Analyse et implémentation du logiciel pilote d'un synthétiseur d'images

Marie-Thérèse Sarrasin

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE AGREE
DE GRENOBLE (C.U.E.F.A)

MEMOIRE

présenté en vue d'obtenir

le diplôme d'ingénieur

en

informatique

par

Marie-Thérèse SARRASIN

ANALYSE ET IMPLÉMENTATION

DU LOGICIEL PILOTE D'UN SYNTHÉTISEUR D'IMAGES

SOUTENU LE : Octobre 1982

JURY

Président : Professeurs L. BOLLIET &

Membres : MM. Y. GARDAN
F. MARTINEZ
J. MERMET

Le travail présenté dans ce mémoire a été réalisé dans le cadre d'un congé-formation d'un an, dans l'équipe "Communication Graphique et Méthodologie de la C.A.O." du Laboratoire IMAG.

Je tiens à remercier chaleureusement :

Monsieur RANCHIN du Conservatoire National des Arts et Métiers qui m'a fait l'honneur d'accepter mon dossier de thèse,

Monsieur le Professeur L. BOLLIET, pour l'aide qu'il m'a apportée durant cette année et pour l'honneur qu'il me fait de présider le jury,

Monsieur F. MARTINEZ, responsable de l'Equipe Communication Graphique du Laboratoire IMAG, qui m'a accueillie dans son équipe et a dirigé mes travaux avec efficacité et compétence. Qu'il soit assuré de ma gratitude pour avoir passé de nombreuses heures en explications et discussions diverses,

Monsieur J. MERMET, directeur de recherche et responsable de l'équipe Méthodologie de la C.A.O., et Monsieur Y. GARDAN, Directeur de l'A.F. MICADO, pour avoir accepté de siéger à ce jury,

Les personnes qui se sont occupées de la frappe et du tirage de ce mémoire.

Que toutes les personnes de l'équipe C.A.O., notamment P. BOULLE, P. VENNIER, G. VITRY, G. BUISSON, J. CARRY, de même que J.P. SERPAGGI, technicien au Laboratoire de Micro Informatique, et F. RENZETTI, bibliothécaire, qui m'ont apporté leur aide et témoigné leur amitié, soient vivement remerciées.

Je n'oublierai pas non plus Madame C. CHALAND, secrétaire de M. BOLLIET, qui m'a levé tous les obstacles administratifs pour que cette thèse ait lieu dans de bonnes conditions. Je rend particulièrement hommage à son dévouement et sa compétence.

A un merveilleux petit oiseau ...

S O M M A I R E

INTRODUCTION

Une image nous permet d'évoquer, beaucoup plus concrètement que des phrases ou des chiffres, la représentation d'objets ou de données. La communication par la perception visuelle d'images, est donc plus agréable et plus efficace.

Il est naturel que de nombreux efforts soient réalisés pour permettre une liaison homme - machine par l'intermédiaire de dessins et d'images, et que des travaux spécifiques essaient d'accroître la quantité d'informations véhiculées par ces images.

Le graphisme est largement employé dans des processus de création, et l'ordinateur est devenu un allié qui permet une conception interactive de nouveaux modèles. Les industries automobile, aéronautique et navale, l'urbanisme, l'architecture ou le génie-civil, bénéficient actuellement largement des techniques graphiques nouvelles, permettant d'alléger le travail des concepteurs, de gagner du temps et de la précision. La création artistique commence aussi à utiliser depuis quelques années, cet ordinateur qui a appris à "dessiner" et à "peindre".

Pour qu'un système apparaisse comme un outil très intéressant, trois facteurs sont à considérer :

- *la facilité de la description* (utilisation de tablettes pour le tracé d'objets, d'opérateurs pour établir les relations entre ces objets, ...),
- *le temps nécessaire à la synthèse* (les utilisateurs souhaitent le temps réel, mais l'élimination des parties cachées et l'affichage des faces visibles requièrent des temps de calcul importants),
- *le degré d'interactivité*, c'est-à-dire les possibilités données à l'utilisateur pour modifier ce qu'il a visualisé.

Dans l'Equipe Graphique du Laboratoire d'Informatique et de Mathématiques Appliquées de Grenoble (IMAG), des travaux ont été développés, dans le cadre de deux projets d'études, pour réaliser un système complet permettant la synthèse d'images réalistes.

Le premier projet : CLOVIS, a décrit les concepts et les primitives de base d'un logiciel mettant en oeuvre une base de données graphiques structurée et banalisée.

Le second a permis la réalisation d'un prototype de terminal : HELIOS, qui effectue, par logique câblée, des opérations de synthèse d'images en temps réel. L'axe privilégié a été la séparation des informations graphiques, si possible jusque dans le matériel, pour permettre une forte interaction.

Les travaux rapportés dans ce mémoire concernent l'étude et la réalisation d'un logiciel INTERFACE entre le logiciel principal utilisant la base de données et le prototype HELIOS. Ce logiciel pilote est destiné à :

- gérer une communication par ligne, avec un calculateur principal contenant l'application graphique et le logiciel principal,
- effectuer le maximum d'opérations de synthèse, non intégrées au terminal,
- piloter le terminal HELIOS en lui communiquant les informations dans le format accepté par le matériel,
- assurer la gestion des bases de données associées au terminal,
- permettre l'utilisation d'outils de dialogue (tablette à digitaliser, clavier, réticule, ...).

Le premier chapitre décrit les étapes nécessaires à la synthèse d'une scène et analyse, aux niveaux matériel et logiciel, les systèmes existants pour tenter de dégager les caractéristiques d'un système de synthèse.

Les deux chapitres suivants présentent l'architecture générale du système retenu à l'IMAG. Le contenu de chaque partie du système est développé. Comme une connaissance approfondie du prototype HELIOS est nécessaire pour piloter ce terminal, il fait l'objet d'un chapitre.

Les caractéristiques du logiciel pilote vu de l'utilisateur, sont définies au chapitre 4. Outre l'intérêt porté aux opérateurs de description et de synthèse, les primitives de dialogue avec le logiciel principal ou les différents postes de travail sont détaillées.

Le cinquième chapitre justifie les options choisies lors de la réalisation. Les algorithmes et aspects intéressants seront aussi présentés.

Enfin, nous analysons les résultats obtenus et indiquons les développements futurs envisageables.

ooo

CHAPITRE 1

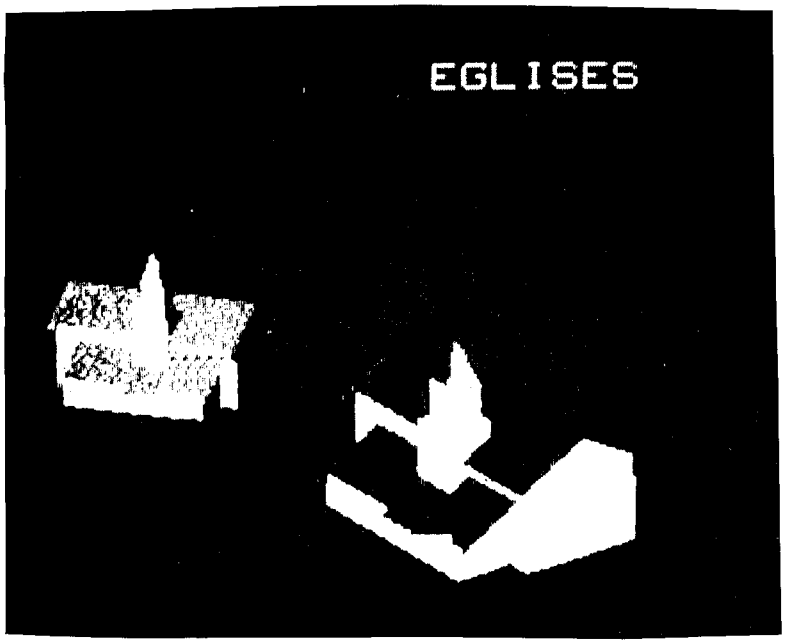
LA SYNTHÈSE D'IMAGES

Nous consacrons ce chapitre aux principaux concepts qui régissent la synthèse d'images, de façon à pouvoir dégager ultérieurement les motivations de nos choix.

1.1. Introduction sur la portée et les difficultés de cette synthèse

L'image est incontestablement un merveilleux moyen de communication. L'intérêt qui lui est témoigné découle de la grande quantité d'informations qu'elle transmet sous une forme synthétique.

La puissance de calcul de l'ordinateur alliée à un dispositif de visualisation, permet, dans de nombreuses applications, d'obtenir et de modifier un produit en cours de réalisation, bien plus rapidement qu'on ne pouvait le faire à travers la construction d'une maquette. Les applications, telles que la conception assistée par ordinateur, trouvent donc dans l'image un outil fondamental comme *support d'information*. Actuellement, dans de nombreux cas, on s'arrête à la création de plans, type "dessin au trait", mais pour un certain nombre d'entre eux, les informations "esthétiques" sont un facteur prépondérant de la conception (architecture, urbanisme, carrosserie, etc ...) et demandent donc la réalisation d'images réalistes synthétisées. Le système doit alors être *capable d'intégrer rapidement les modifications de forme, de position et d'aspect des objets* (photo 1).



1 - Photo d'une église synthétisée sur HELIOS
Deux points de vue différents sont utilisés.

D'autres applications considèrent l'image comme une *finalité* : ce sont des applications à vocation artistique, telles que la production d'images réalistes pour la publicité, le cinéma, les dessins animés, ..., ou à vocation scientifique et pédagogique, telles que l'enseignement assisté par ordinateur, la production de films pour l'industrie, la simulation de conduite. Dans ce domaine, il faudra prendre en compte le réalisme de l'image et le facteur temps réel. Le système devra donc être suffisamment performant pour effectuer très rapidement les modifications et intégrer les paramètres exigés par la génération d'images réalistes (couleur, texture, éclairage, etc ...).

1.2. Les concepts de base de la synthèse d'images

Une image synthétique sera une représentation visuelle obtenue à l'aide d'un ordinateur grâce aux ordres donnés par le concepteur.

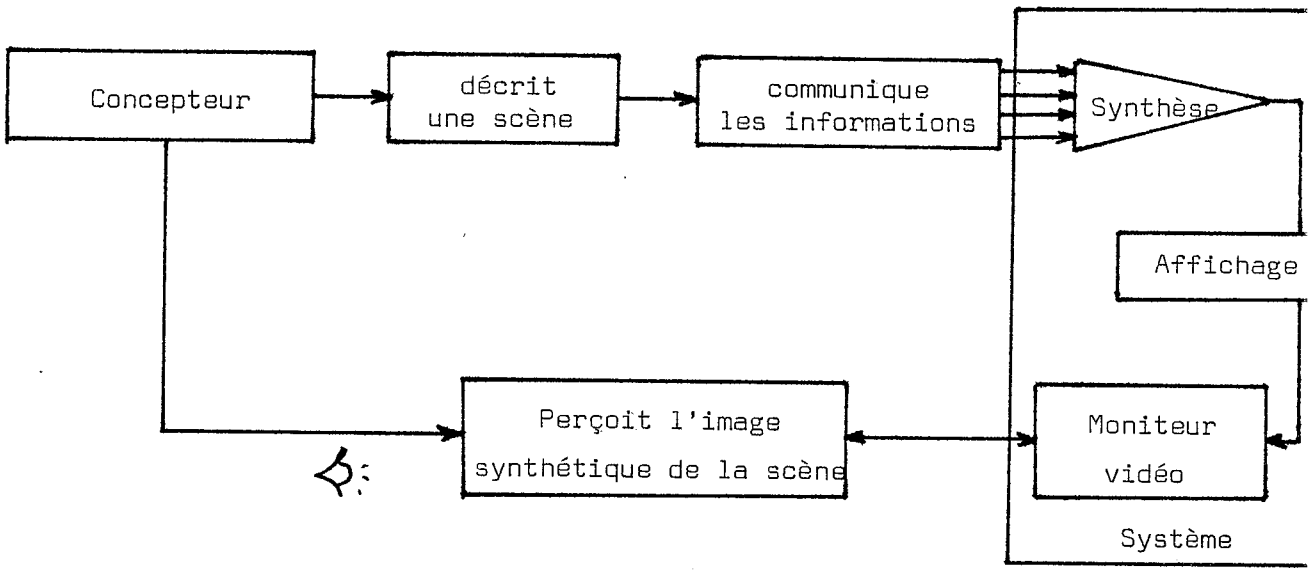


Figure 2 - Le système de synthèse

1.2.1. La notion de classes d'informations

Au premier abord, ce qui nous apparaît dans une image c'est la forme des objets et leur aspect. Mais ceci ne suffit pas à décrire une image et une étude de F. MARTINEZ [MAR 77], [MAR 80], montre que l'on peut partitionner les informations en six classes indépendantes (figure 3).

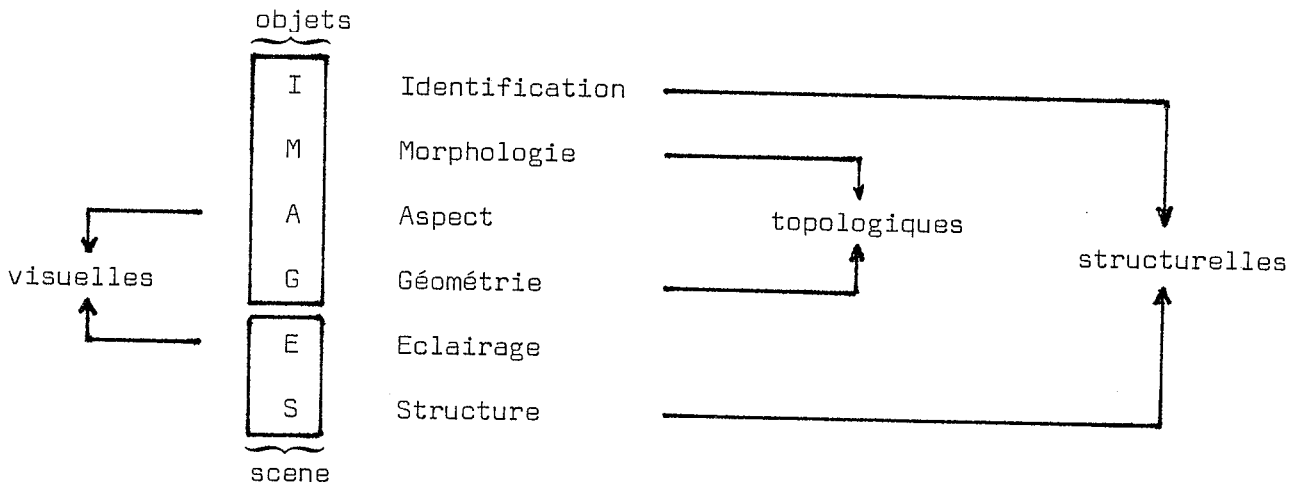


Figure 3 - Les classes d'informations

1. *L'identification* permet de nommer un objet ou un ensemble d'objets ;
2. *la morphologie* exprime la forme d'un objet, indépendamment de sa position dans l'espace ;
3. *l'aspect* précise l'apparence intrinsèque d'un objet, indépendamment des conditions d'éclairage. Il donne des renseignements sur le matériau constituant l'objet : sa couleur, sa texture, son relief, son coefficient de réflexion de la lumière (brillance, transparence ou opacité) ;
4. *les informations géométriques* permettent d'une part de situer l'objet dans l'espace, d'autre part de définir les paramètres de la prise de vue. Les transformations géométriques utilisées peuvent être des translations, des homothéties, des rotations, des projections, des coupages, ... ;
5. *les informations d'éclairage* indiquent la position des sources lumineuses, leur intensité, la lumière ambiante diffusée dans la scène ;
6. *la structure* permet d'exprimer les relations qui lient les objets entre eux, suivant la scène visualisée.

En fin de synthèse, seule importe l'information visuelle véhiculée en chaque point de l'image. *Les informations topologiques* concernant la forme et la position des objets sont déduites alors par le cerveau de façon inconsciente. *Les informations structurelles* ne sont, elles, reconnues que par analogie avec la connaissance que l'on a des objets et de la scène.

Cette perte d'informations au cours des opérations de synthèse, aura de fortes répercussions sur les possibilités d'interaction.

1.2.2. La notion de type d'information

Dans chaque classe d'informations plusieurs types doivent être considérés, qu'ils soient relatifs à la nature ou à la modélisation des informations.

Ainsi les informations morphologiques peuvent-elles être constituées par de nombreux types : segments, cercles, taches, caractères, sphères, polyèdres surfaces courbes, ... et les informations visuelles par des types tels que couleur, brillance, texture, transparence, ...

La synthèse de types d'informations appartenant à des classes différentes est un type d'information appartenant à l'une des classes de départ. (Exemple figure 4).

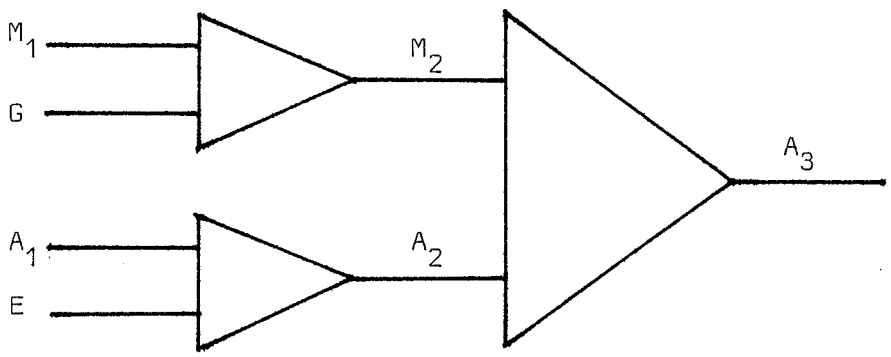


Figure 4 - Exemple de synthèse de type d'information

1.3. Les grandes étapes de la synthèse d'images

Trois étapes fondamentales transforment la vue imaginée en image synthétisée :
- la description d'une maquette de la scène,
- la construction de celle-ci,
- puis sa visualisation.

De façon schématique, deux logiciels permettent au programme d'application d'être indépendant du matériel utilisé (figure 5) :
- le logiciel de description qui permet d'entrer des données (photostyle, tablette, clavier, ...),
- le logiciel de visualisation qui affiche l'image finale sur l'écran.

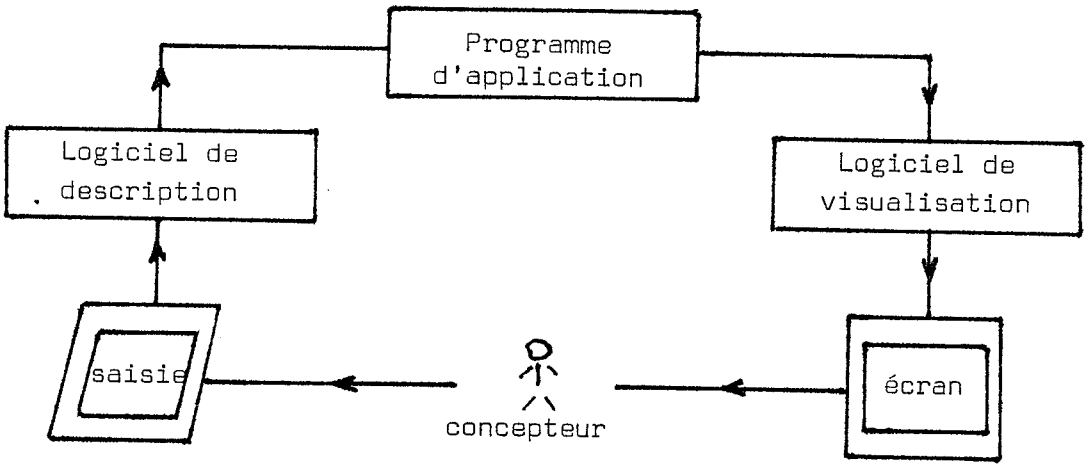


Figure 5 - Circulation des informations

1.3.1. Description de la scène

Cette étape permet de communiquer au système tous les éléments constitutifs de la scène. Elle nécessite la description et la modélisation des divers types d'informations.

Des outils spécifiques devront permettre de décrire les types d'informations morphologiques (M), par exemple un cercle pourra être décrit par deux points : le centre et un point de la circonférence, ou un point (le centre) et une distance (le rayon ou le diamètre).

De même, des outils tels qu'une palette de couleurs, permettront de définir un type d'aspect (A), des fonctions permettront de situer des objets les uns par rapport aux autres (G), de les identifier (I), d'indiquer leur structure (S) ou les conditions d'éclairage (E).

Le modèle de description ainsi obtenu, la maquette, (figure 6), comprend les attributs modélisés des composants de la scène. Une étude récente [SAR 81] décrit les principes et la modélisation de l'aspect intrinsèque des objets et des paramètres d'éclairage. Le lecteur en trouvera un résumé en Annexe 1.

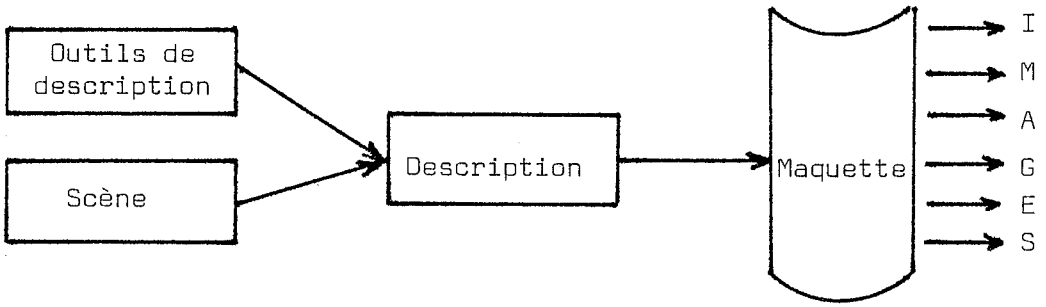


Figure 6 - Description d'une scène

1.3.2. Construction de la maquette

Construire, c'est réaliser des modèles d'objets à partir de la maquette ainsi décrite. Par exemple, si l'on considère le type : fonction à deux variables (M) décrit par la fonction f(x, y), la construction impliquera une approximation de sa surface par un ensemble de surfaces gauches ou de faces polygonales.

Il s'agit donc d'engendrer une information complète à partir des attributs (grâce à des algorithmes, des interpolations, des approximations, etc ...).

1.3.3. Visualisation de la maquette

La visualisation se scinde en deux opérations : la prise de vue et l'affichage.

1.3.3.1. La prise de vue

Considérer les conditions dans lesquelles une maquette sera observée, c'est définir le point de vue, la direction de visée, ... Ce type d'attribut appartient à la classe des informations géométriques, mais s'applique sur toute la maquette. Il sera noté Gv. Il permet de transformer les coordonnées exprimées dans le repère de la maquette (2D ou 3D) en coordonnées exprimées dans le repère de la vue (2D).

La détermination de ce qui est réellement visible nous permet d'obtenir une "vue" (figure 7) : image latente sous une forme non visualisable.

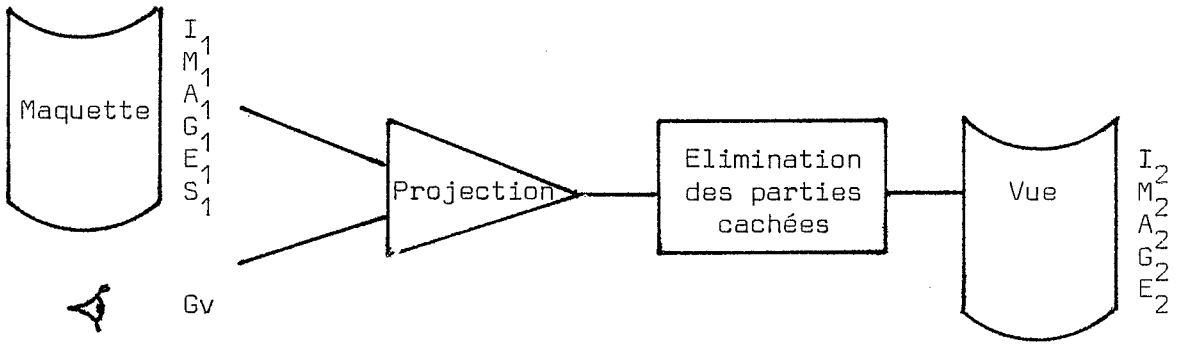


Figure 7 - La prise de vue

1.3.3.2. L'affichage

Cette opération permet de décrire la façon dont est cadrée la vue pour obtenir l'image finale. Ce type d'information, notée Ga, appartient à la classe des informations géométriques. Elle permet de transformer les coordonnées exprimées dans le repère de la vue, en coordonnées écran (figure 8).

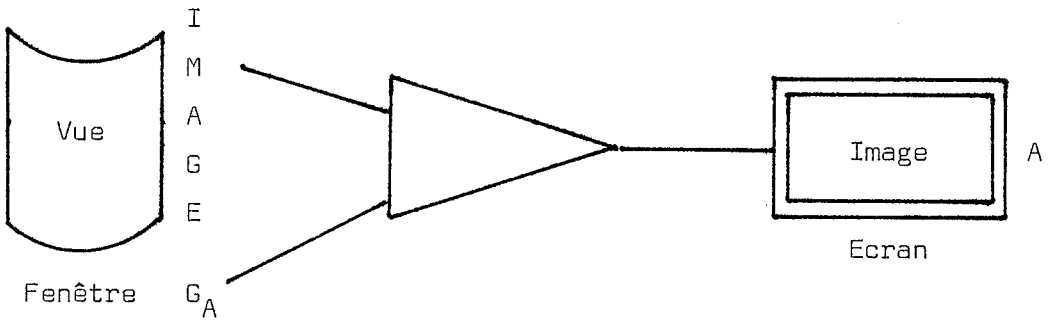


Figure 8 - Affichage d'une vue

L'image ainsi obtenue ne contient pour chaque point de l'écran que la couleur de ce point, toutes les autres informations ayant été synthétisées (figure 9).

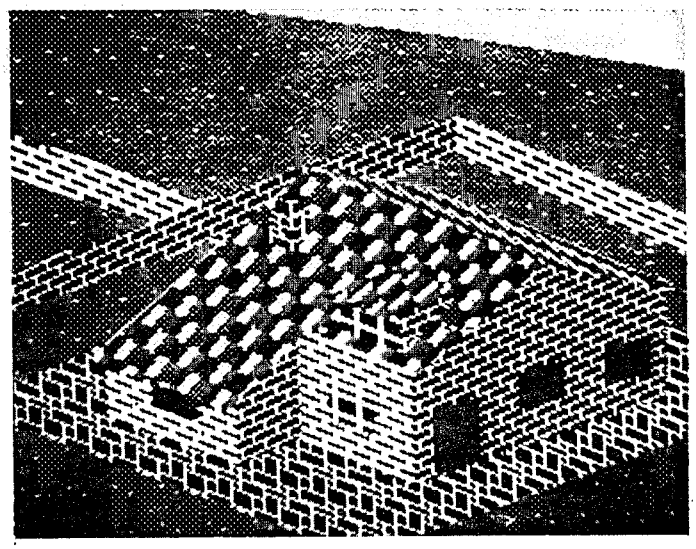


Figure 9 - Image synthétisée sur le système HELIOS

1.3.4. Conclusion

Les différentes opérations de synthèse impliquent un certain nombre d'échanges entre le concepteur et le programme d'application.

Une récente étude [MAR 82] a montré que ces échanges se faisaient à travers quatre processus (figure 10) :

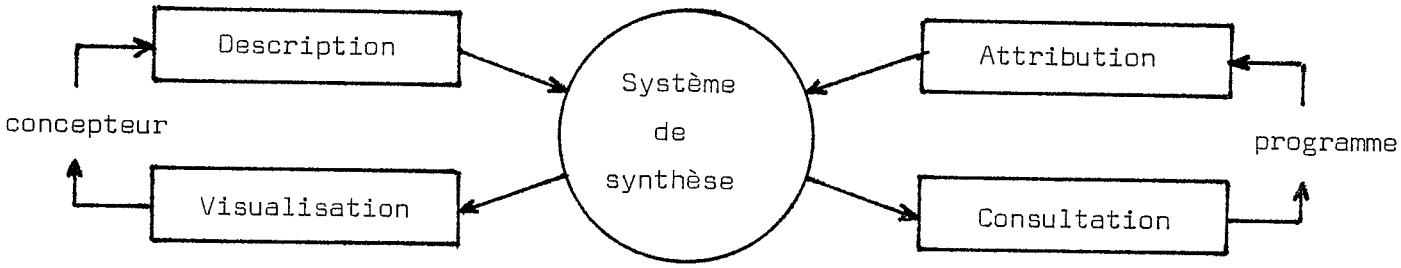


Figure 10 - Les quatre processus d'échange

Ces quatre processus formalisent l'interface entre l'application et le système.

Le processus de description : permet de décrire les informations de base (I, M, A, G, E, S, G_V, G_A) de la scène à l'aide de ce que communique l'utilisateur. Sa mise en oeuvre n'est pas si aisée qu'elle le paraît : en effet les dispositifs de saisie (tablette ...), ne fournissent que des points (information morphologique). Il faudra donc transformer ces informations initiales en attributs du type considéré.

Le processus de visualisation : est capable d'assurer la synthèse des attributs descriptifs des éléments de la scène en une information de couleur pour chaque point de l'image. Pour cela, il aura besoin d'opérateurs de synthèse, de mémorisation (stockage des étapes intermédiaires), de modélisation (transformation d'attribut, d'un type donné en un autre type), de composition (algorithmes d'élimination de parties cachées ...).

Le processus d'attribution : se charge d'affecter à la base de données du système, les calculs effectués par le programme d'application. Les opérateurs sont successivement :

. recherche des éléments concernés,

. affectation proprement dite.

Le processus de consultation : récupère les attributs provenant du système et les communique à l'application.

1.4. Les systèmes classiques utilisés

Nous développons ici les intérêts et les limitations des systèmes classiques. Ils nous permettront de mettre en évidence les difficultés rencontrées pour construire un système interactif idéal.

1.4.1. Organisation générale d'un système

1.4.1.1. Définition [MAR 82]

Un système est composé d'un (ou de plusieurs) synthétiseur(s) : interface entre un univers initial, la scène et un univers final, l'image.

Un synthétiseur est une unité de traitement comprenant au moins :

. un processeur (programmé, micro-programmé ou câblé),

. une ressource locale de mémoire (RAM, disque souple, écran, etc ..),

. un processus de visualisation et d'attribution dans cette mémoire, et éventuellement

. des processus de description et de consultation.

Le synthétiseur le plus simple est le tube cathodique (figure 11).

