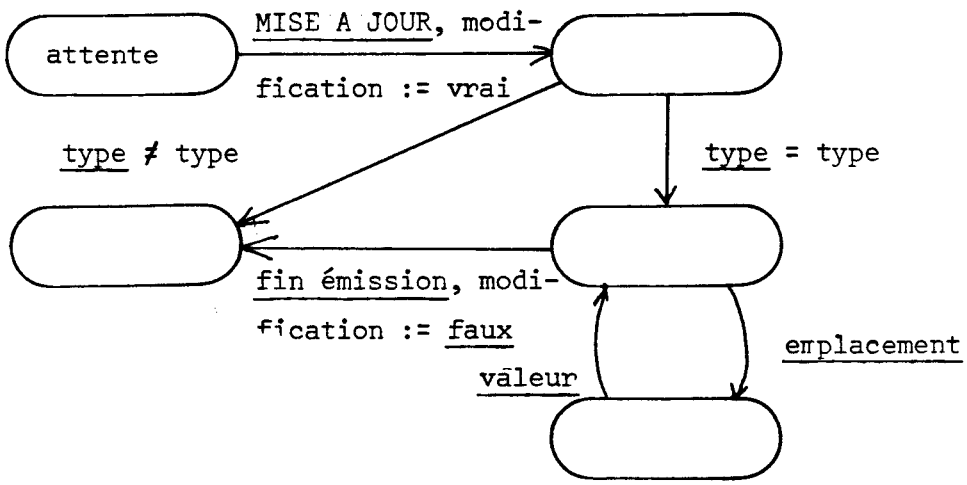
Le gérant garnit sa sous-liste de visualisation avec les valeurs d'attributs qui lui sont envoyées.

Les commandes autres que CREATION ne mettent en jeu que la sélection par nom, elles sont de trois types : mise-à-jour de valeurs d'attributs d'un PE, lecture d'attributs contenus dans un PE et destruction d'un objet.

MISE-A-JOUR (nom d'objet, type, liste des attributs concernés, liste des nouvelles valeurs).

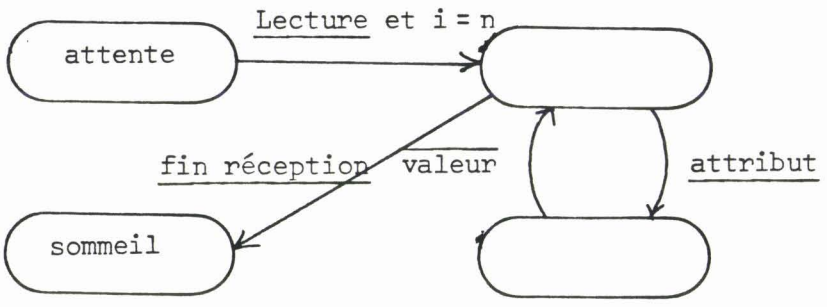
Dans le cas d'un accès à un objet qui n'est pas un élément, le PI doit préciser le type des éléments concernés, la topographie de rangement des attributs n'étant probablement pas la même pour les différents types.

D'autre part les attributs concernés par la modification doivent être spécifiés.



LECTURE (nom d'élément, liste des attributs concernés).

La lecture ne doit porter que sur un seul élément, le type de l'élément sélectionné est considéré comme un attribut et peut à ce titre être lu.



Il faut remarquer qu'on suppose que l'accès par nom fournit dans tous les cas un élément. Dans le cas de la lecture il est commode de s'assurer que le PE existe bien en incluant le nom dans les attributs lus.

Dans le cas d'une mise-à-jour où plusieurs éléments peuvent être sélectionnés, des conflits d'accès à la voie commune de communication interdisent au PEs d'envoyer un quelconque accusé de réception. On peut alors imposer que la mise-à-jour ne porte que sur un élément à la fois.

V.1.3. Prise en compte de l'identification

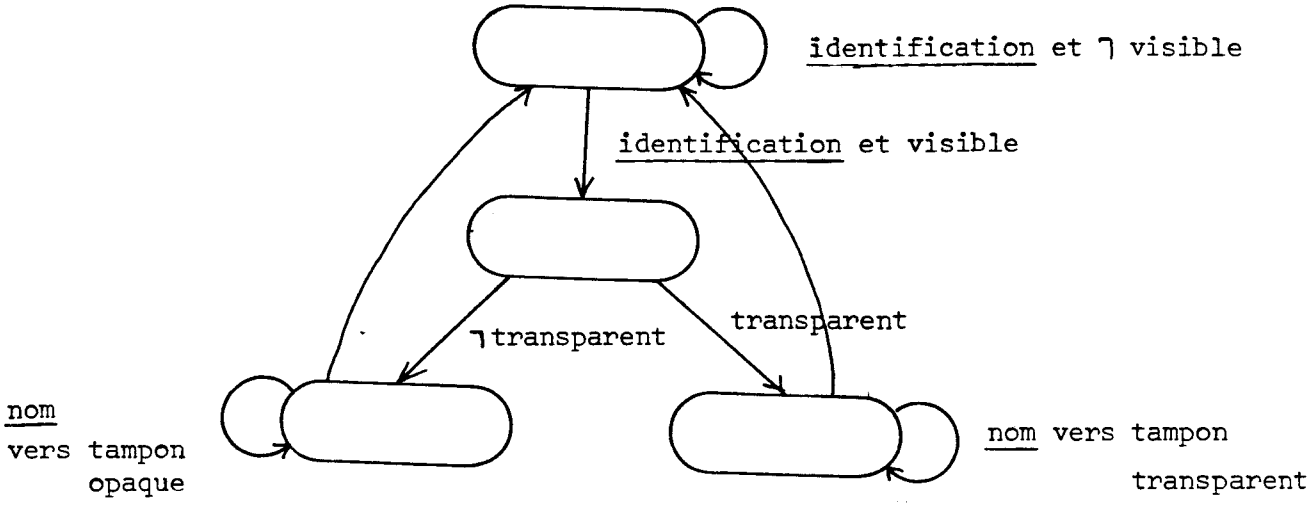
Une demande d'identification émane de l'utilisateur et non du PI. C'est pourquoi le mécanisme de traitement proposé est indépendant de l'exécution des autres commandes et doit pouvoir fonctionner parallèlement.

La sortie du nom d'un élément sur identification dépend de deux indicateurs :

- Un indicateur de "visibilité" de l'élément élaboré par le décideur et transmis au gérant par un registre à décalage.

- Un indicateur de "transparence" qui permet de choisir entre les deux tampons d'identification.

On peut représenter le fonctionnement de cette "commande" par un automate indépendant de ceux présentés précédemment :



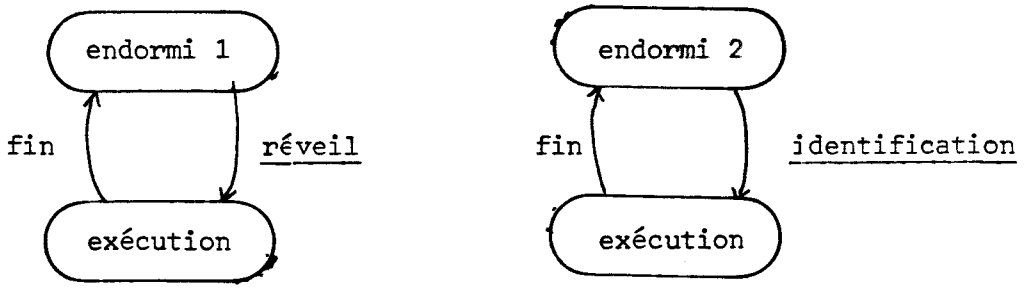
Un tel mécanisme où une même ressource est utilisée par deux processus indépendants (identification et les commandes du PI) se justifie par le fait que le nom d'un élément est constant pendant toute sa durée de vie (i.e. la durée de son affectation à un PE).

V.1.4. Proposition de réalisation du gérant

Ce paragraphe présente une réalisation possible du gérant plus dans un but d'évaluation de ses performances que dans celui d'un câblage effectif.

Vis-à-vis du PI et de l'utilisateur, le gérant possède deux états qui sont "endormi" et "actif". Côté PI le gérant sera "réveillé" et donc mis dans l'état actif par le signal "réveil", il pourra alors lire le bus interne et effectuer le travail demandé, puis il se rendormira de lui-même. Côté utilisateur c'est la demande d'identification qui réveille le gérant qui se rendormira de lui-même en fin de commande.

Deux processus parallèles coexistent donc dans le gérant :



actif 1 correspond aux commandes du PI, elles nécessitent l'emploi du bus interne.

fin 1 est soit envoyé par le PI soit implicite au déroulement de la commande.

Nous proposons de réaliser les états actif et endormi par réactivation et inhibition de l'horloge.

La commande du gérant sera hybride, à la fois câblée et microprogrammée.

L'exécution des commandes émises par le PI (CREATION, DESTRUCTION, MISE-A-JOUR, LECTURE) est dirigée par un microprogramme dont le déclenchement est dicté par le signal réveil. Cette solution permet évidemment une certaine souplesse vis-à-vis du jeu de commandes.

La prise en compte des signaux de plus bas niveau et de l'identification utilise elle des mécanismes câblés séquencés par l'horloge interne du PE.

Le trajet du jeton circulant sera réglé par câblage pour la recherche d'un PE libre et par microprogramme pour sa transmission au PE voisin lorsqu'un nouvel élément est créé.

Certains des mécanismes câblés ainsi que la partie opérative du gérant sont détaillés en annexe.

La suite de ce paragraphe détaille les microprogrammes réalisant les automates des différentes commandes et des accès par nom et par type.

Le gérant est muni d'un registre dont les composantes sont les suivantes :

- Occupé : indique si le PE possède un élément.
- Sélecté : indique si le PE a été sélectionné lors d'un adressage par nom ou par type.
- Modification : indique si la liste d'attributs d'un gérant est en train d'être mise-à-jour (lors de la commande MISE-A-JOUR).
- Egalité : indique, lors d'une identification, si l'élément traité est visible,
- Transparence : indique si l'élément est transparent.

Le microprogramme utilise un microcompteur (mc) pouvant être incrémenté, décrémenté ou chargé avec une nouvelle valeur.

Le contenu du bus de communication est interprété en fonctions des commandes envoyées par le PI et des contenus du registre d'état et du mc :

- Une commande globale (DESTRUCTION, MISE-A-JOUR,...) est en fait une adresse dans le microprogramme.
- Un nom de paramètre d'attribut est une valeur d'index chargée dans le registre d'index (I).
- La valeur d'un paramètre d'attribut est chargée ou lue dans le registre indexé par I (R(I)).

Les dernières notations sont :

- Rt : registre tampon permettant la communication avec le bus de communication interne.
- A, B : les deux entrées d'un comparateur d'égalité.
- C : la sortie de ce comparateur.
- Type : registre contenant le type d'élément traité par le PE.
- Rid : registre à décalage bouclé sur lui-même et qui est chargé lors d'une création avec l'identificateur de l'élément. Ce registre sert au mécanisme de réponse à une identification (voir annexe). Rid est indexé par les premières valeurs de I.

Nous proposons une description fonctionnelle du gérant. Elle comporte la description des mécanismes câblés permettant la mise en condition" du microprogramme (initialisations) dont les actions sont représentées par ":=" et le microprogramme lui-même dont les affectations sont notées "<-".

L'accès par nom suivi de la manipulation peut être décrit par :

```
                "début" & occupé:    I:=0, mc:=0, sélecté:=vrai
                "réveil" & sélecté:  B<-R(I), A<-Rt, I<-I+1,
                                      égalité<-C, mc<-mc+1
adressage                             si égalité alors mc<-mc-1
   par                                           sinon sélecté <- faux
   nom                                                 mc <- attente
                                      fsi
                                     se rendormir
                "fin" & sélecté      mc:=adressage de microinstruction de
                                         chargement
                                     I:=0

chargement
  de la         "réveil" & sélecté:  mc <- Rt, se rendormir
 commande

EXISTENCE       "réveil" & sélecté:  sortir "duplication", mc <- attente
                                     sélecté <- faux, se rendormir

MISE-A-JOUR     "réveil" & sélecté:  modification <- vrai, B <- Rt,
                                      A <- Type
                                     égal <- C, mc <-mc+1
                                     si égal alors mc <- mc+1
                                             sinon sélecté <- faux,
                                                   modification <- faux,
                                                   mc <- attente
                                             fsi
                                     se rendormir
                "réveil" & sélecté:  I <- Rt, mc <- mc+1, se rendormir
                "réveil" & sélecté:  RI <- Rt, mc <- mc-1, se rendormir
                "fin d'émission" & sélecté: modification:=faux,
                                             mc:=attente,
                                             occupé:=vrai
```

Le fonctionnement d'une commande de création est le suivant :

adressage par type

"création" & occupé: modification:=vrai
sélecté:=vrai
sortie /acceptation/
mc:=adresse de chargement

"création" & occupé: sortir /création/

"réveil" & sélecté: mc <- Rt, I <- 0, se rendormir

"réveil" & sélecté: R(I) <- Rt, I <- I+1, se rendormir

"fin d'émission" & sélecté: modification:=faux
sélecté := faux
occupé := vrai
mc := attente

Remarque : Dans le cas d'éléments à nombre variable d'attributs il serait intéressant que le gérant d'un tel élément connaisse ce nombre ainsi que la répartition des attributs dans sa sous-liste de visualisation.

cas LECTURE

- réveil ∧ sélecté : I ← Rt, μC ← μC+1, se rendormir
- réveil ∧ sélecté : Rt ← RI, sortir "validation", μC ← μC-1, se rendormir
- fin réception ∧ sélecté : μC := attente, sélecté := faux

cas DESTRUCTION

- réveil ∧ sélecté : occupé ← faux, sélecté ← faux, μC ← attente, se rendormir

Etant donné le mécanisme de passage du jeton (voir annexe) il est nécessaire que la remise à zéro de "occupé" se fasse sur le front montant de l'horloge interne (H). Une mise à zéro sur front descendant provoquerait une duplication intempestive du jeton.

accès par type

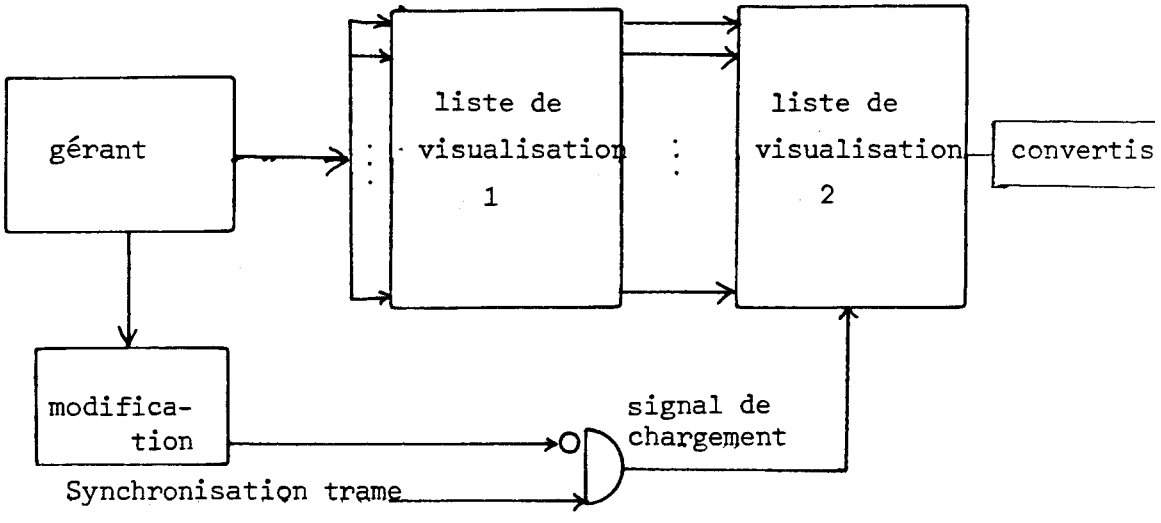
- création ∧ ¬occupé ∧ jeton : modification := vrai, sélecté := vrai,
  sortir "acceptation", armer S1
  mC := adresse de chargement

Création de la liste

- réveil ∧ sélecté : mC ← Rt, I ← 0, se rendormir
- réveil ∧ sélecté : R(I) ← Rt, [Rid ← Rt], I ← I+1, se rendormir
- fin émission ∧ sélecté : modification := faux, sélecté := faux, mC := atten

Remarque : La détection d'un débordement sur le registre d'index permettrait la détection d'éventuelles erreurs de la part du PI ou des communications.

Afin d'éviter le "parasitage" de l'image lors de la création ou de la mise-à-jour d'un élément nous proposons de dupliquer la liste de ses attributs. Le chargement de la seconde liste (représentation dynamique) est effectué à partir de la première à chaque début de trame si l'indicateur de modification n'est pas positionné. Un tel chargement correspond à une initialisation du processus de conversion.



Le coût en bits de la duplication de la liste est étudié en annexe 2.

Ajoutons pour conclure qu'il est possible d'inclure dans ce jeu de micro-programmes de nouvelles commandes telles que INHIBITION, CLIGNOTEMENT, qui ne modifieraient pas beaucoup les mécanismes câblés.

Le gérant est unique pour tous les types d'élément.

V.2. LE CONVERTISSEUR

Chaque PE traite un certain type d'élément. Les calculs effectués par le convertisseur dépendent de celui-ci. Les éléments manipulés sont définis en trois dimensions, le convertisseur doit en déduire une information morphologique et une information d'aspect. Dans la sous-liste de visualisation les attributs d'un élément sont les suivants :

- Géométrie : coordonnées dans l'espace à trois dimensions des points caractéristiques de l'élément définissant sa position.

- Morphologique : regroupe les informations nécessaires au calcul de la forme (les informations dépendent donc de l'algorithme utilisé pour le calcul). Ces informations sont définies dans le repère propre à l'élément qui est équivalent au repère de l'espace de calcul à une translation près.

- Aspect : ces informations peuvent refléter une variation d'intensité, de transparence et comme pour la morphologie s'appliquent à des algorithmes de calcul spécifiques au type manipulé.

Précisons que la morphologie intègre déjà l'effet de perspective (en restant dans l'espace à trois dimensions) et que l'aspect tient compte de divers paramètres tels que l'éclairage et l'orientation de l'élément.

L'espace de calcul des convertisseurs est tridimensionnel (OXYZ) avec (X, Y, Z) appartenant à N^3. Un point (x, y, z) exprimé dans ce repère est dit absolu. Le plan de l'écran est le plan (OXY), l'image étant obtenue par projection orthogonale de (OXYZ) sur (OXY). Y sera le numéro d'une ligne de l'écran, X celui d'une colonne.

Le balayage de l'écran cathodique peut être représenté de la façon suivante :

```
faire toujours
     pour Y=1 à N faire
          pour X=1 à M faire... fait
          fait
     fin toujours
```

où N est le nombre de ligne s et M le nombre de colonnes.

La conversion de tout élément se décompose en deux parties :

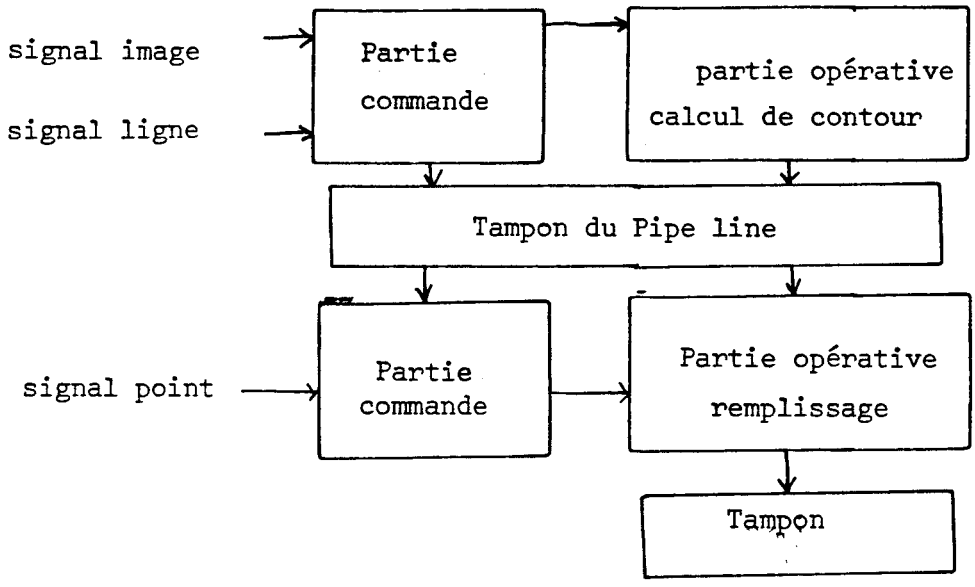
- calcul des bords de l'élément sur une ligne,
- calcul des points appartenant à l'élément entre les deux bords,

qui correspondent en fait au opérations classiques de calcul de contour et de remplissage d'élément. Il faut remarquer que ces calculs ne sont effectués que si l'élément existe sur la ligne considérée. En incluant ces opérations dans la séquence de balayage on obtient :

```
faire toujours
     initialisations;
     pour Y=1 à N faire
          si nécessaire alors calcul de contour
          pour X=1 à M faire
               si nécessaire alors remplissage
               fait
          fait
     fin toujours
```

Le calcul de contour et le remplissage portent à la fois sur la morphologie et sur l'aspect de l'élément ; les algorithmes de calcul de variation ne sont cependant pas toujours les mêmes.

Nous proposons un schéma général de réalisation où le calcul de contour et le remplissage sont agencés en pipe-line et où les parties commandes correspondent aux deux expressions précédentes "si nécessaire alors". Le pipe-line perm un calcul en temps réel.

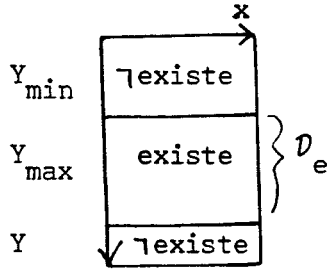


Les parties commandes utilisent des informations :

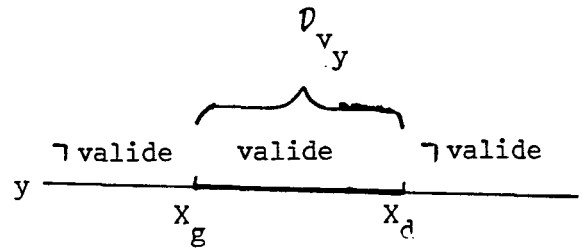
- de synchronisation trame, ligne, point,
- géométriques,
- morphologiques (pas toutes),

et elles produisent des informations "existence" pour le contour et "validité" pour le remplissage. Nous dirons que tout élément possède un domaine d'existence en ligne (De) et des domaines de validité en colonne (Dv_y).

Un élément possède deux extrémités en Y, Ymin et Ymax :



pour une ligne y donnée il possède un bord gauche Xbg et un bord droit Xbd :



Remarquons que : $Dv_y = 0 \iff y \notin De$

La propriété implicite de continuité de De et Dv_y implique qu'aucun élément n'ait des bords "remontant" [MERIAUX 79]. Nous choisirons une condition plus forte en disant que les éléments doivent être convexes, ceci afin d'éviter qu'une rotation d'un élément rende caduque la propriété précédente.

Dans la suite nous nous efforcerons de donner les parties opératives de chaque type de convertisseur, les parties commandes faisant l'objet d'une description fonctionnelle ne préjugeant pas d'une implantation ultérieure. L'outil de description fonctionnelle est une notation algorithmique, chaque algorithme de commande est déclenché par interruption (qui sont les signaux de synchronisation image, ligne et point). Le couple (Xc, Yc) représente la position actuelle du faisceaux d'électrons sur l'écran.

Les trois paragraphes suivants présentent successivement les convertisseurs de :

- segment de droite,
- facette polygonale,
- sphère.

Les paramètres de l'attributs d'aspect seront désignés par le terme générique Ci, invariant de 1 à n s'il y a n paramètres. Ils peuvent être vaviables ou constants, ceci étant précisé par la suite. Les valeurs A, B et N ont la même signification que A', B' et N' dans l'algorithme A2 exposé dans le chapitre trois.

V.2.1. Le convertisseur de segment de droite

Le calcul de la morphologie utilise évidemment une variation linéaire des valeurs. Nous choisissons comme dans la méthode de GOURAUD de faire varier les paramètres d'aspect de façon linéaire eux aussi [GOURAUD].

L'algorithme A2 présenté dans le chapitre trois utilise un temps constant de calcul (1 itération) à chaque pas d'interpolation, et quelque soit la pente du segment. C'est donc exclusivement cet algorithme qui sera utilisé lors d'interpolations linéaires. La "construction" d'un segment de droite se décompose en calcul de contour et remplissage. Ceci paraît inutile mais permet d'utiliser le schéma général de description des convertisseurs.

V.2.1.1. Les attributs

La représentation dynamique d'un segment de droite est la suivante :

- Géométrie : (X1, Y1, Z1) coordonnées absolues du premier point du segment rencontré par le balayage de l'écran.