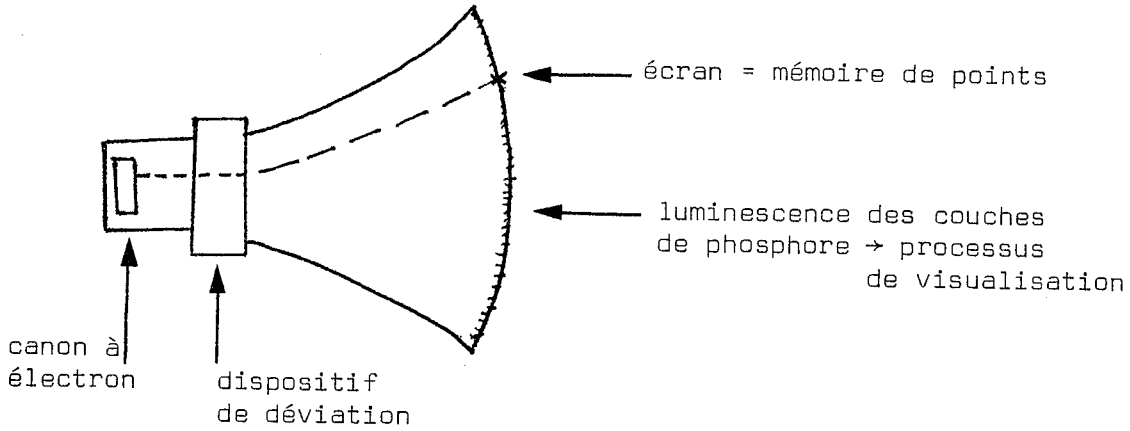


Figure 11 - Principe du tube cathodique

Un système est donc une suite de synthétiseurs dont le dernier est le tube cathodique avec son univers : le point. La plupart du temps, une partie de la synthèse est réalisée par logiciel, une autre par matériel.

1.4.1.2. Influence de la configuration matérielle

a) Organisation sur matériel bas de gamme

La configuration minimale que l'on peut trouver est constituée par :
- un calculateur hôte comprenant logiciel et ressources mémoire,
- une console de visualisation.

Pour en analyser les avantages et les limitations, prenons l'exemple d'une organisation vidéo (figure 12) : la mémoire de trame mono ou multi-plans contient la synthèse de toutes les informations associées à chaque point, exprimée en général par les niveaux des trois couleurs primaires : rouge, vert et bleu.

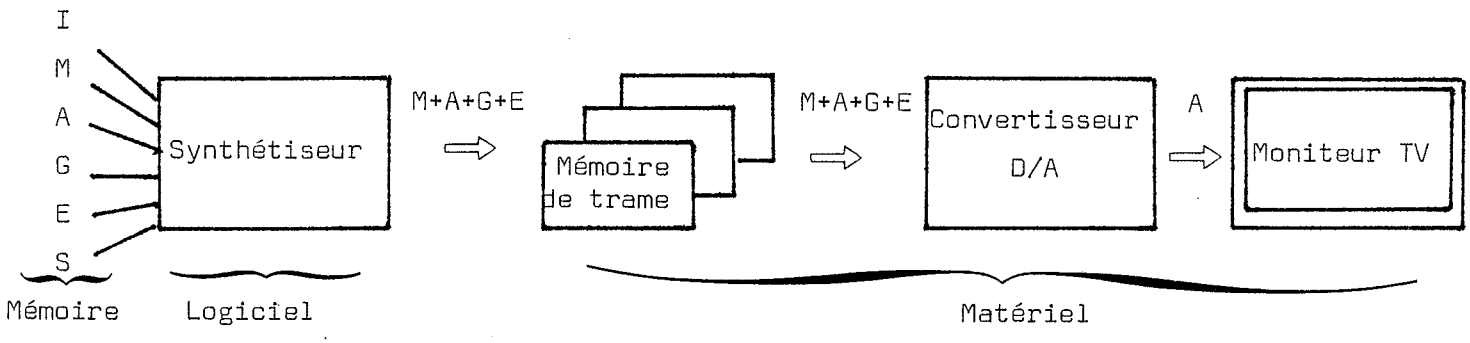


Figure 12 - Organisation vidéo classique

Cette organisation présente les inconvénients suivants :

- toute modification d'un attribut d'un élément de la scène implique l'appel au calculateur principal pour une nouvelle synthèse ;
- l'identification d'un objet à partir d'une désignation sur l'écran est impossible sans avoir recours au calculateur principal du fait que la mémoire de trame conserve une information de couleur et non d'identification.

Elle peut par contre permettre, si le moniteur TV a une bonne définition, d'obtenir des images avec un bon degré de réalisme car les algorithmes logiciels de synthèse sont bien maîtrisés et impliquent peu de perte d'informatic

Mais de nombreuses applications, telles que la conception assistée par ordinateur, ne peuvent se satisfaire d'un tel système du fait de la longueur des calculs nécessaires au niveau logiciel, et de la faible interactivité autorisée.

b) Organisation sur console évoluée

Une console est dite évoluée si elle offre des fonctions locales de description et de synthèse permettant de modifier l'image sans intervention du calculateur principal. Cette console est réalisée à l'aide de deux synthétiseurs : un synthétiseur intermédiaire possédant sa mémoire locale et un synthétiseur final. Elle est connectée au calculateur hôte via un calculateur satellite possédant sa mémoire propre (figure 13). Le calculateur satellite est asservi au terminal ; son logiciel assure l'indépendance programme d'applications - matériel.

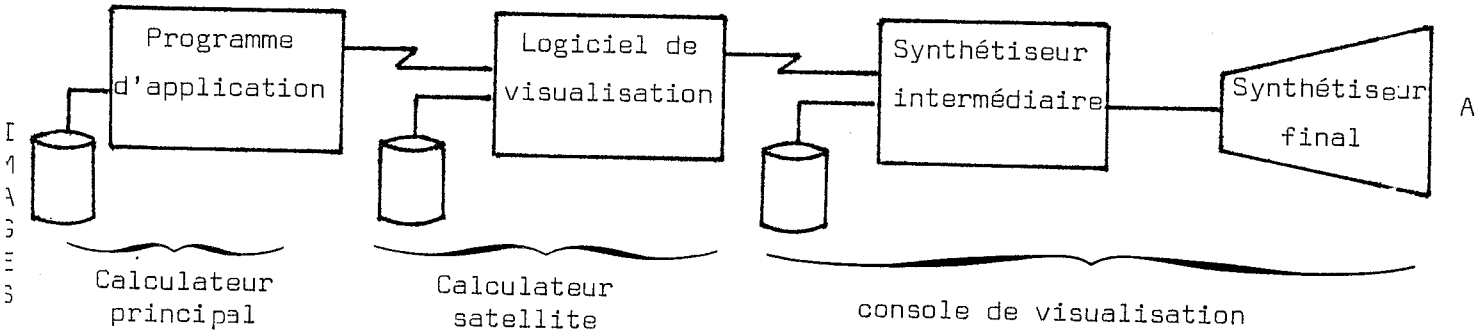


Figure 13 - Console de visualisation évoluée

L'idéal serait que le synthétiseur intermédiaire soit capable de synthétiser toutes les informations (I, M, A, G, E, S) au rythme du synthétiseur final. Mais le peu de temps disponible pour chaque point face à la somme d'opérations nécessaires et aux moyens technologiques dont on dispose, fait que ce synthétiseur n'est actuellement pas réalisable.

Néanmoins, suivant les besoins des applications, un certain nombre de systèmes conservent à ce niveau les attributs qui leur sont nécessaires pour une interaction dans le domaine de la forme ou de l'aspect.

c) Conclusion : comment améliorer le processus de synthèse ?

Entre ces deux organisations, on peut imaginer un ensemble infini de systèmes suivant l'endroit où se situe la synthèse des informations. Pour permettre une meilleure interactivité, deux techniques sont utilisées au vu des informations à privilégier [MAR 82] :

. la mémorisation des attributs synthétisés,

. la pénétration des attributs non synthétisés.

La mémorisation des attributs synthétisés est intéressante avant la synthèse des informations de point de vue (G_V) ou d'affichage (G_A). Quand les attributs de ces éléments changent, la synthèse ne recommence que là où ces paramètres interviennent dans la synthèse (figure 14).

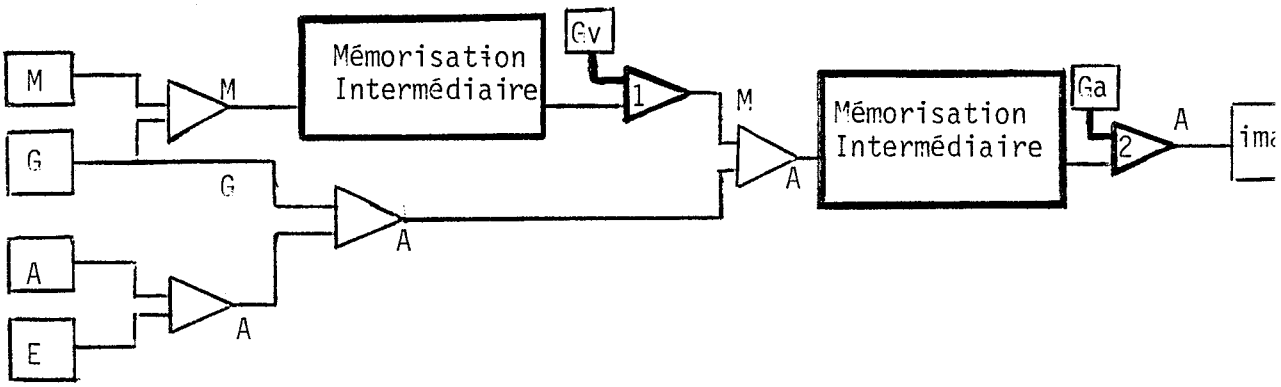


Figure 14 - Exemple de mémorisation des attributs synthétisés

La pénétration des attributs non synthétisés à l'intérieur du processus, permet une modification rapide de ceux-ci (figure 15). Cette pénétration ne peut se faire que pour les attributs indépendants les uns des autres.

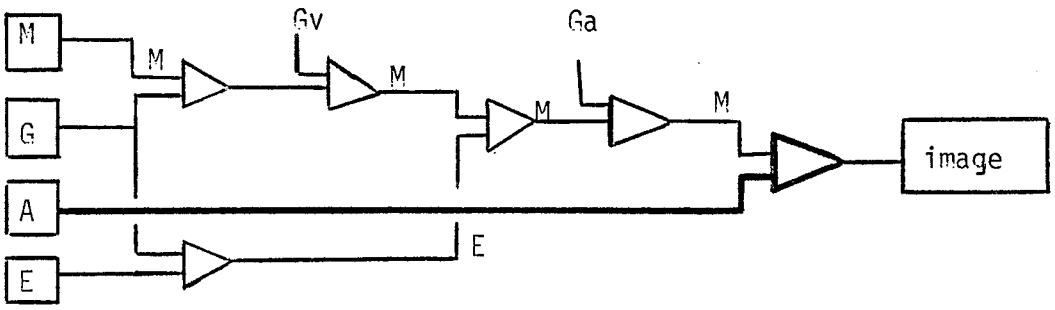


Figure 15 - La modification "aspect", grâce à la pénétration de cet attribut, ne fait intervenir qu'un opérateur de synthèse

Cette synthèse tardive au sein du matériel, ne permet pas non plus d'utiliser un certain nombre d'algorithmes logiciels. En effet, si l'on fait pénétrer les informations d'éclairage, comment simuler par exemple les ombres portées, technique résolue de façon logicielle par Booknight et Kelley [BOK 70] et présentée dans l'annexe 1.

Cette pénétration devra donc être minutieusement étudiée pour limiter le moins possible la qualité de l'image finale. Ceci ne pourra se faire que suivant le besoin des applications concernées : c'est pourquoi les systèmes actuels sont :

- soit destinés au "grand public" et offrent peu de réalisme,
- soit dévolus à un domaine d'application.

1.4.1.3. Architecture des synthétiseurs

La réalisation d'un système implique que soient définis le nombre de synthétiseurs et les possibilités de chacun.

La conception d'un synthétiseur fait appel à plusieurs techniques :

- la réalisation d'un logiciel sur calculateur standard,
- l'utilisation de micro-processeurs standard (logiciel intégré dans PROM) ou spécialement adaptés (intégrés dans boîtier VLSI), ou de microprocesseurs en tranches,
- l'utilisation de la logique câblée.

Notons que quelles que soient les techniques utilisées, des opérations devront être effectuées en parallèle pour permettre une forte pénétration des attributs.

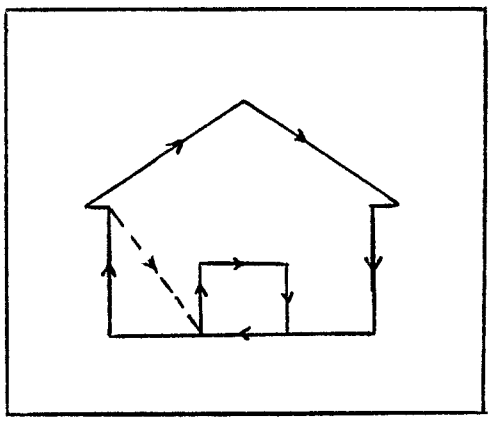
1.4.2. Exemples de systèmes

Les dispositifs d'affichage existant sur le marché font appel pour 99 % d'entre eux aux tubes à rayons cathodiques. Les écrans plats (panneaux à plasma, écran à cristaux liquides ou écran laser) sont encore trop récents pour être réellement utilisés.

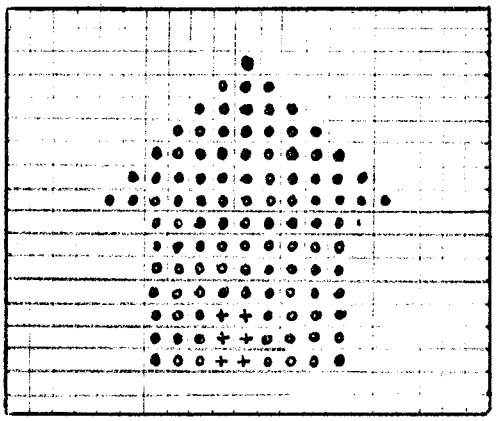
Deux technologies concurrentes (figure 16) existent pour les écrans à rayons cathodiques :

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

. le balayage de trame où le faisceau d'électrons parcourt de gauche à droite toutes les lignes de l'écran (en 40 ms).



faisceau allumé
faisceau éteint



noir couleur a
couleur b pour la porte

Balayage cavalier

la persistance de l'image sur l'écran est dûe à un rafraîchissement continu de l'image grâce à une mémoire d'entretien qui contient la liste des vecteurs à engendrer.

Balayage de trame

le rafraîchissement de l'image est obtenu grâce à une mémoire de trame (mémoire de points contenant pour chacun la couleur), dont la taille dépend directement de la résolution souhaitée

Figure 16 - Technologies de balayage

Le balayage cavalier est plus adapté à la présentation de "dessins au trait" et le balayage de trame à la présentation d'images.

1) Le contrôleur vidéo intégré d'EFCIS

Ce système minimum [MAT 78], intégré dans un boîtier 40 broches, prend en charge les processus de description, visualisation, affectation et consultation pour les éléments simples de type points, vecteurs ou caractères. Il peut être intégré dans un système plus complexe en vue de le décharger des opérations :

- de synchronisation avec le moniteur,
- de gestion de la mémoire d'écran (jusqu'à 512 x 512 points) et du réticule
- de génération de caractères (ils sont programmables en taille et peuvent prendre une des orientations suivantes ↑ ↓ → ← ↖ ↗ ↙ ↘),
- de génération de vecteurs,
- d'interface avec un microprocesseur 8 bits.

Ce contrôleur possède des registres chargeables et éventuellement un photostyle. En annexe 2, une description plus complète des modalités d'exploitation est donnée. Notons cependant qu'il n'effectue que des conversions morphologiques.

2) Le 4010 de TEKTRONIX : un système à tube mémoire

Ici le dispositif d'entretien est intégré au tube mémoire. Les types simples acceptés sont points, vecteurs ou caractères. Une synthèse d'aspect (tiret pointillé) peut être effectuée pour les vecteurs. Le logiciel utilisant ce terminal "bas de gamme" doit donc effectuer toutes les autres synthèses (figure 17).

4) Système comprenant un microprocesseur interne et son logiciel : le 4027 de TEKTRONIX

Le microprocesseur intégré au terminal permet d'accroître le nombre d'opérateurs de synthèse. Ces opérateurs réalisent :

. la génération de points, vecteurs, caractères, sous diverses textures,

. le remplissage de taches à l'aide de textures (matrice 14 x 8 composée de deux couleurs au maximum).

La figure 19 illustre ce système.

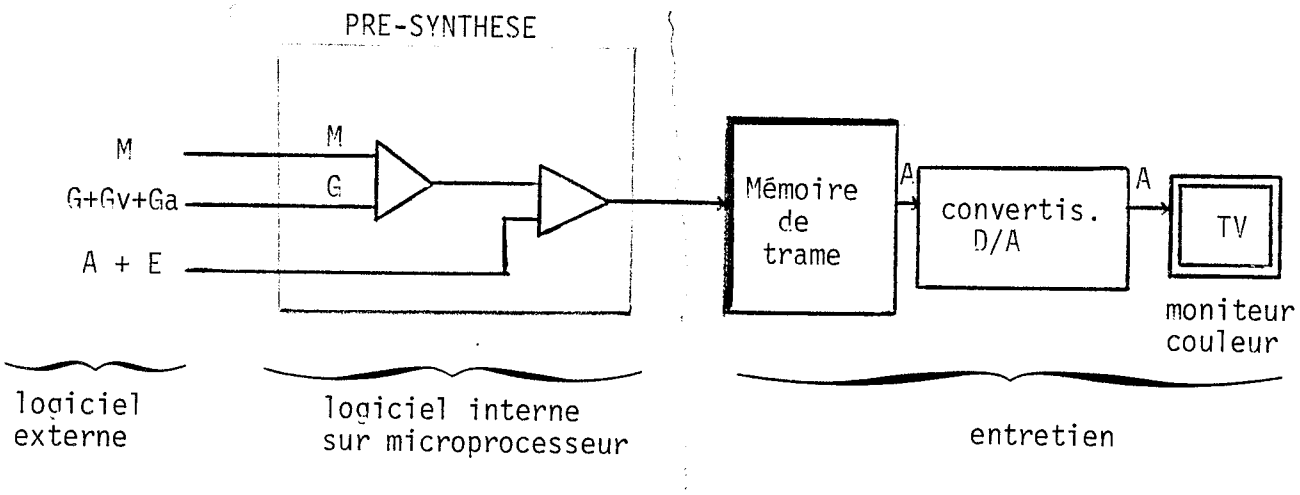


Figure 19 - Structure du 4027 de TEKTRONIX

5) Système comprenant un micro-calculateur : le RAMTEK 9400

Ce système correspond à la tendance actuelle. En effet, il permet d'intégrer de nombreuses fonctions locales. Le RAMTEK 9400 réunit les avantages d'un balayage cavalier et d'un balayage de trame. En effet, l'utilisateur a accès à une mémoire intégrée au terminal contenant un fichier de vecteurs, que l'on peut visualiser dans une fenêtre. Le traitement des informations géométriques (zoom, translation, rotation) et d'aspect (remplissage de polygone avec une gamme de couleurs étendue) est possible grâce à un microprocesseur bipolaire en tranches. La figure 20 nous donne la structure de ce calculateur graphique.

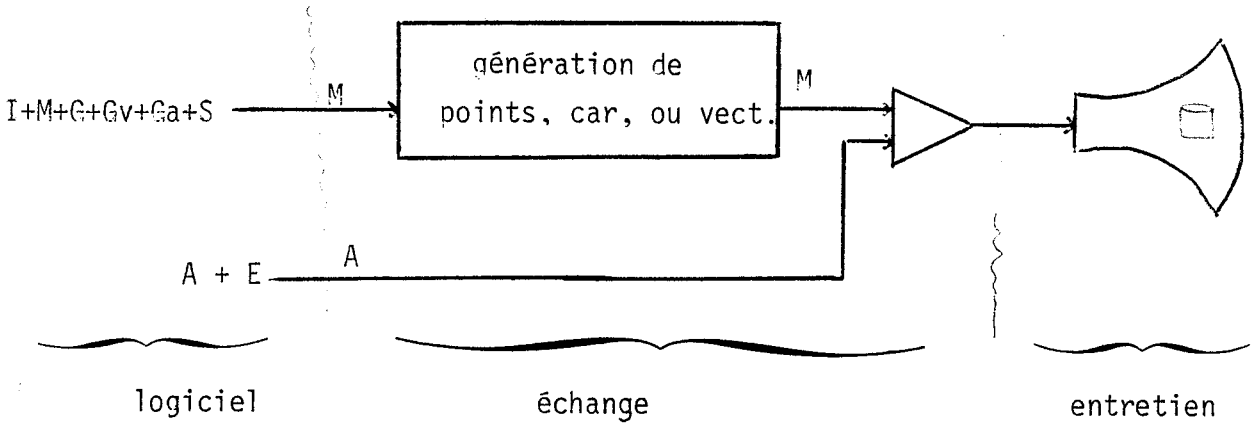


Figure 17 - Représentation schématique du système engendré par l'utilisation du TEKTRONIX 4010

3) Un système à liste de visualisation utilisant un IBM 2250

La mémoire d'entretien du terminal à balayage cavalier comprend directement un programme composé d'ordres graphiques (nommé "liste de visualisation"). Ces ordres graphiques s'appliquent à des points, des caractères ou des vecteurs. L'aspect peut être défini par la texture du trait ou la couleur. Les transformations géométriques acceptées sont les zooms et translations 2D. Ce système permet donc une pénétration importante des attributs géométriques et d'aspect (figure 18).

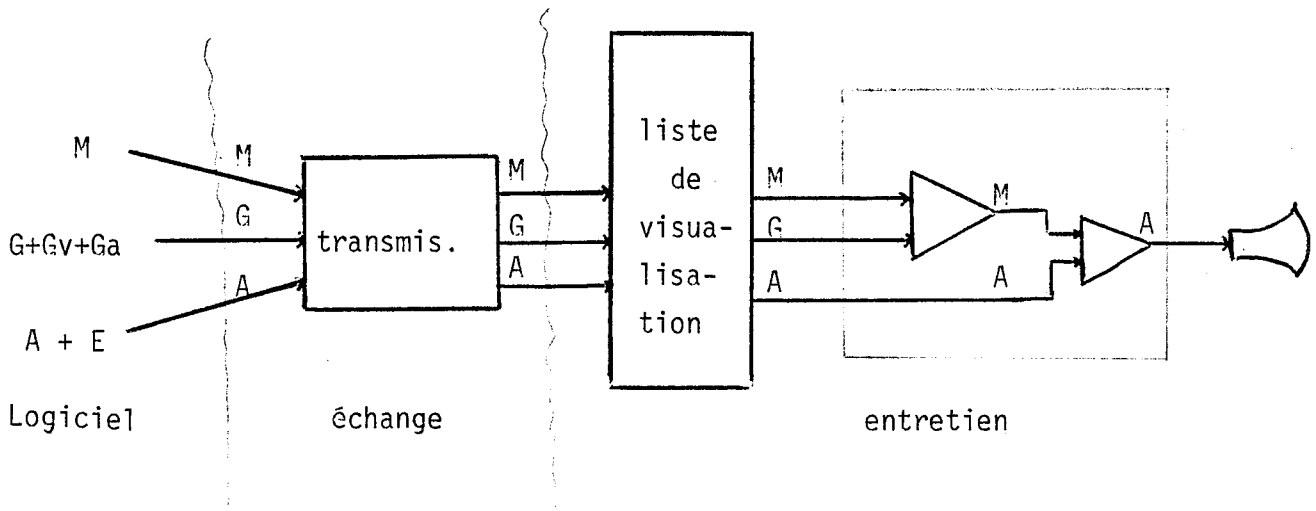


Figure 18 - Représentation schématique d'un système utilisant un IBM 2250

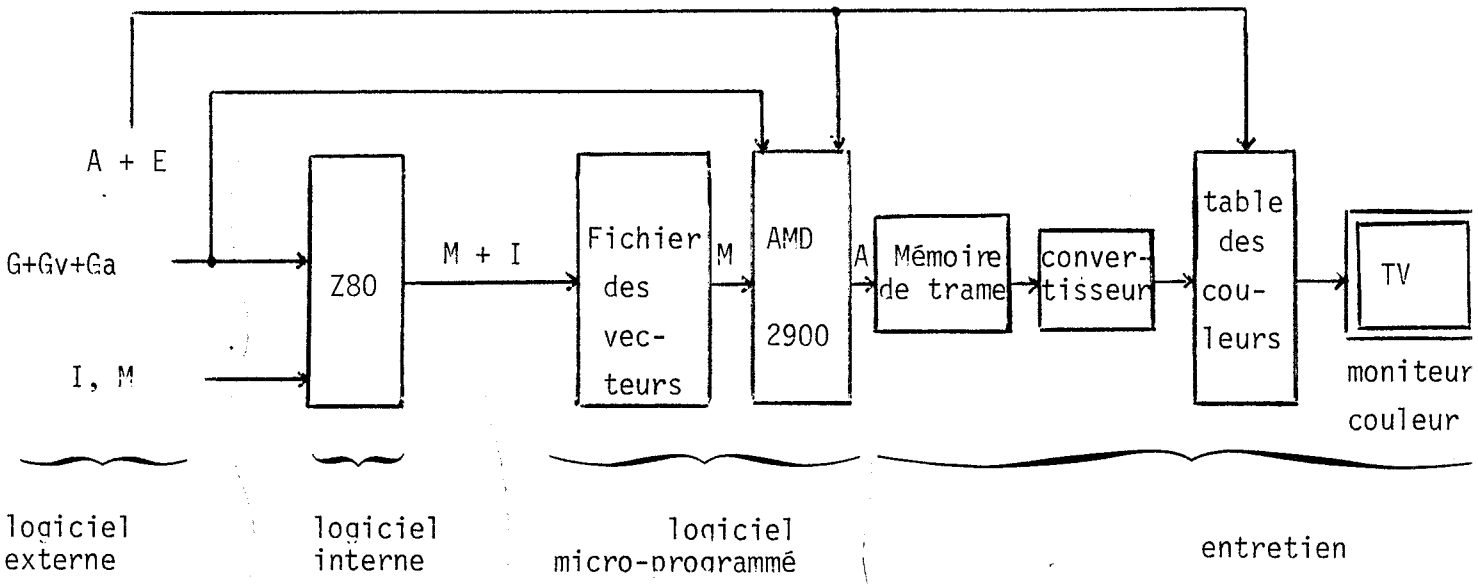


Figure 20 - Architecture du RAMTEK 9400

6) Les systèmes orientés vers des applications particulières ;
exemple : la simulation de conduite

Ces systèmes utilisent souvent plusieurs ordinateurs en parallèle et des circuits très performants spécialement conçus pour résoudre des problèmes spécifiques. La simulation de conduite, pour qui opérations de prise de vue et d'affichage sont importantes, mémorise les attributs synthétisés ou des informations saisies par caméra, au niveau logiciel. Les images synthétiques sont affichées suivant le point de vue sur tel ou tel moniteur couleur. Ces caractéristiques, ajoutées à l'exigence du temps réel, expliquent que les systèmes mis en oeuvre sont impressionnants par leur taille et leur coût.

7) Conclusion

L'étude de ces systèmes nous amène à la constatation suivante : il est illusoire actuellement de créer un système d'un coût correct, qui soit rapide, indépendant des applications, générant des images d'un haut niveau de réalisme et permettant une forte interactivité. Suivant les applications des bons compromis pourront être trouvés soit au niveau matériel, soit au niveau logiciel. Nous étudions dans le paragraphe suivant les différents logiciels existants.

1.5. Les logiciels existants

De nombreux logiciels de base ont été construits et leur étude critique permet de dégager les caractéristiques souhaitables de notre logiciel.

1) Les logiciels fournis par les constructeurs assurent souvent une certaine indépendance vis à vis de l'ensemble du matériel de leur gamme. C'est le cas *d'I.G.L. de TEKTRONIX* [IGL 78]. Il permet la communication de nombreux types géométriques ou morphologiques. Par contre, le processus de visualisation est presque entièrement à la charge du programme d'application, ce qui implique pour chaque application, la ré-écriture du module contenant les primitives d'affichage. Une structuration des diverses primitives proposées aurait pu résoudre ce genre de problème. Un autre inconvénient de ce logiciel est l'absence d'identification par désignation, fonction qui encombre inutilement le programme d'application.

2) Des efforts importants de normalisation ont amené *l'implémentation de logiciels indépendants et configurables*. C'est ainsi que G.K.S. (Graphic Kernel System) a vu le jour [EEK 80].

Les primitives permettent *l'attribution* aux types d'informations suivants :

- informations de morphologie : polygones, sections de points, chaîne de caractères, segments,
- informations d'aspect : visibilité, clignotement, luminosité supérieure, un polygone pourra être représenté rempli, ou simplement avec son contour,
- informations géométriques : transformation de segments, taille des caractères,
- informations d'identité et de structure : grâce à la notion de segments (OPEN-SEGMENT, CLOSE-SEGMENT),
- prise de vue : point de vue dans une fenêtre,
- affichage : clôture.

La consultation peut se faire par récupération de segments complets.

La description, effective pour des informations morphologiques ou d'identité, peut être configurée à l'aide de primitives (écho possible, saisie en continu ou avec attente ...).

La visualisation peut aussi être configurée de la façon suivante :
- immédiate après chaque primitive,
- différée jusqu'à la demande de la prochaine interaction,
- toujours différée jusqu'à un ordre d'affichage.

Ce logiciel, complet au niveau des types d'éléments 2D proposés, fortement configurable tant en description qu'en visualisation, manque toutefois de cohérence (les primitives s'adressent à des éléments ou à des types d'informations différents), et contraint l'utilisateur à structurer ses données graphiques en fonction du déroulement du programme.

3) Un autre logiciel de base, *indépendant du matériel mais non configurable*, a été conçu au Laboratoire d'Informatique et de Mathématiques Appliquées de Grenoble ; il s'agit de GRIGRI [LUC 77], [LLM 78]. Ce logiciel a été commercialisé et c'est fort des enseignements de cette expérience qu'est né le projet CLOVIS.

Ce logiciel permet :

. l'attribution aux types :
  - d'informations morphologiques : points, vecteurs,
  - d'informations d'aspect : texture (tireté ..), visibilité,
  - de géométrie : taille des caractères,
  - d'identité et de structure,
  - de prise de vue : définition d'une fenêtre de la scène projetée,
  - d'affichage : définition d'une clôture sur l'écran ;

. la consultation n'est pas possible ; elle doit se faire au niveau du programme d'application ;

. la description est prédéfinie lors de l'installation du logiciel et ne se fait que pour des informations de morphologie ou d'identification ;

. la visualisation est elle aussi pré-définie ; il s'agit d'une visualisation différée qui associe attribution d'aspect et visualisation proprement dite.

La structuration des données, leur description et leur visualisation en font des processus indépendants du programme d'application. La cohérence est assez bonne, puisque les primitives s'adressent aux mêmes niveaux de structuration, mais l'ensemble est moins complet que le précédent, du fait que deux types d'éléments seulement sont proposés (sections de points et sections de caractères) et que les transformations géométriques ne sont pas considérées. Ce système, s'il est suffisant pour du "dessin au trait", est mal adapté pour la synthèse d'images réalistes.

1.6. Conclusion

Dans ce premier chapitre nous avons fait un tour d'horizon des grandes questions posées par la synthèse d'images.