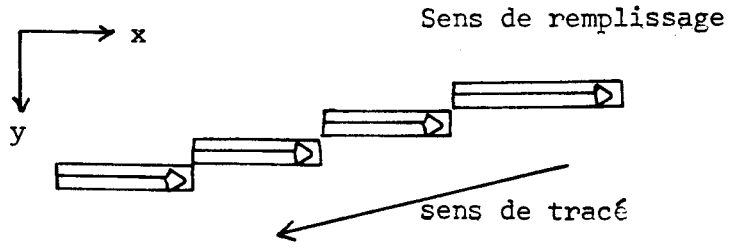
- Morphologie : (DX, DY, DZ) variations en X en Y et en Z du segment de droite. Remarquons que DY doit être positif, ceci étant obtenu par un traitement en amont des PE.

  $(A_{X/Y}, -B_{X/Y}, N_{X/Y})$,
  $(A_{Z/Y}, -B_{Z/Y}, N_{Z/Y})$ qui sont les paramètres nécessaires à l'algorithme A2.

- Aspect : $(Ci, DCi, A_{Ci/Y}, -B_{Ci/Y}, N_{Ci/Y})$ pour chaque Ci variable
  Ci pour les paramètres d'aspect constants.

Remarques : - DY >= 1 d'après II.1.3.4.

- Si $N_{X/Y}$ est strictement négatif alors le mouvement de calcul de remplissage est fait en sens inverse du sens de tracé de la droite :



- Les valeurs A et B sont évaluées avec |DX| puisqu'elles ne servent qu'à choisir le mouvement adéquat, on a donc :

A >= 0 et -B <= 1

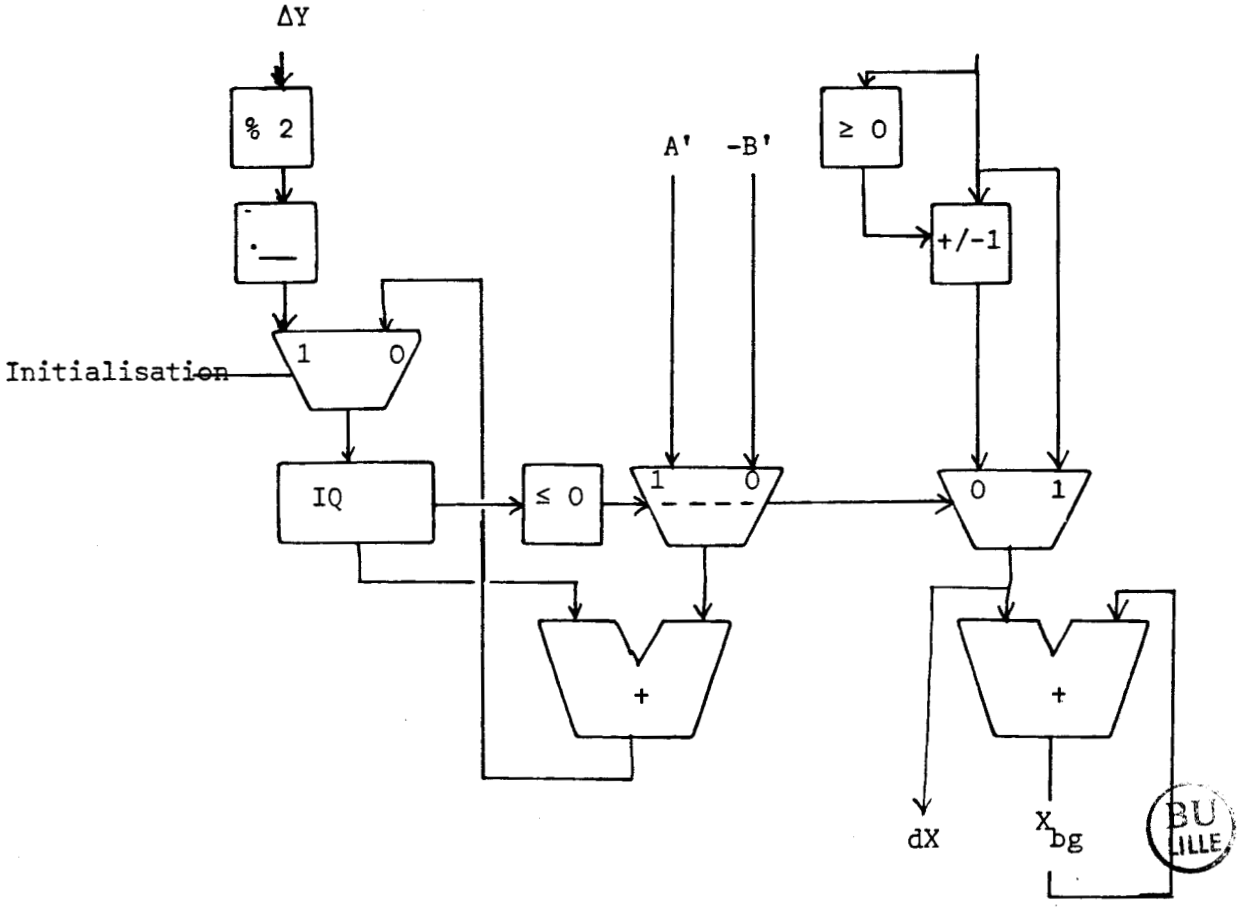
Les conséquences de ces remarques sont les suivantes :

- De ne peut être vide (i.e. Card(De)=1 pour une horizontale)

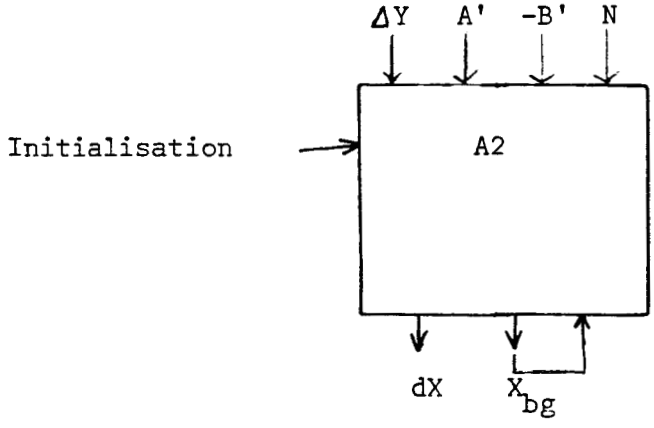
- L'évaluation de Dv_y dépend du signe de $N_{X/Y}$, ce qui influe aussi sur la manière dont est effectué le remplissage.

V.2.1.2. L'interpolateur

Pour des raisons de cadrage, la variable Xbg (bord gauche) qui représente la borne inférieure de Dv_y est initialisée une seule fois par image à X1 - $N_{X/Y}$, ceci étant vrai pour les Ci variables. On peut alors représenter la partie opérative d'un interpolateur de type A2 de la manière suivante :



qu'on remplace dans la suite par le schéma :



qui comporte : 3 multiplexeurs.

1 registre (IQ)
2 additionneurs
1 incrémenteur/décrémenteur
1 complémenteur.

V.2.1.3. Calcul de contour

Nous décrivons ici les opérations que doivent effectuer la partie commande et la partie opérative ainsi que la mise en forme des résultats afin qu'ils soient utilisables par le remplissage.

partie commande

Son rôle est double, elle doit permettre les calculs de contour à chaque ligne et indiquer au remplissage s'il peut être effectué ; c'est la valeur booléenne "existence" qui assure cette fonction. "existence" doit être réévaluée à chaque ligne de balayage :

```
image : Y <- Y1 ; existence <- faux ; Yc <- 0 ;
        Xlim = X1+DX ;
        si DX < 0 alors négatif <- vrai fsi ;
        évaluer A, B, N par rapport à DX pour Z et les Ci

ligne : si non existence alors
              si Yc=Y alors existence <- vrai ;
                            Y = Y1+DY
                            fsi
        sinon
              si Yc=Y alors existence <- faux fsi
        fsi
```

Xlim, Y et Yc sont des registres internes à la partie commande, et Yc est incrémenté à chaque nouvelle ligne. Xlim sera utilisé par le remplissage. L'évaluation de $A_{Z/Y}$, $-B_{Z/Y}$, $N_{Z/Y}$, $A_{Ci/Y}$, $-B_{Ci/Y}$, $N_{Ci/Y}$ est effectuée une fois à chaque début de trame et utilise des diviseurs entiers.

partie opérative

Son rôle est de fournir au remplissage les caractéristiques de Dv_y pour la prochaine ligne. Ces caractéristiques sont le premier point de la ligne appartenant à Dv_y (Xbg) et la longueur de Dv_y (dx).

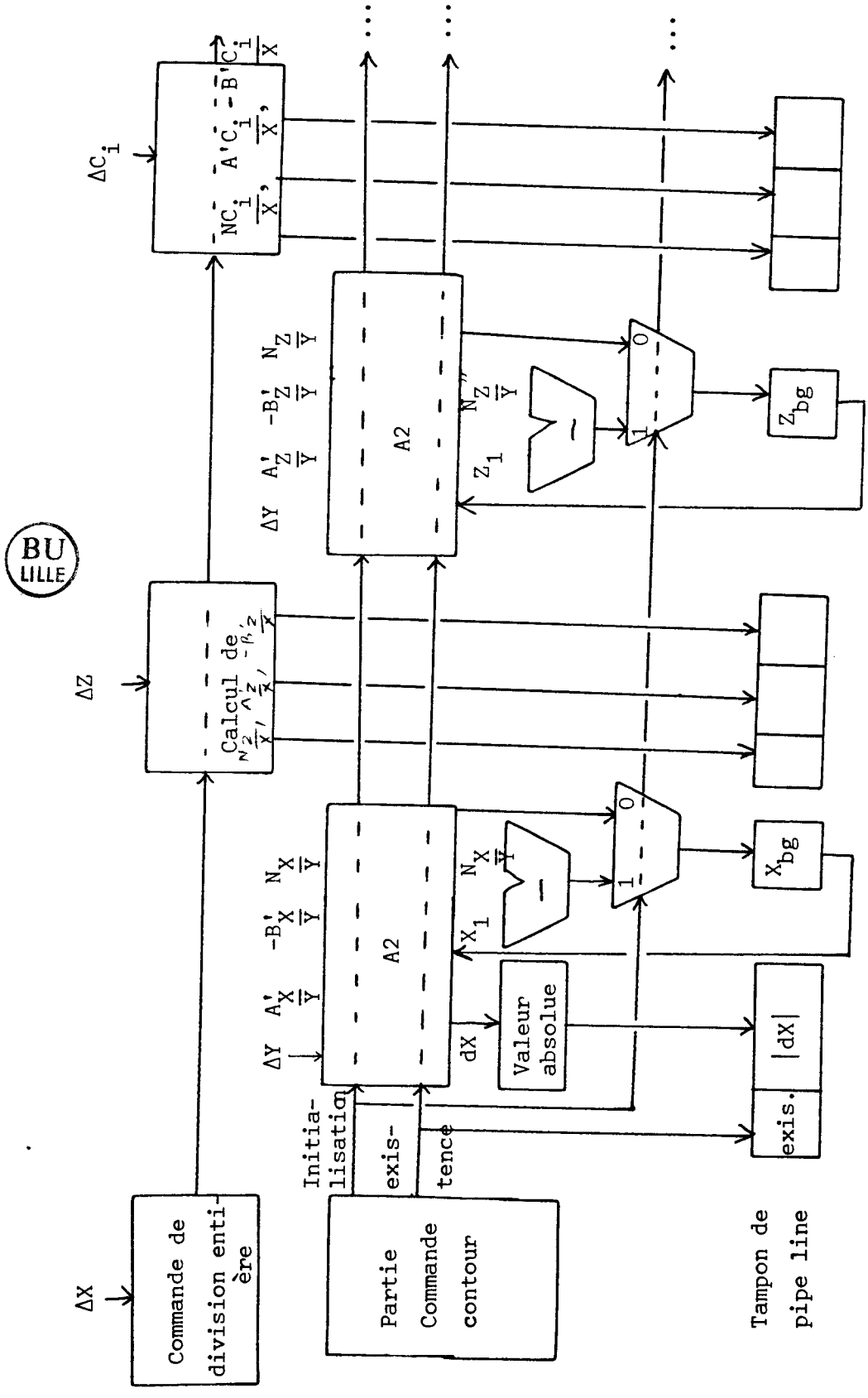
Remarques : - Le calcul des Ci variables n'est pas représenté, il est totalement semblable au calcul en Z.
- Les valeurs des Ci constants sont directement communiquées à la sortie du décideur.

V.2.1.4. Le remplissage

Cette partie travaille avec les informations disponibles dans le tampon du pipe-line. La partie commande utilise la longueur du mouvement à effectuer ($|dx|$), le point de départ de ce mouvement (Xbg) ainsi que l'indicateur d'existence fourni par la partie commande du calcul de contour. La partie opérative effectue l'interpolation des valeurs à fournir (Z et Ci).

```
ligne : si existence alors X <- Xbg; valide <- faux;
                           si |dx| = 0 alors nul <- vrai
                                       sinon nul <- faux
                           fsi
        fsi
```

Schéma de la partie opérative.

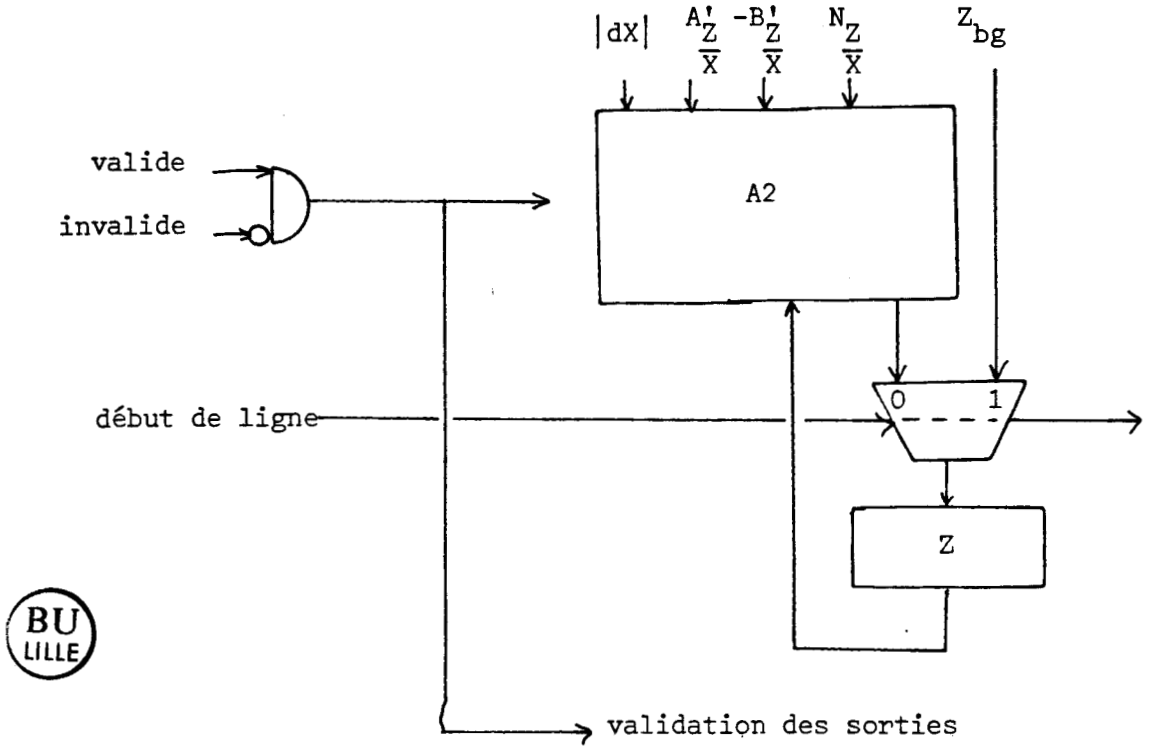


```
point : si existence alors
            si non valide alors
                  si non invalide alors
                        si Xc=X alors valide <- vrai;
                                    X = X+|dx|;
                                    si négatif alors
                                          si Xc xlim alors
                                             invalide <- vrai
                                          fsi
                                    fsi
                        fsi
                  fsi
            sinon
                  si nul ou Xc=X alors valide <- faux
                  sinon
                        si non négatif alors
                              si Xc>Xlim alors invalide <- vrai fsi
                        sinon
                              si Xc=Xlim alors invalide <- faux fsi
                        fsi
                  fsi
            fsi
      fsi
```

X et Xc sont des registres internes à cette partie, le booléen "invalide" permet de supprimer les points qui n'appartiennent pas au segment de droite et dûs à la non convergence de l'algorithme A2.

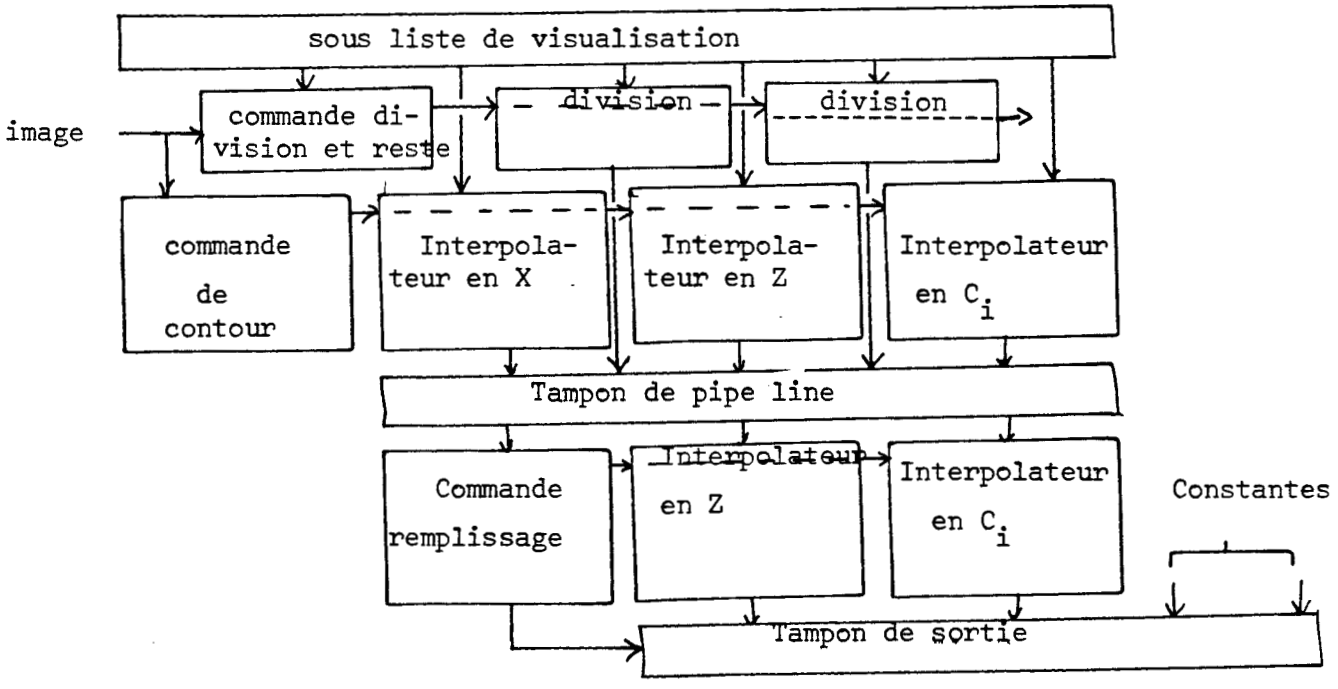
partie opérative

Le registre Z contient la valeur de la profondeur de l'élément au point considéré. Le mécanisme de calcul des Ci n'est pas figuré ici, il est semblable à celui présenté ci-dessous :



V.2.1.5. Conclusion

Le convertisseur de segment de droite peut être schématisé comme suit :



Il faut ajouter que les calculs de $A_{Z/Y}$, - $B_{Z/Y}$, $N_{Z/Y}$ et $A_{Ci/Y}$, - $B_{Ci/Y}$ et $N_{Ci/Y}$ peuvent effectués à l'extérieur du PE, ceci induisant une taille plus importante de la sous-liste de visualisation et des communications PI <-> PE plus longues. Cependant les diviseurs ne seraient plus nécessaires.

Le nombre d'attributs devant être envoyés par le PI au PE est le suivant (non compris le nom de l'élément) :

A = 12 + 5*p + k avec p+1 diviseurs

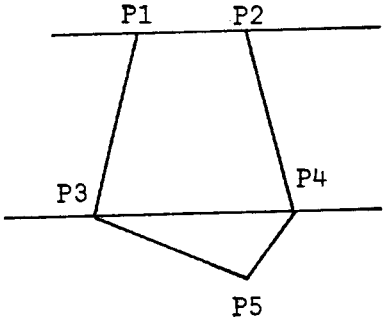
A = 12 + 5*p + k + 3*(p+1) sans diviseur

p est le nombre d'attributs d'aspect variables.
k est le nombre d'attributs d'aspect constants.

V.2.2. Le convertisseur de facette

Les éléments polygones sont supposés convexes, d'autre part comme pour les segments de droite, les attributs d'aspect variables sont donnés comme variant linéairement. Des algorithmes de type A2 sont donc utilisés.

Un polygone est défini par l'ensemble de ses sommets auxquels sont attachées les informations morphologiques et d'aspect. L'ensemble des sommets est divisé en deux sous-ensembles S1 et S2, représentant respectivement le bord gauche et le bord droit du polygone [MERIAUX 79]. Notons que l'intersection de S1 et de S2 peut être non vide :



S1 = {P1, P3, P5}
S2 = {P2, P4, P5}
S1 ∩ S2 = {P5}

de plus Card(S1 ∩ S2) <= 2 puisque les polygones sont convexes.

Pour simplifier l'implantation tout sommet appartenant à S1 ∩ S2 sera dupliqué, la sous-liste de visualisation est alors séparable physiquement en une sous-liste bord gauche et une sous-liste brod droit.

V.2.2.1. Les attributs

Géométrie : (Xi, Yi, Zi)bg i=1, n
(Xj, Yj, Zj)bd j=1, m
ces valeurs sont utiles pour évaluer les domaines de calcul De et Dv et assurer la convergence des calculs (cf. III).

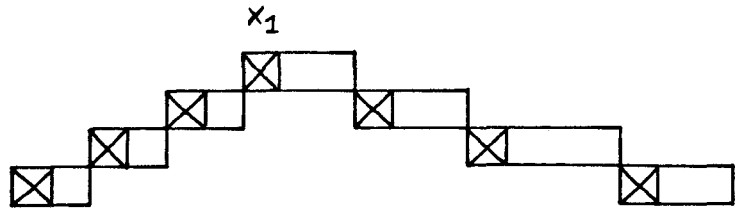
Morphologie : DY = Yn - Y1 hauteur du polygone
(Ai, $i+1_{X/Y}$, - Bi, $i+1_{X/Y}$, Ni, $i+1_{X/Y}$) bg
(Aj, $j+1_{X/Y}$, - Bj, $j+1_{X/Y}$, Nj, $j+1_{X/Y}$) bd et
(Ai, $i+1_{Z/Y}$, - Bi, $i+1_{Z/Y}$, Ni, $j+1_{Z/Y}$) bg
(Aj, $j+1_{Z/Y}$, - B,, $j+1_{Z/Y}$, Nj, $j+1_{Z/Y}$) bd
ceci pour i=1, n-1 et j=1, m-1.

Aspect : Les attributs variables
Ck,
(Ai, $i+1_{Ck/Y}$, - Bi, $i+1_{Ck/Y}$, Ni, $i+1_{Ck/Y}$) bg
(Aj, $j+1_{Ck/Y}$, - Bj, $j+1_{Ck/Y}$, Nj, $j+1_{Ck/Y}$) bd
pour i=1, n-1 et j=1, m-1 et k=1, p

Les attributs constants
Ck.

V.2.2.2. Préliminaires

Le calcul de contour de polygone utilise une interpolation pour chacun des deux bords, on remarque alors que pour un tracé donné il n'est pas nécessaire, pour le remplissage, de ne connaître qu'un point par bord, dans l'exemple les points marqués d'une croix définissent les bords :



C'est pourquoi les paramètres d'interpolation A, B et N pour Z et Ck sont exprimés en fonction de DY.

L'utilisation de l'algorithme A2 lors du remplissage implique que les paramètres d'interpolation A, B et N des attributs Z et Ci sont calculés à chaque fois qu'on "passe" un sommet du bord gauche ou du bord droit. Cependant, si le polygone à n+3 dimensions (en incluant les Ci) est plan ces paramètres sont constants sur l'élément et peuvent être calculés à l'extérieur du PE. Si le polygone n'est pas plan on peut le découper en triangles, créant n-2 éléments, si le polygone avait n sommets. Une autre technique consiste à multiplier ses sommets afin d'obtenir une suite de trapèzes [LERAY 78] :

Remarquons que si un polygone dont tous les sommets ont des ordonnés distinctes est décomposable en trapèzes plans dont les bases sont des ordonnées constantes alors ce polygone est plan. Une facette non plane n'est donc pas forcément décomposable en trapèzes plans.

Ce découpage n'est donc pas utilisable dans le cas général, pour pouvoir évaluer une fois pour toutes les paramètres A, B et N pour un trapèze.

En conclusion, pour ne pas se limiter à des facettes triangulaires il faut soit supposer que les polygones à n+3 dimensions sont plans (ce qui permet d'évaluer A, B et N à l'extérieur du PE) soit doter le convertisseur de diviseurs entiers.

Nous proposons ici une solution générale dans le cas des polygones non plans.

Le dernier problème est celui de la gestion des sous-listes de visualisation bord gauche et bord droit, ou comment accéder aux nouveaux paramètres lorsqu'on quitte un segment pour en traiter un nouveau.

La partie commande peut être libérée des contraintes d'adressage si les sous-listes de visualisation sont contenues dans deux mémoires circulaires à décalage.

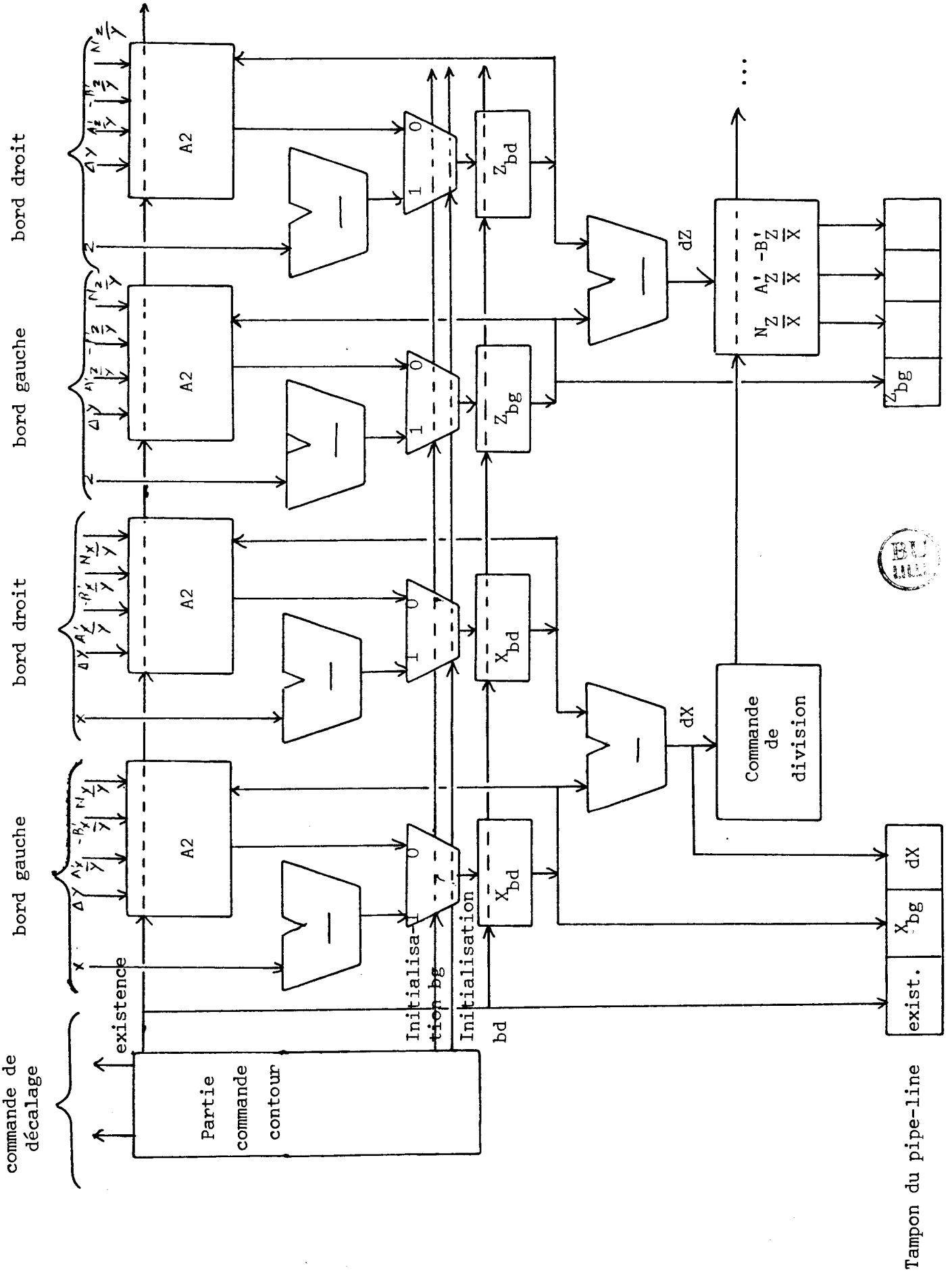
V.2.2.3. Calcul de contour

partie commande

```
image : charger les valeurs initiales dans les registres;
        existence <- faux;
        Y = Y1

ligne : si non existence alors
              si Yc=Y alors existence <- vrai;
                            Y = Y+DY;
                            Ybg <- Y2bg; Ybd <- Y2bd;
                            décaler les deux sous-listes
              fsi
        sinon
              si Yc=Y alors existence <- faux
              sinon
                    si Yc=Ybg alors
                                  réinitialiser la partie bord droit;
                                  Ybd <- Yi;
                                  décaler la mémoire circulante bd
                    fsi
              fsi
        fsi
```

Partie opérative



V.2.2.4. Le remplissage

Le domaine de validité du remplissage pour une ligne y, Dv_y est fonction de trois informations : l'existence ($y \cap Dv_y$), la valeur de dx et le premier point du domaine dans le sens du balayage qui est Xbg. Ces informations sont fournies par le calcul de contour. Comme pour le tracé de segment de droite le cas dx=0 doit être pris en compte. Notons cependant que cette fois dx indique la longueur vraie de Dv_y, il n'est alors plus nécessaire d'utiliser la valeur Xlim et l'indicateur d'invalidité, le schéma de commande se simplifie et devient :

```
ligne : si existence alors X <- Xbg;
                           valide <- faux;
                           si |dx|=0 alors |dx|=1 fsi
        fsi

point : si existence alors
        si non valide alors
               si Xc=X alors valide <- vrai;
                             X <- X+|dx|
               fsi
        sinon
               si Xc=X alors valide <- faux fsi
        fsi
      fsi
```

partie opérative

Celle-ci est la même que pour les segments de droite, de remplissage étant effectué par interpolation linéaire sur les attributs variables.

V.2.2.5. Conclusion

Lors d'une implantation réelle des choix doivent être faits à propos de plusieurs paramètres :

- Le nombre maximum de sommets par bord.
- Quels sont les attributs d'aspect devant être variables.

- Choix de planéité des polygones, doivent-ils être plans, décomposables en trapèzes plans ou non plans. Nous avons vu que ce choix a d'importantes conséquences vis-à-vis de l'architecture (diviseurs entiers) mais aussi vis-à-vis du volume de la communication PI <-> PE et de la capacité de la sous-liste de visualisation.

Le nombre de paramètres A à communiquer pour un polygone non plan à n sommets est le suivant (non compris le nom de l'élément) :

$$A = 3*(n+2) + 1 + 2*3*n + 3*n*p + k$$

géométrie morphologie aspect

$$A = 3*n*(3+p) + k + 7$$

où : p est le nombre d'attributs d'aspect variables.
k est le nombre d'attributs d'aspect constants.

V.2.3. Le convertisseur de sphère

Les calculs morphologiques et d'aspect ne sont bien évidemment plus linéaires. Si pour le calcul de contour nous utilisons l'algorithme circulaire de BRESENHAM, le remplissage utilise pour des raisons d'efficacité les algorithmes d'interpolation polynomiale du second degré présentés en III.3. D'autre part nous prenons comme hypothèse que tous ces calculs se font en deux dimensions, soit dans un plan Z parallèle au plan (XOY) pour le calcul de contour, soit dans un plan Y parallèle au plan (XOZ) pour le remplissage. Cette simplification est logique si le point de vue est supposé être à l'infini.

La principale caractéristique d'une sphère étant sa propriété de symétrie, l'évaluation des deux bords (gauche et droit) se réduit au calcul d'un seul bord. Nous choisissons de calculer le bord droit. Une sphère est un volume fermé, elle est cependant traitée ici comme une surface gauche non fermée : seule la demi-sphère visible est calculée.

V.2.3.1. Les attributs

Géométrie : (X1, Y1, Z1) coordonnées absolues du centre de la sphère.

Morphologie : R rayon de la sphère.

Aspect : $(Xs, Ys)_{Ci}$ coordonnées relatives à (X1, Y1) du point de la sphère où le paramètre variable Ci possède sa plus grande valeur,

(Ci_{max}, Ci_{min}) valeurs maximum (pour $(Xs, Ys)_{Ci}$) et minimum des paramètres d'aspect (Ci_{min} n'est pas vraiment la valeur minimum, en effet Ci peut prendre des valeurs inférieures à Ci_{min}). Les notions de maximum et de minimum sont conventionnelles, on pourrait les inverser.

$(hi, Mi^2, 2*hi*Mi)$ nécessaires pour l'algorithme de calcul d'aspect.

$$hi = Ci_{max} - Ci_{min}$$
$$Mi = R+|Ys|$$

Ck valeurs des attributs d'aspect constants.

V.2.3.2. Préliminaires

Le domaine d'existence morphologique De_m est ici évident. Cependant, il faut ajouter un nouveau domaine d'existence, celui de l'aspect De_a. En effet, dans le paragraphe III.4.3., il apparaît que le domaine de calcul d'aspect peut être plus grand que De_m avec : $De_m \subsetneq De_a$

$$De_m = [Y1 - R, Y1 + R]$$

De_a est fonction de Y1, R et Ys :

$$De_a = [Y1 - R - 2*|Ys|, Y1 + R]$$

Il faudra alors que le De_a d'une sphère vérifie la condition suivante avant d'être envoyé vers un processeur élément :

$$De_a \subset [0, Y1 + R]$$

Ou bien il faudra précalculer les paramètres internes à l'algorithme de calcul de bord et les envoyer au PE ce qui alourdit notablement les procédures de communications internes.

Le même problème de débordement de l'espace de calcul se pose lors du remplissage, une condition nécessaire et suffisante à la cohérence du remplissage est :

X1 - R + 2*Xs >= 0 quand Xs < 0
ou Xs >= 0

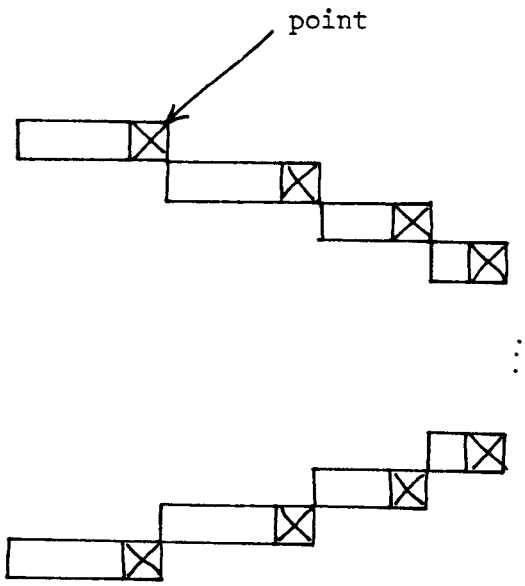
Remarquons que si Ys est nul et Xs est positif, alors $De_a = De_m$ et le remplissage est correct.

V.2.3.3. Calcul de contour

V.2.3.3.1. La morphologie

partie commande

De_m et De_a ne correspondant pas, la commande doit assurer le calcul morphologique et le calcul d'aspect de façon indépendante. En revanche elle ne doit valider les résultats de calcul de bord que si la ligne traitée appartient à De_m. D'autre part le calcul morphologique de bord ne doit fournir qu'un point par ligne, ce point doit être le dernier de Dv_y.



Le point de bord est le dernier à être calculé quand Y <= Y1, et est le premier point calculé quand Y > Y1 où Y est le numéro de la ligne calculée. C'est le booléen "dernier" qui rend compte de cette caractéristique.

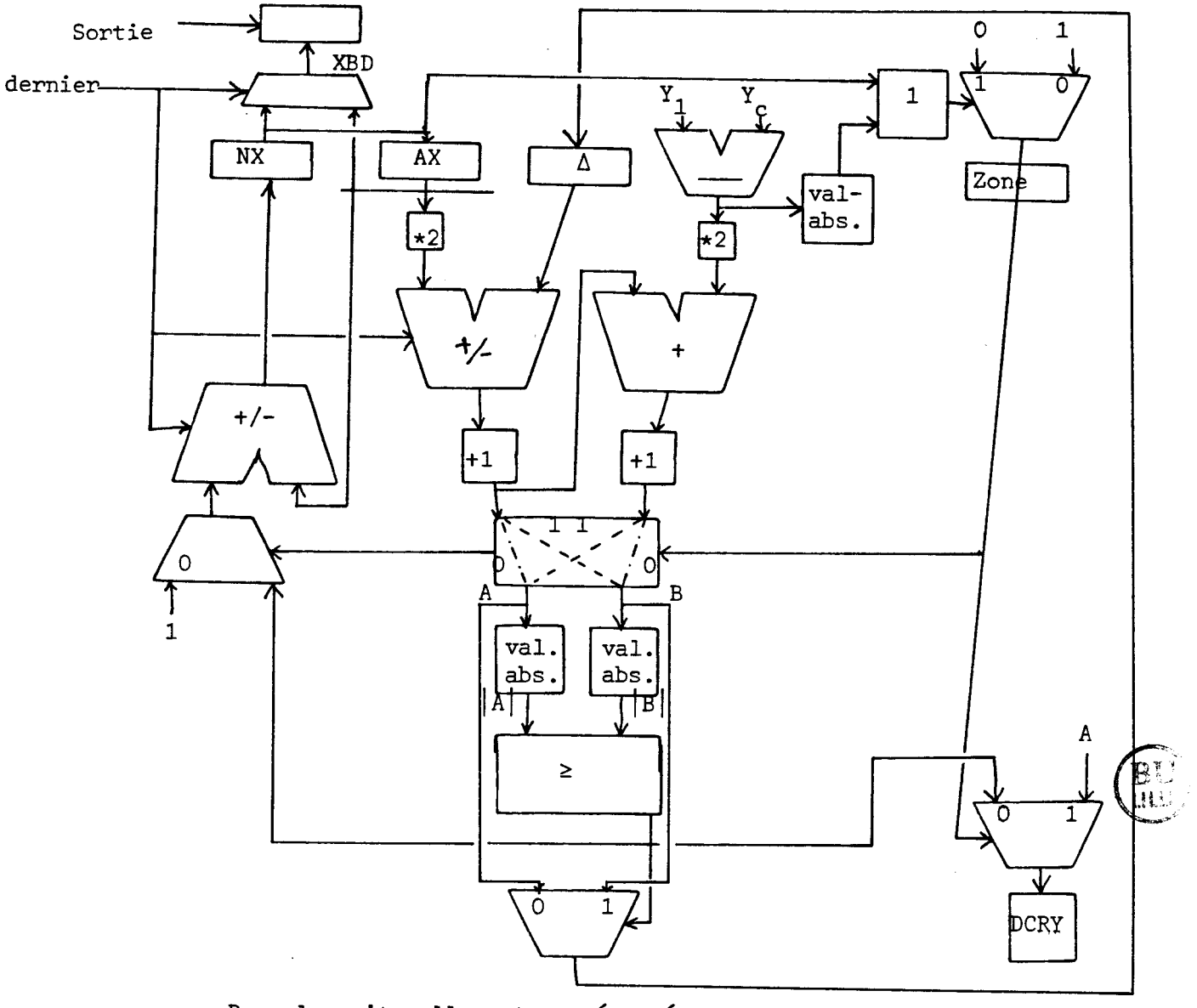
```
image : charger les valeurs initiales;
        existence <- faux;
        dernier <- vrai;
        Ym = Y1-R

ligne : si non existence alors
              si Yc = Ym alors existence <- vrai;
                               Ym = Y1 + R;
                               envoyer premier
              fsi
        sinon si Yc = Ym alors existence <- faux
        sinon si dernier alors
                 tant que non DCRY faire
                          calculer le point suivant NX du cercle de BRESENHAM
                          fait;
                 sortir ancien X(AX);
                 si Yc = Y1 alors dernier <- faux fsi
              sinon
                 sortir NX;
                 tant que non DCRY faire
                      calculer le point suivant NX
                      fait
              fsi
        fsi
fsi
```

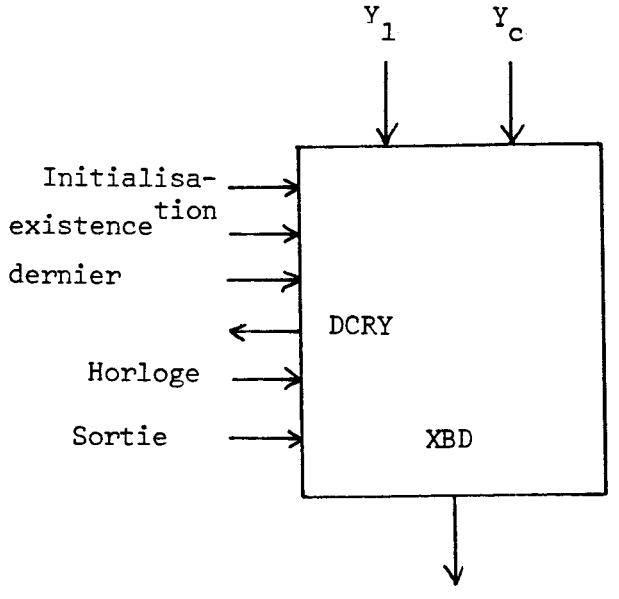
"premier" représente une impulsion qui commande la mémorisation d'une valeur qui doit rester constante ensuite, la valeur mémorisée concerne le calcul de l'aspect.

partie opérative

Elle pourrait être la suivante :



Dans la suite elle est représentée par :



Elle contient : 2 additionneurs/soustracteurs
2 additionneurs
2 multiplexeurs 1 bit
4 multiplexeurs
3 registres
2 comparateurs
3 valeurs absolues
2 bascules

initialisation positionne à zéro les registres de la boîte.

Une partie de la commande est laissée à la partie opérative.

Celle-ci peut être simplifiée si on reporte tout ce qui est commande dans la partie commande (Zône).

La concaténation des booléens DERNIER et ZONE est équivalente à la variable ZONE de l'algorithme du III.2.

V.2.3.3.2. L'aspect

partie commande

Cette partie doit être dupliquée pour tous les Ci puisqu'il possèdent chacun leur $(Xs, Ys)_{Ci}$.

image : charger les valeurs initiales;
existaspect <- faux;
Ya = Y1-R-2*|Ys|;
aspectmin <- faux; aspectmax <- faux;
choix <- faux

```
ligne :  si non existaspect alors
              si Yc = Ya alors existaspect <- vrai;
                        si Ys = 0 alors aspectmin <- vrai;
                                        aspectmax <- vrai;
                                        Ya = Y1 + R
                        sinon si Ys < 0 alors aspectmax <- vrai;
                                              Ya = Y1 - R
                        sinon Ys < 0 alors aspectmin <- vrai;
                                           Ya = Y1 - R
                                           fsi
                        fsi
                  sinon
                        si non (aspectmin et aspectmax) alors
                              si Yc = Ya alors
                                         aspectmin <- vrai;
                                         aspectmax <- vrai;
                                         Ya = Y1 + R
                              fsi
                        sinon
                              si Yc = Ya alors existaspect <- faux;
                                         aspectmin <- faux;
                                         aspectmax <- faux;
                              fsi
                        fsi
                  fsi
```

"choix" détermine le signe de la dérivée.

partie opérative

Nous ne décrivons que la partie "aspectmax" du calcul, le calcul de aspectmin étant symétrique. Cette partie peut être divisée en deux sous-parties : contrôle et calcul.

Le contrôle est le suivant :

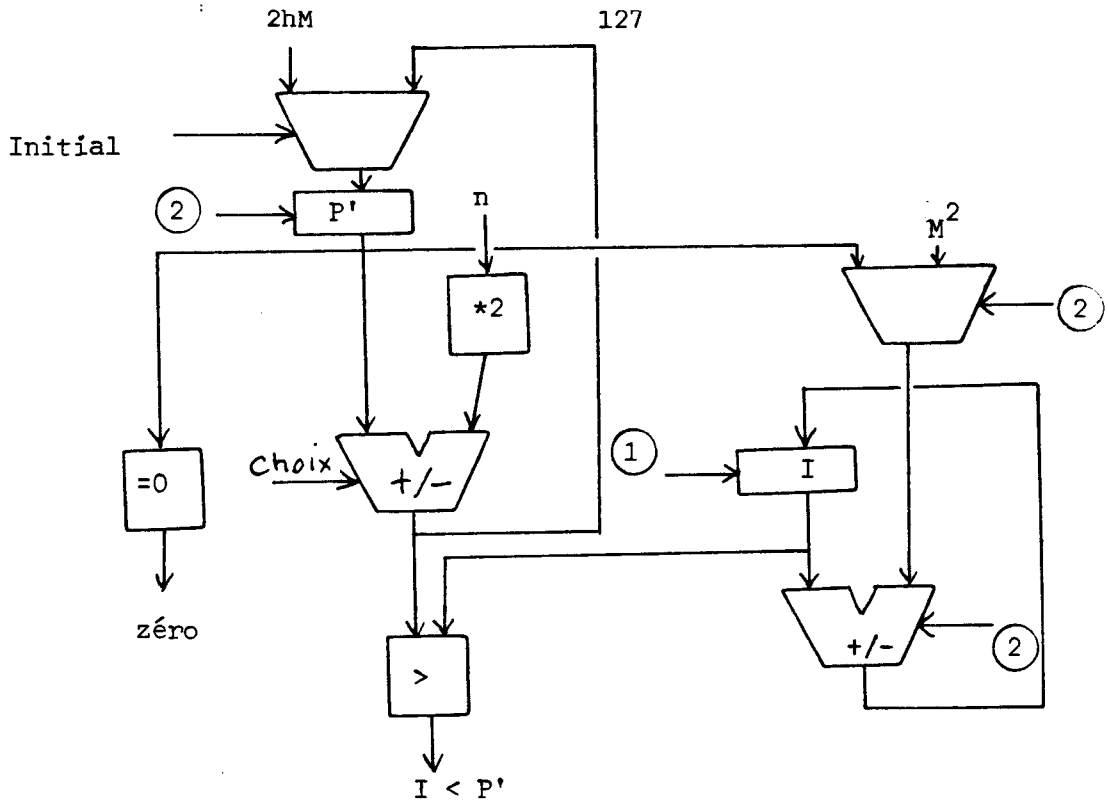
```
aspectmax : si non choix et zéro alors valider sortie P;
                                        choix <- vrai
            sinon
                  tant que I < P' faire
                           activer 1 et 3
                           fait;
                  activer 1 et 2 et valider la sortie de P
            fsi
```

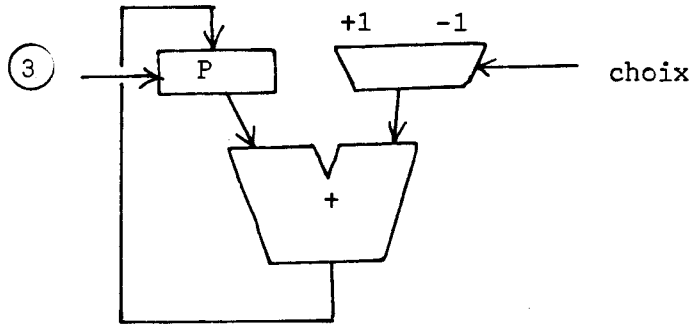
"zéro" indique que la dérivée P' de la courbe est nulle.

La partie opérative utilise l'algorithme P2 (cf. III.3.) avec :

$$M = R + |Ys|$$

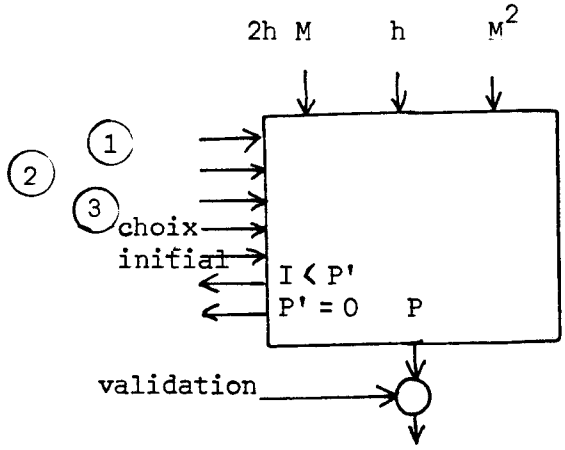
$$h = Ci_{max} - Ci_{min}$$





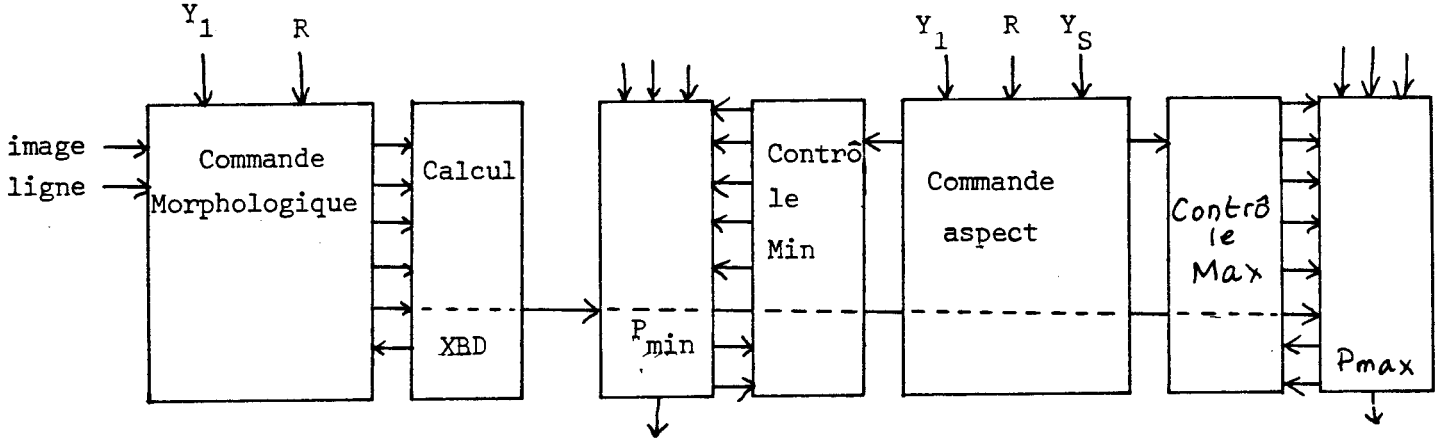
Les registres I et P sont initialisés à zéro au début de chaque image.