Il semble que les systèmes qui se développent actuellement s'appuient sur l'architecture représentée par la figure 21.

Cinq grands facteurs départagent alors les systèmes :

- la puissance des fonctions réalisées par matériel (le degré de pénétration des informations non synthétisées),
- les possibilités de mémorisation intermédiaire des informations déjà synthétisées,
- la structuration des bases de données,
- le type de liaison établie entre calculateur principal et calculateur satellite, et entre calculateur satellite et console de visualisation,
- enfin, l'indépendance entre application et dispositifs extérieurs (tablette, clavier, console de visualisation, ...).

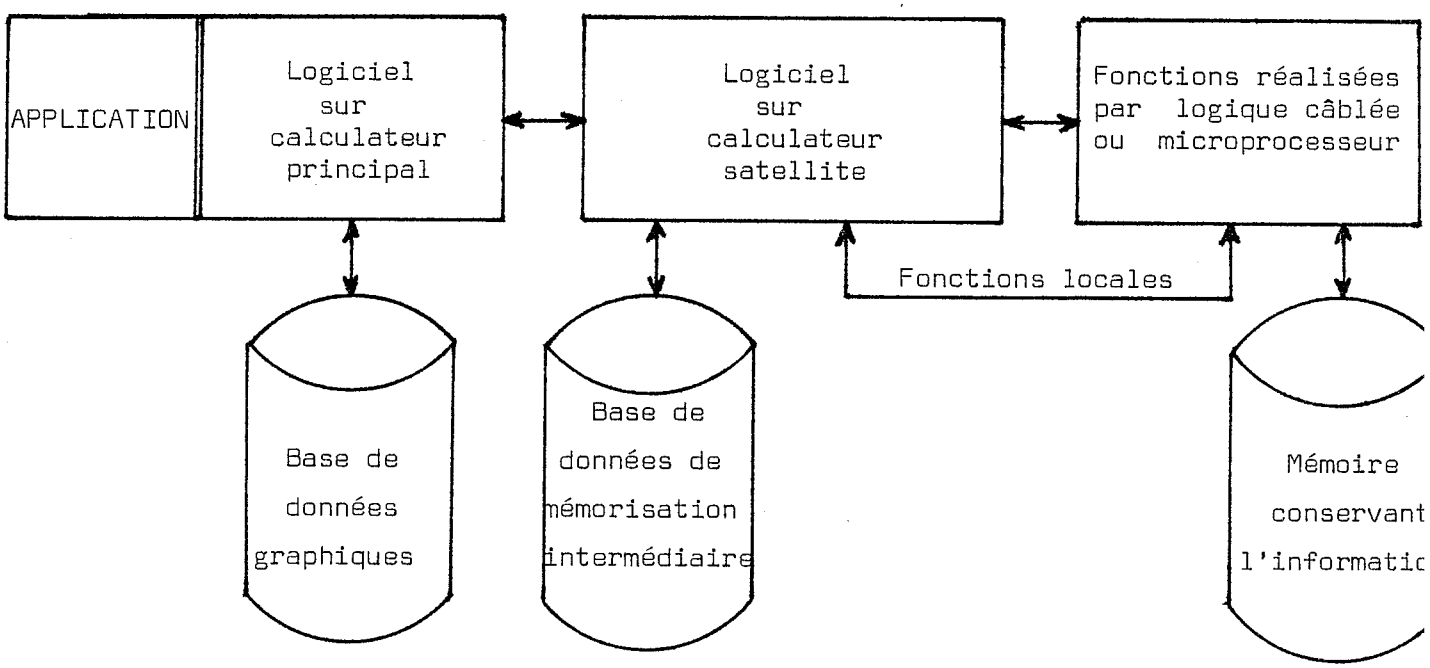


Figure 21 - Schéma général des systèmes qui se développent actuellement

CHAPITRE 2

LE SYSTÈME DE SYNTHÈSE DÉVELOPPÉ À L'IMAG

2.1. Définition du système utilisé à l'IMAG autour d'HELIOS

L'équipe Communication Graphique du Laboratoire IMAG a conçu un système dont le principal domaine d'applications est la Conception Assistée par Ordinateur (CAO). Pour cela, le système doit répondre aux objectifs suivants :

- interaction rapide pour un maximum d'opérations,
- indépendance de l'application par rapport aux représentations graphiques des objets,
- utilisation aisée des outils de dialogue,
- conservation des informations structurelles,
- l'aspect esthétique peut être un facteur prépondérant de la conception ; un haut degré de réalisme est donc nécessaire (possiblité de texture ..),
- des fonctions locales sont souhaitées.

L'idée de base consiste à repousser au maximum la synthèse effective des différents types d'informations jusqu'au logiciel se trouvant sur le calculateur satellite et dans la mesure du possible jusqu'au matériel.

L'architecture proposée est représentée par la figure 22.

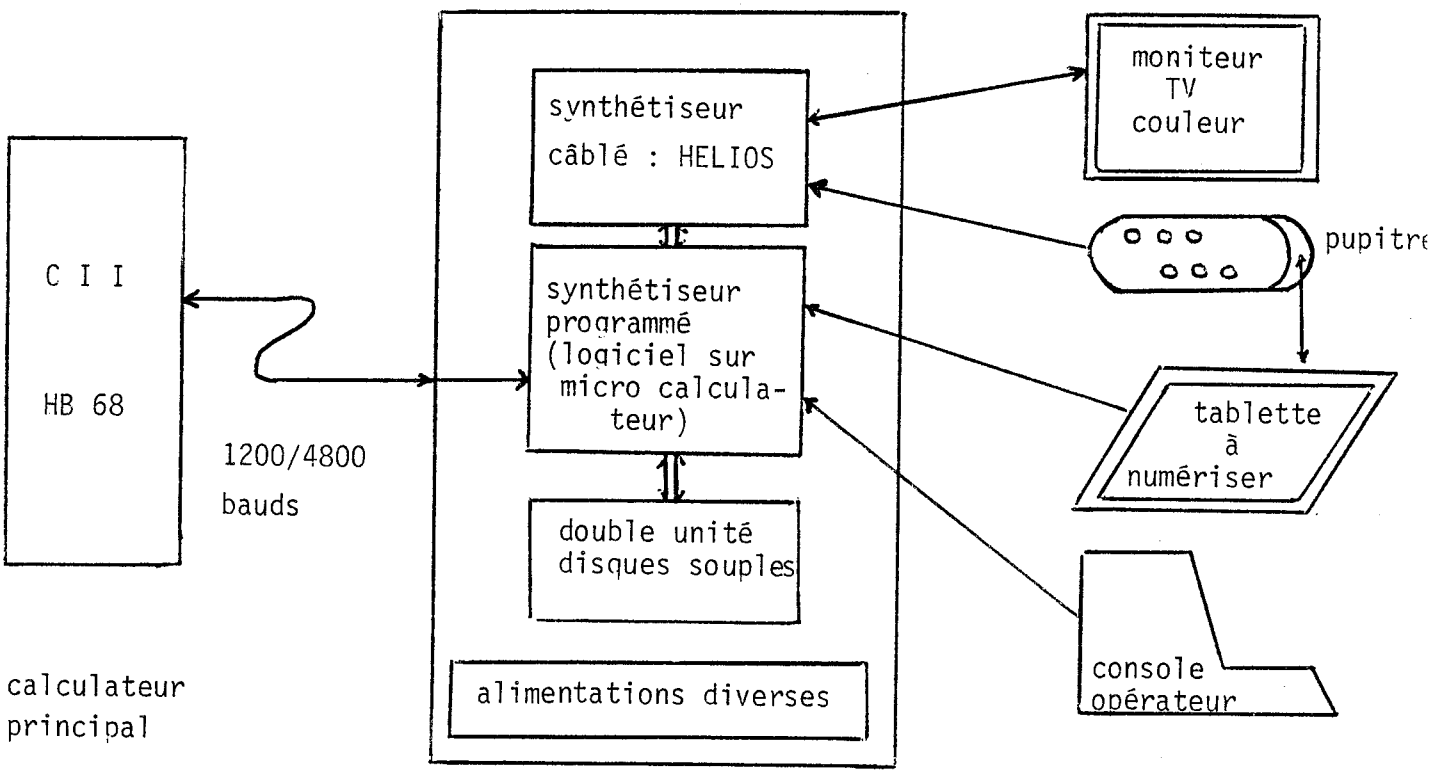


Figure 22 - Architecture du système de synthèse

La synthèse des informations morphologiques, géométriques et d'identificatio réalisée par le logiciel sur le micro-calculateur, permet le remplissage de la mémoire de trame, composée de plusieurs plans d'identification (512 x 512 points). Pour chaque point, c'est le numéro de la face qui est mémorisé. Les informations d'aspect et d'éclairage sont traitées par la logique câblée pour chaque point visible, au rythme du signal vidéo. 4096 couleurs sont possibles, de même que l'attribution d'une texture carrée à chacune des faces. 1024 faces sont synthétisables simultanément ; un réticule intégré au terminal permet la désignation des objets affichés.

La figure 23 schématise les synthèses effectuées. L'intérêt du post-synthétiseur câblé sera démontré au chapitre 3. Le pré-synthétiseur, objet de notre étude, devra permettre l'indépendance du programme d'application vis à vis du matériel, c'est-à-dire qu'il réalisera des synthèses que le matériel ne sait pas faire et transmettra les informations. Sa mémoire locale sera, soit la mémoire centrale, soit l'unité de disques souples.

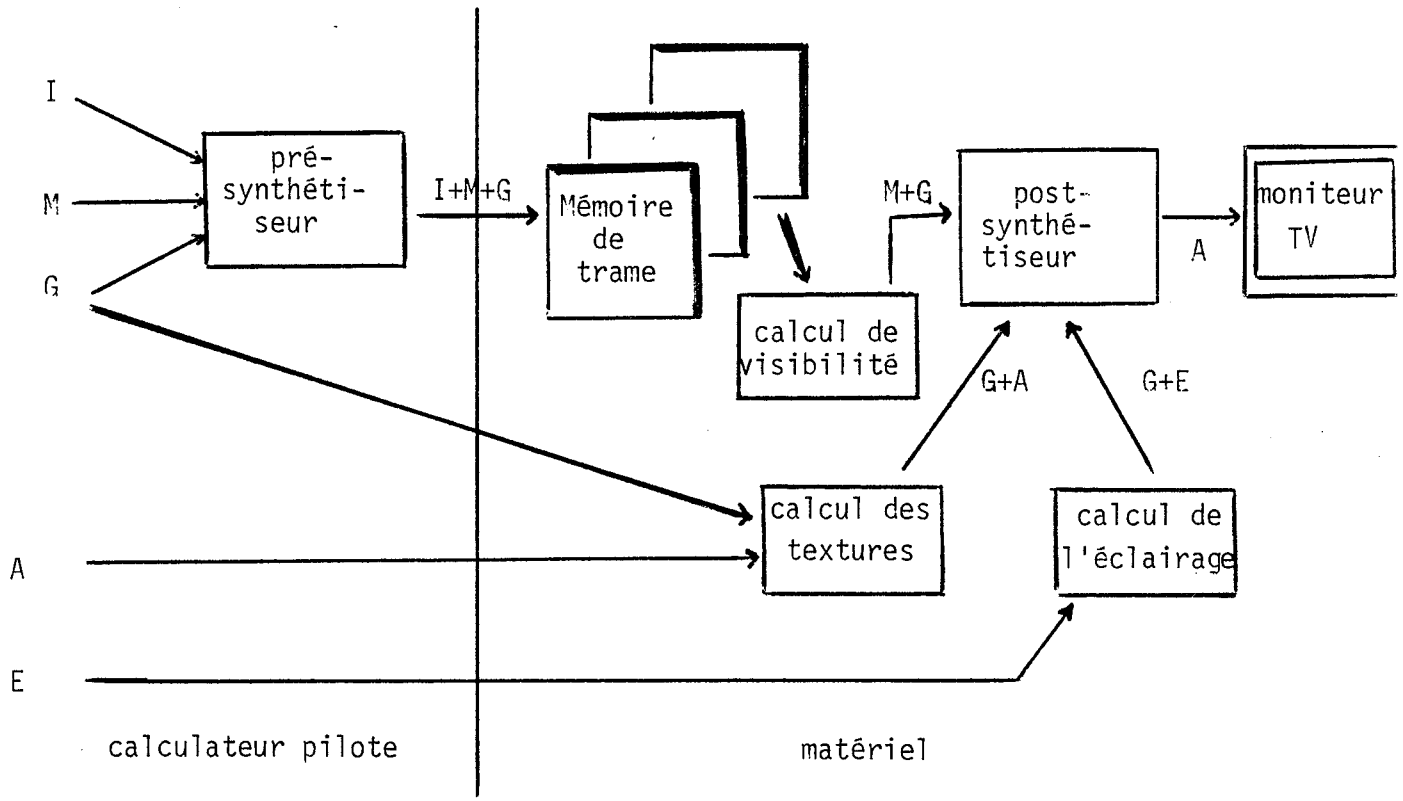


Figure 23 - Schéma de la synthèse sur HELIOS

Les objectifs de ce système de synthèse sont ambitieux. Pour qu'ils soient adaptables aux différentes applications, ce système comporte un certain nombre de paramètres, de façon à limiter l'investissement initial indispensable tout en laissant la porte ouverte à des améliorations. Nous avons choisi une organisation hiérarchique de trois synthétiseurs (figure 24).

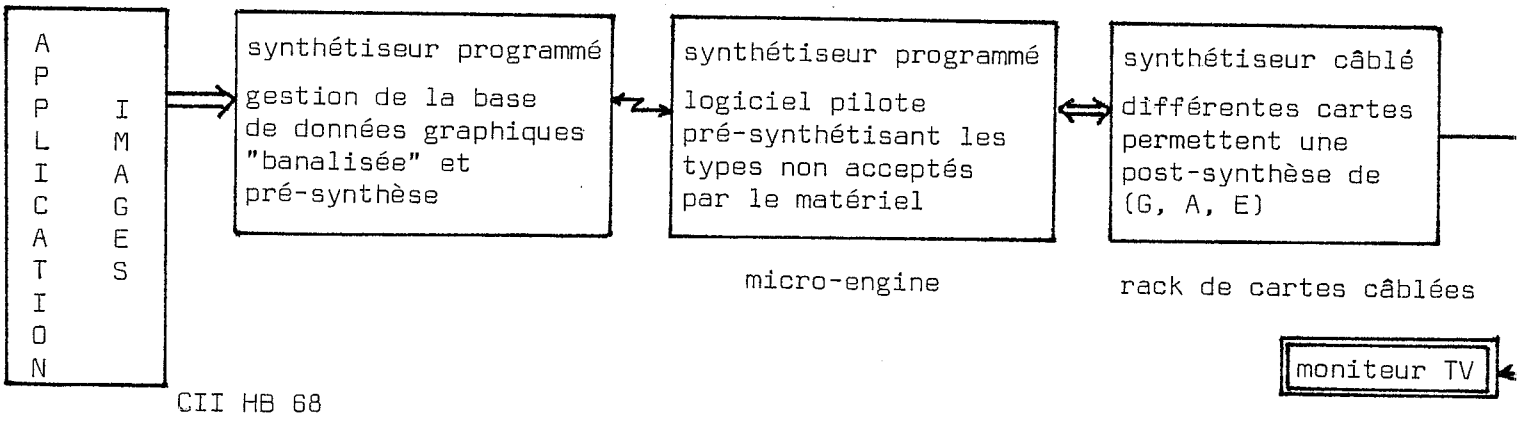


Figure 24 - HELIOS : un système composé de trois synthétiseurs

Nous allons successivement analyser ces trois synthétiseurs pour déterminer leurs univers initial et final. Ceci nous permettra d'en déduire les opérateurs qu'ils doivent réaliser, les bases de données nécessaires à chacun. Il va de soi que l'univers initial de l'un sera l'univers final du précédent et que la communication entre les différents synthétiseurs sera soigneusement étudiée pour minimiser les pertes de temps.

2.2. Concepts du logiciel de base à la lumière des logiciels existants

Cette étude [MAR 80] menée dans le cadre du projet CLOVIS (complexe logiciel pour la visualisation interactive), part de la constatation suivante : deux catégories de systèmes différents pilotent les matériels :

- les uns, systèmes généraux, permettent d'utiliser différents matériels pour différentes applications, mais les possibilités des matériels sophistiqués ne sont pas toujours utilisées ;
- les autres, systèmes spécifiques, sont adaptés à une application ou un matériel ; chaque nouveau matériel implique la construction d'un nouveau logiciel.

Le système logiciel conçu part des enseignements des différents logiciels de base existants. Il devrait permettre de satisfaire toutes les applications de conception assistée sur ordinateur, sur les consoles de visualisation disponibles au Laboratoire (Tektronix 4010, 4114, ... HELIOS).

Les deux couches de logiciel vont prendre en charge la partie du processus de visualisation non pris en charge par le synthétiseur câblé. *La répartition des processus est liée à la puissance de calcul et de mémorisation du calculateur pilote.* De façon symétrique, les deux logiciels accepteront les primitives suivantes :

- ATTRIBUER ( ), qui permet d'associer des attributs à un type d'information donné,
- COLLECTER ( ), qui permet de récupérer l'attribut d'un type d'information donné,
- VISUALISER ( ), qui permet la visualisation des éléments considérés,
- DECRIRE ( ), qui permet la description des attributs d'un type d'information donné.

Le logiciel sera cohérent, c'est-à-dire que les primitives s'adresseront à des entités de même niveau : les identités des éléments.

2.3. Les éléments retenus

2.3.1. Les types d'attributs

1) Des informations morphologiques

. *Les éléments "fils de fer"* pourront être exprimés en 2D ou 3D par :
- une suite de points,
- une suite de segments (origine et extrêmité de chacun),
- une ligne brisée (sommets),
- un cercle (centre et rayon),
- un message (suite de caractères).

. *Les éléments pleins* exprimés en 2D :
- une surface polygonale plane (sommets des polygones composant le contour),
- une surface circulaire (disque : centre + rayon).

. *Les éléments pleins* exprimés en 3D :
- des volumes composés de surfaces polygonales planes (sommets des polygones composant le contour),
- des surfaces gauches (fonction à deux variables du type z = f(x, y)).

2) Des informations d'aspect

. Les éléments "fils de fer" pourront avoir les paramètres suivants :
- le nom de la couleur,
- le nom de la texture de traits (pointillé, tireté court, tireté long, trait plein), ou jeu de caractères utilisés (majuscules, minuscules),
- la visibilité ou le clignotement.

. Les éléments pleins :
- le nom de la texture plane utilisée,
- le nom du modèle de réflexion,
- la visibilité, la transparence, le clignotement.

Notons que les aspects possibles dépendent énormément des possibilités du matériel.

3) Des informations géométriques

L'utilisation des coordonnées homogènes nous permet de représenter n'importe quelle transformation (rotation, homothétie, translation) par une matrice. Pour les éléments 2D ce sera une matrice 3x3, pour les éléments 3D, une matrice 4x4.

4) Des informations d'éclairage

Seule la configuration avec HELIOS admet ce type modélisé par :

. le nom d'une couleur unie,
. l'intensité de lumière ambiante,
. le vecteur directionnel de la source (X, Y, Z dans le repère de l'écran).

5) Des informations de prise de vue

Elles sont représentées par une matrice (4x4 pour maquette tri-dimensionnelle 3x3 pour maquette bi-dimensionnelle), exprimant en coordonnées homogènes les transformations (projections, rotations, ..) et par les limites que l'on impose à la scène.

6) Des informations d'affichage

Elles expriment la clôture dans laquelle l'image est affichée (origine du repère et taille de la clôture mesurée en 1/512ème d'écran).

2.3.2. Le déroulement des processus

Le logiciel se trouvant sur le micro-ordinateur permet des fonctions locales. Pour que ces fonctions locales soient d'un bon niveau, il faut que ce logiciel se charge d'un maximum d'opérations de synthèse. Nos ambitions seront cependant réduites à cause :

. de la taille de la mémoire (64 K) (même augmentée de la mémoire des deux disquettes souples (2 M octets),
. de la vitesse de calcul,
. de la vitesse des entrées-sorties avec la double unité de disquette.

Le partitionnement, qui tient compte de ces impératifs qui découlent de notre expérience, va être étudié dans les paragraphes suivants. Quelques remarques sont à la base de ce partitionnement :

. *Le logiciel pilote* doit s'occuper *de la visualisation et de la consultation du matériel,* permettant ainsi à la couche supérieure d'en être relativement indépendante.

. Le projet CLOVIS a montré qu'une *bonne représentation de la base graphique* était *de forme arborescente.* Celle-ci a surtout l'avantage de permettre l'association d'informations à un noeud quelconque (éventuellement à tout un sous-arbre), ce qui diminue la taille des données. *La gestion de cette base* n'est pas possible au niveau du calculateur pilote ; elle sera donc faite *par le calculateur principal.* Néanmoins, *une base de données intermédiaire* semble souhaitable sur le calculateur pilote. En effet, la liaison entre les deux calculateurs se faisant au maximum à 4 800 bauds, le temps de transmission des informations risque d'être prohibitif. La structuration de cette base locale peut se faire *sous forme linéaire.*

. *L'invocation des processus* doit se faire *sous forme codée pour* les commandes allant au *calculateur pilote* (pour diminuer le temps de transmission), *sous forme de primitives* (pas de problème de transmission) *pour le calculateur principal,* grâce au port parallèle pour le synthétiseur câblé.

Ces différences apparaissent sur la figure 25 :

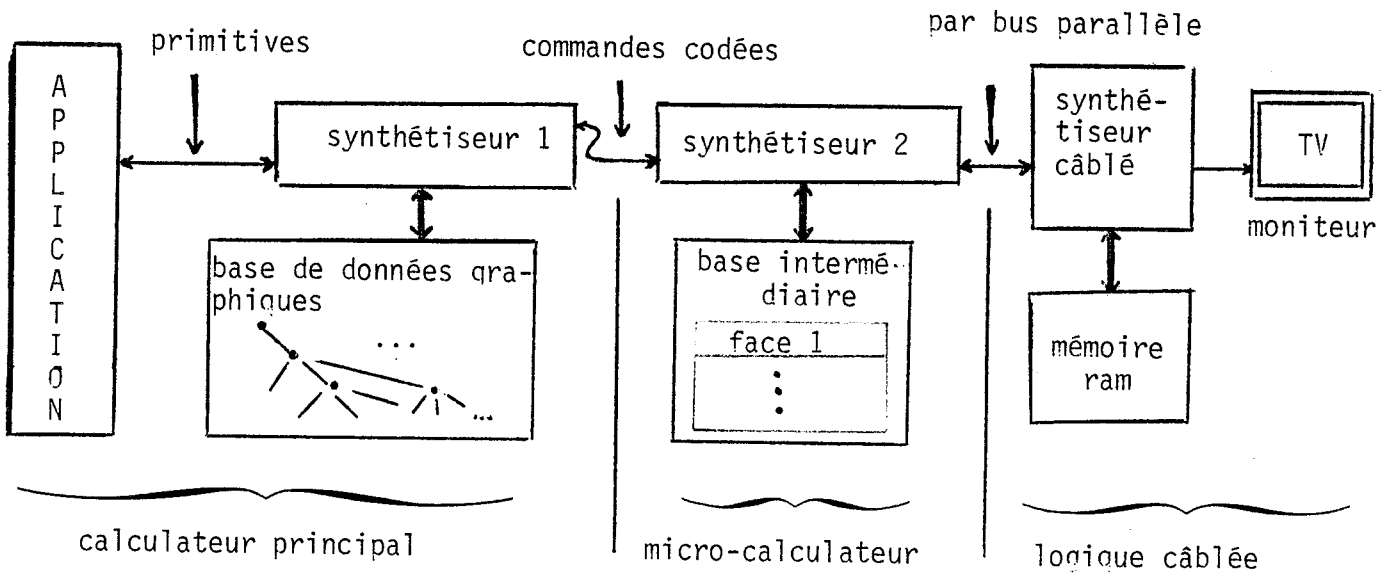


Figure 25 - Différences de transmission et modèles de structuration

2.4. Caractéristiques du synthétiseur programmé : le logiciel principal

Ce logiciel [MAR 80], implanté sur l'ordinateur CII-HB 68 du Centre Interuniversitaire de Calcul de Grenoble, est accessible par le jeu de primitives. Ces primitives sont en rapport avec les quatre processus de synthèse et constituent un noyau minimal. Les primitives de modélisation ne sont développées qu'au fur et à mesure des besoins.

Comme nous l'avions laissé entendre précédemment, le fichier graphique utilisé par CLOVIS a une structure arborescente banalisée, c'est-à-dire que

- le nombre de branches et le contenu des noeuds dépendent entièrement des besoins de la maquette ; ceci permet de considérer des maquettes simples ou complexes ;
- toutes les opérations peuvent être appliquées à l'un quelconque des noeuds, garantissant ainsi une grande cohérence ;
- les attributs peuvent être composés ; ainsi les transformations peuvent-elles être appliquées à un sous-arbre complet ;
- la désignation d'un élément permet de connaître la sous-structure à laquelle il appartient.

L'accès aux différents noeuds de la structure (codés sous forme d'un identificateur de 8 caractères au plus), se fait par l'utilisation de noms :

- relatifs : chemin à suivre à partir d'un noeud quelconque,
- absolus : chemin à suivre à partir de la racine,
- globaux : expression de plusieurs chemins,
- indicés : expression de sous-arbres identiques différenciés par un indice.

On trouvera *la syntaxe* dans [MAR 82] ou [MAR 80] ; elle fait référence à la construction d'un noeud et à son parcours.

La gestion du fichier graphique comporte des actions permettant *la structuration, le parcours, l'affectation, la recherche*. Pour simplifier la notation, il est possible de se définir un "contexte" dans lequel on travaille.

La représentation interne est schématisée par la figure 26 où :
TYPE exprime la classe et le type de l'attribut,
nbnoeuds exprime le nombre de noeuds utilisant cet attribut,
longueur exprime le nombre de mots utilisés pour modéliser l'information de l'attribut.

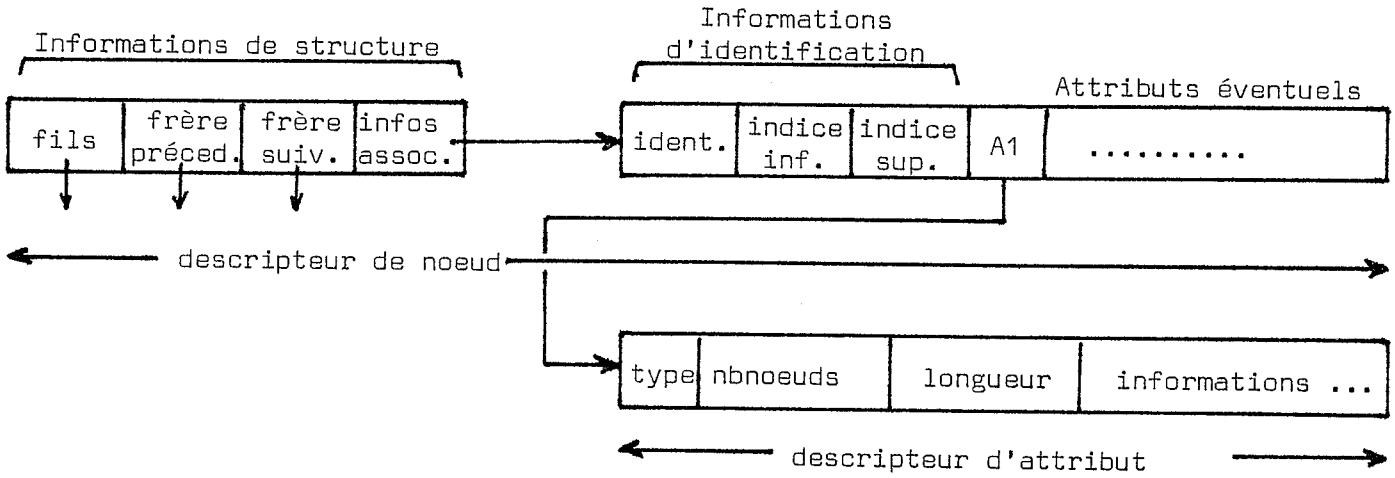


Figure 26 - . Descripteur de noeud pour chaque noeud de l'arbre
. Descripteur pour chaque attribut défini

De plus, pour optimiser les recherches et déterminer quelles sont les modifications ayant lieu depuis le dernier examen, l'unité de communication tient à jour pour chaque noeud un ensemble d'indicateurs permettant d'exprimer l'état de chaque attribut :

1. inexistant,
2. inchangé,
3. modifié,
4. modifié dans le sous-arbre descendant.

Des fonctions (EXISTE et MODIF) permettent de tester leurs états.

L'unité de contrôle gère l'ordonnancement des processus associés aux primitives proposées par le logiciel principal. Un axe de recherche intéressant est l'étude de toutes les situations possibles et l'ordonnancement qui en résulte. Ici l'ordonnancement est pré-défini.

Le processus d'attribution (invoqué par la primitive ATTRIBUER) utilise les opérateurs de l'unité de communication.

Le processus de consultation (invoqué par la primitive CONSULTER) permet de consulter des informations sur les attributs mémorisés.

Le processus de visualisation (invoqué par la primitive VISUALISER) effectue le parcours de l'arbre, afin de déterminer les attributs modifiés (ou détruits). Seules les données modifiées sont recalculées et transmises au calculateur pilote.

Le processus de description explicite permet de décrire de manière interactive les attributs affectés à un élément :
. soit par construction, en dessinant par exemple le contour de la face sur une tablette à numériser,
. soit par référence, en montrant à l'écran l'attribut que l'on veut utilise

Le processus de description implicite permet d'effectuer la description au moment de la visualisation.

2.5. Caractéristiques du synthétiseur programmé : le logiciel pilote

Si nous voulons effectuer en temps réel des opérations de synthèse nombreuse une importante mémorisation des attributs est nécessaire. La capacité de mémorisation relativement faible nous a amenés à privilégier des fonctions locales de haut niveau. Nous décrirons ici les hésitations qui ont été les nôtres lors de cette étude et nous donnerons les caractéristiques générales retenues pour ce logiciel pilote. Les chapitres 4 et 5 décrivent la réalisation effective sur le prototype HELIOS, en définissant les primitives d'accè et le contenu des opérateurs de synthèse.

La définition de l'univers initial de notre synthétiseur dépend des opérateurs de synthèse que nous implémenterons. La rapidité d'exécution de ces opérateurs est liée à la vitesse de transmission des informations à synthétiser. Tourné vers la conception assistée par ordinateur, privilégiant l'esthétique, le système doit intégrer les informations d'aspect et d'éclairage à ce niveau.