Nous remplaçons cet opérateur par une boîte noire :

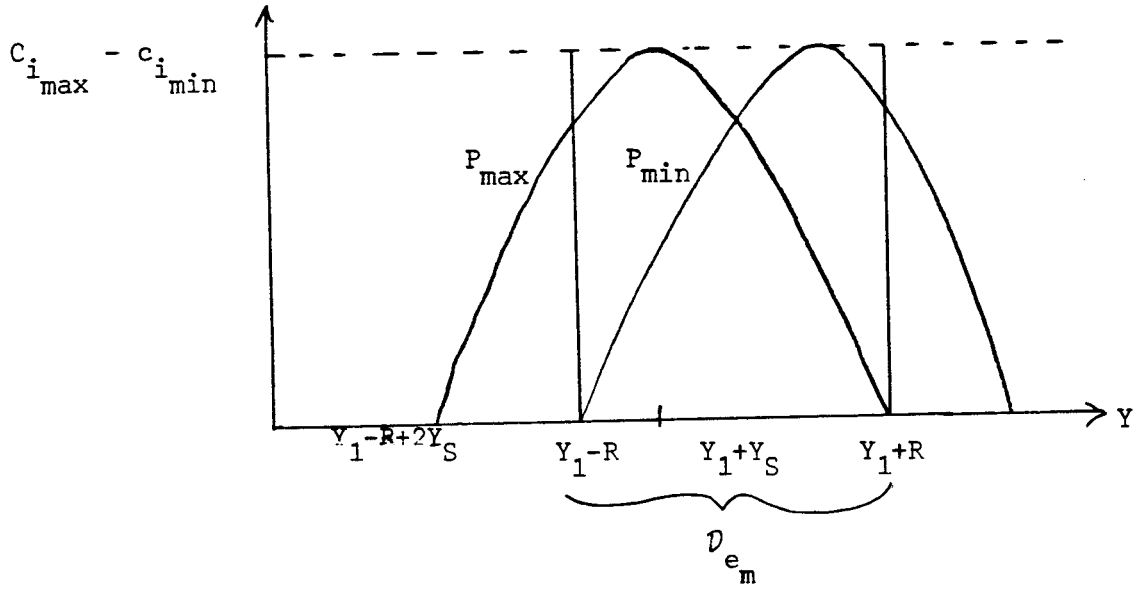


V.2.3.3.3. Mise en forme des résultats

La forme générale de la partie "contour" est schématisée ci-dessous :



La commande s'adressant à tous les blocs de calcul est la commande de sortie. Les courbes Pmin et Pmax sont les suivantes :



Si Ys est négatif, avec Pmin(X) = Pmax(X + 2Ys).

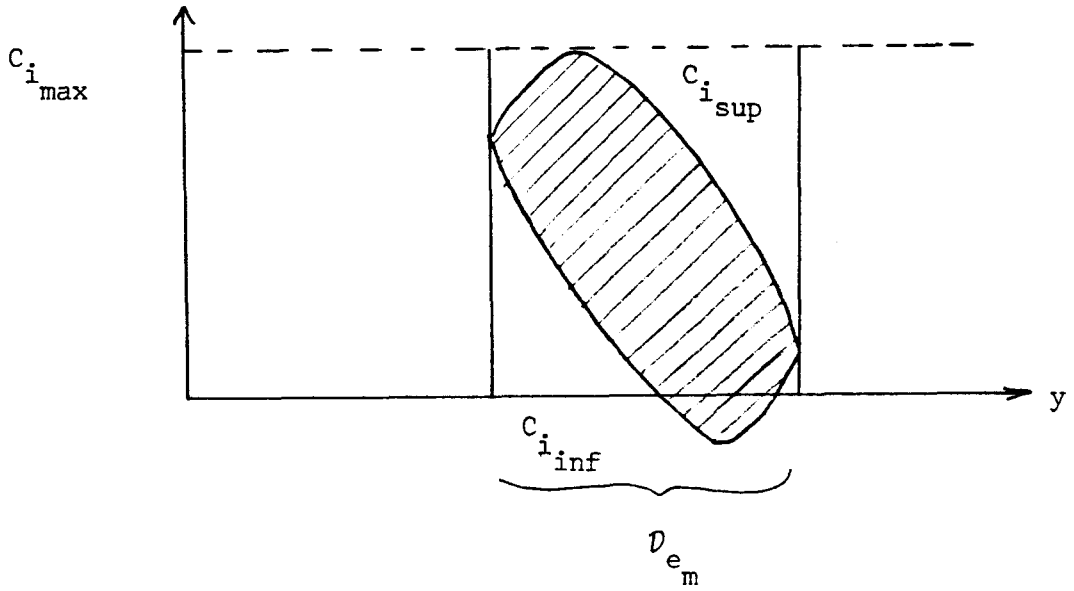
Pour obtenir le domaine recherché il faut transformer ces deux courbes :

$$- Ci_{sup} = Pmax + Ci_{min}$$

$$- Ci_{inf} = Ci_{min} + Pmax(Y1 - R) + Pmin(Y1 - R) - Pmin$$

Pmax(Y1 - R) + Pmin(Y1 - R) représente une valeur de translation, on remarque que Pmax(Y1 - R) > 0 => Pmin(Y1 - R) = 0. Cette translation effectuée sur Ci_{inf} a pour but de fermer le domaine recherché en "recollant" les deux frontières.

Le domaine est le suivant :



Si l'attribut Ci ne peut prendre de valeur négative alors le domaine est tronqué en ramplaçant celles-ci par zéro. Les calculs de Ci_{sup} et Ci_{inf} ne sont effectués que si on se trouve dans De_m, la valeur de translation est donc disponible puisqu'elle est fonction du premier élément de De_m. Cette valeur devra être conservée par la suite.

Connaissant les bornes inférieure et supérieure du paramètre Ci pour toute ligne de remplissage il est alors nécessaire de calculer les paramètres utilisés par le remplissage de l'aspect, c'est-à-dire :

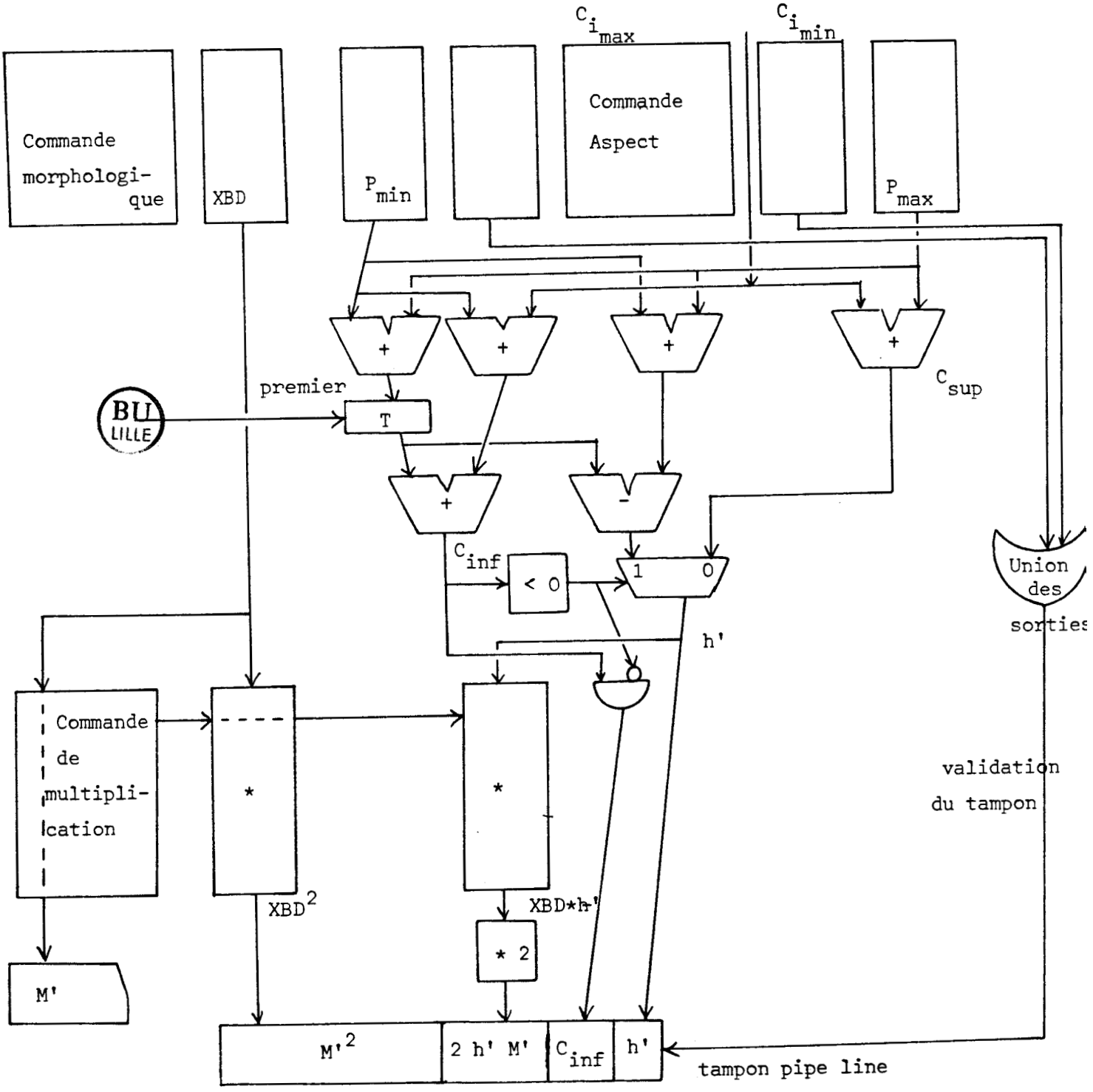
$$h'i = Ci_{sup} - Ci_{inf}$$

$$M'^2 = Xbd^2$$

$$2\ h'i*M' = 2*(Ci_{sup} - Ci_{inf})*Xbd$$

Chaque attribut d'aspect variable nécessite donc une multiplication par ligne de remplissage (l'évaluation de M'^2 est commune à tous les attributs et n'est donc pas prise en compte ici).

En résumé pour la partie contour nous avons :



V.2.3.4. Le remplissage

Comme précédemment nous scindons le remplissage en deux traitements : le calcul morphologique (i.e. le Z de l'élément), et celui d'aspect. Pour les cercles de grand diamètre l'algorithme de BRESENHAM produit de grandes différences de temps de calcul, c'est pouquoi pour le calcul de profondeur (Z) nous utilisons l'algorithme P3 (M' = h) qui présente l'avantage d'une plus grande régularité et de plus faibles temps de calcul. L'algorithme P2 est à nouveau utilisé pour les calculs d'aspect.

Comme lors du calcul de contour nous distinguons deux domaines de validité :

- Dv_m = [X1 - Xbd, X1 + Xbd] validité morphologique

- Dv_a = [X1 - Xbd + 2*Xs, X1 + Xbd] si Xs < 0
  [X1 - Xbd, X1 + Xbd] si Xs >= 0
  validité de l'aspect.

V.2.3.4.1. La morphologie

partie commande

```
ligne : initialisation;
        valide <- faux;
        Xm <- X1 - Xbd

point : si existence alors
          si non valide alors
                si Xc = Xm alors valide <- vrai;
                                Xm <- X1 + Xbd
                fsi
          sinon si Xc = Xm alors valide <- faux
                sinon si zéro et non choix alors
                                valider sortie P;
                                choix <- vrai
                      sinon tant que I < P' faire
                                activer 1 et 3
                                fait;
```

activer 1 et 2 et valider sortie P

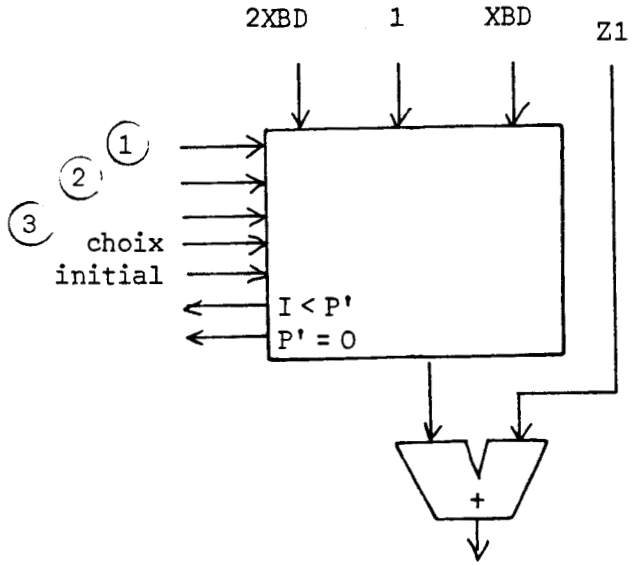
fsi

fsi

fsi

fsi

La partie opérative est semblable à celle de la page -5, seules les constantes entrées sont différentes :



valeur courante de Z

V.2.3.4.2. L'aspect

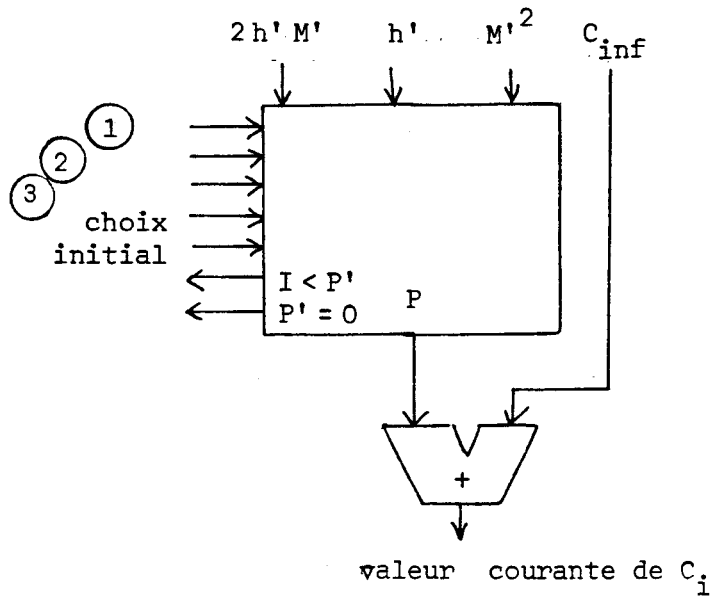
partie commande

ligne : si existence alors initialisations;
valide aspect <- faux;
si Xs < 0 alors
Xa <- X1 - Xbd + 2 Xs
sinon
Xa <- X1 - Xbd
fsi

fsi

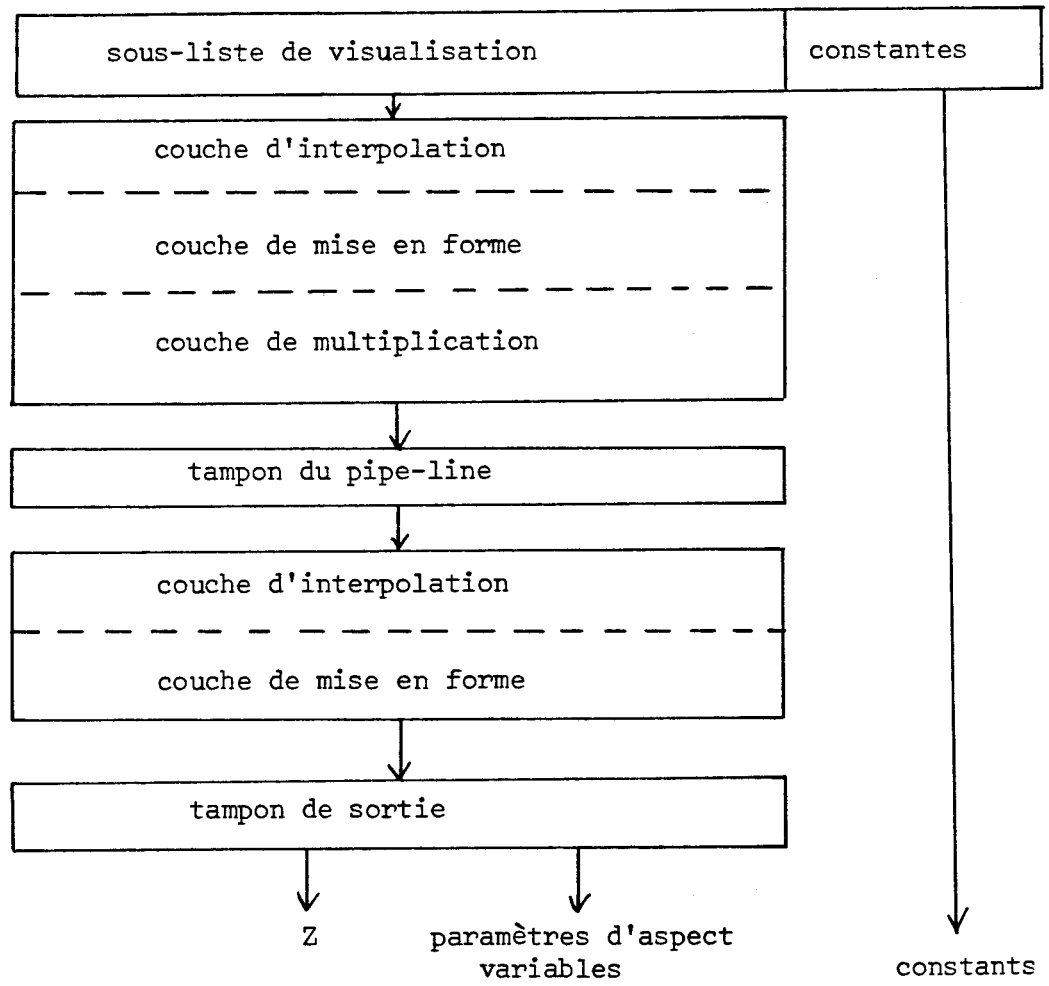
```
point : si existence alors
          si non valide aspect alors
                 si Xa = Xc alors
                          valide aspect <- vrai;
                          si Xs >= 0 alors
                                 sortie <- vrai;
                                 Xa <- X1 + Xbd
                          sinon
                                 Xa <- X1 - Xbd
                          fsi
                 fsi
          sinon si non sortie alors
                       si Xc = Xa alors sortie <- vrai;
                                        Xa <- X1 + Xbd
                       fsi;
                       si non choix et zéro alors choix <- vrai
                       sinon tant que I < P' faire
                                     activer 1 et 3
                                     fait;
                             activer 1 et 2
                       fsi
          sinon si Xc = Xa alors
                        valide aspect <- faux;
                        sortie <- faux
          sinon si non choix et zéro alors
                       choix <- vrai;
                       sortie de P
          sinon tant que I < P' faire
                        activer 1 et 3
                        fait;
                 activer 1 et 2 et valider sortie P
          fsi
    fsi
```

La partie opérative est de la forme :



V.2.3.5. Conclusion

Une vue générale du convertisseur de sphère est présentée :



Le nombre de paramètres à communiquer au PE est le suivant (non compris le nom de l'élément) :

$$A = 4 + 7*p + k$$

où :

p est le nombre d'attributs d'aspect variables.
k est le nombre d'attributs d'aspect constants.

V.3. DECIDEUR ET PROCESSEUR D'UNIFICATION

Les études du décideur et du processeur d'unification (PU) sont regroupées dans le même paragraphe, en effet, nous considérons que le PU est l'unique partie opérative de tous les décideurs, ceux-ci étant les parties commandes. Comme le gérant le mécanisme de décision est commun à tous les types d'élément. Son rôle est de reconnaître si l'élément traité par son PE est "visible" en un point (Xc, Yc) de l'écran. Un élément est visible si les deux conditions suivantes sont satisfaites :

- Il possède la plus faible profondeur Z de tous les éléments.

- Le numéro d'identification de son PE est le plus petit (ceci permet de résoudre le problème de conflit lorsque plusieurs éléments s'intersectent).

L'hypothèse est faite qu'un numéro d'identification est associé de façon unique à un PE.

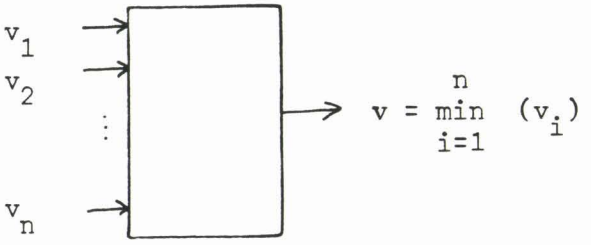
Pour les éléments transparents nous supposerons qu'il n'y a au plus qu'un élément transparent devant l'élément opaque visible, on évite ainsi des conflits éventuels entre éléments transparents.

Nous décrivons tout d'abord un opérateur d'unification puis le fonctionnement général du décideur.

V.3.1. L'unificateur

V.3.1.1. Fonction

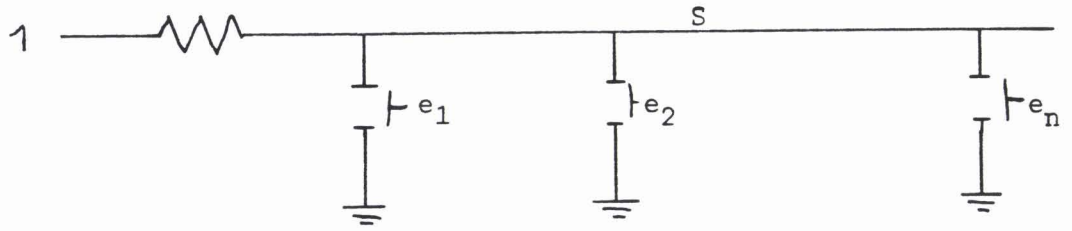
Le rôle de l'opérateur d'unification est d'extraire parmi n valeurs présentées la plus petite valeur :



pour des raisons de modularité et de rapidité nous excluons les solutions utilisant un arbre de comparateurs à deux entrées.

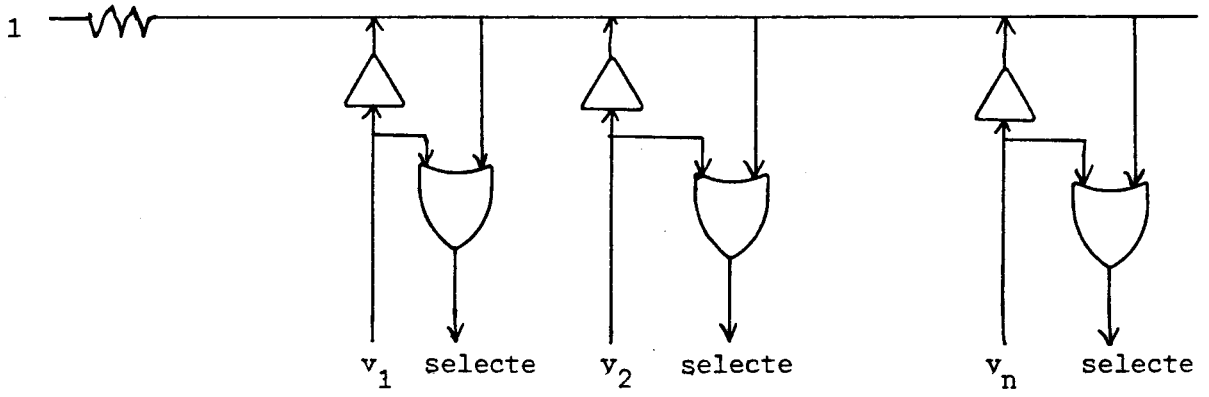
V.3.1.2. Proposition de réalisation

Les solutions présentées utilisent toutes les deux le principe du bus à collecteur ouvert. La particularité d'un tel bus est de réaliser le ET logique des entrées présentes sur chaque fil.



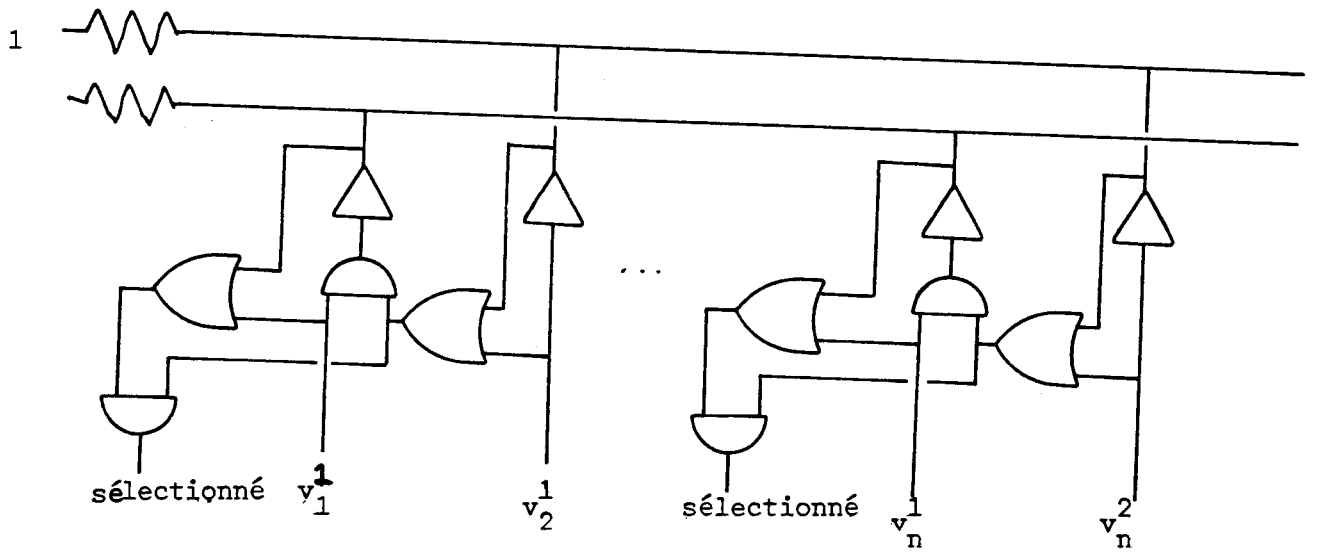
$$S = e_1 \, e_2 \, \ldots \, e_n$$

La première approche est la plus économique en nombre de fils. Elle utilise le bus à la fois comme récepteur des n valeurs et comme émetteur de la plus petite [GEMBALLA 82]. Si v_i est codé sur un bit on a le schéma de principe suivant :



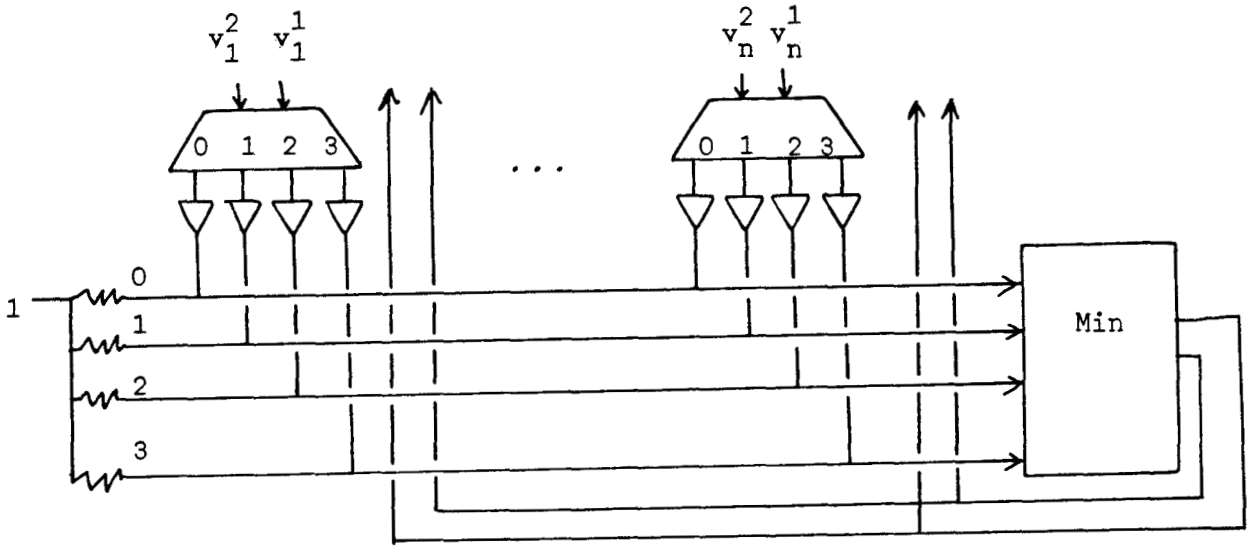
Où les → sont des amplificateurs à collecteur ouvert.

Si v_i est codé sur 2 bits on a alors :



Le temps de sélection est sensiblement proportionnel au nombre de bits formant la valeur v_i. Pour cet inconvénient GEMBALLA propose un fonctionnement en pipe-line par bit. Il faut noter que si v_i utilise p bits, la largeur du bus n'est que de p. Nous verrons cependant que le nombre d'entrées/sorties pour la sélection passe à 2*p dans le cas d'un bus à structure hiérarchique.

La seconde approche, moins économique en nombre de fils mais plus générale, est celle que nous avons développée [DURIF 82]. Elle consiste à décoder chaque v_i en 1 parmi 2^p, un circuit combinatoire sélectionne alors la plus petite valeur et la renvoie à tous les processeurs. Par exemple si p = 2 :



Le temps de stabilisation du bus n'est plus proportionnel au nombre p.

Remarque : Le nombre de fils du bus à collecteur ouvert peut être réduit de 1 en notant que le fil correspondant à la plus grande valeur n'apporte pas d'information. En effet, si aucun des fils n'est activé alors c'est forcément celui-là qui l'est, sinon on choisira la plus petite valeur.

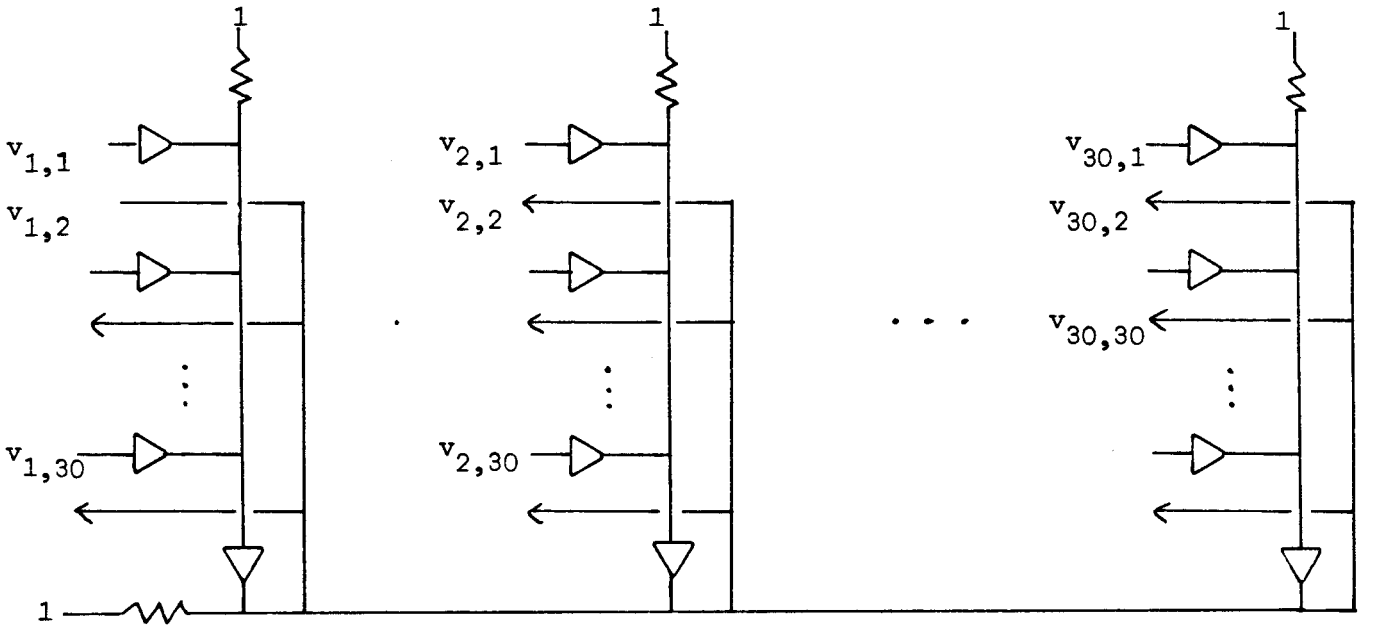
Le nombre d'entrées/sorties est alors de $2^p - 1 + p$. Le cas p=1 est sensiblement équivalent au schéma de GEMBALLA. On peut opposer à cette solution le fait que 2^p peut être très grand, on découpe alors v_i en p/c tranches de c bits et on traite chacune de celle-ci soit de façon purement séquentielle soit en utilisant un pipe-line à p/c étages. Cette dernière solution est une généralisation du schéma de GEMBALLA, et nécessite :

$(p/c)*(2^c + c - 1)$ entrées/sorties

V.3.1.3. Les contraintes

En technologie TTL par exemple le nombre maximum d'entrées que peut supporter un bus à collecteur ouvert est d'environ 30. Le nombre de PE branchés sur un tel bus pouvant être de l'ordre du millier nous proposons une structuration hiérarchique du bus :

Par exemple pour p=1



Ce schéma est commun aux deux approches précédentes.

Le temps de stabilisation d'un bus à collecteur ouvert n'est pas négligeable, nous sommes donc amenés à introduire un pipe-line dans l'opérateur d'unification, chaque niveau de la hiérarchie de bus correspondant à un étage du pipe.

Le paragraphe suivant montre que le décideur peut palier lui aussi au problème de rapidité.

V.3.2. Le décideur

Son rôle est d'établir la visibilité d'un élément pour un point (Xc, Yc) de l'écran. Il dispose pour cela de cinq informations :

- Le bit d'occupation qui commande la validation de tous les traitements et sorties (cf. le gérant). Celui-ci doit en fait être recopié comme c'est le cas pour les représentations dynamiques.

- L'indicateur "valide" de l'élément traité qui valide lui aussi les sorties et les traitements.

- Un indicateur de transparence n'invalidant que les sorties côté unificateur.

- La profondeur Z en (Xc, Yc) de l'élément traité.

- Le numéro d'identification NI du PE qui est constant.

V.3.2.1. Principe de fonctionnement

Soient c, p et n les longueurs d'une tranche, de Z et de NI. Le décideur est alors découpé en (n+p)/c étages de pipe-line.

Le fonctionnement d'un étage i dans les p/c premiers et concernant Z est le suivant :

```
faire toujours
     si visible (i-1) alors
          si non transparent alors envoyer tranche(i) sur le bus
                à collecteur ouvert n° i
          fsi
          si tranche(i) <= tranche reçue alors
               visible(i) := vrai
          sinon
               visible(i) := faux
          fsi
     sinon
          visible(i) := faux
     fsi
fin toujours
```

ceci pour i=1, p/c

Le fonctionnement de l'étage i des n/c restant qui concernent NI est :

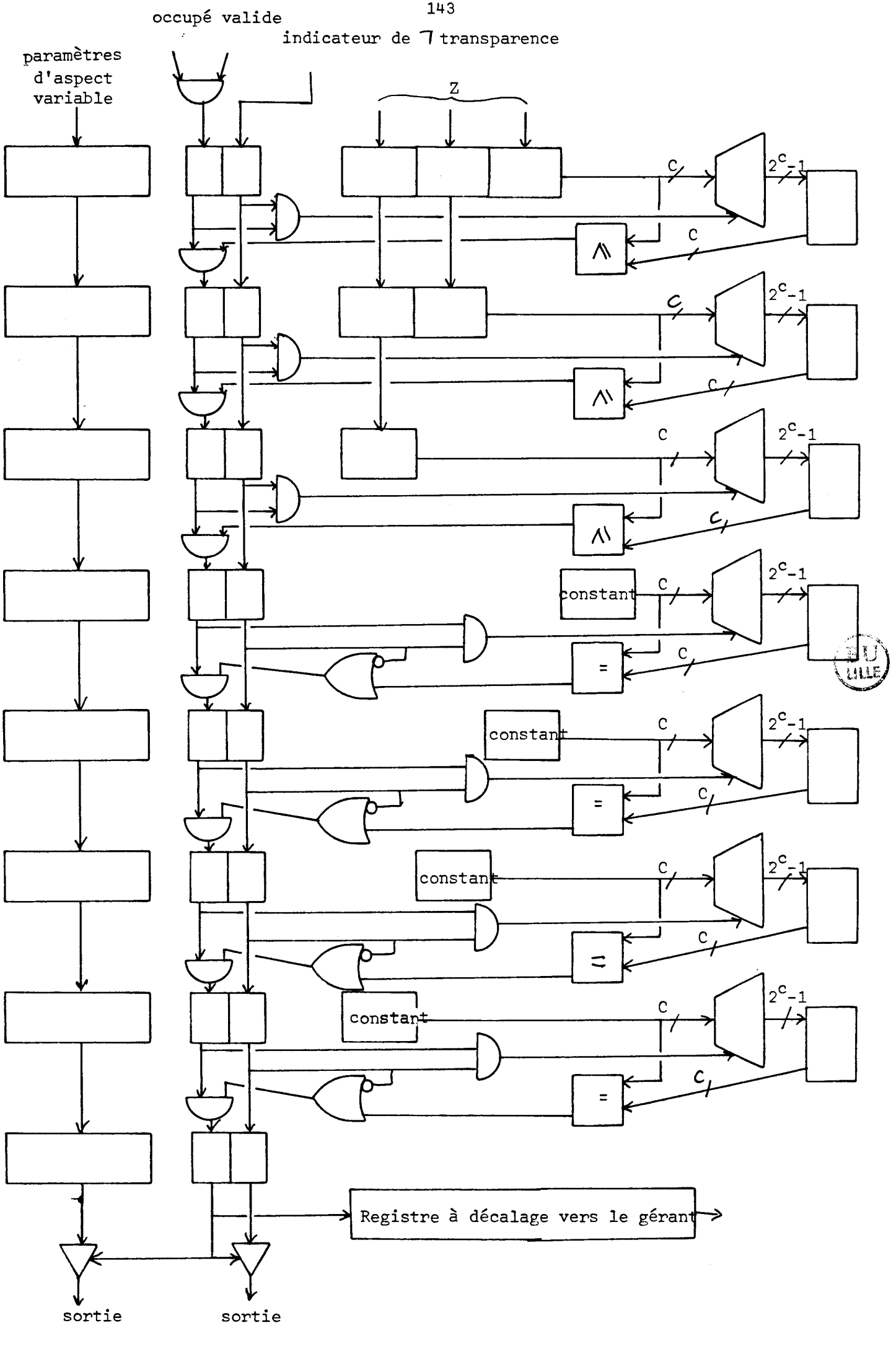
```
faire toujours
     si non transparent alors envoyer tranche(i) sur le bus à
                          collecteur ouvert n° i;
          si tranche(i) <= tranche reçue alors
                    visible(i) := vrai
          sinon
                    visible(i) := faux
          fsi
     sinon
          visible := vrai
     fsi
 sinon
     visible := faux
 fsi
fin toujours
```

ceci pour i = p/c + 1, (p+n)/c

Remarques : - Visible(0) := occupé ET valide
- Visible((n+p)/c) valide les sorties vers la synthèse finale et est envoyé au registre à décalage amenant l'indicateur de visibilité au gérant (cf. V.1.3.). La longueur de ce registre est égale au nombre d'étages du pipe-line restant à parcourir avant l'affichage effectif.

V.3.2.2. Architecture du décideur

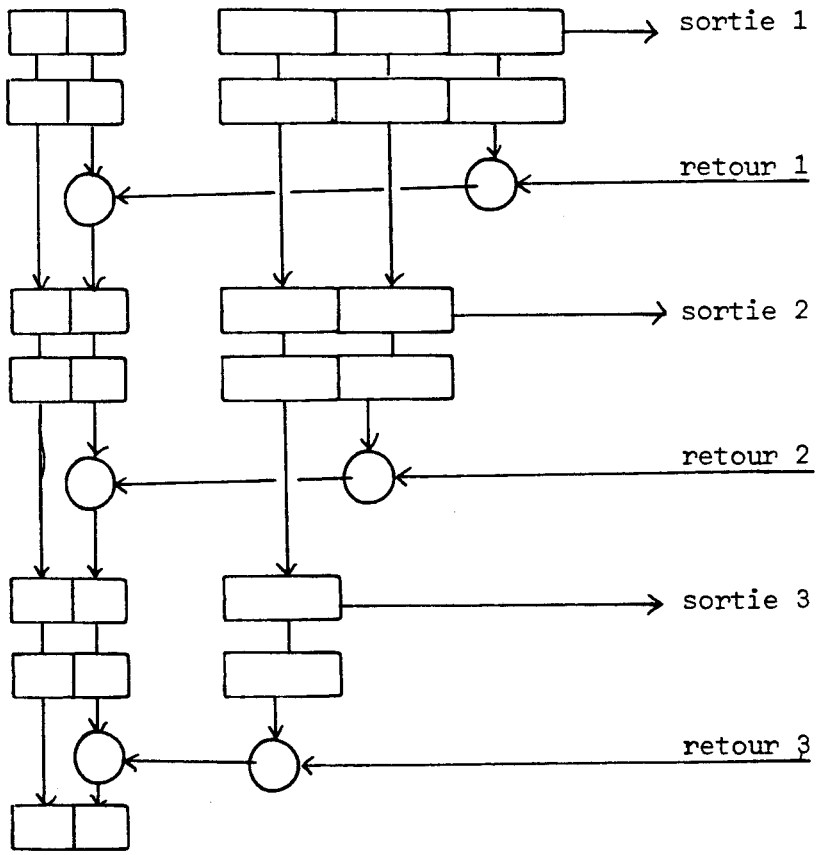
La figure présente le cas où p/c = 3 et n/c = 4 :


occupé valide
indicateur de transparence
paramètres d'aspect variable
Z
C
2^C-1
constant
=
Registre à décalage vers le gérant
sortie
sortie

Remarques : - Il n'y a pas de pipe-line de transmission des paramètres d'aspect constants, cependant leur sortie est elle aussi sanctionnée par visible ((n+p)/c).

- Le décideur possède :
  p/c comparateurs d'infériorité sur c bits
  n/c comparateurs d'égalité sur c bits
  (n+p)/c décodeurs sur c bits
  (n+p)/c + 1 étages de registres.

- L'utilisation d'un pipe-line de k niveaux dans les opérateurs d'unification multiplierait par k le nombre d'étages de registres et le nombre de registres nécessaires à Z et NI sans augmenter la partie combinatoire (décodeurs et comparateurs). Pour k = 2 et pour Z on aurait :



V.3.2.3. Conclusion

Un mécanisme d'unification a été câblé qui comporte un opérateur d'unification et deux décideurs qui ne prennent en compte que Z avec les caractéristiques suivantes :

c = 4 et p = 8

Les décideurs n'utilisent pas le pipe-line précédemment décrit et fonctionnent jusqu'à 10 mégahertz pour une décision complète (50 ns par tranche de 4 bits), ce qui semble montrer qu'un tel mécanisme est viable.

Les sorties vers l'affichage telles qu'elles sont présentées supposent un aiguillage de celles-ci en fonction de l'indicateur de transparence.

V.4. LA SYNTHESE FINALE

Les paramètres d'aspect fournis par le décideur sélectionné ne sont pas forcément directement affichables. Il faut donc traduire les paramètres obtenus en informations affichables par le moniteur couleur qui sont généralement R, V, B (rouge, vert, bleu).

Le second rôle de la synthèse finale est d'effectuer la composition de couleurs lorsqu'un élément transparent se trouve devant un opaque.

Le premier paragraphe présente rapidement trois modèles de couleur, les deux suivants tentent de résoudre la traduction et la composition de couleurs.

V.4.1. Quelques modèles [GERRITSEN 75]

V.4.1.1. RVB (rouge, vert, bleu)

Utilisé par les écrans cathodiques couleurs en synthèse additive il ne permet pas une compréhension ou une manipulation aisée de la couleur. Les trois dimensions R, V et B sont supposée discrètes et prennent leurs valeurs dans [0, n-1].

V.4.1.2. TBN (teinte, blanc, noir)

Ce modèle se déduit de RVB par la transformation suivante :

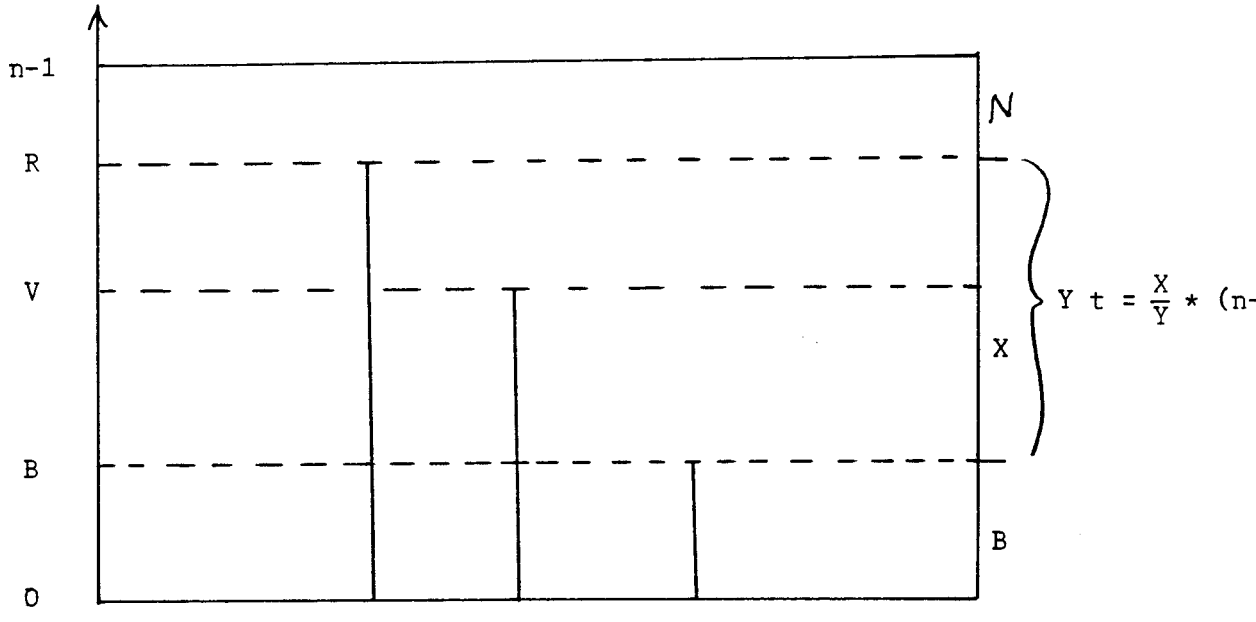
$$B = \text{Min}(R, V, B)$$
$$N = (n-1) - \text{Max}(R, V, B)$$
$$t = \frac{\text{Inter}(R, V, B) - B}{\text{Max}(R, V, B) - B} * (n-1)$$

avec c_j $\text{Inter}(c_1, c_2, c_3) \Leftrightarrow c_i <= c_j <= c_k \quad i \neq j \neq k$

$$\text{et } (i, j, k) \quad [1, 2, 3]^3$$

La transformation apparaît plus clairement sur la figure où on a choisi

$$R <= V <= R :$$



Remarquons que tBN ne suffit pas pour retrouver une couleur RVB, il manque une indication d'ordonnancement de R, V et B. Pour résoudre ce problème deux techniques sont possibles :

- On ajoute un quatrième paramètre O qui prend six valeurs et indique les ordres suivants :

  B <= V <= R
  B <= R <= V
  V <= B <= R
  V <= R <= B
  R <= V <= R
  R <= B <= V

  le modèle possède alors quatre variables t, B, N et o.

- La seconde solution est le modèle classique TBN où T intègre les deux valeurs t et o.

La traduction de TBN ou (toBN) vers RVB est exprimée par :

$$Max = n - 1 - N$$
$$Min = B$$
$$Inter = \frac{t}{(n-1)} * (n-1-B-N) - B$$

L'effet de o étant d'affecter correctement Max, Min et Inter aux fondamentales.

V.4.1.3. TIS (teinte, intensité, saturation) [MERIAUX 82]

Tout en offrant une certaine facilité d'interprétation ce modèle est assez proche de la "réalité physique" (un effet d'ombre pourra être obtenu par une diminution de la saturation). Ce sont les trois paramètres de ce modèle qui sont sous-jacents aux deux paragraphes suivants.

V.4.2. Traduction M → RVB

Nous proposons différents mécanismes permettant le passage d'un modèle de couleurs M à n variables (m_1, m_2,..., m_n) au modèle RVB.

Les mécanismes statiques sont ceux qui utilisent des tables de traduction, les mécanismes dynamiques traduisent M par calcul.

V.4.2.1. Mécanisme statique rechargeable

Il s'agit en fait d'une table de fausses couleurs. Elle est chargée avant tout affichage d'une image avec les triplets RVB qui seront nécessaires à l'affichage. Les n-uplets (m_1, m_2,..., m_n) représente une adresse dans la table.