Pour modéliser *l'aspect*, l'un des types d'information est la texture colorée. Constituer une banque de textures implique la réalisation de modules de description, d'archivage et de consultation de ces textures. Ces trois processus distincts ont leur place sur le micro-calculateur :

- *la banque de données* sera conservée sur disquette,
- *le logiciel de description* et d'archivage [PAY 82] fait l'objet de programmes locaux au calculateur ; ils intègrent la description d'une palette, les opérateurs de composition de couleurs et les algorithmes de description de textures particulières (bois, granité, feuillages, ...). Intégrer cette description au logiciel représente une surcharge et diminue les extensions possibles d'algorithmes par type de texture (unie, à rayures, aléatoire, granulée, bois, ...) ;
- *la consultation* de la banque est intégrée au calculateur (figure 27)

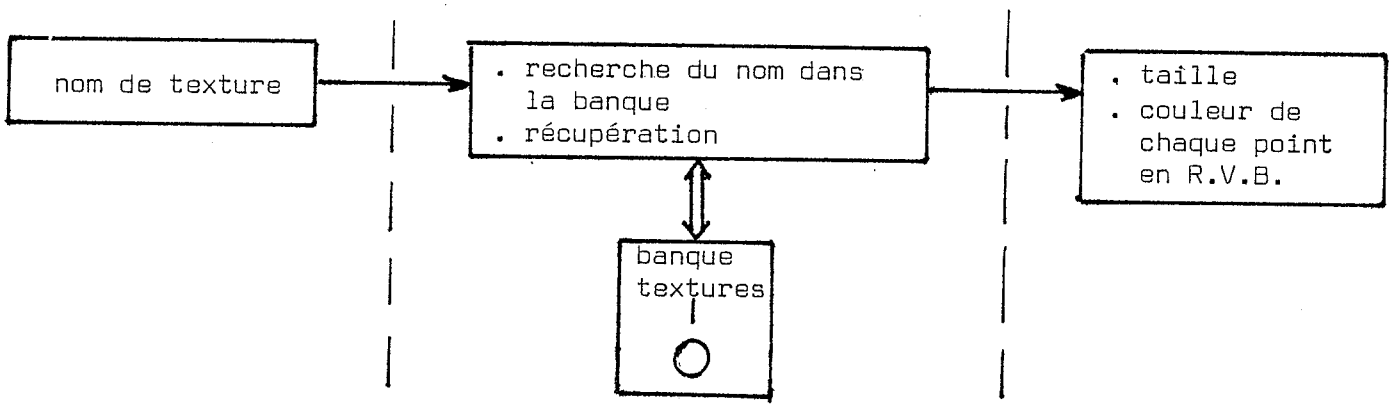


Figure 27 - Module de consultation de la base de données des textures implantée sur disquette

La modélisation de la *réflexion*, de la même façon, fait appel à un programme de description, indépendant du logiciel pilote, à une banque de données de ces modèles et un module de consultation intégré au logiciel.

En début de travail il serait souhaitable de pouvoir initialiser le système en indiquant :
. les textures utilisées,
. les modèles de réflexion autorisés.
Les autres types d'informations d'aspect et d'éclairage sont directement assimilables par le matériel, sauf le clignotement qui ne sera pas autorisé.

Le type d'élément que nous souhaitions implanter est constitué par des *faces planes, superposées, 3D.* Mais l'expérience nous a montré que deux facteurs limitent cette exigence (cf. annexe 3) :
1) les temps de calcul pour l'affichage (la suppression d'une face implique des comparaisons avec les faces précédentes pour afficher les faces visibles et la gestion de différents pointeurs),
2) la place mémoire utilisée.

Aussi, préférant utiliser les différents plans d'identification pour la superposition, avons-nous abandonné cette notion. De plus, c'est après la prise de vue que sont appliqués les algorithmes d'élimination des parties cachées. Ne pouvant avoir de tels algorithmes sur notre micro-calculateur, nous avons donc décidé de considérer simplement *comme éléments de base : le type face plane et le type élément "fil de fer".*

Nous conservons les informations morphologiques sur disquette, car leur lecture est plus rapide que leur transmission par ligne. De plus, elles permettent d'augmenter la puissance des fonctions locales.

Un type d'information concernant l'affichage sera mémorisée sous forme de matrice. Cette matrice 3x3 exprimera en coordonnées homogènes les transformations (rotation sur la surface de l'écran, translation, zoom) de la vue.

Le logiciel assure donc :

- la transmission des informations jusqu'au matériel, la consultation de ces informations,
- la visualisation des éléments enregistrés à la demande,
- la description des coordonnées à l'aide de points (obtenus au clavier, à la tablette à numériser ou au réticule intégré dans HELIOS).

Les programmes locaux complémentaires permettent :

- la description de textures et de modèles de réflexion,
- la récupération d'images sur disquette,
- la génération d'objets (exemple : objets de résolution) ou de surfaces courbes : la modélisation se faisant sous forme de faces polygonales planes.

2.6. Le prototype HELIOS : synthétiseur câblé

Ne trouvant sur le marché aucune console de visualisation répondant aux objectifs qu'elle s'était fixés, l'équipe Graphique du Laboratoire IMAG a construit son propre prototype, grâce à une étude menée par F. FERREIRA et F. MARTINEZ [FER 81], [MAR 81] et l'aide du Laboratoire de Micro-Electronique. Des fonctions de synthèse d'éclairage, d'aspect, ont été intégrées.

L'univers final est la couleur de chaque point communiquée au moniteur TV.
L'univers initial comporte des types appartenant aux classes d'information suivantes :

- identification - morphologie - géométrie - aspect - éclairage.

L'élément de base est la surface plane.

L'attribut de type "identification" est le numéro de face.
Les attributs d'aspect sont pour chaque face :
- la texture
- la visibilité
- la réflexion

L'attribut géométrique est représenté par une matrice exprimant la projection du repère 3D de la face dans le plan de vue.

Les attributs d'éclairage sont représentés par la description de la source lumineuse (position, couleur, lumière ambiante diffusée).
Les attributs associés à l'affichage permettent de décrire la fenêtre associée à l'écran (translation, zoom).

Les opérateurs intégrés, décrits plus précisément au chapitre 3, réalisent par logique câblée les processus suivants :
. attribution et consultation des attributs de l'univers initial,
. description et consultation, à l'aide d'un réticule intégré, des coordonnées écran d'un point,
. visualisation en temps réel des faces enregistrées.

Trois banques de données sont nécessaires :
. la mémoire de trame représentant un ou plusieurs plans d'identification,
. la mémoire de textures,
. la banque des modèles de réflexion.

Nous verrons au chapitre 3 comment sont modélisés les divers types d'information intégrés à ce synthétiseur. Notons que pour réaliser les divers processus décrits, tout a été fait par logique câblée.

2.7. Conclusion

La description de ce système nous permet de montrer qu'il est à la fois complexe et cohérent. De nombreuses possibilités existent et leur développement est en cours. Le logiciel pilote, réalisé en PASCAL sur le micro-ordinateur "micro-engine" de WESTERN DIGITAL, sera défini plus complètement au chapitre 4. Les fonctions locales envisagées en fontun outil pratique pour la conception assistée par ordinateur, mais la véritable dimension de cet outil est bien l'utilisation du système tout entier.

★ ★

CHAPITRE 3

PRÉSENTATION D'HELIOS

Pour décrire toute la partie d'interface logiciel pilote - HELIOS, nous avons besoin de bien connaître le principe de fonctionnement de la partie matérielle et en particulier ses moyens de communication avec l'extérieur. Ce troisième chapitre nous permettra donc de faire connaissance avec ce prototype pour l'utiliser de façon aisée et performante [FER 81], [MAR 81], [MAR 82].

3.1. Principe de fonctionnement

On distingue deux modes de fonctionnement (figure 28) :

- le mode rafraichissement de l'écran où HELIOS calcule au rythme vidéo l'image nécessaire au rafraichissement d'un moniteur couleur à balayage de trame, grâce aux données stockées dans ses mémoires et registres ; ce mode correspond au *processus de visualisation.*

- le mode accès extérieur où des accès extérieurs viennent modifier ou lire le contenu des mémoires ou registres. Il intègre les *processus d'attribution et de consultation. Le processus de description* est réalisé à l'aide d'un réticule intégré au terminal.

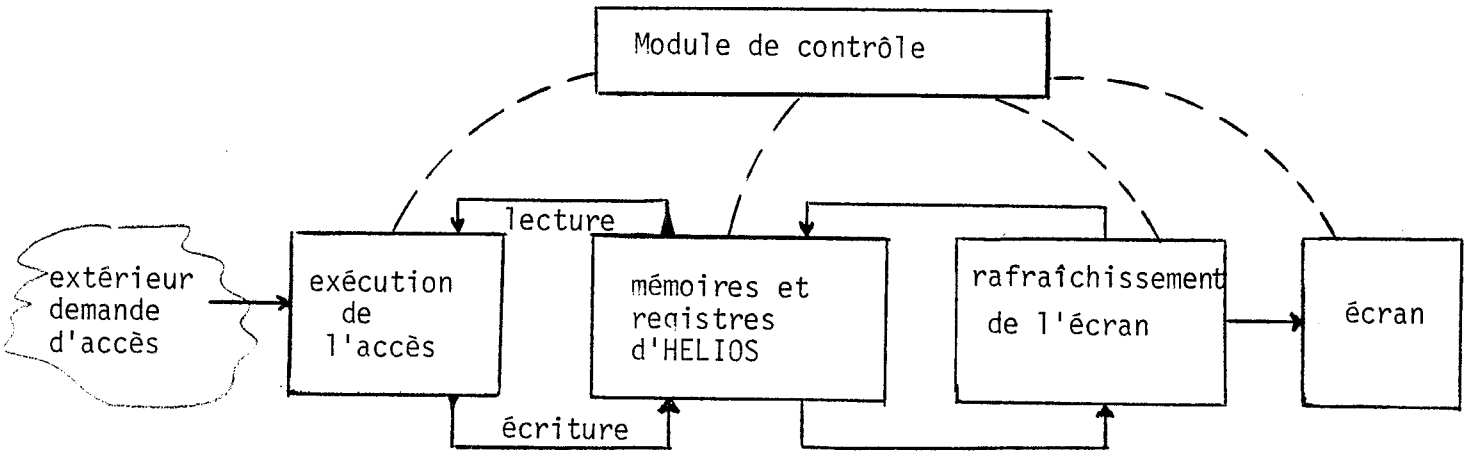


Figure 28 - Les modes de fonctionnement avec les signaux de contrôle

3.2. Le processus de visualisation

Nous avons défini au § 2.6. les éléments de l'univers initial du prototype : ce sont des faces planes acceptant des attributs d'aspect (texture, réflexio visibilité), de géométrie (repère tri-dimensionnel attaché à la face) ; des informations d'éclairage et d'affichage sont aussi définies.

Le processus de visualisation assure :
. la pénétration des attributs (A) et (E),
. la mémorisation des attributs synthétisés.

Il a besoin :
.d'opérateurs de synthèse des attributs de base (▷)
. d'opérateurs de composition inter-éléments (▱)
et peut être représenté par la figure 29 :

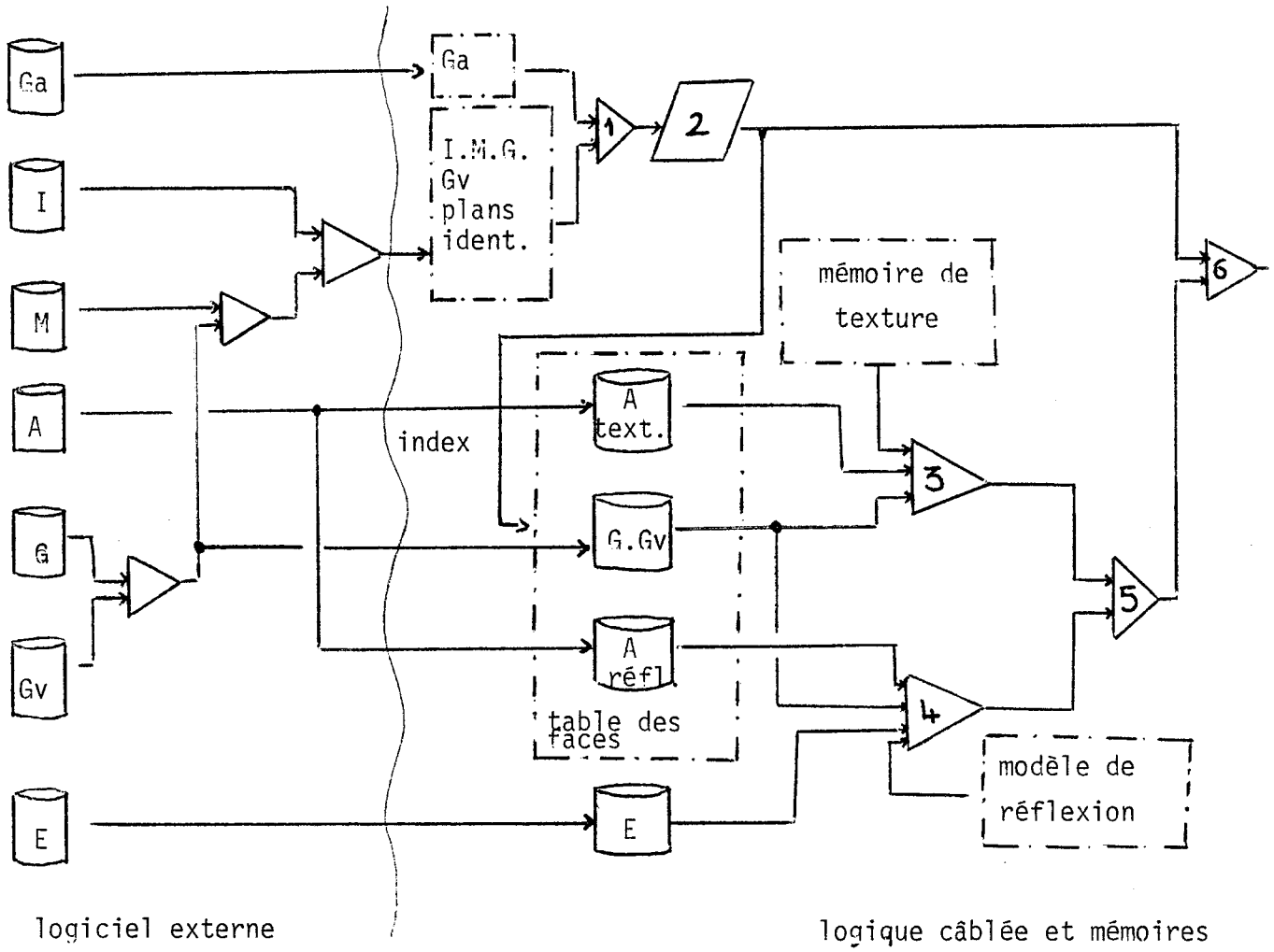


Figure 29 - Schéma de la synthèse sur HELIOS

3.2.1. Les structures de données associées

1) Les plans d'identification

La mémoire de trame est composée de différents plans, qui contiennent les données d'identification de l'image à visualiser (figure 30) :

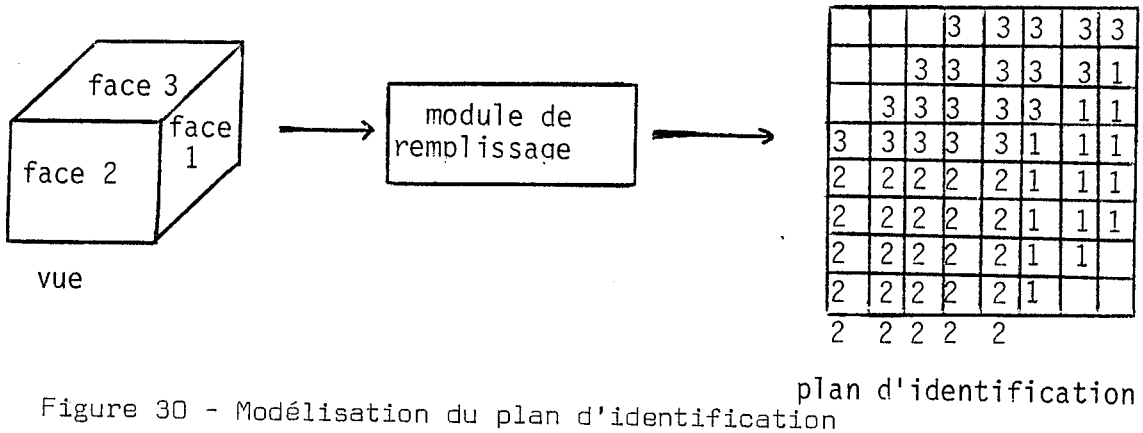


Figure 30 - Modélisation du plan d'identification

Ces données sont représentées sur 10 bits permettant donc 1024 faces possibles (de 0 à 1023).
Le nombre de plans est limité à 7, plus un plan de fond, constitué d'un numéro de face unique et permettant la visibilité du fond pour tout l'espace où les numéros de faces ne sont pas attribués.

2) Les paramètres d'affichage

Pour chaque plan d'identification, on pourra définir une fenêtre grâce à deux registres qui mémorisent :

- la taille de celle-ci (zoom à 4 niveaux en x et y),
- le coin supérieur gauche d'une fenêtre à côtés parallèles aux bords de l'écran (translation en x et y de 0 à 512).

3) La table des faces

Elle contient des informations définissant pour chacune des 1024 faces (figure 31) :

- son matériau,
- son repère dans l'espace 3D.

adresse	contenu			
0				
⋮				
n	brillance	pointeur texture	normale	matrice projection
1023				

Figure 31 - Contenu de la table des faces

. La brillance :

quatre niveaux de brillance sont possibles (codés de 0 à 3). Lors de l'initialisation du système, il faut charger le modèle de réflexion qui définit les niveaux utilisés (figure 32).

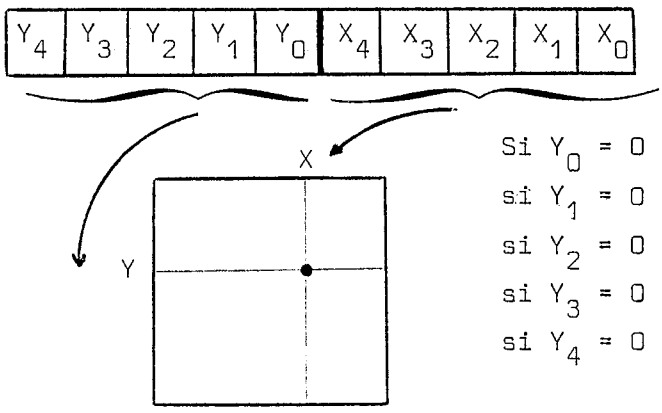
brillance	0	1	2	3
codage	mat	semi-mat	satiné	brillant

Figure 32 - Exemple de codage

. Le pointeur de texture :

une texture colorée est modélisée dans une matrice carrée, stockée dans la mémoire de texture. Elle a une taille qui varie de 16x16 à 512x512. Le pointeur de texture (codé sur 10 bits) définit (figure 33) :

- le début de la texture
- la taille de celle-ci

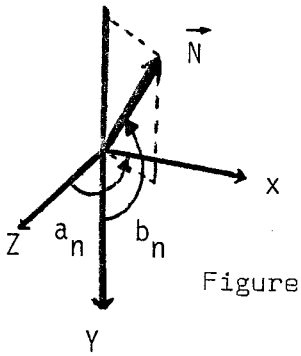


Si Y_0 = 0 ou X_0 = 0 alors taille 16x16 sinon
si Y_1 = 0 ou X_1 = 0 alors taille 32x32 sinon
si Y_2 = 0 ou X_2 = 0 alors taille 64x64 sinon
si Y_3 = 0 ou X_3 = 0 alors taille 128x128 sin
si Y_4 = 0 ou X_4 = 0 alors taille 256x256 sin
taille 512x512

Figure 33 - Pointeur de texture

. Le vecteur normal :

il est représenté en coordonnées sphériques par rapport au repère associé à l'écran (figure 34). Cinq bits sont réservés pour représenter chacun des deux angles, ce qui autorise 1024 directions dans le demi-espace avant de l'écran.



a_n varie de $-\pi/2$ à $\pi/2$

b_n varie de 0 à π

Figure 34 - Vecteur normal

. La matrice de projection :

la projection de la texture prend en compte le repère de la face dans l'espace 3D, c'est-à-dire que pour chaque point de l'écran, il faut déterminer le point correspondant dans le plan de la face. Une matrice de changement de base, précalculée par logiciel, est enregistrée pour chaque face. Cette matrice est représentée sur 22 bits (figure 35).

	a		b			c			d	
	exposant	mantisse	expos.	signe	mant.	expos.	signe	mant.	exposant	mantisse
bits →	1	4	1	1	4	1	1	4	1	4

valeur coefficient = (-1) ** signe * mantisse ** (2 * exposant - 3)

Figure 35 - Matrice de projection

4) La mémoire de texture

De taille 512x512, elle permet de mémoriser des textures carrées de tailles différentes (figure 36). La notion de texture comporte trois originalités :

. chaque point exprime la couleur intrinsèque de la face indépendamment des conditions d'éclairage. La couleur est modélisée en R, V, B, et codée sur 12 bits :

codage	11 10 9 8	7 6 5 4	3 2 1 0
valeur	0 ⟶ 15	0 ⟶ 15	0 ⟶ 15
	VERT	ROUGE	BLEU

. la texture est supposée définie dans le plan de la face,

. toutes les textures mémorisées peuvent être utilisées simultanément.

Chaque texture est donc une simulation visuelle de l'aspect d'un matériau donné. Pour toutes les faces utilisant ce même matériau, une seule texture est nécessaire.

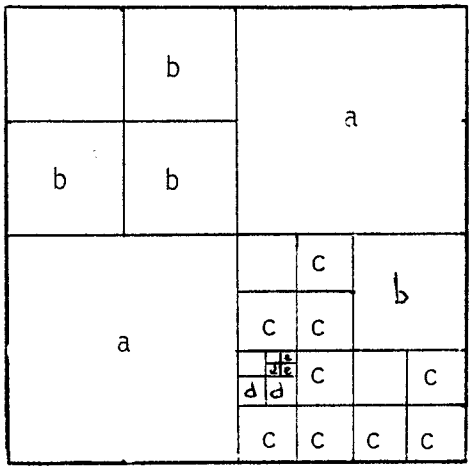


Figure 36 - Configuration de la mémoire de texture avec :
a : texture 256x256
b : texture 128x128
c : texture 64x 64
d : texture 32x 32
e : texture 16x 16

5) Les modèles de réflexion

Quatre modèles de réflexion peuvent être enregistrés, correspondant aux quat types de brillance que l'on veut utiliser.

Chaque modèle est une table exprimant à la fois la réflexion diffuse et la réflexion spéculaire. Chaque valeur de la table est exprimée entre 0 et 1 et donne le coefficient de réflexion en fonction de l'angle (figure 37).

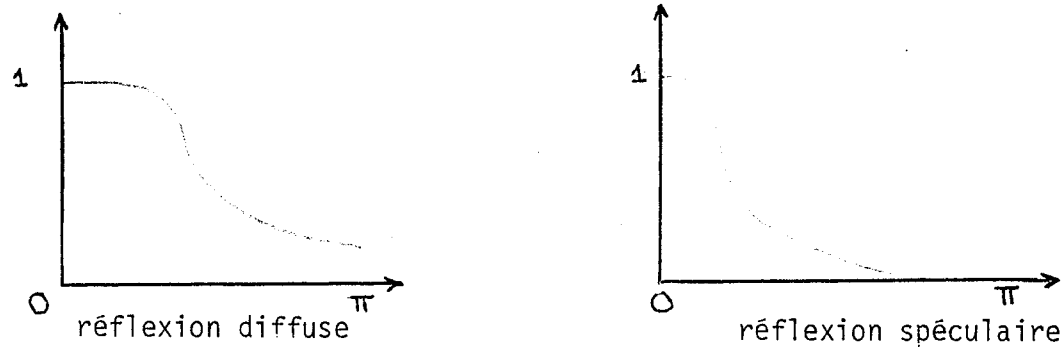


Figure 37 - Modèle pour un matériau "brillant"

6) Les paramètres d'éclairage

. La couleur de la source est codée en R,V,B sur 12 bits :

11	10	9	8	7	6	5	4	3	2	1	0
VERT				ROUGE				BLEU			

. la position de la source est donnée en coordonnées sphériques. Elle est considérée à l'infini et admet 2048 directions possibles :
l'angle as est codé sur 6 bits (0 : - π/2 ... 63 : 2π)
l'angle bs est codé sur 5 bits (0 : 0 ..... 31 : π)

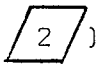
. la lumière ambiante diffuse codée sur 4 bits, peut prendre 16 niveaux différents,
. la position médiane entre la source et l'observateur est calculée par le système et codée en coordonnées sphériques comme la position de la source.

7) La table de visibilité des faces

Chaque plan possède une table de visibilité des faces, de taille égale au nombre de faces possibles (soit 1024). Elle contient l'indicateur de visibilité de chacune d'entre elles, codé sur 1 bit (0 : invisible, 1 : visible). Cet indicateur pourrait être porté à 2 bits dans les extensions futures, avec le codage (0 : invisible, 1 : visible, 2 : transparent), car c'est à ce niveau que l'on pourrait facilement simuler la transparence par matériel.

3.2.2. Les opérateurs de visualisation

Ils sont représentés sur la figure 33 par des numéros allant de 1 à 6. Nous allons les examiner.

1) Le calcul de visibilité () et ()

Il est composé de deux opérateurs :

. la fonction fenêtre (opérateur de synthèse )

. l'étude de visibilité (opérateur de composition )

. La fonction fenêtre :

permet de déterminer la partie de la vue du plan que l'on va afficher en fonction de la translation, du zoom et du mode de coupage demandé sur ce plan (figure 42). L'espace d'adressage de chaque vue peut s'étendre de 0 à 1023. Cette fonction permet un fonctionnement en configuration réduite (512x512, 256x256, 128x128, 64x64). Le coin de la fenêtre peut défiler sur la vue (en temps réel) pour permettre de cadrer une zone intéressante, un zoom permet alors de modifier la taille de la fenêtre et le mode de coupage permet la visibilité ou l'invisibilité de la vue en dehors de la fenêtre. Ces possibilités combinées (translation des plans les uns par rapport aux autres, juxtaposition, superposition), permettent une certaine animation. Cette fonction (figure 38) s'exécute en parallèle pour chaque plan d'identification.

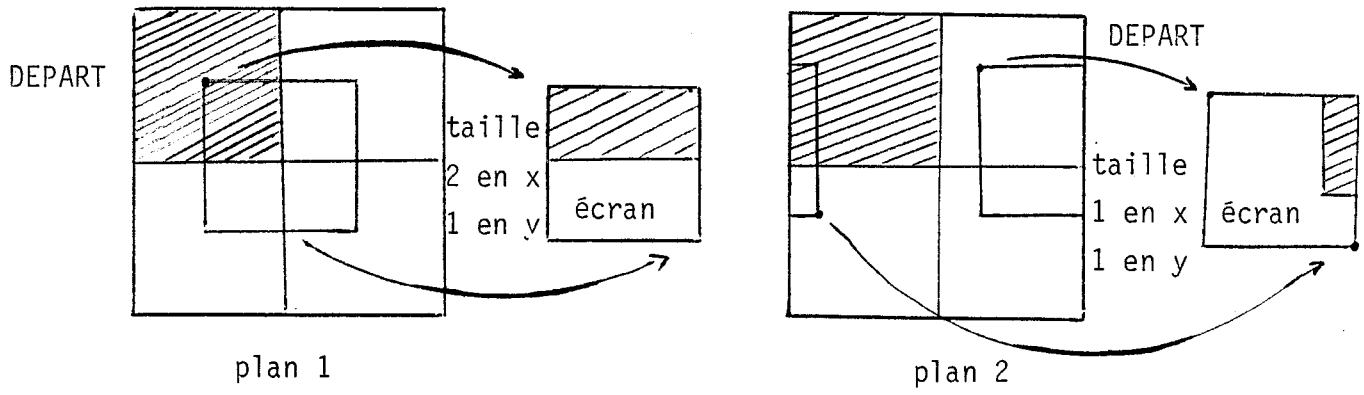


Figure 38 - Exemple de fenêtre avec mode de coupage invisible

. L'étude de visibilité :

l'existence de plusieurs plans nécessite la détermination pour chaque point de l'écran, de celui qui doit être visualisé. C'est donc le point de convergence des opérateurs de synthèse parallèles précédents. L'étude prend en compte dans l'ordre :

1) la priorité des plans d'identification (figure 39) numérotés de 0 à 7

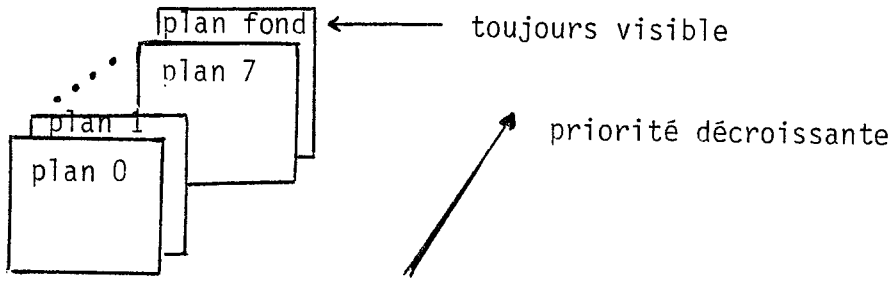


Figure 39 - Priorité de visualisation des plans

2) Le mode de coupage de la fenêtre,

3) la table de visibilité des faces.

Pour chaque point de l'image, si le plan i est visible, on regarde la table des faces ; si la face est invisible, on passe au plan i+1, ceci jusqu'au plan de fond qui contient le numéro d'une face toujours visible. Le résultat de ce calcul est donc *le numéro d'une face visible*.

2) Le calcul des textures

Cet opérateur de synthèse se décompose en deux opérations : projection et pavage.

. La projection des textures :

la texture est définie dans le plan de la face. Il faut donc déterminer pour chaque point de la vue (Xe, Ye), le point (Xf, Yf) correspondant dans le plan de la face. Ceci est réalisé grâce à la matrice de changement de base, précalculée par logiciel et enregistrée dans la table des faces. On obtient :

$$\begin{vmatrix} Xf \\ Yf \end{vmatrix} = \begin{vmatrix} a & b \\ c & d \end{vmatrix} \times \begin{vmatrix} Xe \\ Ye \end{vmatrix}$$

N'ont été envisagées que les projections considérant le point de vue à l'infini. Par ailleurs, seuls les coefficients b et c sont signés, ce qui implique qu'il ne peut y avoir de retournement de la face. Si cela est, le logiciel pilote doit effectuer une symétrie des vecteurs de base. Une vue de profil, cas limite, provoque des erreurs importantes dans l'échantillonnage de la texture, aussi n'est-il pas apparu nécessaire d'admettre des coefficients dont les valeurs absolues sont supérieures à 7,5.

. Le pavage des textures :

après avoir obtenu les coordonnées dans le plan de la face, il faut calculer le point résultant dans la mémoire de textures. Ceci est réalisé grâce au pointeur de texture de coordonnées (Xa, Ya) enregistrées dans la table des faces. Les coordonnées (Xb, Yb) dans la mémoire de texture sont obtenues par :

$$\begin{vmatrix} Xb \\ Yb \end{vmatrix} = \begin{vmatrix} Xa \\ Ya \end{vmatrix} \times \text{taille texture} + \begin{vmatrix} Xf \\ Yf \end{vmatrix} \times \text{modulo taille}$$

La lecture du point (Xb, Yb) nous donne *la couleur associée.*

Le pavage réalisé par matériel évite les problèmes évoqués par la figure 40

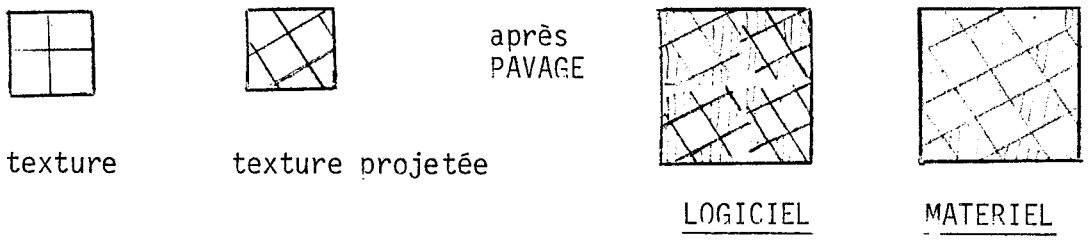


Figure 40

3) Le calcul de la réflexion

Le module d'éclairage retenu (proche de l'algorithme de B.T. PHONG [PHO 75]) ne considère que les phénomènes de réflexion diffuse et spéculaire. L'éclairement d'un point est approché par la somme de la somme de trois termes :

$$E = f(s) + g(m) + L$$

f(s) simule la diffusion du matériau, il est fonction de l'angle d'incidence s entre la normale et la direction de la source,

g(m) simule la réflexion ; il est fonction de l'angle m entre la normale et la direction médiane entre la source et l'observateur,

L simule la lumière ambiante.

Les opérateurs élémentaires sont donc le calcul des angles m et s et la modélisation de la réflexion.

· Le calcul des angles :

fait appel à la normale à chaque face mémorisée dans la table des faces et la position de la source mémorisée dans les paramètres d'éclairage. Les calculs faits ont été un point délicat de l'étude et le lecteur trouvera dans [FER 81], [MAR 82] les détails de cette réalisation.

· Le calcul de la réflexion :

fait appel aux angles calculés précédemment, aux paramètres d'éclairage, à la brillance enregistrée dans la table des faces, et aux modèles de réflexion pré-enregistrés [FER 81], [MAR 82]. Il permet d'obtenir les coefficients de réflexion diffuse Rd et speculaire Rs.

4) Le calcul de l'éclairage et de l'affichage

Il se fait grâce :

- à la couleur Ct du point issu du calcul des textures,
- aux coefficients de réflexion diffuse Rd et spéculaire Rs,
- à la couleur Cs de la source et à l'intensité L de la lumière ambiante enregistrée dans les paramètres d'éclairage.

La couleur finale C est obtenue comme suit :

$$C = ((L + Rd) * Ct + Rs * B) * Cs$$

où B représente le blanc pur.
Ceci se fait à l'aide des trois équations représentant les composants (R,V,B).
Le résultat pour chaque point est donc une couleur qui est transmise au terminal TV par un module d'affichage.

3.3. Le processus de dialogue intégré : le réticule

La description sur HELIOS se fait à l'aide d'un réticule intégré de couleur (parmi les 4096 couleurs admises). Pour modifier l'emplacement du réticule il faut :

. choisir sa couleur,

. appuyer sur un ensemble de touches directement relié au matériel (fig. 41)

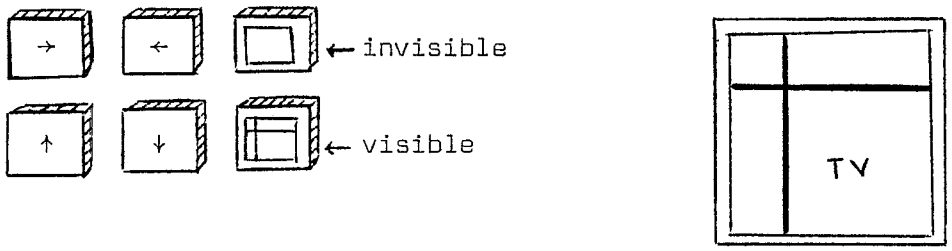


Figure 41 - Touches de contrôle du réticule

Les coordonnées du point désigné par le réticule sont récupérées en lisant un registre. Elles sont définies dans le repère de l'écran (0 ... 511).

3.4. Processus d'attribution et de consultation

L'attribution (écriture des données dans les mémoires ou registres) et la consultation (lecture de ces données), sont assurées par deux processeurs spécialisés :

. le processeur de contrôle et d'interface,

. le processeur de communication.

HELIOS apparaît comme un ensemble de modules programmables (figure 42).

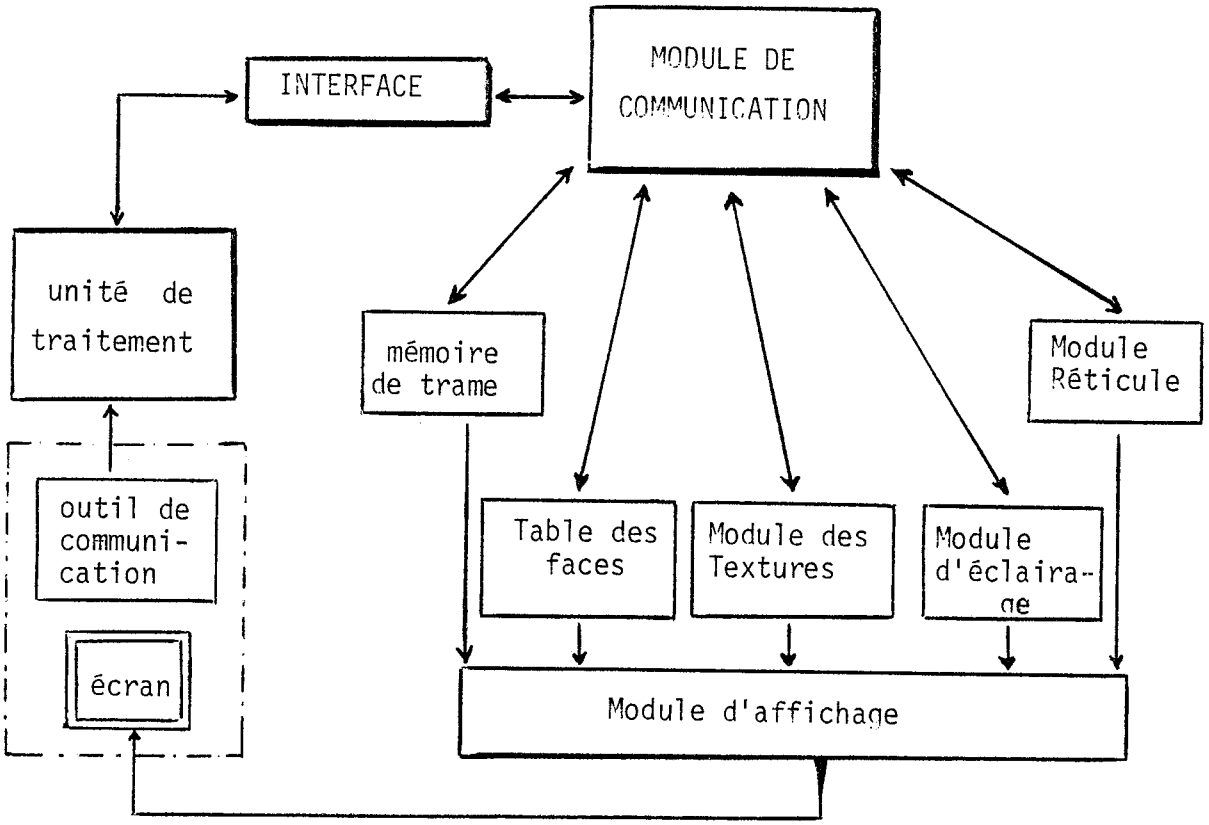


Figure 42 - Architecture d'HELIOS

1) Le module d'interface

Il assure la compatibilité entre la largeur du bus de données d'HELIOS et le bus de l'unité de traitement associé, c'est-à-dire que n'importe quelle unité de 8, 16, 24 ou 32 bits peut être connectée à HELIOS. Cette connexion est une connexion parallèle et les entrées-sorties se font simplement :

LECTURE(adresse registre d'HELIOS, variable lue)

ECRITURE(adresse registre d'HELIOS, variable à écrire).

2) Le module de communication

Il permet l'accès aux différents registres et mémoires d'HELIOS. Ce processeur utilise pour cela sept registres dont les plus importants sont :

- le registre module qui indique la nature de l'accès (lecture/écriture), la mémoire ou le registre à accéder (table des faces, mémoire de texture, ...), le processeur concerné par l'accès ;

- le registre de compactage qui contient un nombre nb(0 à 511) qui permet de répéter une action nb fois sans intervention du calculateur pilote. Ceci est intéressant pour l'affectation de numéros de faces identiques au plan d'identification ou d'informations identiques à des faces de la table des faces ;

- le registre adresse sert de tampon au bus d'adresse interne du terminal. A chaque accès (lecture/écriture), il est incrémenté automatiquement.

- le registre entrée définit l'information à écrire sur 1, 2 ou 3 octets (format compatible avec la mémoire concernée). Il sert de registre tampon quand l'écriture intervient nb fois ;

- le registre sortie contient l'information lue sur 1, 2 ou 3 octets.

Ces registres permettent une communication plus aisée avec HELIOS. La modélisation des paramètres de ces registres qui conditionnent l'accès, est donnée par F. FERREIRA [FER 81].

Toutes les données attribuées pourront être consultées par l'opération LECTURE (adresse mémoire ...). Pour connaître un numéro de face contenu dans l'image affichée, on pourra par exemple positionner le réticule puis lire la mémoire concernée.

3.5. Conclusion

La réalisation de ce prototype a été faite avec une configuration réduite (1 plan d'identification 256x256). De nombreux artifices ont dû être trouvés pour qu'une image complète puisse être affichée dans le temps d'une trame :

- *l'affichage asynchrone* qui permet le calcul d'un point et le remplissage d'un tampon en 125 ns, alors que le vidage de ce tampon peut se faire en 70 ns ;

- *l'utilisation d'une architecture pipe-line* où le traitement est décomposé en étapes élémentaires ; mais le temps d'accès aux plans d'identification et à la mémoire de texture est de l'ordre de 300 ns. Quatre points sont donc lus ou écrits à chaque accès, ce qui provoque des problèmes pour les textures verticales n'ayant pas quatre points consécutifs de même couleur ;

- *l'utilisation du parallélisme de la synthèse*. Le calcul des textures et le calcul de la réflexion sont fait parallèlement.

CHAPITRE 4

LE LOGICIEL PILOTE IMPLANTÉ SUR HÉLIOS

VU DE L'UTILISATEUR

Pour prendre en compte au maximum les possibilités du matériel, il faut prévoir un ordonnancement de synthèse pour chaque type de console de visualisation. Il serait souhaitable d'arriver à définir un schéma de synthèse de façon automatique qui, en fonction des opérations réalisées par le dispositif d'affichage, engendre celles qui doivent l'être par programme. Le logiciel mériterait alors vraiment son nom de logiciel pilote. Ces travaux dépassent néanmoins l'objectif de ce mémoire.

Notre logiciel pilote n'accepte qu'un seul dispositif de visualisation : le prototype HELIOS. Nous avons donc défini quel devait être le cheminement des informations ot où devaient avoir lieu les opérations de synthèse non intégrées dans le matériel.

Les différents modules du logiciel, ceux décrivant la synthèse ou ceux permettant la gestion des données, ou certains modules d'interface, pourraient être ré-utilisés pour d'autres dispositifs de visualisation ; c'est pour cela que nous donnerons, dans ce chapitre, les principales primitives générées.

4.1. Présentation et principe de fonctionnement

Nous avons défini les caractéristiques de notre logiciel au chapitre 2. En tant que synthétiseur, il a comme univers initial, des faces planes. Le processus de visualisation comprend quelques opérateurs de synthèse et peut être représenté par la figure 43.

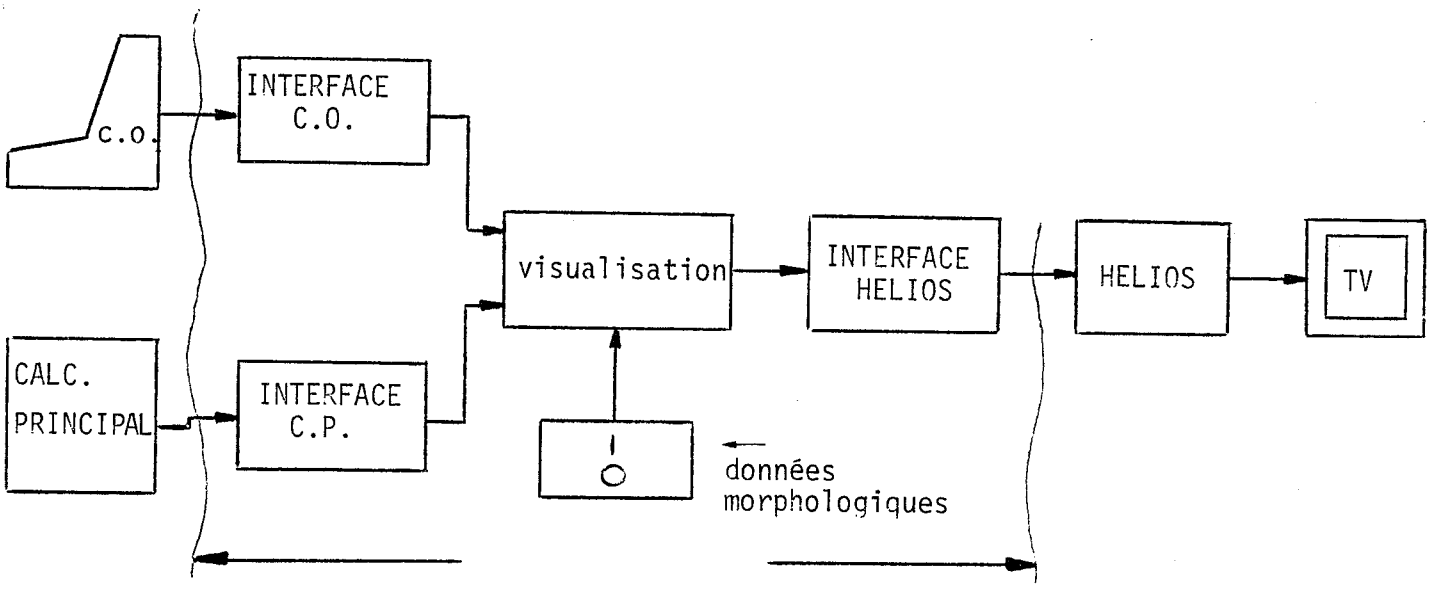


Figure 43 - Le processus de visualisation

Le processus d'attribution et de consultation (figure 44) doit permettre :

. la gestion des données morphologiques,

. la gestion des informations d'aspect (texture, réflexion),

. la communication et la consultation des informations du matériel.

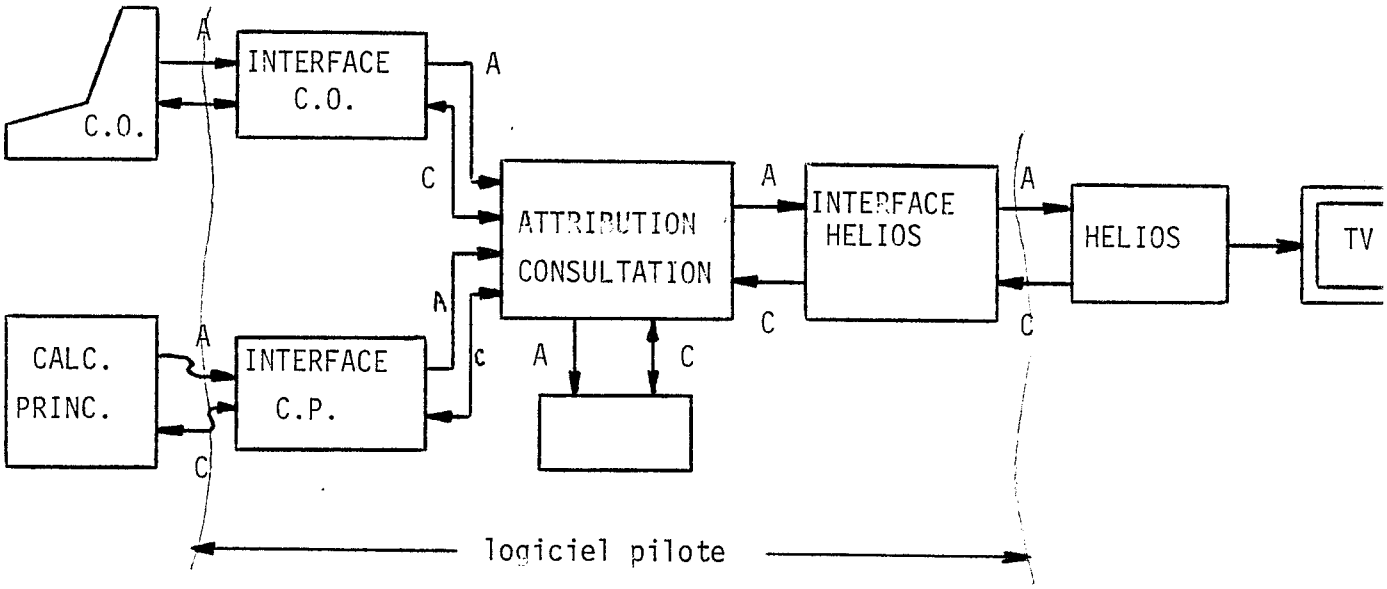


Figure 44 - Processus d'affectation (A) et de consultation (C)

Le processus de description permet la récupération de points 2D grâce à la tablette à digitaliser, au réticule câblé d'HELIOS, ou à la console opérateur (figure 45).

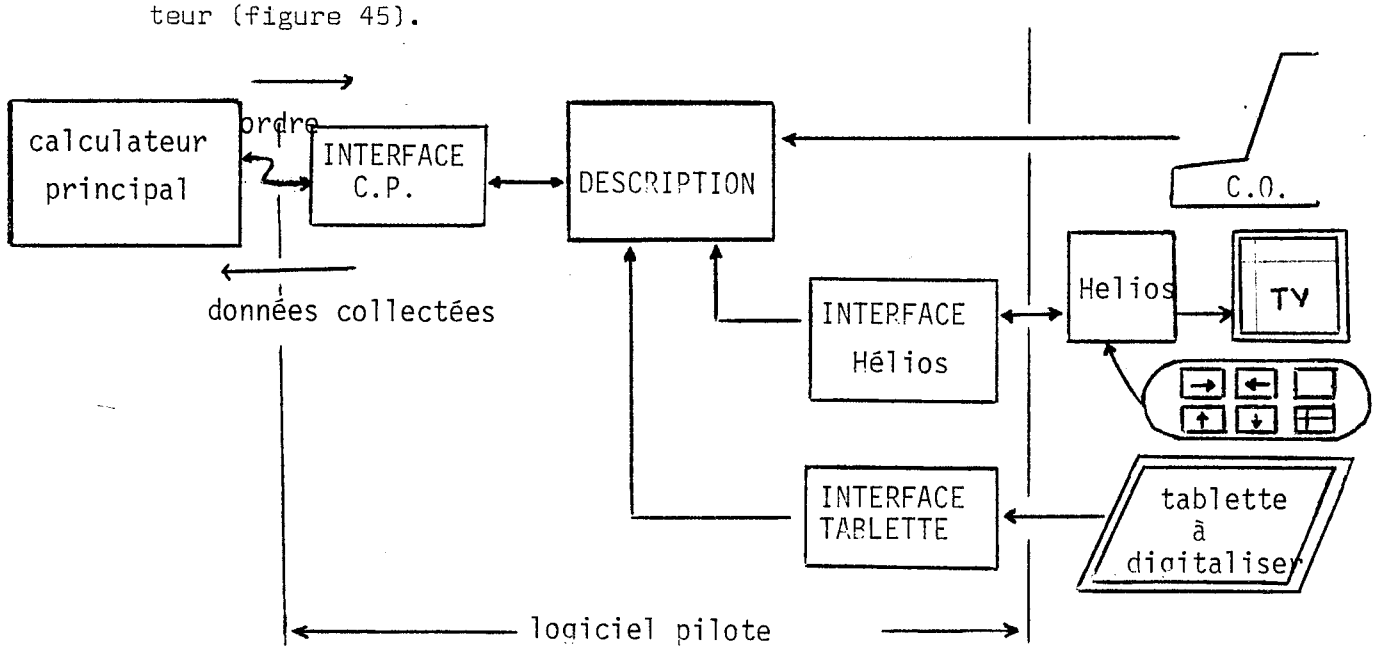


Figure 45 - Processus de visualisation

Des programmes locaux permettent la description de textures [PAY 82] et leur archivage sur disquettes (idem pour les modèles de réflexion).

Trois modes de fonctionnement sont prévus pour le logiciel pilote :

. *le mode "connexion"* :
qui permet une liaison directe MULTICS - CONSOLE OPERATEUR. Le logiciel pilote est transparent et renvoie simplement à MULTICS (port B du Micro-Engine) les informations saisies sur la console opérateur (connectée au port A du Micro-Engine) et inversement, les informations venant de MULTICS sont renvoyées à l'écran de la console opérateur (figure 46) : ceci permet par exemple de lancer des travaux.

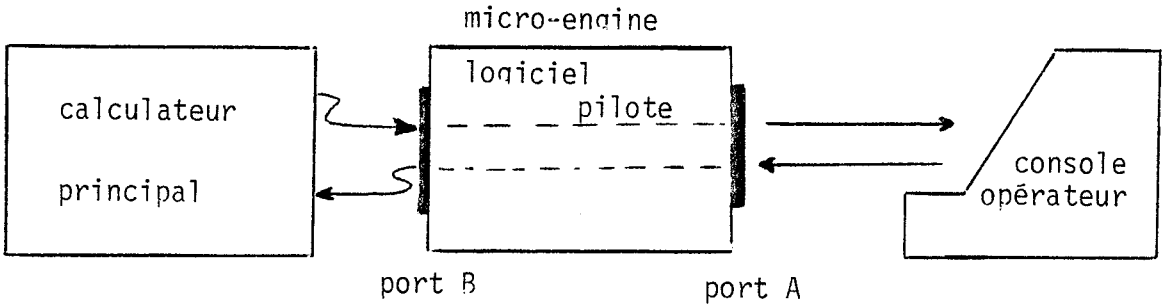


Figure 46 - Mode de fonctionnement "connexion"

. *Le mode "opération"* :
est le mode normal de fonctionnement (figure 50). Le terminal est commandé par le logiciel du calculateur principal. Tous les dispositifs (console, terminal HELIOS, tablette), sont sous la responsabilité du logiciel pilote. Quand le calculateur principal émet une commande (sous forme codée), le logiciel pilote émet, après exécution de la commande, un acquitement indiquant la fin de l'opération et éventuellement les résultats demandés. l'acquittement est soit "FIN CORRECTE", soit "message ERREUR".

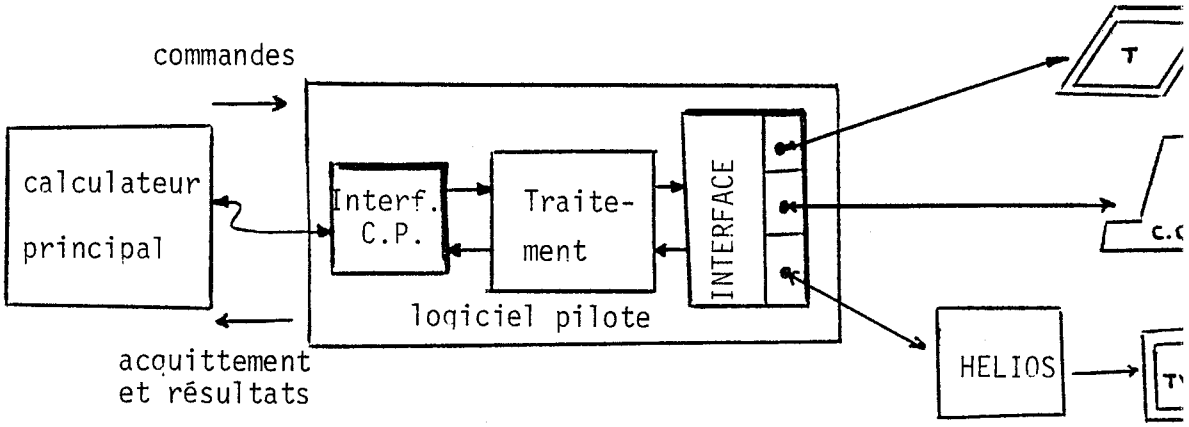


Figure 47 - Mode de fonctionnement "opération"

. *Le mode de fonctionnement "local" :*
les commandes sont émises à partir de la console opérateur. Tout se passe comme si la ligne calculateur principal - Micro Engine était coupée. En fin de traitement d'une commande, l'acquittement va à la console opérateur (figure 48).

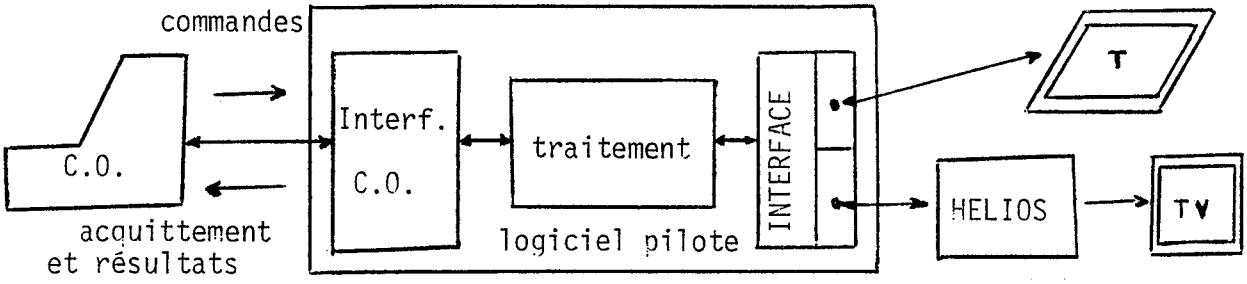


Figure 48 - Mode de fonctionnement "local"

Pour exécuter les divers processus dans les différents modes de fonctionnement proposés, on a recours à *une unité de contrôle* (figure 49) qui :

. analyse les commandes émises,

. met en fonction les divers processus,

. appelle les fonctions d'interface quand cela est nécessaire.

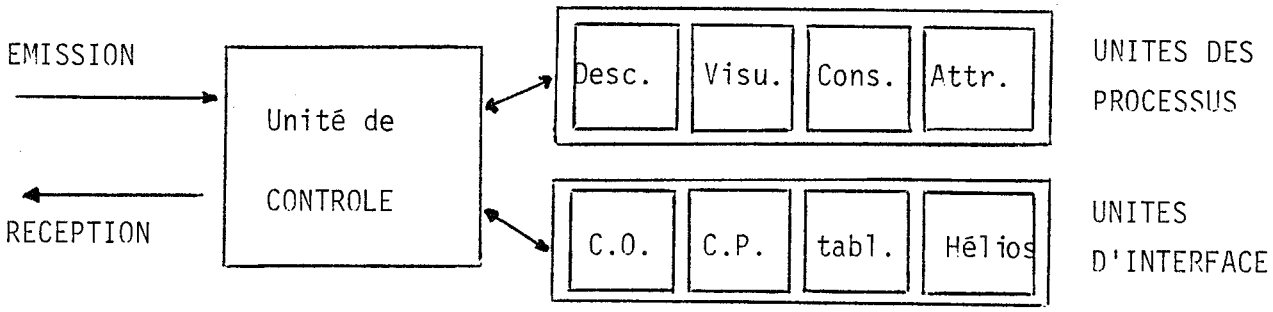


Figure 49 - Unité de contrôle

4.2. Définition des types d'informations manipulés

Les éléments de base auxquels l'utilisateur peut affecter des attributs sont les suivants :

- face plane 3D,
- éléments "fil de fer" 2D composés de :
  - suite de segments ou ligne brisée,
  - messages (suite de caractères).

4.2.1. Les informations d'identification (I)

Chaque élément est défini par un entier. Cet entier est communiqué par le logiciel principal, en fonction de la structure arborescente. Lors d'opérations de consultation, c'est cet entier qui est renvoyé au logiciel principal. Le logiciel pilote n'accepte que 1024 éléments différents ; en effet, à chaque élément, il fait correspondre une face plane d'HELIOS (dont les numéros varient de 0 à 1023). Cette attribution des faces planes d'HELIOS se fait de façon séquentielle (figure 50) dans l'ordre de la communication des éléments grâce au module de gestion de la table des éléments.

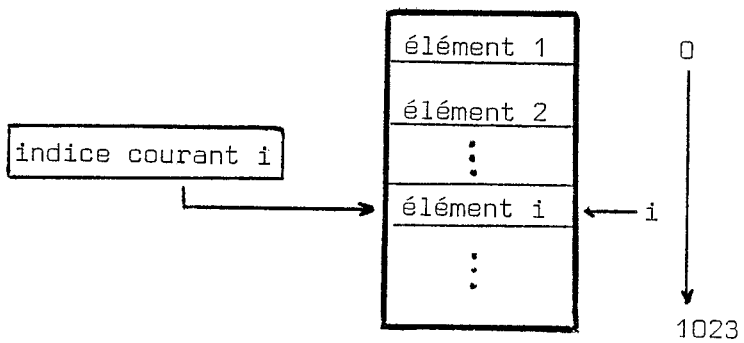


Figure 50 - Table logicielle des éléments

4.2.2. Les informations morphologiques (M)

A chaque élément est associé le type morphologique auquel il fait référence :

F : face

S : suite de segments disjoints

L : ligne brisée

M : message.

La forme des objets est décrite par une suite de coordonnées (x, y) représentant l'objet dans la vue. Les messages sont définis par :

. les coordonnées représentant le coin gauche de départ du message,

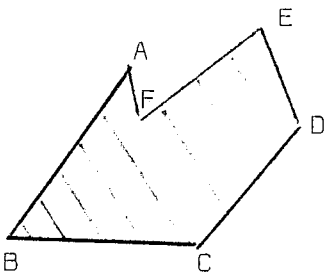
. les caractères ASCII qui composent ce message.

Ces informations sont conservées sur disquette de façon à :

. minimiser les transferts d'information (calculateur principal - Micro-Engine),

. pouvoir afficher (ou ré-afficher après modification) des éléments à la demande.

Les figures 51, 52 et 53 illustrent quelques exemples de la souplesse du système.