Cette solution ne peut être efficace que si la table est de taille réduite, c'est-à-dire que celle-ci ne contient pas toutes les couleurs possibles. Le problème principal est alors de réaliser une fonction de modification des n-uplets en un index correct. Ceci reporte la complexité de la traduction au niveau des adresses, nous ne retiendrons pas cette solution.

V.4.2.2. Mécanisme statique pur

Cette fois la table contient toutes les couleurs possibles, il n'y a plus de mécanisme de traduction d'adresse mais la dimension de la table devient prohibitive :

si M = TIS et n est le nombre de bits pour chacune des composantes R, V et B alors

$$T = [0, 6*(2^n - 1)]$$

$$I = [0, 2^n - 1]$$

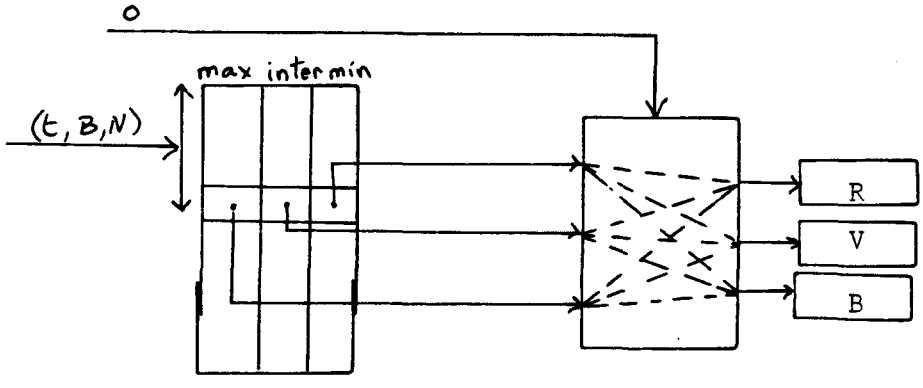
$$S = [0, 2^n - 1]$$

Ce qui porte la capacité de la table à 186K*15 bits avec n = 5 et pour seulement 32K couleurs distinctes !

Les paragraphes suivants proposent dans le cas des modèles TBN et TIS deux solutions complémentaires de réduction de la capacité.

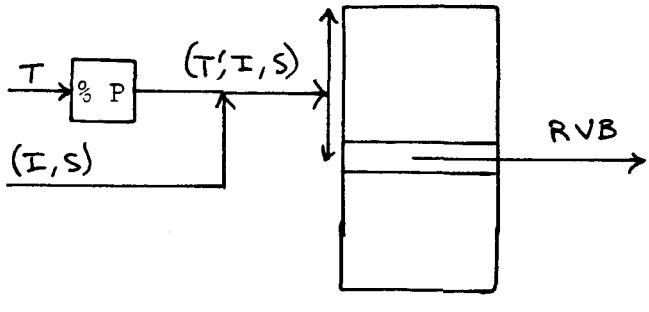
V.4.2.3. Traduction semi-statique

Pour les modèles TBN et TIS on décompose T en t et o, une adresse dans la table est alors (t, I, S) ou (t, B, N), la capacité de la table est réduite par 6 (31K*15 bits dans l'exemple précédent), la valeur de o commande un multiplexeur d'attribution des trois valeurs aux fondamentales :



V.4.2.4. Réduction de la définition

On se propose de réduire les domaines de définition des paramètres d'aspect constants au sens des convertisseurs. Cette réduction ne pénalise en rien la qualité de l'image obtenue si les paramètres constants des éléments n'ont pas de valeur en dehors des domaines réduits. En s'appuyant sur le modèle TIS (TBN) on peut choisir T (ou t) comme paramètre constant. Il est alors possible de réduire par p le domaine de T(t) :



La taille de la table est réduite d'un facteur p.

La table peut être une mémoire vive permettant par chargement de disposer virtuellement de toutes les teintes produites par les convertisseurs.

La composition des deux dernières solutions avec n = 5 et p = 2 donne une taille de table pour l'exemple de 16K*15 bits.

V.4.2.5. Traduction dynamique

Elle n'a pas été étudiée. Elle correspond à câbler les algorithmes de traduction.

V.4.2.6. Conclusion

Les mécanismes proposés sont indépendants du reste du terminal, ce sont eux qui donnent une signification aux attributs d'aspect en les interprétant. Il est possible dès lors de changer de modèle en modifiant la table de traduction (TIS et TBN).

D'autres réductions peuvent être effectuées à condition de s'attacher à un modèle de couleurs. Par exemple pour TIS la quantité Min se déduit facilement de la valeur de Max par Min = 2*I - Max, ce qui diminue de un tiers la taille du mot mémoire.

L'oeil étant plus sensible aux différences d'intensité qu'aux variations de teintes il est envisageable de favoriser la définition de la première au détriment de la seconde.

Enfin citons HELIOS qui à partir d'un vecteur (R, V, B) et d'une intensité produit la couleur à afficher : I*(R, V, B) [FERREIRA 81], [MARTINEZ 82].

V.4.3. Composition de couleurs

Il convient de distinguer trois types de composition de couleurs [GERRITSEN 75] :

- La composition additive qui est celle réalisée par les écrans cathodiques et plus généralement lors de la fusion de plusieurs faisceaux lumineux.

- La composition soustractive effectuée par le mélange de couleurs (cas de la transparence).

- La composition partitive obtenue par juxtaposition de pièces uniformément colorées.

A partir des mêmes couleurs chacune de ces compositions donne un résultat différent. Seul le cas de la composition par transparence est étudié. Nous proposons de séparer cette composition en deux parties :

- Prise en compte des superpositions d'intensités lumineuses (il faut employer un modèle possédant un paramètre d'intensité). Ceci correspond au cas de la synthèse additive.

- Prise en compte du phénomène d'absorption par l'élément transparent. Cette absorption dépend de la couleur de l'élément : un rouge laisse passer la composante rouge de la couleur filtrée ; cette partie correspond à une synthèse soustractive et nous choisissons de l'effectuer dans le modèle final RVB.

Pour l'"addition" des intensités nous nous en tiendrons au mécanisme proposé par NEWELL, NEWELL et SANCHA [NEWELL 72] ; si I_t et I_o sont les intensités des éléments transparent et opaque, l'intensité résultante I est la suivante :

$$\text{si } I_t < I_o \text{ alors } I = I_t$$

$$\text{sinon } I = I_t * tr + I_o * (1-tr) \quad tr \in [0, 1]$$

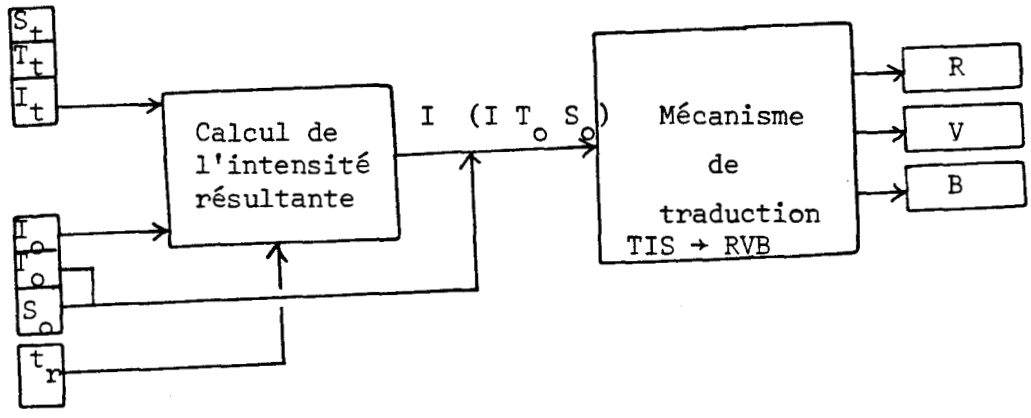
où tr est un coefficient de transparence.

Dans une réalisation câblée cet algorithme utilise deur multiplieurs, un additionneur et un comparateur. Il est plus commode d'utiliser une valeur entière pour tr :

$$tr \in [0, T] \quad \text{avec } T = 2^n - 1$$

En ce qui concerne la composition soustractive des couleurs deux politiques ont été étudiées et simulées.

La première consiste à remplacer l'intensité I_o de l'objet opaque par la nouvelle intensité calculée ; c'est alors la teinte de ce dernier qui est affichée. Ce mécanisme implique que l'objet transparent n'a pas de couleur (comme un verre incolore). La simulation montre que cette approximation est relativement satisfaisante. Le schéma général de la synthèse finale est alors le suivant :



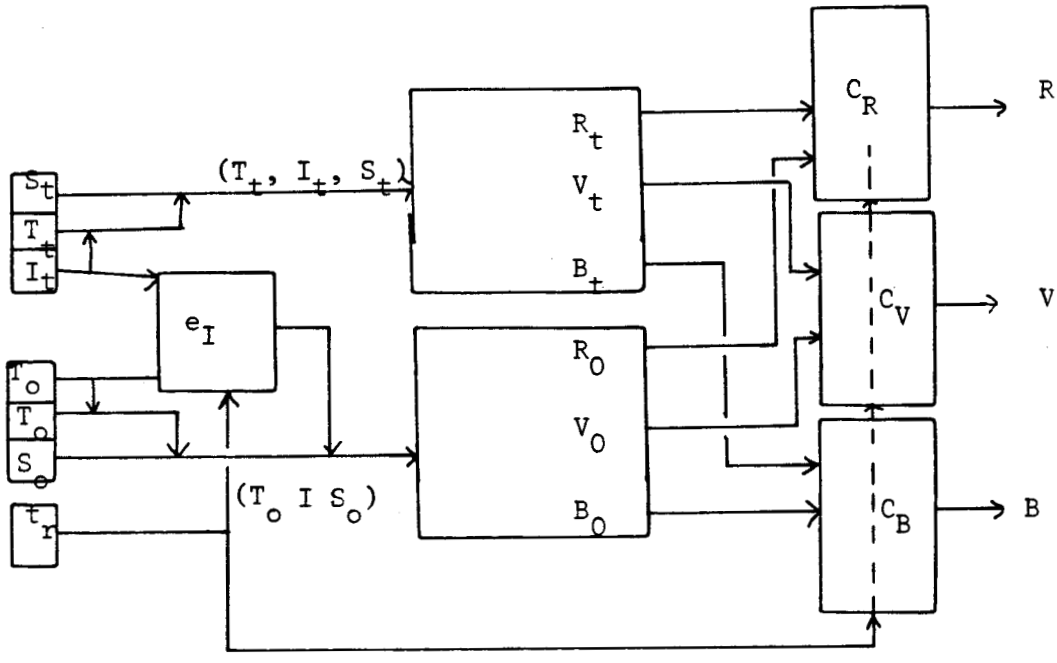
La deuxième approche consiste à tenter d'effectuer la synthèse soustractive en tenant compte de l'objet transparent. La méthode choisie est la suivante :

- (I_o, T_o, S_o) est remplacé par (I, T_o, S_o)
- On obtient alors (R_o, V_o, B_o) pour l'objet opaque et (R_t, V_t, B_t) pour l'élément transparent.

- La couleur résultante est obtenue par trois équations de la forme générale :

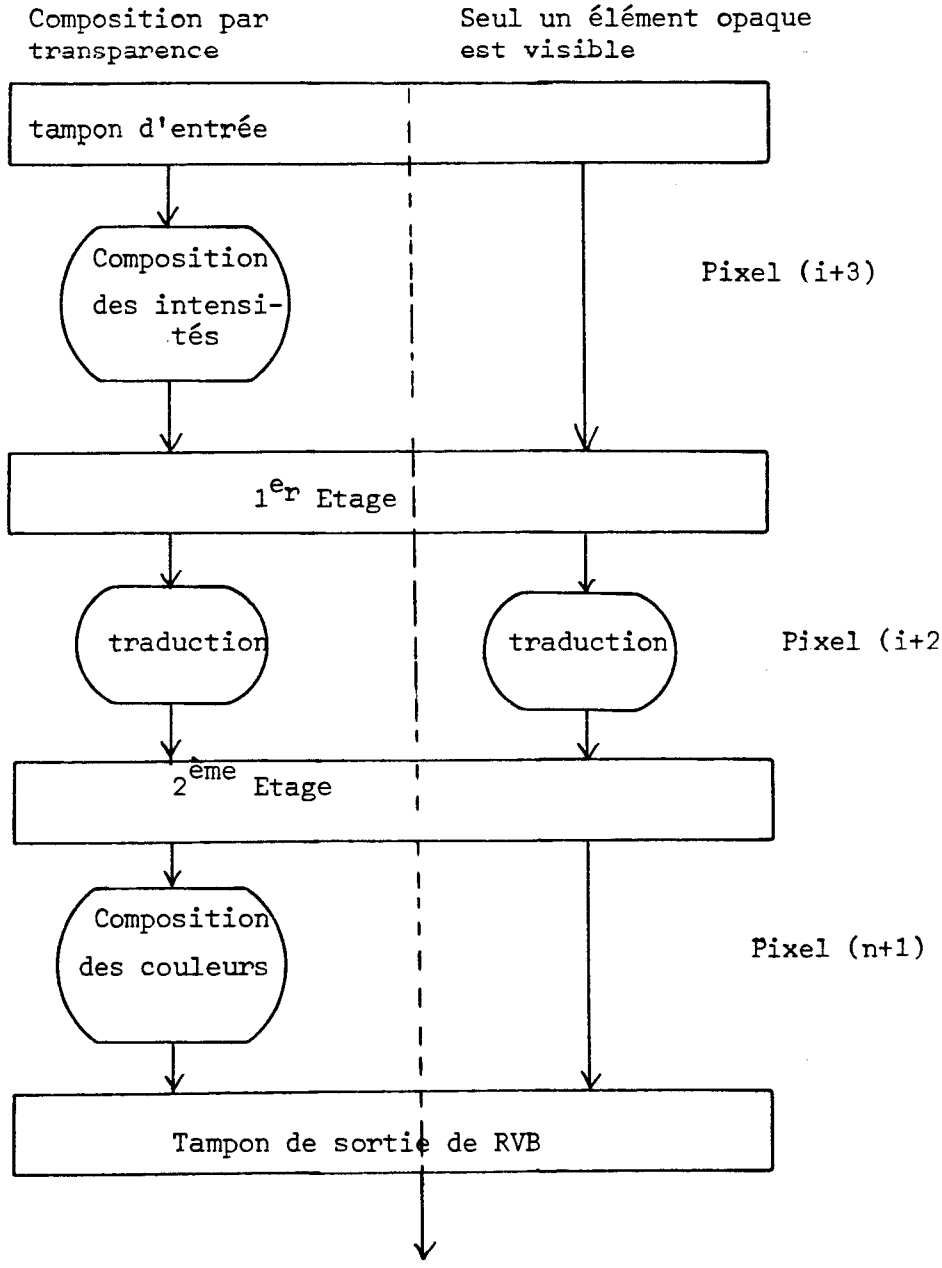
$$x = \frac{x_t * tr + x_o * (T + 1 - tr)}{T + 1}$$

Comme précédemment l'effet obtenu par simulation est relativement satisfaisant. Le schéma de composition est plus complexe que dans la première approche, en particulier il est nécessaire de dupliquer (physiquement ou par multiplexage temporel) le mécanisme de traduction TIS → RVB :



Les boîtes de composition C_R, C_V et C_B contiennent chacune deux multiplieurs, un additionneur et un complémenteur à 2^n, elles sont les mêmes que la boite C_I.

Pour conclure remarquons que la composition par transparence n'est pas le cas général, le cas le plus fréquent étant celui où seul un élément opaque est visible, il faut alors emprunter un autre chemin de traduction. Cependant l'opérateur de transparence étant relativement complexe il est nécessaire de pipe-liner cette synthèse finale :



L'algorithme de composition peut bien entendu être raffiné sans que le coût du système global en soit profondément modifié, en effet ce mécanisme est unique dans le système. C'est le nombre d'étages de ce pipe-line qui détermine la longueur du registre à décalage amenant l'indicateur de visibilité au gérant (cf. V.3.2.1).

CONCLUSION

Seuls quelques systèmes de synthèse d'images réalistes permettent actuellement une vraie animation en temps réel. Ce sont avant tout les simulateurs de conduite dont le prix est suffisamment élevé pour les réserver à des organismes dont le budget est conséquent (en particulier l'armée et quelques compagnies aériennes).

Le système décrit dans les pages précédentes, par sa flexibilité et son extensibilité, permettrait d'atteindre des coûts bien inférieurs aux produits actuels. Il faut souligner qu'il ne prend en compte que quelques traitements de synthèse et suppose un processeur hôte effectuant la description de l'univers et sa manipulation en termes de calculs de perspective, de clôture de l'image, d'évaluation de l'aspect en des points caractéristiques des objets et de gestion de ceux-ci.

Plusieurs voies de développements visant à augmenter l'autonomie et les capacités de traitement des PE sont envisageables :

- Donner aux PE la possibilité d'appliquer une texture à leur élément. Un procédé primaire pourrait consister à modifier l'intensité calculée suivant une loi de variation (cyclique, pseudo-aléatoire,...).

- Certains traitements de synthèse sont peu complexes lorsqu'ils sont appliqués à des éléments possédant de "bonnes" propriétés. Par exemple des transformations géométriques telles que translation, rotation et un calcul de perspective peuvent être appliqués à des sphères. L'intégration de tels traitements dans les PE paraît peu coûteuse en encombrement (compte tenu de celui déjà prévu, cf. annexe) et n'affecterait pas les performances étant donné le temps disponible (un cycle de rafraîchissement complet ou le temps séparant deux trames).

- Un inconvénient typique de cette architecture et le faible taux de participation de chaque PE à l'élaboration de l'image (le De d'un élément peut ne représenter que quelques lignes de la trame). On peut alors imaginer de confier plusieurs éléments à un PE à condition que les De des éléments soient disjoints. Cette solution est séduisante car elle a l'avantage de ne modifier que la taille de la représentation dynamique et la complexité du gérant.

A l'extérieur des processeurs-éléments on peut aborder :

- La prise en compte de l'aliassage côté affichage. Une technique pourrait être d'utiliser un tampon d'une ou plusieurs lignes de trame et d'effectuer un filtrage sur une fenêtre de plusieurs pixels. Cependant le registre à décalage transportant l'indicateur de visibilité de la sortie du décideur jusqu'au gérant serait ralongé de la longueur du tampon. Une autre solution consiste à calculer en surésolution et utilise elle aussi un registre tampon et pose donc le même problème.

- La réalisation du PI qui peut être un multiprocesseur.

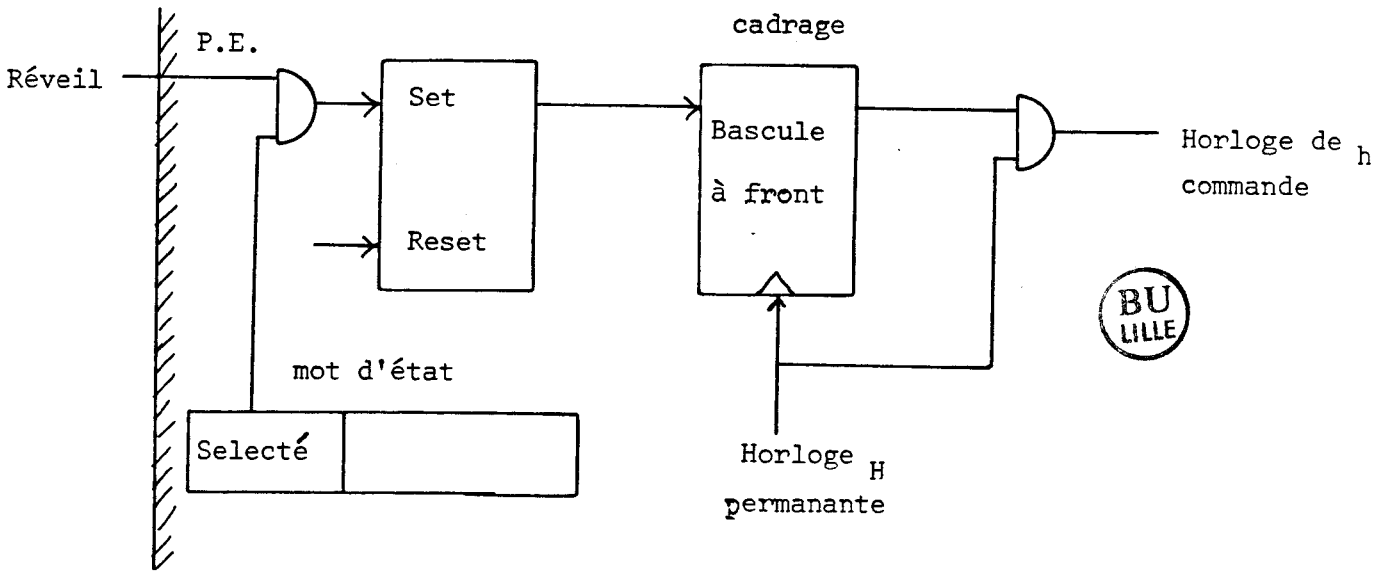
- L'étude de mécanismes de composition de couleurs plus réalistes dans leurs effets.

Les différentes machines de synthèse d'images utilisant le découpage par objets n'ont à notre connaissance pas fait l'objet de réalisation matérielles. L'architecture décrite ici s'inscrit dans cette classe de propositions. Nous pensons cependant que celle-ci peut connaître d'intéressants développements par son adaptation aux représentations des données souvent utilisées dans les systèmes de synthèse.

ANNEXE 1

A.1. MECANISME DE REVEIL D'UN GERANT

Le signal "réveil" émis par l'interface réveille un gérant à condition que celui-ci ait été sélectionné au préalable.

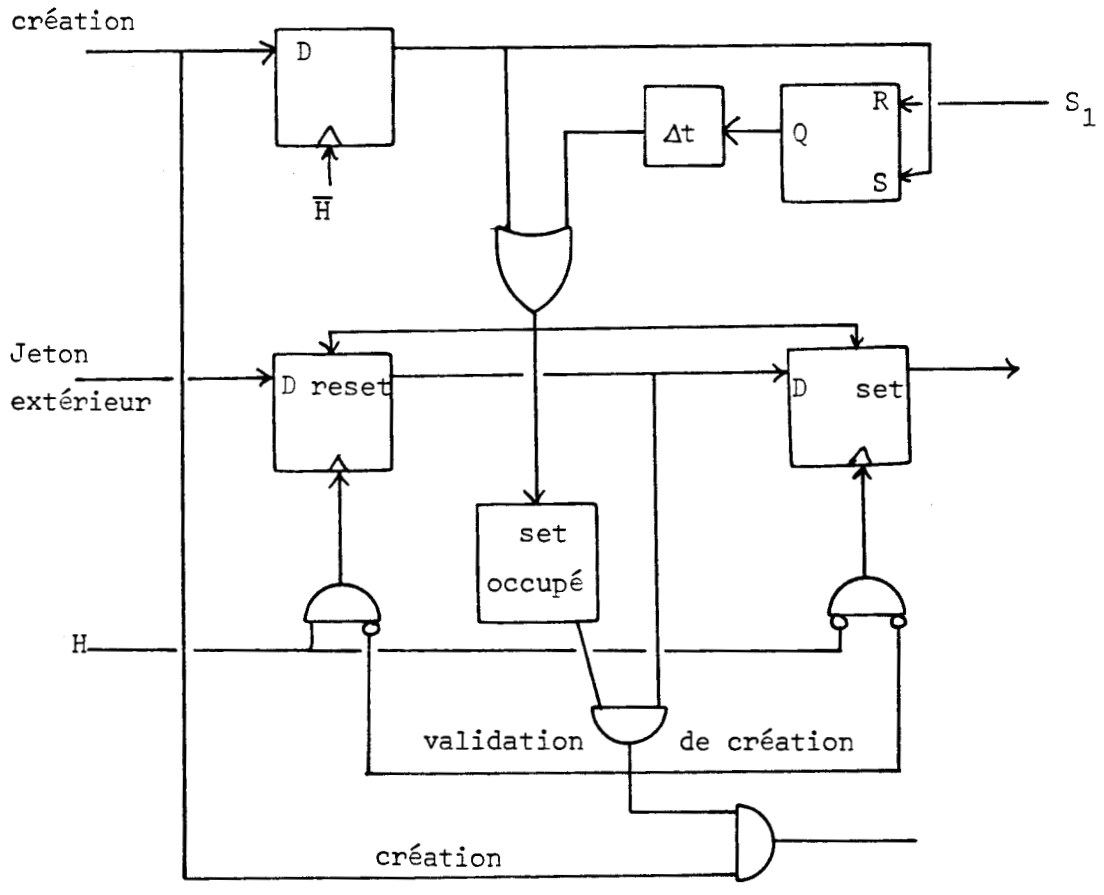


A.2. MECANISME DE SELECTION PAR JETON CIRCULANT

Le circuit câblé gérant le jeton réagit de la manière suivante :

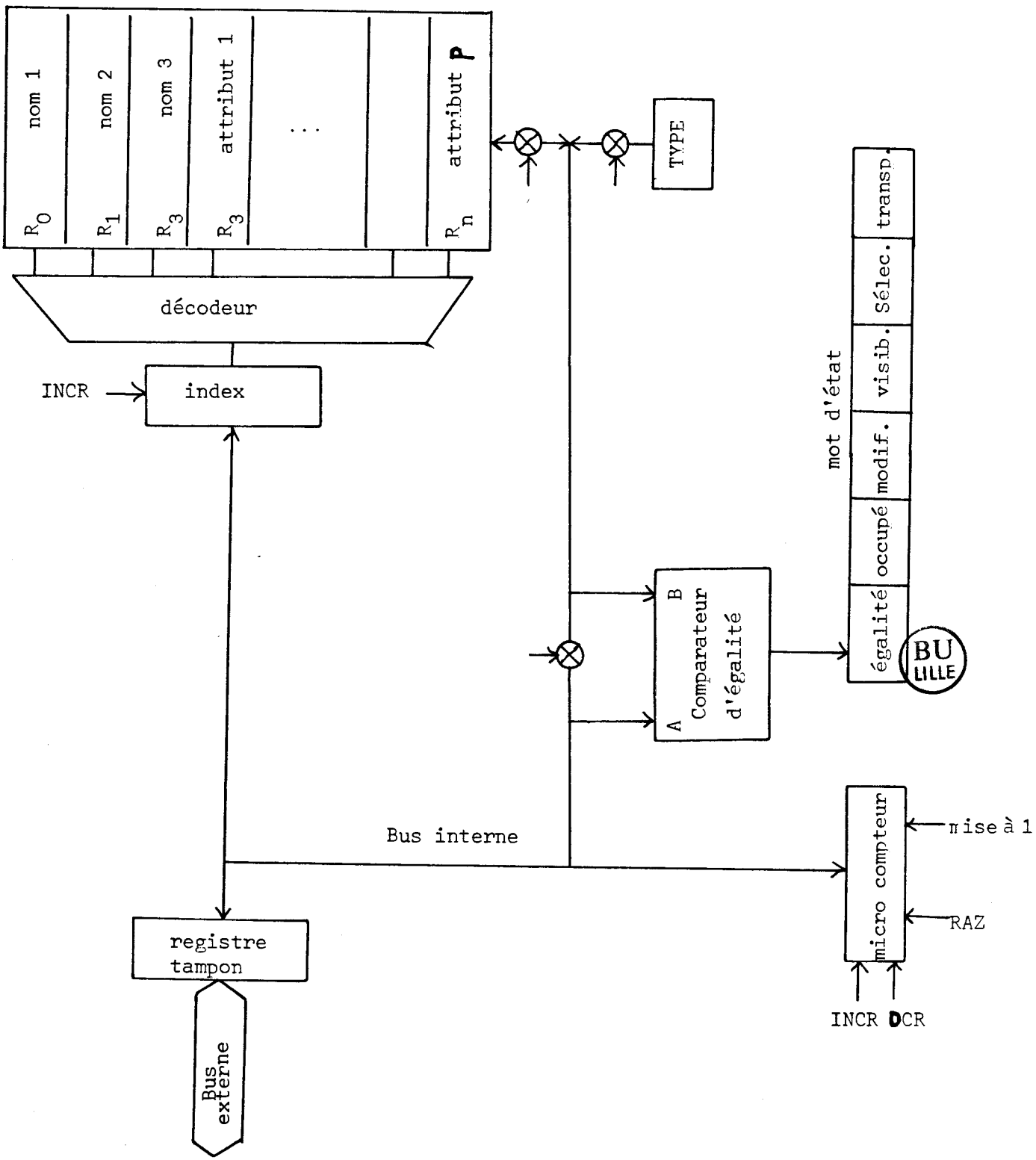
- Si le PE est occupé il est transparent vis-à-vis du jeton et le passe à son voisin.

- Si le PE est libre et qu'il n'a pas de jeton il attend que son voisin le lui passe puis le garde en attendant d'être occupé.



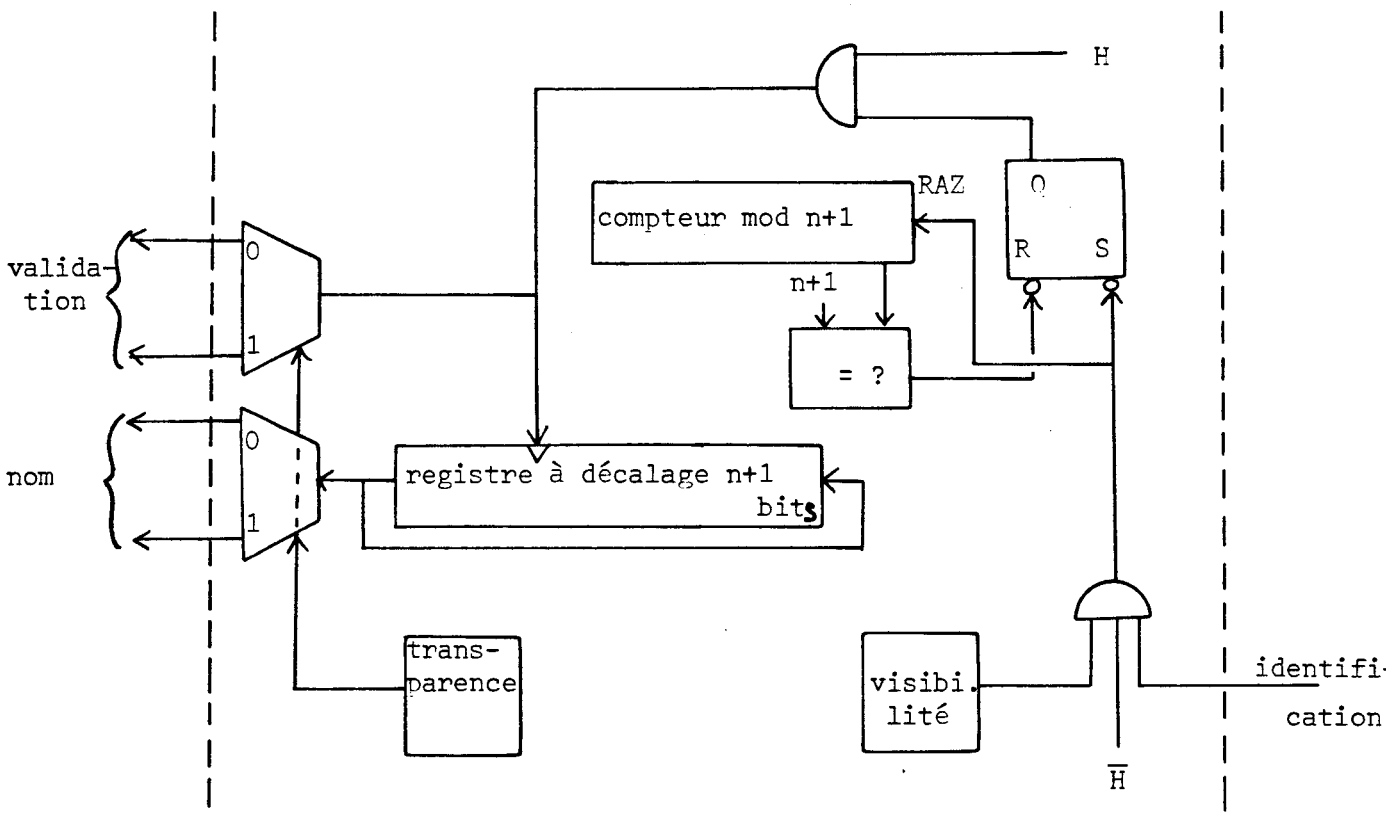
Le microprogramme peut alors utiliser la demande de création validée en envoyant "acceptation" et activer le mécanisme de passage du jeton à son voisin.

A.3. LA PARTIE OPERATIVE D'UN GERANT



A.4. MECANISME D'IDENTIFICATION

n est le nombre de bits d'un identificateur d'élément, le registre à décalage contient alors n+1 bits dont le bit de poids fort est initialement positionné à un :



ANNEXE 2

PERFORMANCES ET FAISABILITE

A.1. INTRODUCTION

A.2. LES PERFORMANCES

2.1. Performances de calcul

2.1.1. Calculs de contour
2.1.2. Calculs de remplissage
2.1.3. Evaluation de visibilité
2.1.4. Traduction des couleurs
2.1.5. Composition de couleurs

2.2. Performances des communications internes

2.2.1. Création
2.2.2. Mise à jour
2.2.3. Lecture
2.2.4. Destruction
2.2.5. Estimation de l'animation possible

A.3. LA FAISABILITE DU PROCESSEUR-ELEMENT

3.1. Nombre de pattes

3.2. Encombrement

3.2.1. Encombrement des processeurs-élément
  3.2.1.1. Encombrement des parties opératives
  3.2.1.2. Encombrement des parties commande
  3.2.1.3. Conclusion
3.2.2. Encombrement du gérant
3.2.3. Encombrement du décideur

3.2.4. Récapitulation

3.3. Conclusion

A.1. INTRODUCTION

L'étude de performances montre dans quelle mesure l'architecture présentée répond aux buts initiaux : suppression de la mémoire d'image et animation en temps réel. L'étude de faisabilité concerne la possibilité d'intégration sur une puce d'un PE.

Au sujet des performances nous distinguons trois parties :

- Les performances de conversion qui rendent compte des calculs effectués par les convertisseurs et l'unification et qui conditionne la suppression de la mémoire d'image.

- Les performances des communications internes, c'est-à-dire entre le PI et les PE, qui conditionnent les possibilités d'animation et de réalisme de l'image.

- Les performances en amont du système décrit qui concernent les précalculs de l'ordinateur hôte et le débit de la liaison entre celui-ci et le système. Elles conditionnent une animation en temps réel ou différé, par exemple en utilisant un disque rapide [MARTINEZ 82]. Aucune hypothèse n'ayant été faire quant à cette partie, ses performances ne seront pas étudiées.

La faisabilité s'intéresse d'une part au nombre de broches nécessaires au composant supportant un PE, d'autre part à la densité du circuit en terme de nombre de portes.

Les évaluations supposent que le support d'affichage est un écran cathodique couleur à balayage non entrelacé fonctionnant à 25 Hertz et permettant une définition informatique de 512*512 pixels.

Les caractéristiques temporelles sont alors les suivantes :

- Cycle de trame de 40 ms.
- Cycle de ligne de 64 microsecondes.
- Temps d'affichage d'un pixel 100 ns environ.

Le cas d'un écran à trame entrelacée sera envisagé en conclusion. Les temps de traversée des circuits et leurs occupations approximatives sont extraits de [TTL 76] et [INTEL 81].

A.2. LES PERFORMANCES

A.2.1. Performances de calcul

Il s'agit de montrer que les différents "étages" du pipe-line de synthèse fonctionnent suffisamment vite :

- t(calcul de contour) <= 64 microsecondes
- t(remplissage/pixel) <= 100 ns
- t(évaluation de visibilité) <= 100 ns
- t(traduction de couleur) <= 100 ns
- t(composition de couleurs) <= 100 ns

où t(op) est le temps nécessaire à la réalisaton de "op".

A.2.1.1. Calcul de contour

Segment de droite

- A chaque début de trame une division entière doit être effectuée. Si elle est séquentielle elle n'utilise qu'une dizaine de cycles d'horloge.

- Pour chaque ligne un pas d'interpolation doit être effectué qui représente la traversée de deux additionneurs, deux multiplexeurs et le chargement d'un registre ce qui semble très faisable en 64 microsecondes.

- Pour chaque ligne la partie commande réévalue le booléen "existence" en utilisant un comparateur d'égalité et un additionneur, ce qui ne pose pas de problème non plus.

facette_polygonale

- La partie commande du calcul de contour possède une "profondeur" maximale réalisée avec une comparaison d'égalité et une addition.

- La partie opérative doit pour chaque ligne évaluer une division entière et un pas d'interpolation A2.

Donc les 64 microsecondes sont suffisantes.

sphère

L'interpolation circulaire de BRESENHAM ne possède pas la propriété des algorithmes précédents qui est de fournir un temps de calcul constant quelque soit le segment de droite envisagé. En effet le temps d'évaluation du bord de la prochaine ligne varie avec le rayon de la sphère et la ligne considérée. Le nombre d'itérations peut cependant être majoré du fait de l'hypothèse selon laquelle une sphère ne peut être découpée et doit donc être complètement incluse dans l'écran. Le rayon de toute sphère est alors inférieur à 256 (taille de l'écran/2) et le nombre d'itérations est majoré par 25 (nombre de points calculés sur la première ligne de la sphère).

Nous avons montré dans le chapitre trois que les algorithmes Pi n'utilisent que deux itérations au maximum par ligne.

Le calcul morphologique et le calcul d'aspect sont distingués comme dans le chapitre cinq. A ces deux calculs s'ajoutent la "mise en forme" et une multiplication.

a) Calcul morphologique

La partie commande a comme profondeur :

égalité # 20 ns
25 itérations de calcul
égalité # 20 ns
1 multiplieur # 20 ns

La partie opérative

1 additionneur/soustracteur	# 50 ns
1 incrémentation	# 40 ns
1 aiguilleur	# 5 ns
1 valeur absolue	# 50 ns
1 comparateur	# 50 ns
1 multiplexeur	# 5 ns
1 incrémenteur/décrémenteur	# 40 ns
total	# 290 ns

Il semble donc nécessaire d'accorder une microseconde par itération, ce qui conduit à un temps maximum de calcul de bord d'environ 25 microsecondes.

b) Calcul d'aspect

partie commande :

égalité ?	# 20 ns
égalité ?	# 20 ns
addition, deux itérations	
1 additionneur/soustracteur	# 50 ns
1 multiplexeur	# 5 ns

partie opérative :

addition/soustraction	# 50 ns
comparaison	# 50 ns
total	# 300 ns

Une microseconde est à nouveau suffisante pour une itération.

c) Mise en forme et multiplication

La mise en forme nécessite deux additionneurs et un multiplieur. Il reste une trentaine de microsecondes pour effectuer ces opérations ce qui est suffisant.

Les calculs de contour ne posent pas de problème du point de vue des performances. Il est alors envisageable de compacter les parties opératives en sérialisant les calculs.

A.2.1.2. Calculs de remplissage

segment de droite

partie commande :

égalité ?	#	20 ns
comparaison	#	50 ns

partie opérative :

2 multiplieurs	#	10 ns
1 additionneur	#	40 ns

Les calculs de commande de superposant aux calculs opératifs, la vitesse est suffisante.

facette polygone

Les résultats sont identiques à ceux du segment de droite.

sphère

Morphologie et aspect.

partie commande :

égalité # 20 ns
2 itérations

partie opérative :

1 addition/soustraction	# 50 ns	200 ns
1 comparaison	# 50 ns	

mise en forme :

1 addition # 40 ns

Il paraît nécessaire ici d'effectuer un pipe-line sur les itérations.

A.2.1.3. Evaluation de visibilité

Nous avons montré au Chapitre V qu'il est possible de "pipe-liner" à la fois les décideurs et les comparateurs, au dépend de l'encombrement des décideurs. Il semble donc possible de réaliser les vitesses de comparaison désirées (environ 100 ns par pixel). Il faut trouver un compromis entre le débit de l'affichage et l'encombrement des décideurs.

Rappelons qu'un opérateur d'unification muni de deux décideurs a été câblé et fonctionne jusqu'à 10 mégahertz, il effectue une sélection sur une valeur codée sur 8 bits en deux cycles de comparaison et ceci sans pipe.

V.2.1.4. Traduction de couleur

On trouve, par exemple, dans le catalogue Intel des mémoires PROM et RAM satisfaisant aux besoins de vitesse d'accès (PROM 2K*8 : 35 ns, RAM 4K*1 : 55 ns) [INTEL 81].

V.2.1.5. Composition de couleurs

Ces opérations peuvent être agencées en pipe-line. En ce qui concerne les multiplications elles peuvent être réalisées avec des circuits spécialisés ou des ROM rapides.

A.2.2. Performances des communications internes

Après une évaluation du temps de communication nécessaire pour chaque commande et pour chaque type d'élément nous tentons en fixant certains paramètres d'en déduire quels sont les types d'animation possibles en temps réel. On distinguera les trois sortes d'animations suivantes :

- Animation d'objets dans l'image avec fond et point de vue fixes.

- Animation de l'éclairement des éléments avec géométrie et point de vue fixes.

- Animation du point de vue.

Il faut commencer par nommer les paramètres intervenant dans les volumes d'information à transmettre. Nous supposons pour simplifier que les longueurs d'un champ d'identificateur et d'un code de commande sont inférieurs à la largeur du bus de communication interne.

Les paramètres sont :

- i : nombre de champs d'un identificateur d'élément.
- k : nombre d'attributs d'aspect constants.
- p : nombre d'attributs d'aspect variables.
- e_D : nombre de PE de type segment de droite.
- e_F : nombre de PE de type facette polygonale.
- e_S : nombre de PE de type sphère.
- n : nombre maximum d'arêtes par bord de facette.
- t : temps qui s'écoule entre l'envoi d'un paramètre par le PI et le rangement effectif de celui-ci par le gérant concerné. Nous supposons que t est constant quelque soit le paramètre envoyé. t dépend du nombre de couches d'amplification et de la période de l'horloge des PE.
- h : période de l'horloge d'un PE, h est égal à 100 ns ce qui correspond à la durée d'affichage d'un pixel.

- S : temps de traversée d'un PE pour l'information "jeton" dans le cas d'un accès par type. En fait S = h dans le cas d'un mécanisme synchrone.

- t_n : temps nécessaire à un adressage par nom :

$t_n = t*(2+i)$

où 2t correspond aux commandes asynchrones début et fin et i*t au temps de présentation de l'identificateur. Cette valeur est maximum puisqu'il est possible de désigner un objet en n'envoyant que le premier champs du nom par exemple.

- t_t : temps nécessaire à un adressage par type :

$t_t = (e_D)*S + t$ (si $e_D = e_F = e_S$)

Cette évaluation ne rend pas compte des gains de temps obtenus avec la technique du jeton (cf. IV.3.1.2), en particulier lors de la création initiale où tous les PE sont libres, le temps de sélection se réduit à trois t pour chaque création successive.

Les temps de communication des différentes commandes sont donnés dans la suite.

A.2.2.1. Création

$t_{\text{segment de droite}} = (17 + 8p + k + i)*t + t_n + t_t$

$t_{\text{facette}} = (9 + 6n*(3+p) + k + i)*t + t_n + t_t$

$t_{\text{sphère}} = (6 + 7p + k + i)*t + t_n + t_t$

A.2.2.2. Mise à jour

Soit q le nombre de paramètres à modifier.

$t_{\text{segment de droite}} = 2t + 2qt + t + t_n$

avec $1 \leq q \leq 15 + 8p + k$

$$t_{facette} = 2t + 2qt + t + t_n$$

avec $1 \leq q \leq 7 + 6n*(3+p) + k$

$$t_{sphère} = 2t + 2qt + t + t_n$$

avec $1 \leq q \leq 4 + 7p + k$

A.2.2.3. Lecture

Le temps est pour tous les éléments de $(2q+1)\ t + t_n$

avec : $1 \leq q \leq 15 + 8p + k$ segments de droite
$1 \leq q \leq 7 + 6n*(3+p) + k$ facette
$1 \leq q \leq 4 + 7p + k$ sphère

A.2.2.4. Destruction

Le temps de destruction est le même pour tous les types d'élément.

$$4t \leq t_{destruction} \leq (3+i)*t$$

A.2.2.5. Estimation de l'animation possible

Nous fixons les paramètres suivants :

- 40 ms sont disponibles pour effectuer un ensemble de commandes en temps réel.
- t = 0.2 microsecondes.
- n = 2.
- p = 2 par exemple intensité et transparence.
- k = 2 par exemple teinte et saturation.
- i = 3.
- S = 100 ns.
- $e_D = e_F = e_S = 1000$.

Notons que l'animation d'objets et l'animation de point de vue nécessitent les mêmes commandes, en particulier la commande de mise à jour devra à chaque mouvement remplacer tous les attributs autres que l'identificateur. En revanche une animation de l'éclairement utilise uniquement la mise à jour et ne modifie que les attributs d'aspect. Nous donnons deux tableaux, le premier concerne l'animation d'objet ou de point de vue, le second l'animation d'éclairement.

Ces tableaux présentent le nombre de commandes exécutables en 40 ms sur chaque type d'élément, et leur temps d'exécution.

animation géométrique

	création	mise à jour	lecture	destruction
segment de droite	660 60	2640 15	2640 25	30.000 et plus en utilisant la structuration en objets 1.2
facette	570 70	1320 30	1320 30	
sphère	660 60	4000 10	4000 10	

Soulignons que les temps de création ne rendent pas compte des gains apportés par la technique du jeton.

animation de l'éclairement

	mise à jour
segment de droite	4000 9
facette	4000 10
sphère	4000

Pour les sphères il est possible de ne modifier que le point d'intensité maximum sans changer les autres paramètres d'aspect.

A.3. LA FAISABILITE DU PE

Ce paragraphe doit conclure quant à la possibilité d'intégrer un PE sur une seule puce. Pour cela nous évaluons d'une part le nombre de "pattes" nécessaires aux communications, d'autre part l'encombrement d'un PE en nombre de portes.

Le nombre de pattes est la somme de quatre termes :

- alimentation(s) et horloge(s),
- communications internes (PI <--> PE),
- les bus du décideurs,
- les bus de sortie à destination de l'affichage.

L'encombrement est évalué pour chaque partie opérative et chaque partie commande. Le nombre de portes des unités fonctionnelles (registres, additionneurs,...) est extrait des schémas logiques proposés par Texas Instrument [TTL 76], il ne tient donc pas compte des possibilités de réduction offertes par une intégration directe. Les hypothèses faites sur les paramètres sont les mêmes que précédemment. Ajoutons que les PE sont numérotés de 0 à 4095 (12 bits).

A.3.1. Nombre de pattes

Cette étude est commune aux trois types de PE :

- L'alimentation et l'horloge utilisent au moins trois pattes.

- Pour les communications internes il semble raisonnable de réserver dix fils au bus de données. Les commandes demandent treize fils, en incluant les renvois d'identificateurs transparent et opaque sur identification. Sans multiplexage cette partie comporte donc 23 pattes.

- La largeur des bus du décideur dépend de la politique de comparaison adoptée. Le schéma de GEMBALLA nécessite le minimum de fils (42) [GEMBALLA 82]. En revanche cette solution à 21 étages de pipe-line (9 + 12) présente le désavantage de doubler le nombre de portes par rapport à un découpage des valeurs (profondeur et priorité) en tranches de trois bits par exemple. Un tel découpage implique une largeur du bus de :

$$\frac{9+12}{3} * (2^3 - 1 + 3) = 70 \text{ fils.}$$

- En supposant que chacune des quatre informations (T, I, S, tr) destinées à l'affichage est codée sur huit bits, le débit du bus doit alors être de 320 mégabits/seconde. On peut envisager un bus de largeur 32 ou réduire cette valeur en réduisant la précision des paramètres S et tr (par exemple à 5 bits). La largeur du bus devient alors égale à 26.

Le tableau suivant résume les différentes possibilités :

	Schéma de GEMBALLA (tranches de un bit)	Tranches de trois bits
Réduction de tr et S à 5 bits	94	122
T, I, S tr sont tous codés sur 8 bits	100	128

Des boîtiers munis de broches sur leurs quatre côtés existent et offrent les nombres de pattes indiqués dans le tableau.

A.3.2. Encombrement

L'étude porte sur le convertisseur, le gérant et le décideur.

A.3.2.1. Encombrement des processeurs-élément

A.3.2.1.1. Encombrement des parties opératives

Les évaluations présentées se déduisent des schémas proposés dans les Chapitres IV et V sous les hypothèses précédentes. Nous présentons les résultats bruts obtenus en nombre de portes.

- Les "interpolateurs" :

A2	Circulaire	P2	P3
550	1226	1253	758

- Les diviseurs utilisés pour les calculs de facette, ils incorporent les calculs de reste de division entière et de complément du reste et sont au nombre de trois :

  1 diviseur # 500 portes

- Les multiplieurs qui sont utilisés dans les calculs d'aspect des sphères sont au nombre de quatre :

  1 multiplieur (9 bits * 9 bits) # 190 portes

- Les couches de mise en forme :

  facette : 4 soustractions # 400 portes,
  sphère : 2 registres, 10 additionneurs et deux soustracteurs
  # 1628 portes

- Les tampons du pipe-line :

segment de droite	facette	sphère
1944	2862	2214

A.3.2.1.2. Encombrement des parties commandes

L'étude porte sur chacun des trois types d'élément pour le calcul de contour et le calcul de remplissage. Les algorithmes réalisés par ces parties commande étant relativement simples nous proposons de les câbler. Seule l'étude de la première commande est détaillée :

Segment de droite

Calcul de contour : l'examen de l'algorithme employé fait apparaître les registres nécessaires et les opérateurs employés :

Bascule	"existence"	6 portes
Registres	"Y" "Xlim"	2*9*6 portes
Registre à autoincrément	"Yc"	90 portes
Additionneurs dix bits		2*100 portes
Comparateur d'égalité 9 bits		46 portes
Total		450 portes

Calcul de remplissage # 500 portes.

Facette

Calcul de contour : il inclut le registre 9 bits à décalage qui commande les multiplications :

460 portes.

Calcul de remplissage 500 portes.