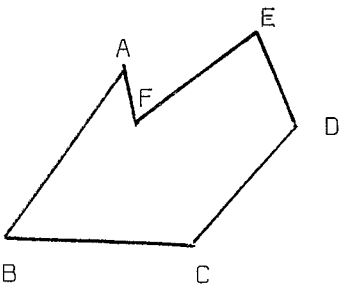


type : F nb de sommets : 6

coordonnées :

xa	ya
xb	yb
xc	yc
xd	yd
xe	ye
xf	yf

} de tous les points dans l'ordre du contour



← le changement du type de F en L (seul le contour apparaît)

Figure 51

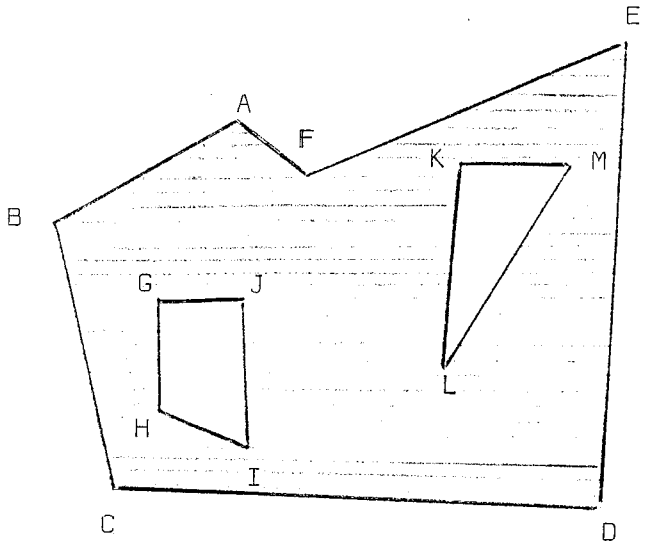


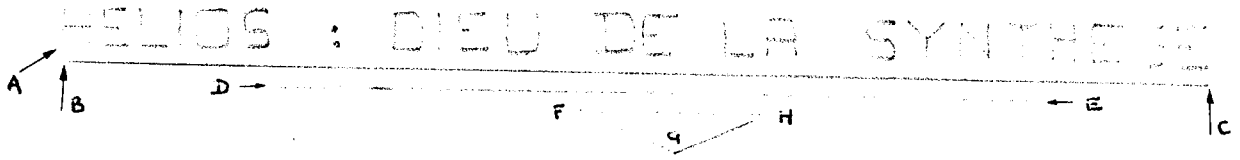
Figure 52

type : F nb de sommets : 13+2

coordonnées :

xa	ya
xb	yb
xc	yc
xd	yd
xe	ye
xf	yf
xa	ya
xg	yg
xh	yh
xi	yi
xj	yj
xg	yg
xl	yl
xk	yk
xm	ym

on répète le sommet de départ pour tous les polygones non terminaux



3 éléments :

* type : M coordonnées xa, ya + suite de caractères
* type : S nb de sommets : 4 coordonnées xb yb xc yc xd yd xe ye
* type : F nb de sommets : 3 coordonnées xf yf xg yg xh yh

Figure 53

Pour décrire la forme de la pièce (figure 54), on pourrait diminuer le nombre d'éléments, en considérant comme un seul élément les polygones A, B et C qui ont les mêmes informations d'aspect et de définition dans le repère. Cette notion a toutefois été abandonnée. En effet, dans le programme principal, l'élimination des parties cachées engendre des faces différentes. Il faudrait donc les recomposer, ce qui demanderait de longs calculs.

Pour minimiser la place occupée par ces informations, celles-ci seront compactées.

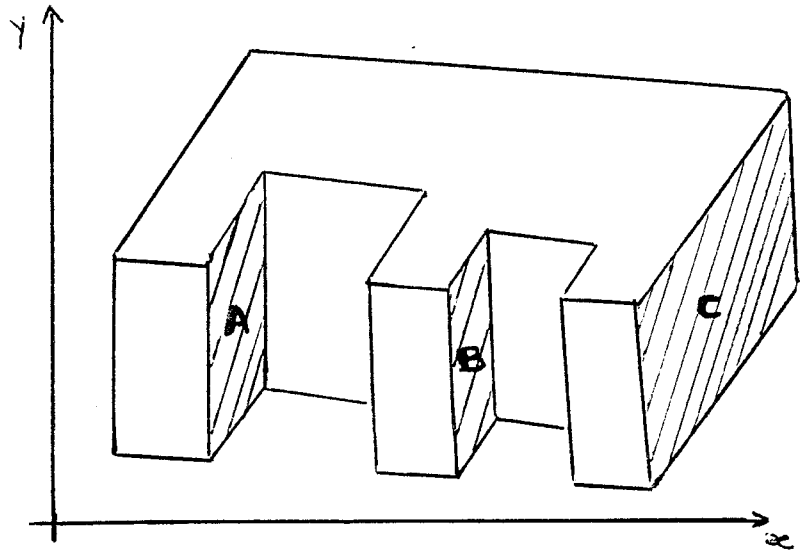


Figure 54 - Forme d'une pièce

4.2.3. Les informations d'aspect (A)

. Le type texture (T) :

est composé d'un nom défini par un identificateur ayant au maximum 8 caractères. Ces textures sont choisies parmi celles enregistrées à l'initialisation du matériel, ou celles disponibles dans la banque de données associée sur disquette. L'utilisation de textures différentes est limitée par la taille de la mémoire qui les contient.

Dès qu'une nouvelle texture est utilisée par le synthétiseur, il y a :
- recherche dans la banque de données (sur disquette) des points la décrivant,
- affectation de ces points à la mémoire de texture et du pointeur associé à la table des faces (HELIOS),
- enregistrement dans une "table des textures" (figure 55) de son nom et du pointeur de la mémoire du matériel

nom texture	ciel	grès	car.vert	....
pointeur	i	j	k	

Figure 55 - La table des textures

Quand une texture existe dans la table des textures, il suffit de communiquer son pointeur à la ou les faces concernées dans la table des faces d'HELIOS.

- Le type brillance (B) :

  est composé d'un nom de 8 caractères maximum ; il permet d'indiquer la réflexion du matériau. Il est choisi parmi les 4 types enregistrés lors de l'initialisation du matériel (exemple : mat).

- Le type visibilité (V) :

  permet d'attribuer V à une face visible, I à une face invisible (après extension du matériel T à une face transparente). Ce type d'information peut être communiqué pour les différents plans d'identification (exemple : face 8 invisible sur plan 0, visible sur plan 1).

Les informations de brillance et de visibilité sont conservées par HELIOS.

4.2.4. Les informations géométriques (G)

Certaines informations affectent les conditions d'affichage de l'image finale, d'autres ne concernent que les éléments.

1) Informations concernant l'affichage de la vue (V)

Une matrice 3x3 (en coordonnées homogènes) concernant la vue peut être appliquée à toutes les coordonnées des points de la vue. Elle est communiquée par le logiciel principal, ou obtenue lors de transformations géométriques de la vue. Toute transformation géométrique (rotation, translation, zoom) peut être exprimée par une matrice en coordonnées homogènes :

rotation de θ : $\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$

translation de m en x et de n en y : $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & n & 1 \end{pmatrix}$

homothétie de facteur k : $\begin{pmatrix} k & 0 & 0 \\ 0 & k & 0 \\ 0 & 0 & 1 \end{pmatrix}$

(La matrice est 3x3 car nos coordonnées sont en 2D)

L'utilisation de coordonnées homogènes est intéressante. En effet, toute combinaison de transformations peut être exprimée par un simple produit matriciel. Nous ne conservons donc, lors de chaque transformation, que *la matrice 3x3 résultante*.

Les informations suivantes peuvent donc être communiquées :

- la matrice de vue est exprimée par 6 réels (la dernière colonne est toujours (0,0,1)) ;

- une rotation (R) est représentée par un entier exprimant en degrés l'angle de rotation. Cet angle peut varier entre 0 et 360 ° et un signe positif indique le sens de rotation des aiguilles d'une montre ↻ , un signe négatif, le sens inverse ↺ . Cette rotation s'effectue dans le plan de l'écran et ne sera effective que lors du prochain ordre d'affichage "AFFICHER (VUE)", le matériel ne sachant par réaliser de rotation ;

- une translation (T) est représentée par deux entiers exprimant le déplacement en x et en y. Ce déplacement s'exprime par rapport au coin supérieur gauche de l'écran (figure 56).

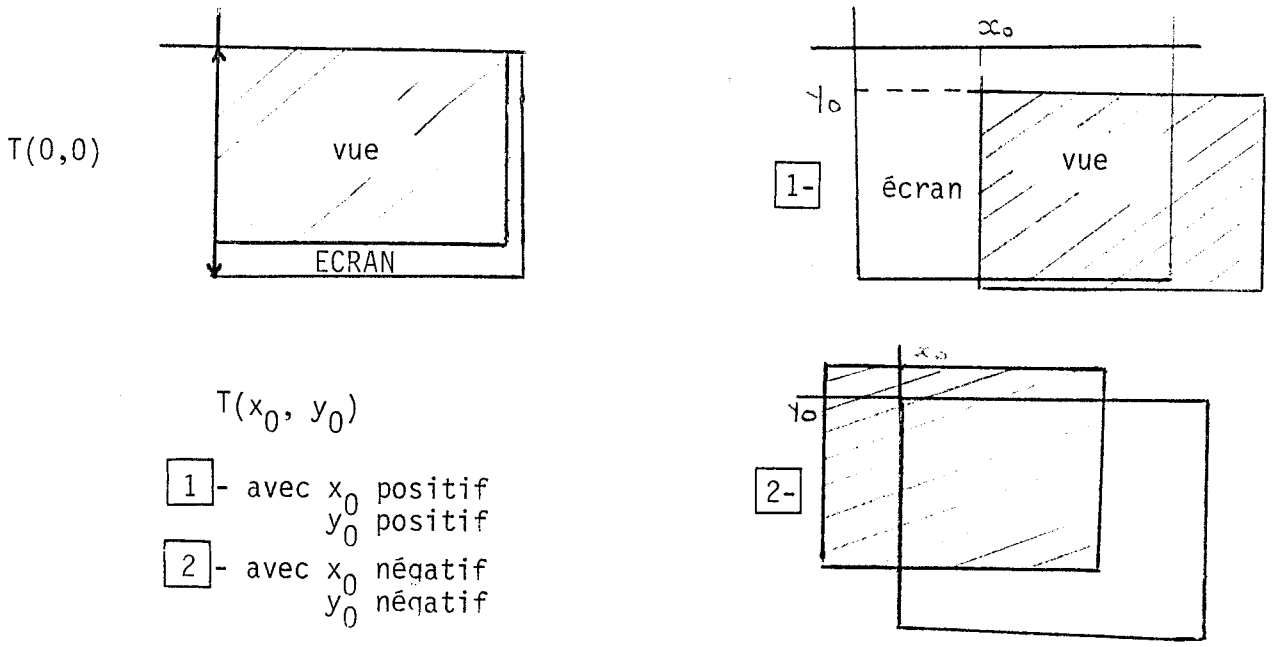


Figure 56 - Translation

Si l'utilisateur souhaite effectuer une translation en temps réel (R), il communique le ou les plans d'identification concernés (caractères 0, 1, 2, 3, 4, 5, 6, 7 ou T(ous)). Le logiciel transmet alors ces informations au matériel.

Si l'utilisateur préfère une translation différée (D), le logiciel effectue le produit matriciel et l'image ne sera affectée que lors du prochain "AFFICHER vue".

. Une homothétie ou zoom (Z) est exprimée par deux réels donnant le facteur de grossissement en x et en y (un réel compris entre 0 et 1 diminue la taille de la vue). De même que pour la translation, le zoom se réalise en temps réel grâce au matériel, ou en différé à la demande de l'utilisateur.

S'il se réalise en temps réel, le facteur de grossissement peut alors être représenté plus simplement par deux caractères Zx et Zy. Le codage sera le suivant :

0 → *1

1 → *2

2 → *4

3 → *8

4 → *16

Ce sont les possibilités admises par le matériel. En revanche, il affecte un ou plusieurs plans d'identification.

S'il se réalise en différé, le zoom peut être exprimé par des réels. Le logiciel de base effectue alors le produit matriciel et l'image ne sera affectée que lors du prochain affichage.

. Le mode de coupage de l'image permet de dire par plans d'identification ce qu'il y a sur l'image en dehors de la vue définie (photo 57). Le codage du coupage se fait à l'aide d'un caractère :

0 : permet de faire apparaître le plan de fond,

1 : permet de dupliquer la vue.

Aucune information de prise de vue n'est conservée puisque nous n'avons que des informations morphologiques 2D et que nous n'intégrons pas d'algorithmes d'élimination de parties cachées.

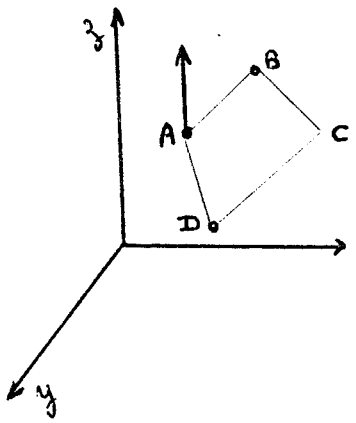
Aucune transformation géométrique attachée aux éléments eux-mêmes n'est prise en compte pour les mêmes raisons que ci-dessus.

2) Les informations concernant le repère des éléments

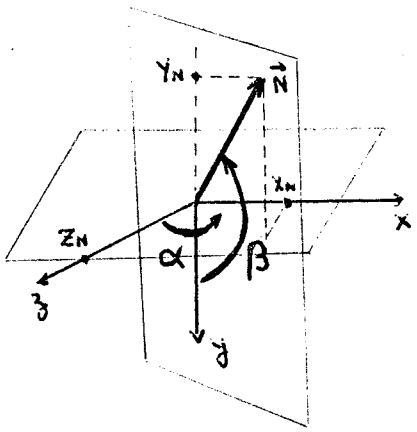
Ces informations ne concernent que les éléments "faces 3D". Les types d'attributs concernent la normale à la face et la matrice de projection.

- La normale (N) à la face est définie (figure 58) :
  - soit par les coordonnées cartésiennes du vecteur normal (3 réels X_N, Y_N, Z_N),
  - soit par les coordonnées sphériques de ce vecteur (2 réels : alpha, béta),
  - soit par les coordonnées cartésiennes de trois points de la face (9 réels : (x, y, z) pour chacun des trois points).

Un module de calcul permet à partir des trois points d'une face, d'obtenir le vecteur normal en coordonnées cartésiennes, puis en coordonnées sphériques, seules admises par le matériel (α, β).



Exemple 1 : normale définie par 3 points (A, B, D)

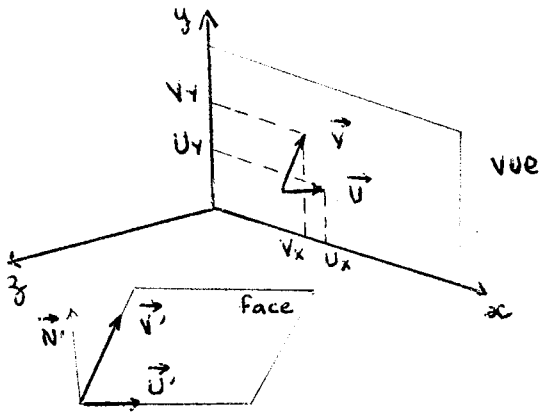


Exemple 2 : normale définie par le vecteur normal (X_N, Y_N, Z_N) ou (α,β)

Figure 58 - Description de normale

. La matrice de projection (M) est obtenue à partir du repère 3D associé à la face ($\vec{U}'$, $\vec{V}'$, $\vec{N}'$) grâce à la projection des vecteurs de base $\vec{U}$, $\vec{V}$ du plan de la face, dans le plan de la vue (figure 59).

La matrice de projection est exprimée par *les quatre réels* Ux, Uy, Vx, Vy. Celle-ci est inversée, puis transmise au matériel.



$$M = \begin{vmatrix} Ux & Vx \\ Uy & Vy \end{vmatrix}$$

Figure 59 - Matrice de projection

3) Les informations concernant l'arrangement des éléments dans la vue (P)

Il existe plusieurs plans d'identification et les éléments peuvent appartenir à l'un quelconque des plans. Cet attribut est représenté par un caractère : le numéro du plan (0, 1, 2, 3, 4, 5, 6, 7 ou F). F représente le plan de fond. Cette information est conservée sur disquette avec les informations morphologiques de l'élément.

4.2.5. Les informations d'éclairage (E)

Elles affectent toute la vue et on distingue trois types :

- la couleur de la source (C) :

  elle est définie par le nom (8 caractères maximum) d'une couleur unie appartenant à la banque de données des textures. Un module du logiciel transforme ce nom en couleur (R, V, B) transmissible au matériel.

- La position de la source (P) :

  elle est définie par les composantes (3 réels x, y, z) du vecteur directionnel unitaire de la source lumineuse, exprimées dans le repère absolu de l'écran.

- L'intensité de la lumière ambiante (L)

  est représentée par un entier variant entre 0 et 15.

Toutes ces informations sont conservées par le matériel.

4.3. Définition des opérateurs

Compte tenu des informations que nous avons retenues, deux opérateurs de synthèse sont à l'heure actuelle implémentés dans le logiciel pilote :
- la composition des informations morphologiques et des informations d'affichage non prises en compte par le matériel,
- le remplissage des plans d'identification.

Les autres opérateurs réalisés sont des opérateurs de calcul (produit de matrices, calcul de normale, ...), ou d'habillage (gestion de table de correspondances, transmission de données, ...) permettant à l'utilisateur de faire abstraction des composants de base des matériels.

4.3.1. Organisation générale de la synthèse

Les deux logiciels, le logiciel pilote et le logiciel principal, se partagent actuellement le début de la synthèse (figure 60). Nous aurions aimé que le logiciel pilote intègre toutes ces fonctions, mais la taille de la mémoire centrale du micro-ordinateur et des périphériques associés (disquettes souples) ne l'ont pas permis.

La forte pénétration des attributs dans le matériel demande que le logiciel pilote prenne en compte pour la visualisation :

. la synthèse des attributs non synthétisés (OPERATEUR DE SYNTHESE)

. le codage des attributs
. la transmission au matériel

} INTERFACE HELIOS

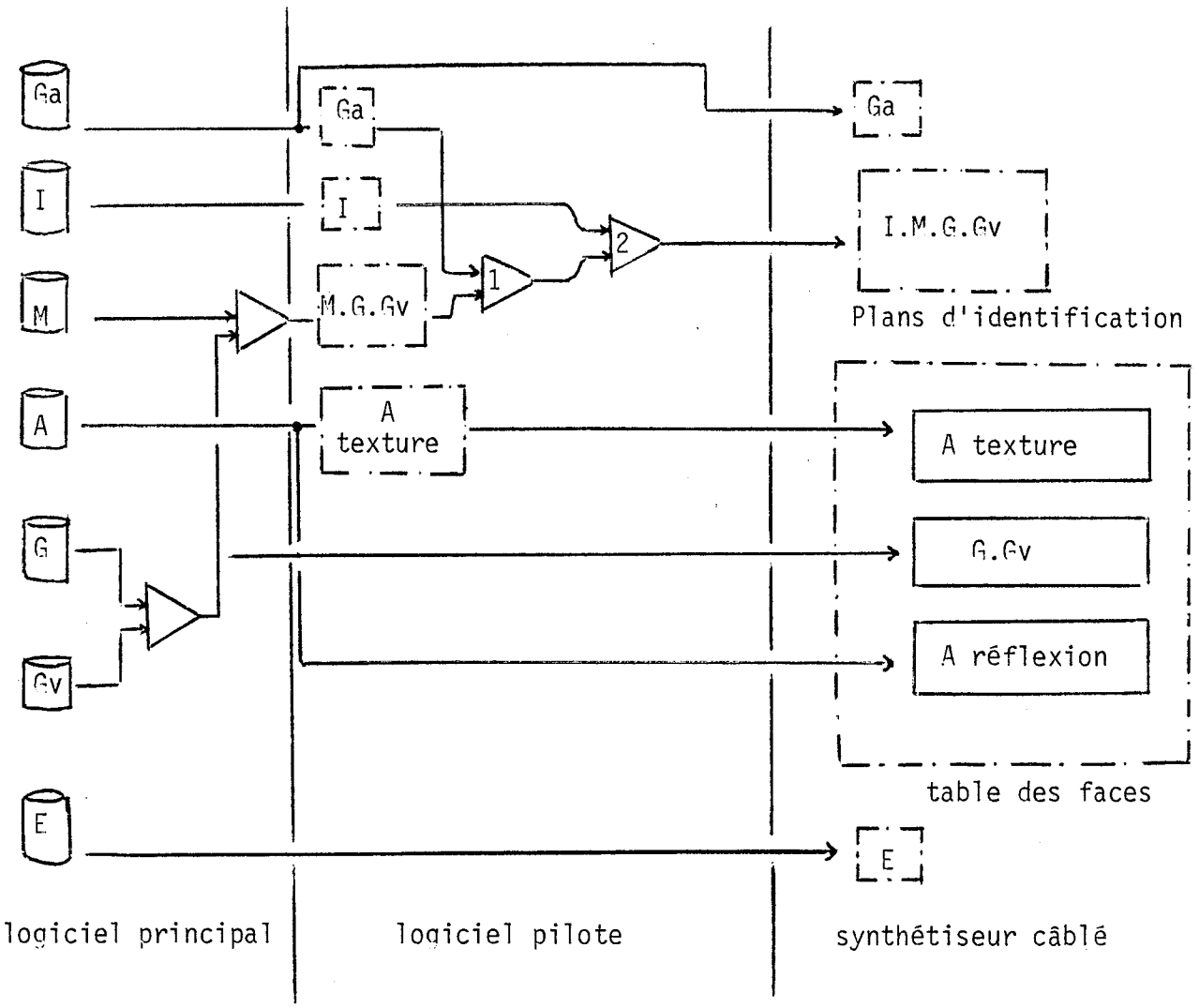
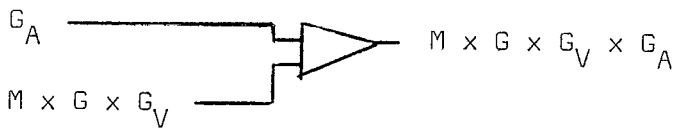


Figure 60 - Début de la synthèse effectué par le logiciel

4.3.2. Les opérateurs de synthèse

1) L'opérateur 1 :



Il permet de transformer les coordonnées exprimées dans le repère de la vue en coordonnées exprimées dans le repère de l'écran :

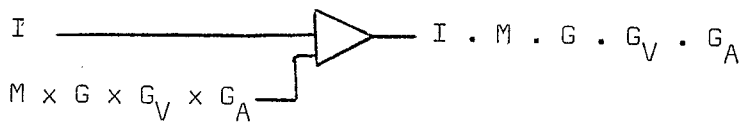
$$(x_e, y_e, z_e) = \begin{pmatrix} a_0 & a_1 & 0 \\ b_0 & b_1 & 0 \\ c_0 & c_1 & 1 \end{pmatrix} \times (x, y, 1)$$

matrice exprimant les transformations 2D subies par la vue

coordonnées des points

seuls x_e et y_e sont réellement calculés.

2) L'opérateur 2 :



Il permet de communiquer à chaque point de l'écran, le numéro de face associé à chaque élément. Cet opérateur est loin d'être trivial. En effet, la définition des informations morphologiques est faite sous forme de description des points composant le contour des éléments. Trois modules sont nécessaires :

- l'un effectuant le remplissage des caractères,
- le deuxième effectuant la numérisation des segments,
- le troisième permettant le remplissage des tâches polygonales éventuellement trouées.

Nous développerons au chapitre suivant les principes de calcul et les problèmes posés par la réalisation logicielle effective de ces modules.

4.3.3. Les opérateurs de mémorisation interne

Le logiciel pilote prend en charge la gestion des différents fichiers et tables de correspondance internes. Les primitives décrites pourront être utilisées par les différents modules d'interface. En aucun cas ces opérateurs n'ont un effet sur l'image.

1) Gestion de la "table des éléments"

Elle permet :

. la mise à jour de la table des éléments (correspondance numéro officiel des éléments ↔ numéro de face HELIOS) ;

. *la création d'une liste des faces valides.* De nombreuses faces ont souvent les mêmes attributs et le prototype HELIOS permet la compression d'informations pour des faces consécutives. Pour utiliser ces possibilités du matériel, nous allons tenir à jour une liste des faces valides sur laquelle s'effectuent les communications d'information.
La primitive AFFECFAC lit les identifieurs (entiers) communiqués sous la forme NB1 A NB2 ␣ NB3 ␣ NB5 A NB6 etc ..., recherche si ces entiers existent dans TFLIDENT (table des éléments), les crée éventuellement et affect le numéro de faces correspondant à FACVALID (table des faces valides).
La primitive SUP supprime les identifieurs communiqués en les remplaçant par NIL. Ils ne seront réutilisés que lorsque la table des éléments sera pleine.

2) Gestion de la table des textures

La communication du type d'aspect texture fait appel à la primitive :
. AJOUTEXT (nom texture) qui recherche le pointeur de texture à associer à nom texture (éventuellement en créant la texture dans la mémoire de texture),
. FACETEXT qui affecte ce pointeur de texture à toutes les faces valides (FACVALID).

3) Gestion de la structure de données des informations morphologiques et géométriques

Compte tenu de la place qu'elles occupent, les informations morphologiques sont stockées sur disquette. Elles se trouvent sur la même disquette que les programmes, l'autre unité de disquette étant réservée à la banque de données des textures. Le fichier a besoin, si toutes les faces sont utilisées, d'environ 128 Ko. Cette place consécutive est réservée sur la disquette, car l'attribution des fichiers est séquentielle.

L'accès à la structure de données se fait par le numéro de face (ACCES DIRECT), et les mouvements de la tête de lecture sont réduits par l'attribution séquentielle des numéros de face.

Les primitives sont les suivantes :

- ENTREEPV (TYPMORPH, NOFACE, NBSOM, COORD-SOMMET), permet d'entrer les informations morphologiques sur cette base de données pour les faces 3D, les segments disjoints et les lignes brisées ;
- ENTREMES (NOFACE, NBCAR, PHRASE, COIN-INF-DROIT (X, Y), permet cette entrée d'informations pour les messages ;
- ECRFP (NOFACE-DEB, NOFACE-FIN, NOPLAN), permet l'attribution du type d'information géométrique : numéro de plan associé ;
- REMPI permet la lecture séquentielle des éléments et appelle l'opérateur de remplissage des plans d'identification ;
- ECRFACE (NOFACE), permet la lecture directe d'une face (+ remplissage).

4.3.4. Les opérateurs de communication

C'est à eux que fait appel l'unité de contrôle pour régir les échanges calculateur pilote ↔ dispositifs extérieurs.

Seuls sont considérés comme extérieurs, le calculateur principal, le prototype HELIOS, la tablette à numériser et la console opérateur. L'ajout d'autres dispositifs demanderait l'écriture d'un module d'interface spécifique.

Nous allons dans les paragraphes suivants, détailler ces interfaces. Elles pourront éventuellement être utilisées par d'autres programmes locaux (description de textures, description d'objets, ...).

Quatre interfaces sont implémentées :

. *l'interface avec le calculateur CII HB 68 :*
  les échanges se font grâce à une ligne téléphonique série V24 (à 1200 ou 4800 bauds), reliée au port B du micro-calculateur ;

. *l'interface avec le prototype HELIOS :*
  un bus relie directement le module d'interface d'HELIOS et le port parallèle du micro-calculateur ;

. *l'interface avec le console opérateur*
  (reliée au port série A) ;

. *l'interface avec la tablette à numériser*
  (reliée au port série B).

Elles permettent la visualisation sur HELIOS, la description (de coordonnées) la consultation d'éléments du système (l'identification de faces par exemple)

4.4. Interface avec le calculateur principal

Elle permet la communication entre le calculateur principal et le micro-calculateur. La ligne de transmission n'accepte que des caractères à une vitesse de 1200 ou 4800 bauds. Pour minimiser le temps de transmission, les échanges sont codés.

- Chaque commande émise par le calculateur principal en direction d'un des processus du logiciel est encadrée par deux caractères :

  début de commande → [ESC] commande [LF] ← fin de commande

  Tout ce qui n'est pas entre ces deux caractères est envoyé directement à la console opérateur.

- De même, l'acquittement émis par le calculateur pilote, en fin de processus, se fait comme suit :

  [ESC] messages sur 2 caractères (réponses) [LF]

  Le message sera soit OO : commande bien exécutée
  XX : un message d'erreur.

- Une commande comprend une primitive suivie d'un certain nombre de paramètres.

- Une réponse comprend le nombre de paramètres (entier), le type de codage (E(ntier, R(éel ...) et les paramètres effectifs.

4.4.1. Codage des primitives et des types d'informations

- *Les primitives* seront codées par *un caractère* :
  A : attribuer (type information) : communiquer ou supprimer
  V : visualiser
  D : décrire coordonnées
  C : collecter type information
  I : initialiser

La primitive I comprend l'initialisation d'HELIOS (modèle d'éclairage, mémoire de texture, table des faces ..) et l'initialisation des tables du logiciel.

. *Le type d'information* est défini par *deux caractères :*

- le premier indique *la classe d'information :*
  - I : identité des éléments
  - M : morphologie
  - A : aspect
  - G : géométrie
  - E : éclairage
  - V : affichage de la vue
- le second (quand il est nécessaire) indique *le type :*

pour les informations morphologiques :	F : faces 3D
	S : segments disjoints
	L : ligne brisée
	M : message
pour les informations d'aspect :	T : texture
	B : brillance
	V : visibilité
pour les informations géométriques :	N : normale
	M : matrice de projection
	P : plan

4.4.2. Codage des dispositifs de dialogue appelés

Il ne correspond qu'aux actions DECRIRE ou IDENTIFIER et se fait à l'aide d'un caractère :

T : accès à la tablette à numériser,

R : accès au réticule intégré à HELIOS,

C : accès au clavier de la console opérateur

? : permet de différer le choix de l'un des dispositifs ci-dessus, au moment de l'exécution de la commande. Ce choix (T, R, C) est alors proposé à l'utilisateur au moyen d'un menu.