Sphère

Contour géométrique # 350 portes (en 1 exemplaire)

Contour d'aspect # 720 portes (en 2 exemplaires)

Remplissage # 780 portes (en 3 exemplaires)

A.3.2.1.3. Conclusion

Le tableau résume les résultats obtenus pour chacun des convertisseurs:

segment de droite	facette	sphère
7000	11852	17460

Ces valeurs sont élevées mais peuvent être réduites en faisant les remarques suivantes :

- La définition des paramètres d'aspect variables est supposée être la même que celle des paramètres géométriques, on peut espérer réduire notablement les encombrements en réduisant leur définition.

- Dans une première approche il est raisonnable de supprimer un des paramètres d'aspect variables qui est la transparence et de ne laisser varier que l'intensité.

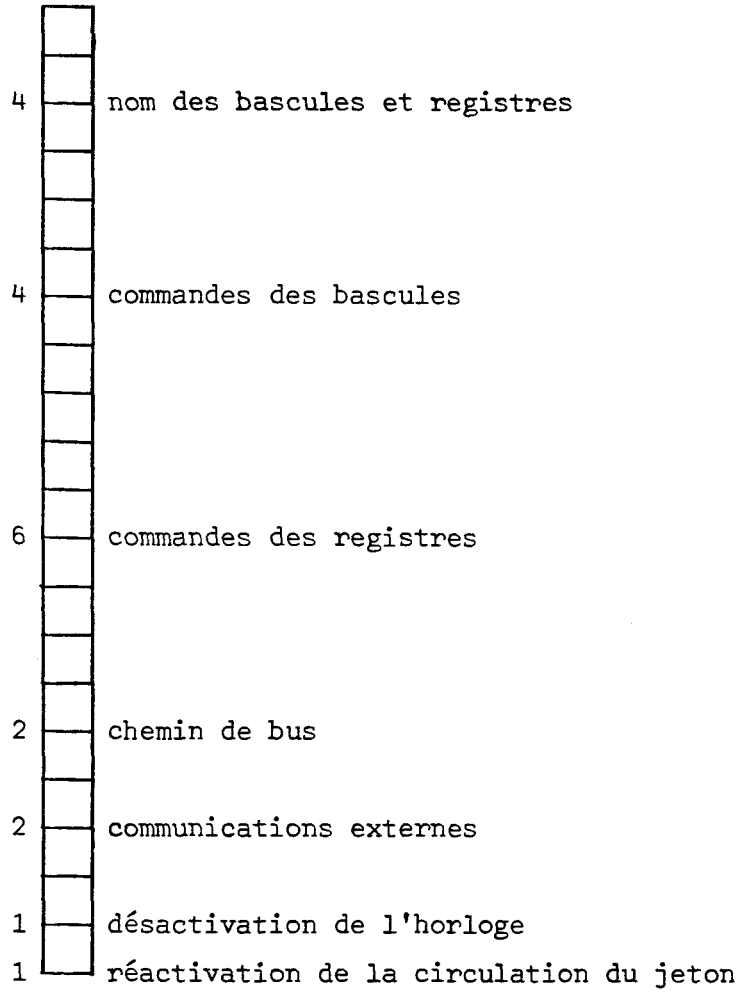
En réduisant la définition de l'intensité à huit bits et en supposant la transparence constante on obtient le nouveau tableau approximatif suivant :

Segment de droite	facette	sphère
5300	8430	10200

A.3.2.2. Encombrement du gérant

Les gérants des trois types de PE ne se différencient que par leur nombre de registres contenant les attributs des éléments. L'étude évalue la taille du microprogramme, l'encombrement de la partie opérative, et l'occupation des mécanismes câblés.

Le nombre de microinstructions formant les microprogrammes s'élève à environ 25. Un format horizontal est adopté pour leur codage qui demande 20 bits :



On majore alors la taille de mémoire morte du microprogramme par 32*20 positions.

Cette mémoire est munie d'un microcompteur de 5 bits et d'un décodeur (5 → 32).

Si on majore par 256 le nombre de registres formant la sous-liste de visualisation du gérant, le registre I est un compteur à huit bits. Dans le cas des sphères et des segments de droite cinq bits suffiront à I. Dans le premier cas le nombre de portes du décodeur s'élève à 450, dans le second à 45.

Les mécanismes annexes de réveil, de sélection, de sortie série peuvent être estimés à 400 portes.

La taille des sous-listes de visualisation dépend des types :

- segment de droite # 2070 portes,
- facette # 4360 portes,
- sphère # 1350 portes.

Les tailles des gérants des différents types sont résumées ci-dessous :

segment de droite	facette	sphère
2800	5500	2100

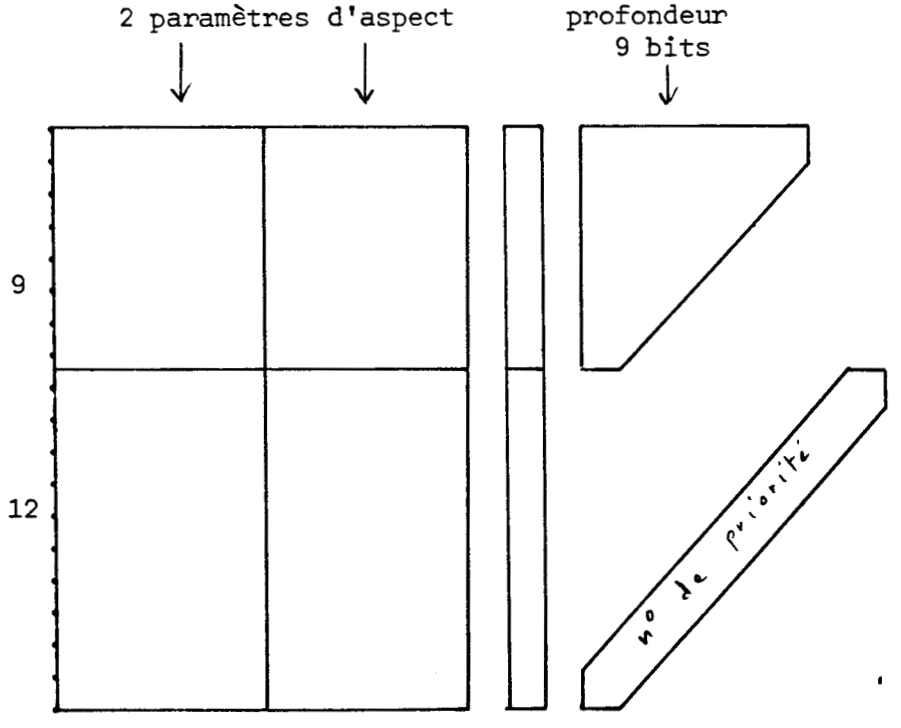
+ 640 positions de mémoire morte.

Les mêmes réductions que dans le paragraphe précédent amènent les résultats suivants :

segment de droite	facette	sphère
2600	4850	1670

A.3.2.3. Encombrement du décideur

La solution parallèle par tranches de 1 bit (GEMBALLA) demande 21 étages de pipe-line :



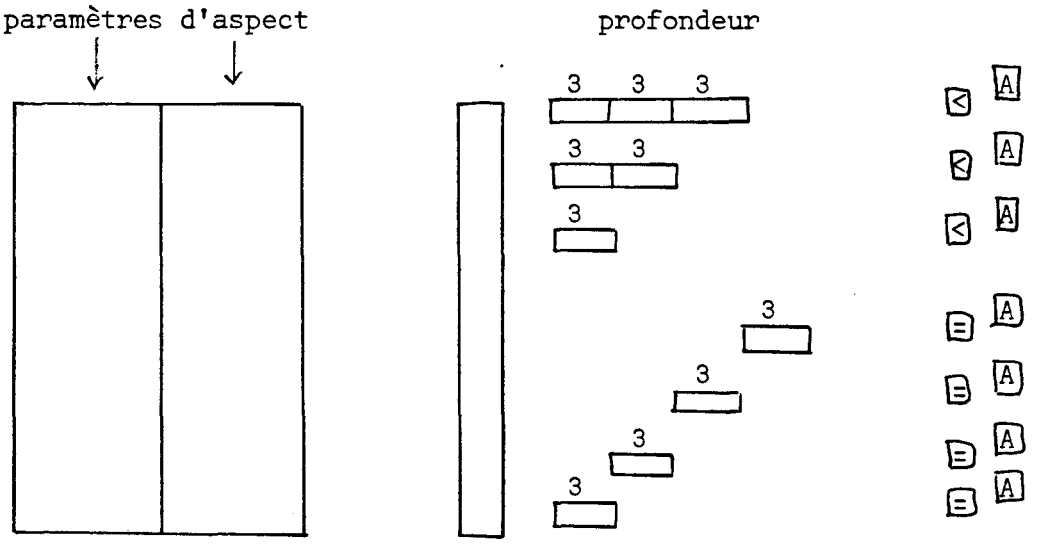
en utilisant 8 bits par paramètre d'aspect on obtient en terme d'occupation par des registres le nombre de portes suivant :

$$21 * (2 * 8 * 6 + 6) + 6 * (9 * 5 + 12)$$

c'est-à-dire 2484 portes.

La logique de décision ajoute une centaine de portes, d'où le nombre total de portes 2600.

La solution parallèle par tranches de trois bits suppose cette fois 7 étages de pipe-line mais une logique de décision plus complexe :



La mémorisation nécessite :

$$6 * (7 * (2 * 8 + 1) + 3 * 2 * 3 + 12) \# 900 \text{ portes}$$

La partie combinatoire comporte :

7 décodeurs 3 → 8 ('A') à 16 portes,

3 comparateurs d'infériorité ('<') à 31 portes.

4 comparateurs d'égalité ('=') à 16 portes.

elle comporte donc 269 portes et avec la mémorisation # 1200 portes.

En supposant la transparence constante, on obtient respectivement pour les deux méthodes 1600 et 900 portes.

A.3.2.4. Récapitulation

Le tableau ci-dessous indique les encombrements en nombres de portes de chacun des types de PE et dans chaque cas de réduction :

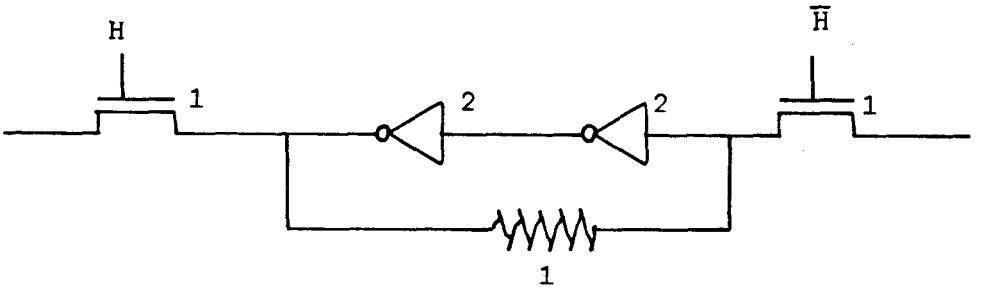
	Segment de droite	facette	sphère	nombre de broches
- sans réduction - décideur parallèle tranches de 1 bit	12.400	20.000	22.200	100
- sans réduction - tranches de 3 bits	11.000	18.600	20.800	128
- transparence constante - tranches de 1 bit	9.500	14.900	13.500	100
- transparence constante - tranches de 3 bits	8.800	14.200	12.800	128

A.3.3. Conclusion

Deux remarques s'imposent :

- L'encombrement des registres pour chaque de PE représente une grande partie de l'encombrement global.

- La cellule de base (1 bit) de tous registres a été évaluée à six portes, or par exemple en technologie NMOS une porte représente de trois à quatre transistors ce qui donne entre 18 et 24 transistors par registre de 1 bit. Dans la même technologie une cellule de mémoire pourrait être réalisée de la manière suivante :



C'est-à-dire avec sept transistors, ce qui est équivalent à 2 ou 3 portes logiques. D'autre part certains points de mémorisation peuvent être dynamiques réduisant encore l'encombrement.

Les estimations sont donc largement surévaluées.

Certaines limitations peuvent réduire les exigences précédentes :

- Utiliser un écran de 256 * 256 pixels.

- Sérialiser les parties dont la vitesse n'est pas critique, en particulier les calculs de contour.

- Supprimer les diviseurs des PE de type facette en supposant celle-ci planes.

Ajoutons que les performances de calcul sont évaluées sur la base d'un rythme de rafraîchissement de l'écran de 25 hz. Avec l'utilisation d'écrans fonctionnant à 50 Hz il faut utiliser des technologies plus rapides ou envisager une réduction de la définition.

Au niveau des composants on peut espérer d'ici 1985 l'apparition de réseaux prédiffusés offrant 20.000 portes bien qu'il convienne de rester prudent à ce sujet, la complexité de conception d'un tel réseau augmentant de façon exponentielle avec le nombre de portes.

En revanche les possibilités offertes par les "circuits à la demande" à haute, voire très haute intégration semblent répondre aux besoins précédemment décrits avec cependant un coût de développement beaucoup plus élevé que pour les réseaux prédiffusés.

Le problème des écrans entrelacés peut être facilement résolu en itérant deux fois le processus de calcul de contour sur chaque ligne.

Enfin on a pu remarquer que les parties opératives des calculs de contour et de remplissage, au moins dans le cas des segments de droite et des facettes polygonales, sont répétés à plusieurs exemplaires tous identiques. Les méthodes de conception des circuits intégrés évoluant vers une structuration systématique des plans de masse, la remarque précédente milite en faveur d'une intégration [MEAD 79].

BIBLIOGRAPHIE

A. OUVRAGES DE BASE

[CEI 82], [FOLEY 82], [GARDAN 83], [MEAD 79], [MORVAN 76], [NEWMAN 79].

B. L'ALIASSAGE

[CROW 77], [GUPTA 81], [WHITTEL 83].

C. LES REALISATIONS

[CLARK 80], [CLARK 82], [CORDONNIER 81], [DURIF 82], [FERREIRA 81], [GEIB 83], [GEMBALLA 82], [GRUIA 81], [GUPTA 81], [LERAY 78], [LERAY ?], [PILLER 80], [REDJIMI 82], [SPROULL 83], [WEINBERG 81].

D. LES MODELES

[ACQUAH 82], [CARLBOM 80], [CARLBOM 82], [KILGOUR 81], [LUCAS 77], [MARTINEZ 82].

E. LES LOGICIELS ET REPRESENTATIONS DE DONNEES

[ALLAL 82], [BOULLE 80], [BRADIER 81], [CATMULL 75], [FREEMAN 61], [GOURAUD 71], [GRAVE 80], [KHESSAIRI ?], [PARENT 82], [PELERIN 82], [WARNOCK 69].

F. CALCUL INCREMENTAL

[BRESENHAM 65], [BRESENHAM 77], [CARREZ 77], [CEI 82], [EARNSHAW 77], [LOCEFF 80], [PITTEWAY 67], [SPROULL 82], [YASUHITO 79].

G. L'ECLAIREMENT

[BLINN ?], [GOURAND 71], [PHONG 75], [WHITTED 80].

H. LA COULEUR ET L'ASPECT

[CONRAC 80], [GERRITSEN 75], [JOBLOVE 79], [KAY 79], [MERIAUX 82], [NEWELL 72], [OLEJNIK 82], [SMITH 78].

I. DIVERS

[INTEL 81], [TTL 76].

J. ACQUAH, J. FOLEY, J. SILBERT, P. WERNNER "*A conceptual model of raster graphics systems*". Computer Graphics, Vol. 16, n° 3, Juillet 1982

J. ALLAL. "*Etude d'algorithmes de traitement et de manipulation d'images pour un multiprocesseur associatif parallèle* (MAP)". Mémoire de D.E.A. Lille 1982.

J.F. BLINN. "*Models of light reflection for computer synthesized pictures*".

P. BOULLE. "*Etude et réalisation d'algorithmes pour la visualisation de scènes composées de facettes planes*". Thèse de Docteur Ingénieur. Grenoble, Septembre 1980.

A. BRADIER. "*Eléments pour la représentation de données graphiques dans un contexte interactif*". Mémoire de D.E.A. USTL 1981.

J.E. BRESENHAM. "*Algorithm for computer control of a digital plotter*". IBM System Journal, Vol. 4, n° 1, 1965.

J.E. BRESENHAM. "*A linear algorithm for incremental digital display of circular arcs*". Communication of the ACM, 20(2), Février 1977.

I.B. CARLBOM. "*System architecture for high performance vector graphics*". Ph. D. Thesis, Dept. of Computer Science, Brown University, Providence, R.I., 1980.

I.B. CARLBOM, J. MICHENER. "*Quantitative analysis of vector graphics system performance*". ACM transactions on graphics, Vol. 2, n° 1, Janvier 1983.

C. CARREZ. "*Control of an incremental plotter by a microprocessor*". Publication interne n° 94, USTL, Septembre 1977.

E.E. CATMULL. "*Computer display of curved surfaces*". Proc. IEEE Conf. on Computer Graphics, Pattern Recognition and Data Structure, Mai 1975.

CEA-EDF-INRIA. "*La réalisation des logiciels graphiques interactifs*". Collection de la direction des études et recherches d'Electricité de France. Editions Eyrolles, 1982.

J.H. CLARK. "*A VLSI Geometry processor for graphics*". Computer, Juillet 1980.

J.H. CLARK. "*The Geometry engine : a VLSI Geometry system for graphics*". Computer Graphics, Vol. 16, n° 3, Juillet 1982.

CONRAC. "*Raster graphics handboock*". 1980.

V. CORDONNIER, MOUSSU. "*The MAP project : an associative processor for speech processing*". Publication interne. USTL. 1981.

F.C. CROW. "*The aliasing problem in computer generated shaded images*". Communication of the ACM, Vol. 20, n° 11, Novembre 1977.

P. DURIF, M. MERIAUX. "*Une architecture pour la synthèse d'images sans mémoire de trame*". Actes du Congrès AFCET, 1982.

R.A. EARNSHAW. "*Line generation for incremental and raster device*". Computer Graphics Siggraph ACM, Vol. 11, n° 2, Eté 1977.

N.F. FERREIRA. "*Conception et réalisation d'un système interactif pour la synthèse d'images réalistes : Hélios*". Thèse de Docteur Ingénieur, Grenoble, 1981.

J. FOLEY, A. VAN DAM. "*Fundamentals of interactive Computer Graphics*". Addison Wesley, 1982.

H. FREEMAN. "*On the encoding of arbitrary geometric configurations*". ITEC, Juin 1961.

Y. GARDAN, M. LUCAS. "*Techniques graphiques interactives et CAO*". Hermes, 1983.

J.M. GEIB, E. DELATTRE. "*Partage réparti de composants matériels dans le cadre d'une machine à processeurs multiples*". Publication interne USTL, à paraître.

R. GEMBALLA R. LINDNER. *"The multiple-write bus technique"*. IEEE Computer Graphics, Septembre 1982.

F. GERRITSEN. *"Présence de la couleur"*. Dessain et Tolre, Paris, 1975.

H. GOURAUD. *"Continuous shading of curved surfaces"*. IEEE Transactions on Computers, C-20(6), Juin 1971.

M. GRAVE. *"Etude d'un noyau de système de synthèse d'images . Application à la visualisation de scènes tridimensionnelles"*. Thèse de Docteur Ingénieur, USTL, 1980.

R. GRUIA CATALIN, K. TAKAYUKI. *"VLSI perspective of real time hidden-surface elimination"*. Computer aided Design, Vol. 13, n° 2, Mars 1981.

S. GUPTA, R.F. SPROULL. *"Filtering Edges for gray-scale displays"*. Computer graphics, Vol. 15, n° 3, Août 1981.

S. GUPTA, R.F. SPROULL, I.E. SUTHERLAND. *"A VLSI Architecture for updating raster-scan displays"*. Computer Graphics, Vol. 15, n° 3, pp. 71-78, Août 1981.

INTEL. *"Component Data Catalog"*. Janvier 1981.

G.H. JOBLOVE, D. GREENBERG. *"Color Spaces for computer graphics"*. SIGGRAPH, 1979.

K.D.S. Greenberg. *"Transparency for computer synthetized images"*. Siggraph, p. 158, 1979.

M. KHESSAIRI, A.A. JERRAYA. *"Lucie, Langage Universitaire de Conception de circuits Intégrés pour l'Enseignement"*. ENS IMAG, Grenoble.

A.C. KILGOUR. *"A hierarchical Model of a graphic system"*. Computer Graphics, Avril 1981.

P. LERAY. *"Génération en temps réel d'images synthétiques tridimentionnelles"*. Journée d'Etude AFCET, Avril 1978.

P. LERAY. *"Réalisation d'un système de génération synthétique d'images en temps réel"*.

M. LOCEFF. *"The line"*. IEEE Computer, Juin 1980.

M. LUCAS. *"Contribution à l'étude des techniques de communication graphique avec un ordinateur - Eléments de base des logiciels graphiques interactifs"*. Thèse d'Etat, Grenoble, Décembre 1977.

F. MARTINEZ. *"Vers une approche systématique de la synthèse d'image. Aspects logiciel et matériel"*. Thèse d'Etat, Grenoble, Novembre 1982.

C. MEAD, L. CONWAY. *"Introduction to VLSI System"*. Addison Wesley, 1979. Traduction française : *"Introduction aux systèmes VLSI"*. Inter System design, 1982.

M. MERIAUX. *"La programmation des couleurs en informatique graphique"*. Publication interne du L.A. 369; Lille, 1982.

P. MORVAN, M. LUCAS. *"Introduction à l'infographie interactive"*. Larousse Université, 1976.

M.E. NEWELL, R.G. NEWELL, T.L. SANCHA. *"A solution to the hidden surface problem"*. ACM National Meeting, 1972.

NEWMAN, SPROULL. *"Principles of interactive computer graphics"*. Mac Graw-Hill. 2nd Edition, 1979.

R. OLEJNIK. *"Réalisme dans les images générées par ordinateur"*. Mémoire de D.E.A. USTL, 1982.

C. PARENT. *"Etude et spécification d'un éditeur d'images. Application à la bureautique"*. Thèse de 3ème Cycle, USTL, Octobre 1982.

M. PELERIN. *"Conception et réalisation d'outils de synthèse d'images nécessaires à l'élaboration d'un logiciel graphique sur MAP"*. Mémoire de D.E.A. Lille 1, 1982.

BUI-TUONG, PHONG. *"Illumination for Computer generated Pictures"*. Communications of the ACM 18(6), Juin 1875.

E. PILLER, H. WIDNER. *"Real-Time Raster Scan Unit with Improved picture Quality"*. Computer Graphics, Vol. 14, n° 1, 2, Juillet 1980.

M. PITTEWAY. "*Algorithm for drawing Ellipses or hyperbolae with digital plotter*". Computer Journal 10(3), Novembre 1967.

M. REDJIMI. "*MAP*". Mémoire de D.E.A., USTL, 1982.

A.R. SMITH. "*Color Gamut Transform pairs*". Siggraph 1978. du 21 au 25 Août 1978, pp. 12-19.

R.F. SPROULL. "*Using Program transformations to Derive line-drawing Algorithms*". ACM Transactions on Graphics. Vol. 1, n° 4, Octobre 1982.

R.F. SPROULL, I.E. SUTHERLAND, A. THOMPSON, S. GUPTA, C. MINTER. "*The 8 by 8 display*". ACM Transactions on Graphics, Vol. 2, n° 1, Janvier 1983.

TEXAS INSTRUMENT. "*The TTL Data book for design engineers*". Seconde édition, 1976.

J.E. WARNOCK. "*The hidden line problem and the use of halftone display*". in Pertinent Concept in Computer Graphics. University of Illinois press, 1969.

R. WEINBERG. "*Parallel Processing Image Synthesis and anti-aliasing*". Computer Graphics, Vol. 15, n° 3, Août 1981.

T. WHITTED. "*An Improved Ihlumination Model for shaded display*". Graphics and Image Processing, ACM, Vol. 23, n° 6, Juin 1980.

T. WHITTED. "*Anti-aliased line drawing Using Brush extension*". Computer Graphics, Vol. 17, n° 3, Juillet 1983.

YASUHITO SUENAGA, TAKAHIKO KAMAE, TOMONORI KOBAYASHI. "A *high-speed algorithm for the generation of straight lines and circular arcs*". IEEE Transactions on Computer, Vol. C-28, n° 10, Octobre 1979.

RESUME

La production ou synthèse d'image par ordinateur conduit à un très grand nombre de calculs. Pour réduire les temps de calcul les concepteurs de machine utilisent le caractère répétitif des opérations effectuées pour y introduire le parallélisme. On distingue trois grandes causes de parallélisme : le découpage du support d'affichage, le découpage fonctionnel, le découpage par objet. C'est à cette dernière que se rapporte la machine présentée. Elle ne possède pas de mémoire de trame et doit permettre la création et l'animation d'image réaliste en temps réel. L'univers est décomposé en éléments et chacun d'entre-eux est affecté à un processeur-élément.

Une étude préliminaire décrit plusieurs algorithmes de calculs incrémentaux et compare leur performance. La suite propose des mécanismes d'adressage des processeurs éléments, décrit leur fonctionnement interne ainsi qu'un opérateur d'unification de leurs sorties permettant l'affichage final. L'intégration des processeurs-éléments à l'aide de circuits à haut degré d'intégration amène enfin à une étude des performances et de faisabilité.

MOTS CLES : Synthèse d'image, parallélisme, animation, temps réel, processeur élément, calcul incrémental.

Image synthesis, parallelism, animation, real time, element processor, incremental computing.