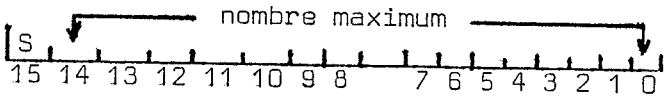
4.4.3. Codage des paramètres

Les paramètres transmis sont soit des caractères
soit des entiers
soit des réels

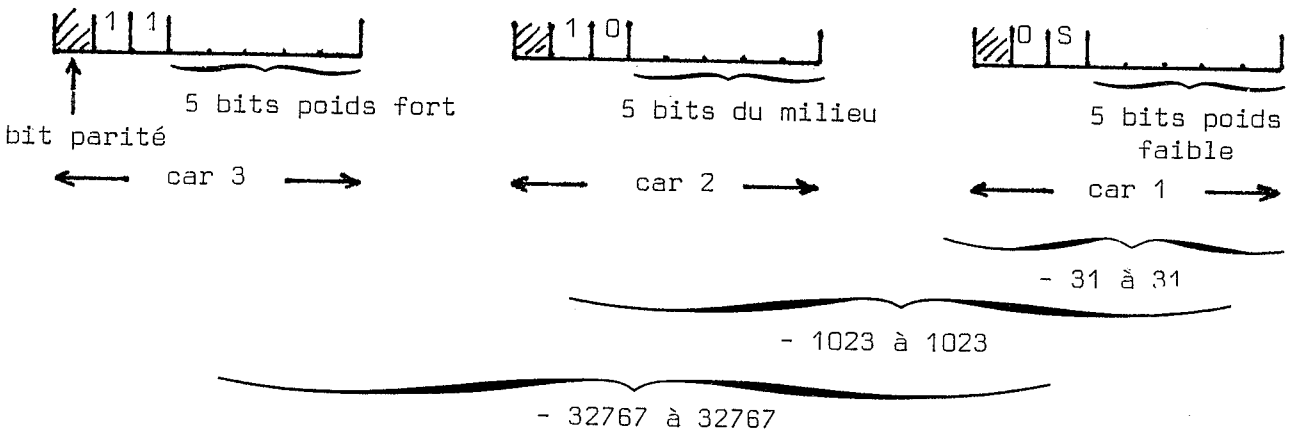
Le codage des réels et des entiers utilisé dans les machines, ne peut être transmis tel quel sur la ligne, car il utilise le bit de parité. Nous générons donc notre propre codage des entiers et des réels. Le codage par caractère (du style 1289 ou -1.567) demande la transmission de nombreux caractères pour les paramètres eux-mêmes et doit avoir en plus un caractère indiquant la fin du nombre. Il n'a donc pas été retenu.

4.4.3.1. Codage des entiers

Le micro-calculateur code ses entiers sur 16 bits acceptant des nombres compris entre -32767 et 32767.



Notre codage doit également permettre d'accepter des nombres compris entre -(2**15)-1 et (2**15)-1.
Il se fait sur 1, 2 ou 3 octets suivant la taille du nombre :



S = signe : 0 ← +
1 ← -

Une procédure de codage doit être implantée sur le calculateur principal qui engendre les caractères comme suit :

```
procédure COD-ENTIER
   SI |ENTIER| < (2 ** 5) alors génération car1
   SINON SI |ENTIER| <  (2 ** 10) alors génération car2 et car1
         SINON generation CAR3, CAR2 et CAR1
fin
```

L'interface permet *le décodage par analyse des bits 6* (et 5 éventuellement) de chaque caractère (figure 61)

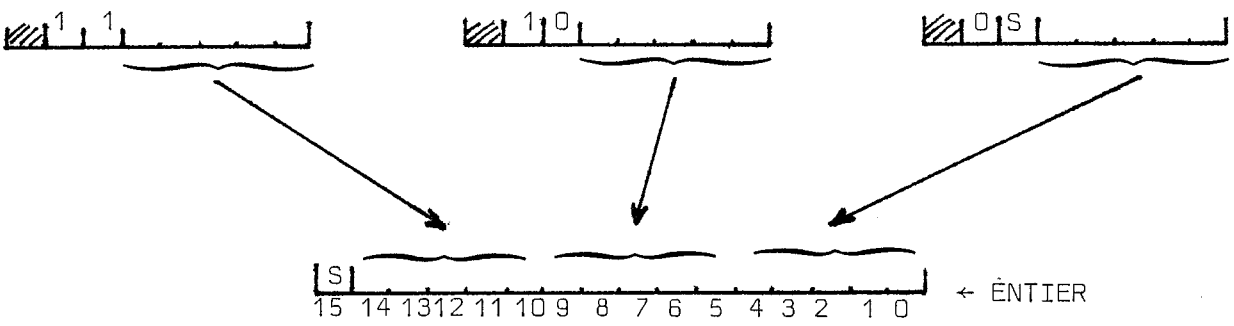
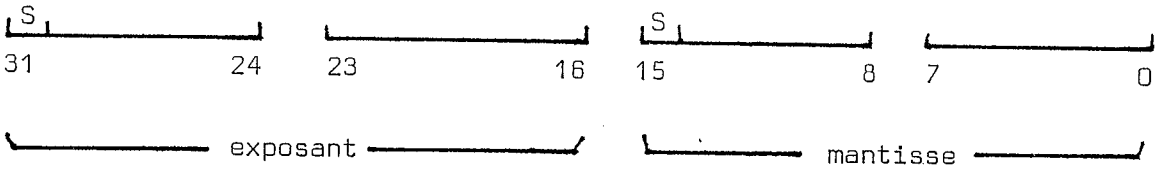


Figure 61 - Décodage d'un entier

```
FONCTION ENTIER
DEBUT
|  lecture (CAR)
|  ENTIER [0..15]←0
|  tant que CAR [6] = 1 faire
|  |  si CAR [5] = 1 alors ENTIER [11..14]←CAR [0..4]
|  |  SINON ENTIER [5..10]←CAR [0..4] finsi
|  |  lecture(car)
|  fin tant que
|  ENTIER [0..4]←CAR [0..4]
|  ENTIER [15]←CAR [5] (*signe*)
FIN
```

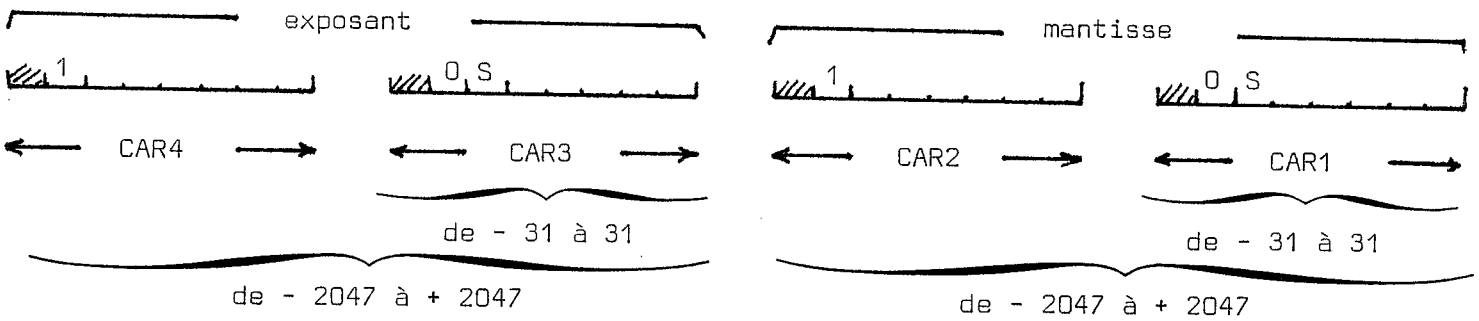
4.4.3.2. Codage des réels

Le codage des réels sur le micro-calculateur se fait sur 32 bits ; l'exposant et la mantisse du réel sont en fait codés chacun sur un entier :



Comme au paragraphe précédent, nous pourrions coder chaque entier sur 3 octets. Mais les réels que nous utilisons sont le plus souvent des coordonnées de faible valeur absolue.

Le codage adopté est donc le suivant : 1 ou 2 octets pour l'exposant suivi d'un ou 2 octets pour la mantisse :



La procédure implantée sur le calculateur principal engendre à partir des réels, les caractères comme suit :

```
PROCEDURE COD-REEL
|   si |mantisse| > 2047 alors |mantisse| = 2047
|   si |exposant| > 2047 alors |exposant| = 2047
|   si |exposant| > 31 alors génération CAR4 puis CAR3
|                      sinon génération CAR3
|   si |mantisse| > 31 alors génération CAR2 puis CAR1
|                      sinon génération CAR1
FIN
```

Le décodage de ces caractères, inclus dans le logiciel pilote, est effectué par analyse du bit 6 de chaque caractère (figure 62).

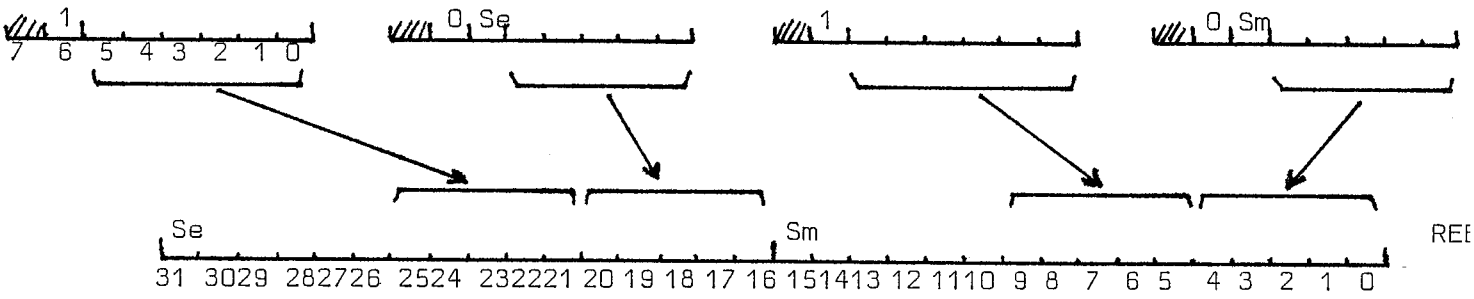


Figure 62 - Décodage d'un réel

```
FONCTION REEL
DEBUT
|  lecture (CAR)
|  REEL [0..31] ← 0
|  POUR J ← 0 à 1 faire
|  |  si CAR [6] = 1 alors  REEL [21 - (15*j) ... 26 - (15*j)] ← CAR [0..5]
|  |                        LECTURE (CAR)
|  |  finsi
|  |  REEL [31 - (15*j)] ← CAR [5]
|  |  REEL [16 - (15*j) .. 21 - (15*j)] ← CAR [0..4]
|  |  lecture (CAR)
|  FINPOUR
FIN
```

Nous ne détaillons pas ici la liste exacte des paramètres affectés à chaque type d'information. Le lecteur trouvera en annexe 4 un descriptif complet des commandes, types d'information et paramètres associés, de même que quelques exemples (avec photos du résultat).

4.5. Interface avec HELIOS

Cette unité prend en charge le codage particulier nécessité par la logique câblée d'HELIOS. Elle permet donc :
. l'attribution des informations aux registres ou mémoires,
. la consultation de celles-ci.

En utilisant cette unité, les programmes locaux peuvent accéder à HELIOS sans connaître son principe de fonctionnement interne. Il suffit pour cela d'utiliser les primitives décrites ci-dessous et qui font référence aux principaux processus offerts directement par le matériel.

4.5.1. Attribution des faces

1) Morphologie

|| SEGMENT (NO-PLAN, NOFACE, Y, X-GAUCHE, X-DROIT)

écrit un segment horizontal (ligne Y) dans le plan d'identification NO-PLAN.

2) Aspect

|| TEXTURE (nom-texture)

attribue la tecture de nom "nom-texture" aux faces appartenant à la table des faces valides (FACVALID). La correspondance entre le nom de la texture et son pointeur dans la mémoire de texture est effectuée par le module de gestion des textures.

‖ REFLEXION (nom-modèle)

attribue un modèle de réflexion, parmi ceux définis lors de l'initialisation du système, aux faces appartenant à la table des faces valides (FACVALID).

‖ VISIBILITE (ind-visibilité)

attribue la visibilité ou la transparence aux faces appartenant à la table des faces valides.

3) Géométrie

‖ NORMALE (alpha, beta)

attribue une direction perpendiculaire exprimée en coordonnées sphériques, aux faces valides (FACVALID).

‖ PROJECTION (U_x, U_y, V_x, V_y)

- calcule l'inverse de la matrice si le déterminant n'est pas nul et se charge de la symétrie du repère si les coefficients diagonaux sont négatifs. Les coefficients obtenus sont codés selon le format demandé par HELIOS (cf. 3.2.1.) :

```
si valeur > 7,5 alors valeur ← 7,5 finsi
si valeur > 1,875 alors
   e ← 1
   m ← arrondi (valeur *2)
sinon
   e ← 0
   m ← arrondi (valeur *8)
finsi
```

- attribue ces coefficients aux faces valides (FACVALID).

4) Eclairage

|| POSITSC (ALPHA, BETA)

. calcule la direction médiane entre la source et l'observateur,
. code la direction de la source et la direction médiane,
. affecte la position de la source ainsi codée au matériel.

|| COULEURSC (nom-couleur)

. transforme le nom de la couleur en code couleur R,V,B,
. le transmet au matériel.

|| AMBSC (lum-ambiante)

affecte l'intensité de la lumière ambiante.

5) Géométrie de l'affichage

|| FENETRE (NO-PLAN, X-SUP-GAUCHE, Y-SUP-GAUCHE, taille-x-taille-y, coupage)

positionne la fenêtre du plan d'identification NO-PLAN.

4.5.2. Consultation des faces

Ne pourront bien sûr être consultées que des informations déjà fournies à HELIOS.

1) Identification

|| IDENT (numéro-face)

permet de récupérer le numéro d'une face identifiée à l'aide du réticule.

2) Aspect

|| LEC-TEXTURE (numéro-face, nom-texture)

donne le nom de la texture, nom-texture, associée à la face numéro-face.

|| LEC-REFLEXION (numéro-face, nom-modèle)

donne le nom du modèle de réflexion utilisé pour la face numéro-face.

|| LEC-VISIBILITE (numéro-face, état)

communique l'état de la face (visible, invisible, transparente).

3) Géométrie

|| LEC-NORMALE (numéro-face, codage-alpha, codage-beta)

donne le codage des angles alpha et beta. Cette primitive peut permettre des comparaisons entre normales, mais le codage ne permet pas de retrouver les coordonnées sphériques exactes.

|| LEC-MATRICE (numéro-face, codage a, b, c, d)

donne le codage des coefficients de la matrice de projection.

Il ne semble pas utile de consulter les autres informations.

4.5.3. Visualisation et description des faces

Sur HELIOS, la visualisation des faces est implicite et aucun dispositif de description des attributs n'est disponible sur le synthétiseur câblé.

4.6. Interface avec les dispositifs de dialogue

Les seuls dispositifs de dialogue actuellement gérés par le logiciel pilote sont :

. le réticule intégré au synthétiseur câblé d'HELIOS,

. la tablette à numériser,

. la console opérateur.

4.6.1. L'attribution

Ce processus n'est défini que pour le réticule.

1) Morphologie

La logique câblée permet, de façon implicite, la définition d'un segment horizontal et d'un segment vertical.

2) Géométrie

POSITION-RETIC (nom-dispositif, X, Y)

permet de centrer le réticule sur un point dont les coordonnées sont définies dans le repère écran.

3) Aspect

ASPECT-RETIC (nom-dispositif, visibilité, nom-couleur)

permet de faire apparaître ou disparaître le réticule et de lui attribuer une couleur dans la gamme des unis de la banque de données des textures.

4.6.2. Consultation

1) Géométrie

‖ CONSULTER-COOR (nom-dispositif, x, y)

permet de récupérer les coordonnées (x, y) d'un point.

2) Aspect

‖ CONSULTER-ETAT (nom dispositif, état) ETAT = 0 : valide
1 : invalide

permet de savoir si le dispositif est valide, c'est-à-dire si la récupération de coordonnées continue ou pas. En effet, à l'aide de touches, on peut arrêter la consultation quand un nombre indéfini de coordonnées a été demandé.

4.6.3. Description et visualisation

La description des coordonnées d'un point se fait à l'aide :
. des touches de la console opérateur,
. des touches disponibles sur la console d'interaction d'HELIOS,
. du crayon associé à la tablette à numériser.

‖ DECRIRE-COORD (nom-dispositif, attente)

cette primitive enclenche le processus de description et le paramètre attente nous indique si les coordonnées sont à récupérer en continu ou après une attente de validation.

△ △ △

CHAPITRE 5

RÉALISATION

Nous évoquerons ici tout ce qui nous a paru intéressant et original dans la réalisation de ce logiciel pilote.

Presque toutes les primitives décrites dans le chapitre précédent ont pu être implémentées. Celles qui ne l'ont pas été, faute de temps, ne posent aucun problème algorithmique.

. *Le micro-calculateur* utilisé est le *"Micro Engine" de Western Digital,* relié au *lecteur de disquettes REMEX 40* (double face, double densité). C'est une machine à mots de 16 bits interprétant directement le P-code fourni par la compilation de programmes PASCAL ou ADA.

Le système fourni comporte :
- un éditeur de programmes d'utilisation aisée,
- un compilateur PASCAL UCSD,
- une gestion des fichiers sur disquettes permettant l'accès séquentiel ou direct,
- un éditeur de liens rassemblant des unités compilées séparément,
- divers utilitaires (formattage de disquettes, etc ...).

Quatre ports d'entrée-sortie sont également disponibles :
- un port série A affecté à la console opérateur (9600 bauds) ou à la connexion avec le calculateur principal (4800 bauds),
- un port série B affecté aux périphériques locaux (tablette, imprimante, etc ...),
- un port spécial pour l'unité de disques souples,
- un port parallèle que nous avons relié au synthétiseur câblé HELIOS.

. *Les périphériques locaux* actuellement disponibles sont :
- la console opérateur HAZELTINE 1520,
- la tablette à numériser SORED (50 cm x 50 cm), qui permet la récupération de points avec une résolution de 0,025 mm et qui comporte un clavier de 5 touches,
- une imprimante DIABLO 1641.

. Les programmes ont été réalisés en PASCAL UCSD. Ils ont tous été testés en local.

5.1. Gestion de la table des éléments

Cette table permet de faire la correspondance entre les identifieurs communiqués (et connus dans la base de données arborescentes) et les faces disponibles sur HELIOS (0 ... 1023)

Cette table TFLIDENT comporte donc 1024 éléments et est organisée séquentiellement.

La création et la suppression d'éléments se font par une banale affectation
CREATION (IDENTIFIEUR, indice) : TFLIDENT [indice] ← IDENTIFIEUR
SUPPRESSION (indice) : TFLIDENT [indice] ← NOM-NIL

La création d'éléments se fait séquentiellement. Un indice TFIND nous donne le nombre de faces utilisées. Les faces supprimées ne sont réutilisées que quand TFIND = 1024, de façon à ce que les identifieurs communiqués se retrouvent le plus souvent possible avec des numéros de faces consécutifs (ce qui facilite le compactage des faces, accepté par HELIOS) (figure 63).

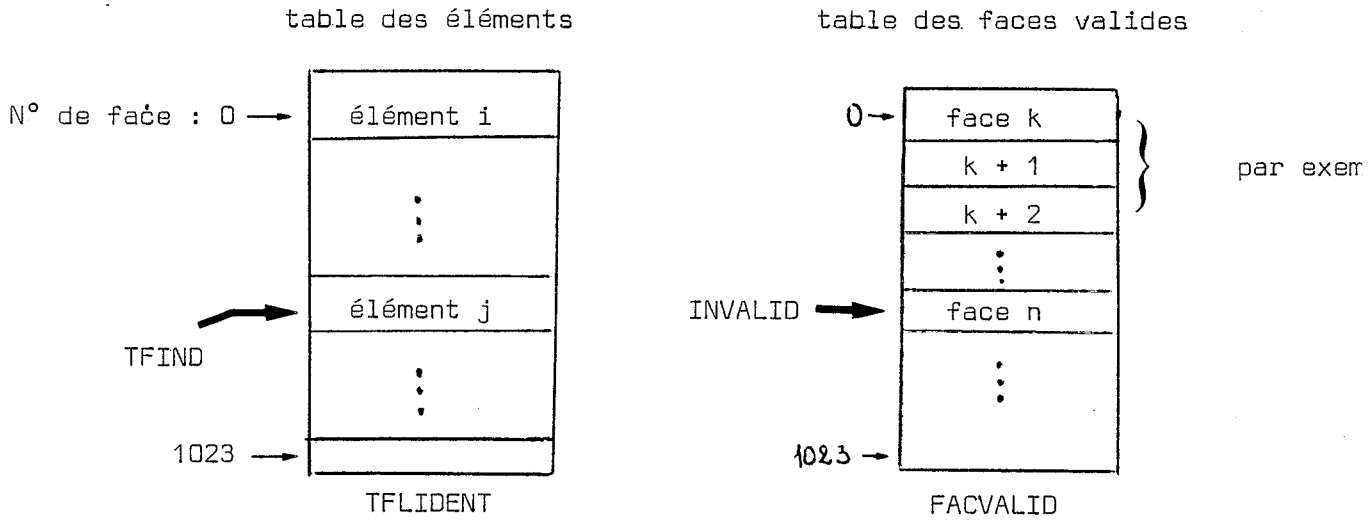


Figure 63 - Table des éléments et des faces valides

1) Communication d'éléments

La création des éléments, de même que la création des faces valides, va se faire lors de l'attribution du type identification (communiqué par exemple par l'ordre : A I NB1 A NB2 NB3 ...).

La procédure AFFECTER recherche dans TFLIDENT si l'identifieur communiqué existe (s'il n'est pas dans la table, elle le crée) et communique à FACVALID l'indice de TFLIDENT (figure 64).

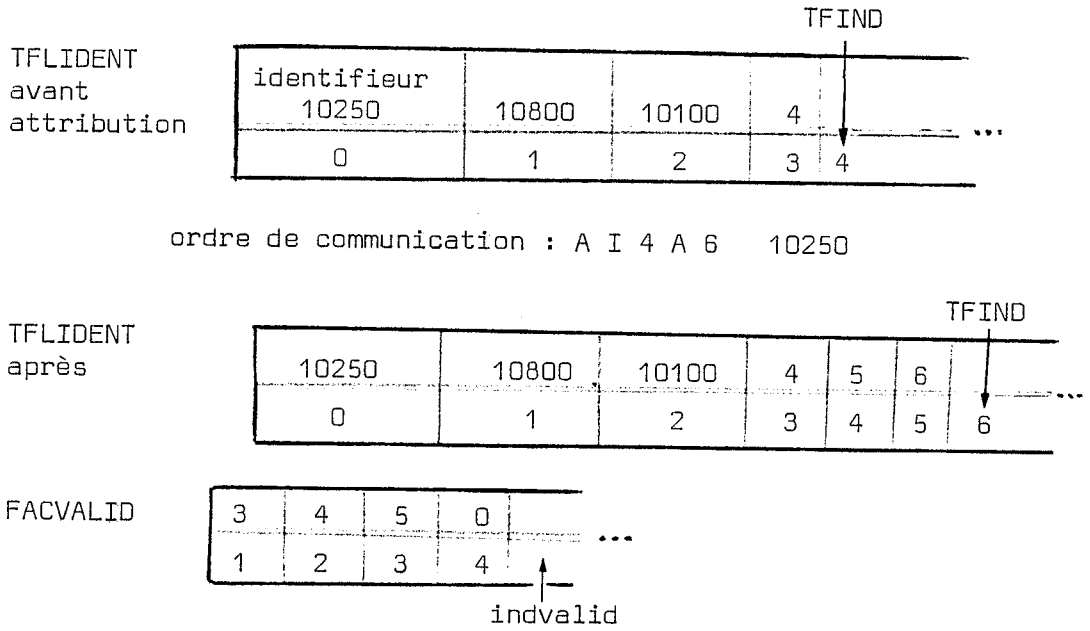


Figure 64 - Exemple d'affectation

Quand on arrive à la fin de la table des éléments (TFIND = 1024), on regarde si des éléments ont été supprimés et on utilise les indices de ces éléments rangés dans une table RESERVE (figure 65).

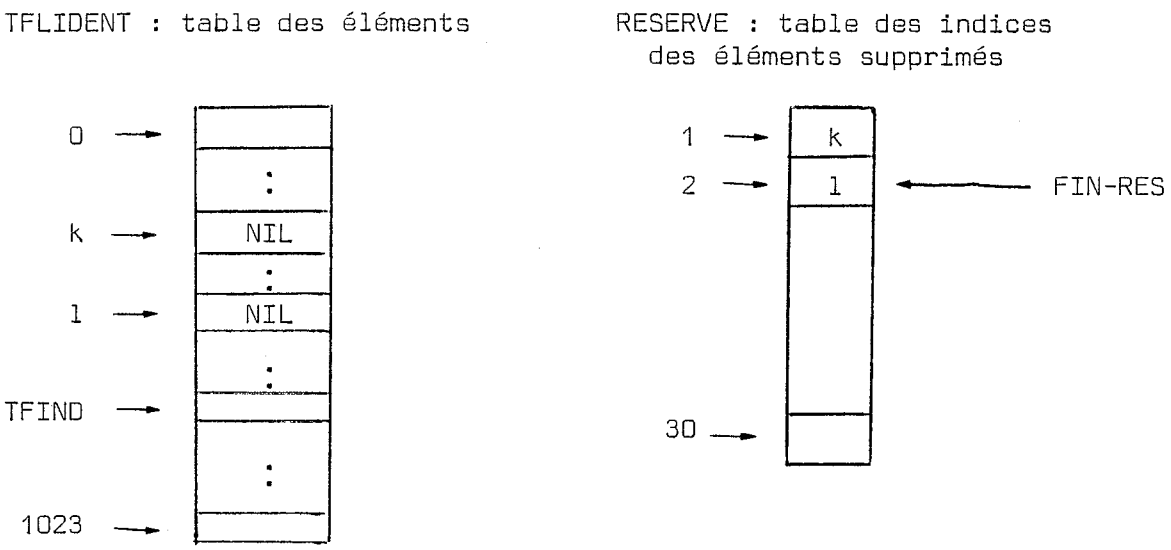


Figure 65 : Utilisation d'une table réserve pour éviter le parcours de TFLIDENT quand TFIND = 1024

2) Suppression d'éléments

Cette suppression va :

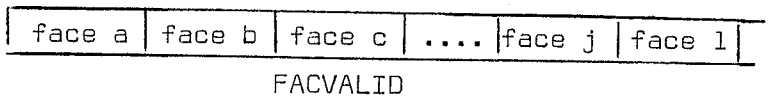
. supprimer l'élément dans la table des éléments,

. supprimer tous les attributs de cet élément, pour cela il suffit de rendre invisible la face correspondante et de mettre le pointeur de texture à 0.

```
SUPPRIMER ELEMENT (élément)
|   recherche TFLIDENT (élément, indice)
|   SUPPRESSION (indice)
|   dans table des faces d'HELIOS  face ← invisible, pointeur texture ← 0
|   communication à la table RESERVE de "INDICE"
fin
```

3) Communication d'attributs aux faces valides

La table des faces valides (FACVALID) comprend la liste de toutes les faces auxquelles doivent être communiqués les attributs. Comme HELIOS permet en une seule opération, d'affecter les attributs à plusieurs faces consécutives, on regroupe les faces consécutives :



FACVALID

si les faces c à j sont consécutives, il faut communiquer l'attribut en une seule opération :

▲ COMMUNIQUER ATTRIBUT (face c, face j, attribut)

Pour cela, une primitive BORNE (i, j) parcourt la table des faces valides et retourne les indices i et j de début et de fin :

```
COMMUNIQUER ATTRIBUT FACES VALIDES
|  i ← 1   j ← 1
|  tantque j < INVALID faire
|  |  BORNE (i, j)
|  |  COMMUNIQUER ATTRIBUT (TABVALID[i], TABVALID[j], attribut)
|  |  i ← j + 1
|  |  j ← i
|  fin tantque
fin
```

5.2. Gestion des tables de texture

Un nom de texture doit être traduit par un pointeur de texture pour être compréhensible par HELIOS. Deux tables existent donc :
TTLNOM qui contient les noms de texture sur 8 caractères maximum,
TTLADR qui contient les pointeurs de texture associés.

Quand on communique un nom de texture, il faut associer à toutes les faces valides, dans la table des faces intégrée à HELIOS, ce numéro de pointeur (figure 66).

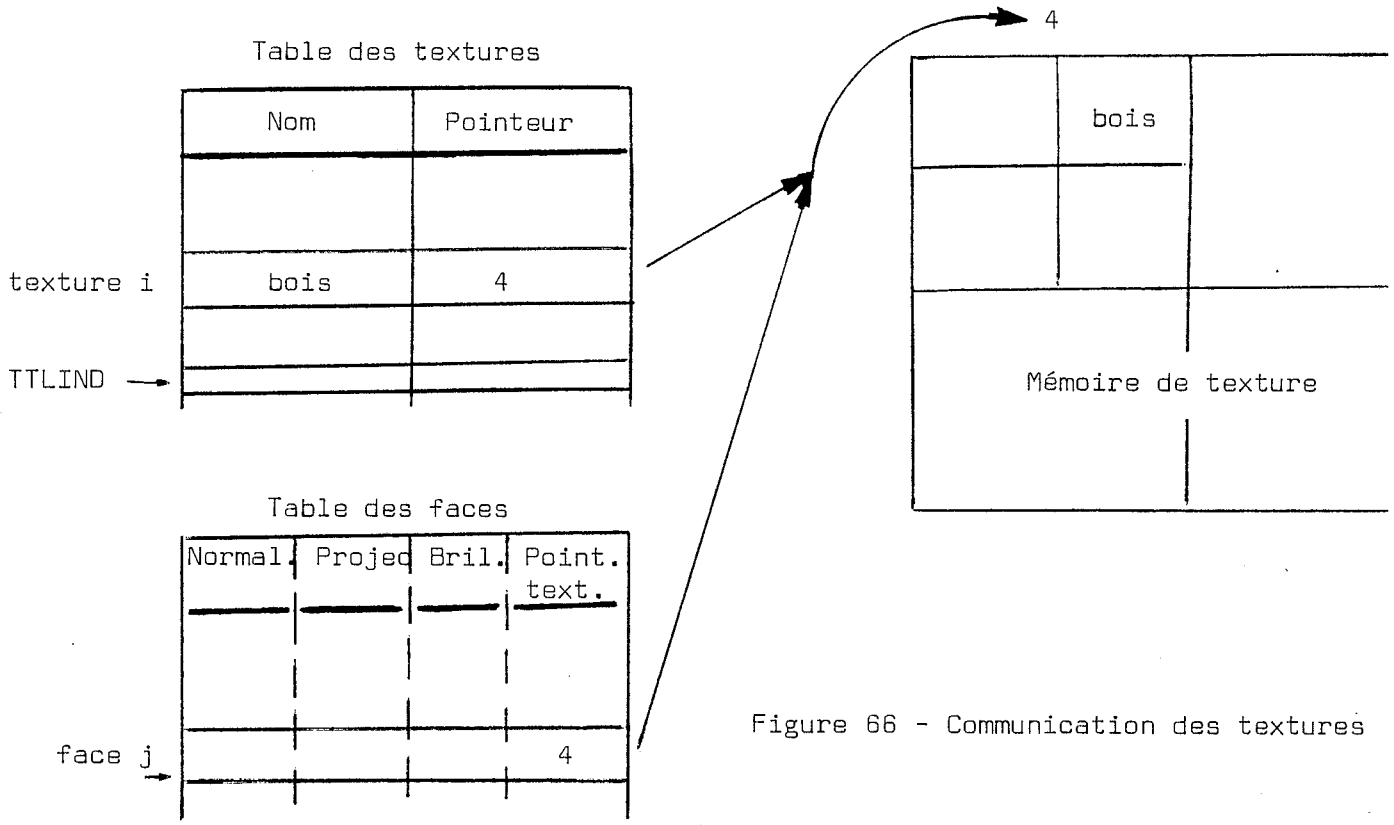


Figure 66 - Communication des textures

A l'initialisation du travail, la table des textures et la mémoire des textures sont remplies à l'aide d'un fichier indiquant les textures utilisées pendant ce type de travail.

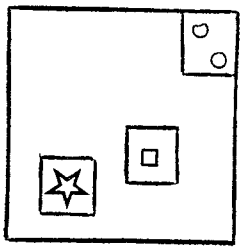
La table des textures est remplie de façon séquentielle ; TTLIND indique le nombre de textures utilisées.

1) Remplissage de la mémoire de texture

L'utilisation d'une nouvelle texture implique la recherche d'une place libre dans la mémoire de texture. Le nom de texture appartient ou non à la banque de données des textures, implantée sur disquette.

La primitive BDTIOUT (nom-texture, taille) recherche dans le catalogue de la banque de données la texture, nommée "nom-texture" ; si celle-ci existe, la primitive retourne la taille, sinon taille = 0.

Grâce à cette taille, une primitive doit rechercher dans la mémoire de texture un pointeur disponible correspondant à la taille. La mémoire de texture doit être remplie de façon méthodique pour qu'un nombre maximum de textures y prennent place (figure 67).



3 textures de taille 64

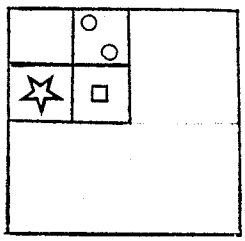


Figure a

Figure b

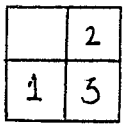
Nota : la figure b a un remplissage plus intéressant que la figure a. En effet, dans la figure a, on ne peut rajouter une texture de taille 128.

Figure 67 - Remplissage de la mémoire de texture

Pour le rangement des textures, deux ensembles sont utilisés :

- COMPLET comprend les pointeurs déjà affectés (il est initialisé à VIDE au départ),
- LIBRE comprend les pointeurs qu'on peut affecter (il comprend au départ tous les pointeurs possibles de 0 à 1023).

Pour occuper l'espace au maximum, les textures vont être rangées dans cet ordre :



La fonction récursive ci-dessous [MAR 82] permet de trouver le pointeur à utiliser suivant la taille de la texture : TAILLE.

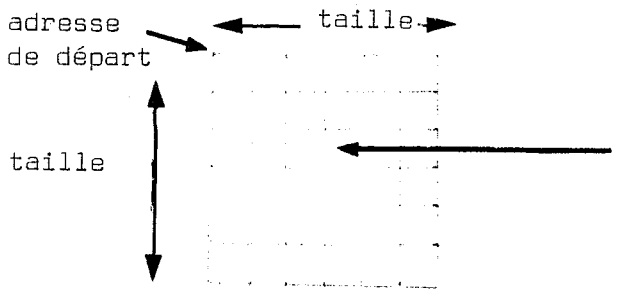
```
FONCTION POINTEUR LIBRE (i, j, +) : logique
DEBUT
   POINTEUR LIBRE ← faux
   si COMPLET (i, j) alors
      si (t = taille) alors
         si libre (i, j) alors
            pointeur libre ← vrai
            complet (i, j) ← vrai
            libre (i, j) ← faux
            pointeur texture ← (i, j)
         finsi
      sinon
         place ← vrai
         libre (i, j) ← faux
         t ← t div 2
         si pointeur libre (i, j, t) alors
            si pointeur libre (i + t, j, t) alors
               si pointeur libre (i, j + t, t) alors
                  si pointeur libre (i + t, j + t, t) alors
                     pointeur libre ← faux finsi
      finsi
   finsi
fin
```

Comme nous avons une mémoire de texture de 256 x 256, les pointeurs disponibles vont de 0 à 15 en x et de 0 à 15 en y. Les pointeurs impairs permettent d'affecter les plus petites textures : les 16 x 16. L'appel de cette fonction se fait par :

- POINTEUR LIBRE (0, 16, 256) ;

Si cette fonction délivre la valeur vraie, pointeur-texture comprend le numéro du pointeur libre.

La procédure BDTOUT écrit les points composant la texture, à partir de ce pointeur (qui comprend adresse et taille) :



chacun des petits carrés devra être rempli par un code couleur en (R V B)

La banque de données des textures a compacté les points de la texture suivant son type (unie, à rayures, ...). Actuellement quatre types différents sont acceptés :

. unie,
. rayures verticales (16),
. multi-point,
. point à point.

En effet, même si dans la mémoire de texture, pour une texture unie, la taille minimum est 16 x 16, un seul point contenant le code RVB de la couleur suffit, dans la banque de données des textures. Différentes procédures (une par type) permettent le remplissage suivant le compactage adopté.

2) Réorganisation des textures

Au cours d'un travail, de nombreuses textures sont utilisées et il est possible qu'à un moment donné la mémoire de texture soit pleine, malgré un rangement optimum. Pour cela, il faut pouvoir réorganiser la mémoire de texture et ne conserver que les textures réellement utilisées, certaines ayant pu être créées et non utilisées, ou supprimées en cours de travail.

Quand la mémoire de texture est pleine, l'utilisateur choisit :

. soit de RE-ORGANISER,

. soit d'ANNULER la communication des textures.

Les différentes étapes de la ré-organisation sont les suivantes :

■ création d'un tableau TFLADR qui contient le pointeur de texture associé à chaque face,

■ création d'un tableau TFLNBF qui contient le nombre de faces utilisées par texture,

■ si la réorganisation des possible (∃ i / TTLNBF (i) = 0) alors
  - suppression des textures dans la table des textures telle que TTLNBF (i) = 0 par TTLNOM [i] ← NOM-NIL
  - R.A.Z. des pointeurs de texture dans la table des faces,
  - initialisation de la mémoire de texture, des ensembles COMPLET, LIBRE
  - pour chaque texture de la table :
    . recherche de la taille par BDTOUT
    . recherche d'une place libre dans la mémoire de texture par POINTEUR LIBRE
    . écriture dans mémoire de texture par BDTOUT
    . affectation du pointeur dans la table des faces d'HELIOS (par l'intermédiaire de l'ancien pointeur conservé dans la table des textures et dans TFLADR)
    . affectation du pointeur à la table des textures

  finsi

Le temps mis pour la RE-ORGANISATION dépend du nombre, de la taille et du type de texture. Il risque d'être assez long tout de même, et le traitement a été implémenté de façon à ce que l'utilisateur ne soit pas bloqué dans la suite de ses travaux. Il devrait néanmoins être rare. Les deux premières étapes auraient pu se trouver lors de la communication de l'attribut texture. Nous ne l'avons pas fait, de façon à ne pas pénaliser l'utilisateur qui ne se sert jamais de ré-organisation.

5.3. Gestion des données morphologiques et géométriques

Toutes les informations morphologiques sont conservées ; par contre, seul le numéro de plan associé à une face est actuellement conservé, comme information géométrique. Néanmoins, si à l'avenir on voulait permettre des translations ou des rotations d'objets, il suffirait d'agrandir le fichier pour pouvoir mémoriser les transformations géométriques de chaque face.

Le fichier qui se trouve sur l'unité de disquette (n° 4) a comme nom MORPH. Un programme local CREATMORPH permet de l'initialiser comme un fichier comprenant des enregistrements de 34 réels.

1) Description de la base de données

Libellé	type de variable	place occupée
type morphologique	caractère	
nb de sommets ou nb caractères	entier	1 réel
numéro du plan associé	caractère	
si message		
x coin inférieur gauche	réel	1
y coin inférieur gauche	réel	1
message		
caractère 1 à 4	caractère	1
⋮		⋮ } 30 réels au max.
n-3 à n		1
si non message		
pour chaque sommet (15 au maximum)		
x } dans repère vue	réel	1 } 30 réels au max.
y } dans repère vue	réel	1

2) Accès aux données

Deux accès sont autorisés :

. *l'accès séquentiel* correspond à des entrées-sorties sur des enregistrement consécutifs,

. *l'accès direct* permet la lecture ou l'écriture du nième enregistrement.

Les enregistrements sont rangés dans l'ordre des numéros de faces (0 .. 1023 Un accès direct revient donc à un accès indexé sur le numéro de face.
La combinaison des deux accès : positionnement de la fenêtre de lecture sur la iième face, puis lecture séquentielle, permet de simuler le séquentiel indexé quand cet accès est utile (exemple : lecture des faces consécutives composant le iième objet en vue de son affichage).

Quelles procédures accèdent aux données :

. en lecture-écriture, l'accès peut se faire par ENTREEPV, ENTREMES qui entrent les données morphologiques, ECRFP qui entre les données géométriques ;

. en lecture, ECRFACE (NOFACE) permet un accès direct, REMPI accède séquentiellement à toutes les faces.

5.4. Remplissage des faces dans les plans d'identification

Pour visualiser une scène, deux possibilités s'offrent à l'utilisateur pour afficher les faces décrites :

. soit demander l'affichage complet de la scène,

. soit grâce à la mémorisation des informations synthétisées (M. G. G_V. G_A) sur disquette, l'affichage d'une liste de faces.

Les types d'éléments que nous aurions aimé avoir en entrée étaient constitués par des faces planes, superposées. Le lecteur trouvera en annexe 3 une implémentation de ces éléments. Par manque de place mémoire (pour les programmes et les données), nous avons abandonné ce type, pour des éléments plus simples, composés de faces planes non superposées.

Nous décrivons ici comment a été implanté l'opérateur de synthèse (I + M. G. G_V. G_A), c'est-à-dire comment réaliser le remplissage des plans d'identification en fonction des informations morphologiques (éventuellement modifiées par des transformations géométriques).

Trois remplissages vont être successivement évoqués : celui des caractères, celui des segments, et celui des surfaces polygonales.

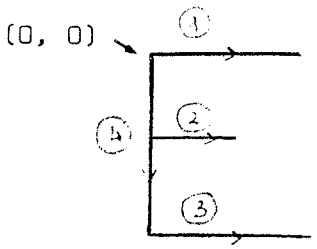
5.4.1. Les caractères

Les caractères transmis sous forme de caractères ASCII doivent être traduits par des points représentant ces caractères dans les plans d'identification. Comme nous n'avions pas de générateur de caractères, il a fallu pré-enregistrer la forme des caractères. Une table de correspondance nous donne donc la représentation graphique des caractères ASCII. Elle pouvait être organisée de deux façons différentes au moins :

1) chaque caractère est représenté par plusieurs segments orientés

L'orientation se fait comme suit : 1 → 3 ←
(ainsi, par deux bits, toutes les 2 ↓ 4 ↑
directions peuvent être désignées).

Les segments sont définis par leur origine (déplacement en x et en y par rapport au coin supérieur gauche du caractère) et le nombre de points, ce qui demande 12 bits au maximum pour représenter x, y, d. Chaque caractère comprend en moyenne 5 segments. Comme un segment peut être représenté par 14 bits, chaque caractère est représenté sur 70 bits environ (figure 68).



segment	direction	x	y	d
1	1	0	0	7
2	1	3	0	3
3	1	8	0	7
4	2	0	0	9

Figure 68 - Représentation du caractère E en 7 x 9

2) Chaque caractère est représenté par une matrice de points

Chaque point correspond à un booléen auquel on affecte la valeur vrai quand il est rempli, faux quand il est vide (figure 69).

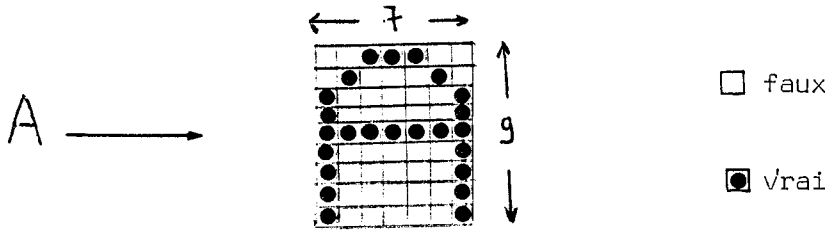


Figure 69 - Représentation du caractère A en 7 x 9

Cette représentation prend (7 x 9) soit 63 bits pour chaque caractère. Le seul avantage de cette méthode vient du fait que la place occupée ne dépend pas de la représentation graphique des caractères. Nous avons implanté cette solution avec des caractères 7 x 9, les caractères 5 x 7 s'avérant beaucoup trop petits. Les matrices de points ont été définies pour les caractères majuscules (voir photo n° 70) et indexées suivant l'ordre des caractères ASCII.

Photo n° 70

L'accès au coin supérieur gauche d'une lettre quelconque, notée CAR, se fait par TABC7x9[ORD(CAR), 1, 1]

L'algorithme de remplissage prend en compte les possibilités de compactage d'HELIOS.

```
procédure REMPC7x9 (CAR, COINSUPGAUCHE (x, y), NO-PLAN, NO-FACE)
DEBUT
  si CAR ∈ TABLEAU DES CARACTERES DECRITS alors
    pour J ← 1 à 9 faire
      n ← 0
      xx ← x
      pour i ← 1 à 7 faire
        si TABC7x9 [ORD (CAR), i, j] alors n ← n + 1
        sinon
          si n ≠ 0 alors écriture de n points par
             SEGMENT (y + j - 1, xx, xx + n - 1)
             n ← 0
          sinon xx ← xx + 1 finsi
        finsi
      fin pour
      si n <> 0 alors écriture des n points par
         SEGMENT (y + j - 1, xx, xx + n - 1)
    finpour
  finsi
FIN
```

5.4.2. Les éléments "fil de fer"

Deux types morphologiques permettent de décrire des "éléments fil de fer" : les éléments disjoints (S) et les lignes brisées (L).

La représentation graphique se fait à partir des sommets des éléments. Ces sommets sont pris deux à deux comme suit :

- pour les segments disjoints (j, j + 1), (j + 2, j + 3) ...
- pour les lignes brisées (j, j + 1), (j + 1, j + 2) ...

Pour tracer un vecteur, compte tenu du fait que le matériel n'admet que des segments horizontaux, il faut calculer les segments horizontaux qui composent ce vecteur et trouver une approximation telle que l'oeil puisse reconstruire le segment de droite représenté.

Dans la littérature, plusieurs algorithmes ont été proposés [BRE 65], [EAR 7 [LUC 74], [BRE 82], pour l'inscription point par point d'un segment quelconque. Leur caractéristique principale est d'éviter les opérations de multiplication et de division et de travailler en entiers. Pour notre part, nous avons tenu compte de ces remarques et implémenté un algorithme de numérisation, proche de celui de M. LUCAS [LUC 74], mais qui prend en compte les possibilités de compactage du matériel. Schématiquement le problème est le suivant (figure 71) :

```
Pour chaque ligne y ← yd à yf faire
|   . calcul de l'origine et du nombre de points
|   . affichage du segment
finpour
```

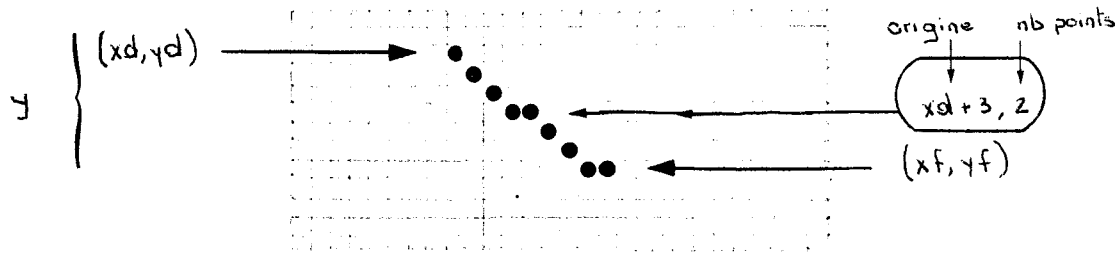
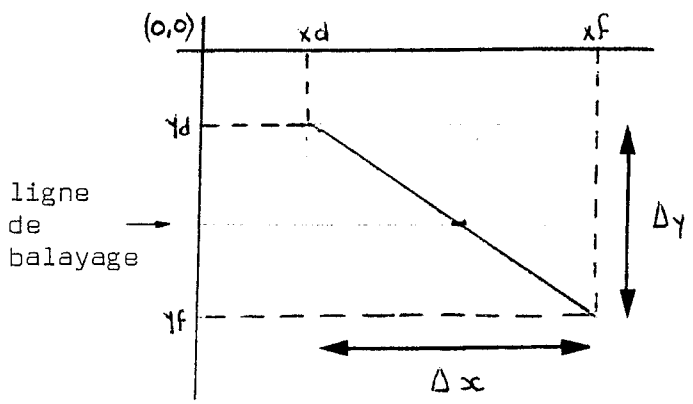


Figure 71 - Numérisation d'un vecteur

Pour optimiser le tracé et n'utiliser que des additions et des soustractions sur des entiers, l'algorithme utilise les paramètres suivants :



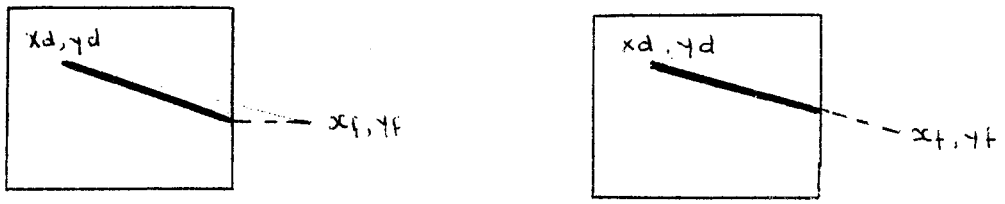
- xd, yd ← point de départ
- xf, yf ← point final
  si yf > yd alors le point final deviendra le point de départ et inversement
- Δx = valeur absolue de (xf - xd)
- Δy = yf - yd
- ix représente l'incrément en x ; il est déterminé par le sens de la pente et prend les valeurs 1, 0, o -1 suivant le signe de (xf - xd)
- h est un paramètre permettant de savoir quand xd doit être incrémenté suivant la pente.

La procédure NUMERIS donne, pour chaque ligne de balayage, le segment horizontal à afficher, c'est-à-dire détermine le x de gauche et le x de droite en fonction des paramètres Δx, Δy, ix, h, xd et xf. Celle-ci est utilisée par la procédure principale qui se charge de l'initialisation des paramètres.

La procédure principale, REMPVEC, travaille comme suit :

```
PROCEDURE REMPVEC
DEBUT
|  si yf <> yd alors
|     si yf < yf alors permutation finsi
|     initialisation de Δx, Δy, ix, xdeb, xfin, h (← 0)
|     pour j ← yd à yf faire
|        NUMERIS (Δx, Δy, ix, h, xdeb, xfin, xgauche, xdroite)
|        si xgauche <= borne sup de l'écran alors
|           si xdroite > borne sup de l'écran alors
|              xdroite ← borne sup de l'écran finsi
|           écriture du SEGMENT HORIZONTAL par SEGMENT (j, xgauche, xdroite)
|        finsi
|     fin
|  sinon écriture du segment horizontal :
|        si xf > borne sup. de l'écran alors xf ← borne sup de l'écran
|        SEGMENT (yd, xd, xf)
|  finsi
fin
```

Quand les coordonnées (x ou y) d'un point sont en dehors de l'écran, nous tronquons, après la numérisation et non à l'initialisation, de façon à ne pas provoquer l'erreur schématisée par la figure 72.



x est tronqué, à l'initialisation à la borne supérieure de l'écran

x n'est pas tronqué

Figure 72 - Génération d'un vecteur dont l'une des bornes est en dehors de l'écran.

L'algorithme de numérisation utilise les résultats obtenus à la ligne j pour calculer les valeurs des paramètres de la ligne j + 1. Cette implémentation se réalise seulement à l'aide d'additions et de soustractions et pourrait facilement être câblée ou micro-programmée.

```
PROCEDURE NUMERIS
DEBUT
|  XD ← XDEB
|  si Δx >= Δy alors
|  |  h ← h - Δy
|  |  tant que (valeur absolue (h) < valeur absolue (h + Δx))
|  |  et (XDEB ≠ XFIN) faire
|  |  |  incrémentation de XDEB
|  |  |  h ← h - Δy
|  |  fin tant que
|  |  XF ← XDEB
|  |  si XD > XF alors permutation XD et XF
|  |  XGAUCHE ← SD
|  |  XDROITE ← XF
|  |  incrémentation XDEB
|  |  h ← h + Δx
|  sinon
|  |  XF ← XDEB
|  |  si XD > XF alors permutation XD et XF
|  |  XGAUCHE ← XD
|  |  XDROITE ← XF
|  |  h ← h + Δx
|  |  si valeur absolue (h) > valeur absolue (h - Δy) alors
|  |  |  incrémentation de XDEB
|  |  |  h ← h - Δy
|  |  finsi
|  finsi
fin
```

Les résultats obtenus, même par logiciel, sont satisfaisants au niveau du temps de réponse. La qualité du dessin dépend, quant à elle, de la définition et est correcte en 512 x 512.

5.4.3. Remplissage d'une tâche

5.4.3.1. Analyse des algorithmes existants

1) Tâche dont le contour est défini point par point

Quand le contour d'une tâche est connu par l'ensemble des points qui le composent, il est possible de connaître un point à l'intérieur de cette tâche à remplir. L'algorithme progresse alors d'une manière identique au coloriage manuel : ligne par ligne, en revenant ensuite sur les zones oubliées [LIE 78] (figure 73).

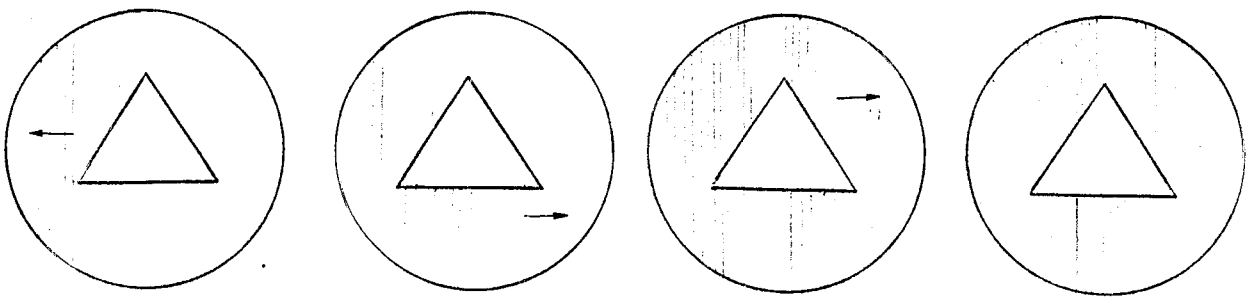


Figure 73 - Coloriage d'une tâche

Le principal problème rencontré est le remplissage de tâches disjointes ; on ne peut alors utiliser cet algorithme. Une autre méthode consiste à analyser le contour, selon des techniques semblables à celles utilisées pour l'extraction de contours en traitement d'images, afin d'obtenir les informations nécessaires au remplissage [PAV 78].

2) Tâche dont le contour est défini de façon mathématique ou algorithmique

Actuellement, des algorithmes permettent de générer rapidement des cercles, des ellipses [BRE 82], à l'aide de comparaisons, de décalages ou d'additions. Il suffit alors de partir du sommet du cercle et de calculer pour chaque ligne horizontale la frontière gauche et la frontière droite.

3) Tâche définie par des contours polygonaux

Une façon simple de décrire un objet est de l'approximer par un certain nombre de facettes polygonales.

a) **Un algorithme simple** part d'une arête quelconque et inverse tout ce qui est à droite (figure 74).

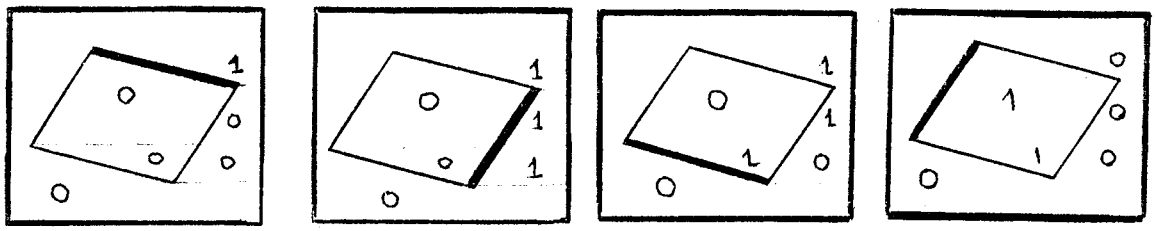


Figure 74 - Traitement de chaque arête sur un exemple

Le codage de chaque point sera obligatoirement fait sur deux bits, la codification pouvant être faite comme suit :

point externe au polygone : 00
contour droit : 10
contour gauche : 01
point interne : 11

Cet algorithme a été câblé et permet un remplissage très rapide pouvant se trouver au niveau du matériel.

b) **Un autre algorithme de remplissage** peut être le suivant (figure 75) :

- tri des arêtes dans l'ordre décroissant des coordonnées y de chaque sommet (tri de haut en bas),
- calcul des incréments en x,
- remplissage à partir du sommet du polygone dès qu'on rencontre un autre sommet (=> fin de traitement d'une arête) ; les intersections des arêtes si elles existent, sont triées de gauche à droite [LUC 77], [MER 79].

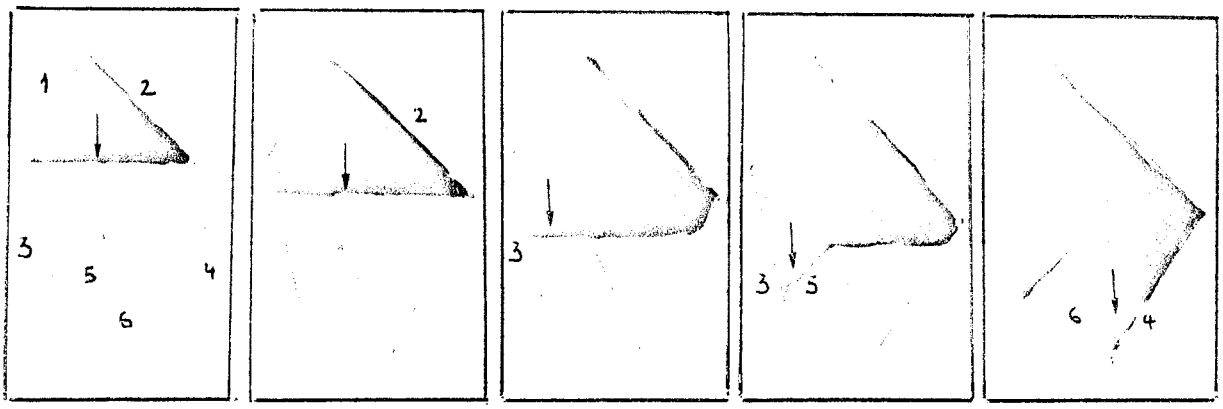


Figure 75 - Numérotation des arêtes et remplissage à chacune des étapes

Il conviendra mieux que le précédent, dans un traitement par logiciel. En effet, ne seront remplies que les zones "utiles" de la figure.

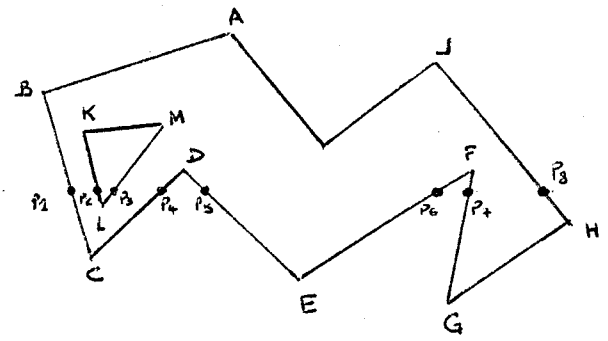
5.4.3.2. Implémentation d'un algorithme

Les seuls types morphologiques acceptés comme "éléments pleins", dans ce synthétiseur, sont les faces polygonales planes. Par ailleurs, HELIOS ne peut faire l'affichage que si la mémoire de trame contient tous les points correspondant au contour et à l'intérieur des polygônes. Ces points sont représentés dans les divers plans d'identification par le numéro de face associé aux polygônes.

Nous avons choisi d'implémenter le dernier algorithme décrit dans le paragraphe précédent, qui effectue le remplissage ligne par ligne. Il est particulièrement adapté à HELIOS qui permet l'écriture de plusieurs points consécutifs d'une même ligne en une seule opération.

De façon schématique, la méthode est la suivante (figure 76) : pour chaque ligne :

- détermination de l'ensemble des arêtes l'intersectant,
- tri de ces arêtes de gauche à droite,
- calcul des points d'intersection,
- affichage des vecteurs horizontaux reliant les points d'intersection deux à deux.



ARETES EN CAUSE : JH, FG, EF, DE, CD, BC, KL, LM
ARETES ORDONNEES : BC, KL, LM, CD, DE, EF, FG, JH
INTERSECTIONS : P1, P2, P3, P4, P5, P6, P7, P8
AFFICHAGE DES VECTEURS : P1-P2, P3-P4, P5-P6, P7-P8

Figure 76 - Remplissage d'une ligne de balayage

Pour éviter les calculs redondants, l'algorithme utilise les astuces suivantes :

- les arêtes intersectant la ligne i + 1 se déduisent des arêtes intersectant la ligne i :
  - en enlevant d'abord les arêtes qui se terminent à la ligne i,
  - puis en rajoutant celles qui commencent à la ligne i + 1 ;
- le calcul des intersections des arêtes avec la ligne de balayage i + 1 est déduit de la ligne i, grâce à la mémorisation des pentes de toutes les arêtes, ainsi que des divers paramètres de la numérisation.

La description d'un polygône est identique à la description d'un ensemble de lignes brisées. Quand un polygône comporte un ou plusieurs trous, le sommet non terminal de chaque ligne brisée est répété en fin de ligne brisée les lignes brisées sont données dans n'importe quel ordre. La détermination des arêtes se fait donc comme suit :

```
FIN-POL ← 1
POUR i ← 1 à NB SOMMET-1 faire
|   ARETE [i] ← (sommet i, sommet i + 1)
|   si sommet i + 1 = FIN-POL alors
|   |   i ← i + 1
|   |   FIN-POL ← i
|   finsi
finpour
```

L'algorithme implémenté est décrit dans la procédure REMPFACE :

```
PROCEDURE REMPFACE
DEBUT
  . initialisation des tableaux des arêtes entrantes et sortantes à NIL
  . on ordonne les sommets en y dans LISTEY et calcule YMIN et YMAX
  . détermination des paramètres de numérisation des arêtes par le par-
    cours de(s) ligne(s) brisée(s) et numérotation des arêtes
    - pour chaque sommet, on note les numéros d'arêtes entrantes et sortantes
    - pour chaque arête on détermine les paramètres qui permettront la
      digitalisation par la méthode des cumuls (Δx, Δy, ix, h, xdeb, xfin)
  . éventuelle troncature de ymax aux bords supérieurs de l'écran
  pour y ← ymin à ymax faire
    tantque des sommets sont sur cette ligne de balayage (LISTEY)
      mise à jour de la liste ordonnée des arêtes courantes (TABCOUR)
      qui comprend les arêtes de début et de fin qui limitent chaque
      portion de segments
    fin tantque
    pour j ← i à fin TABCOUR
      i ← TABCOUR [j]
      NUMERIS (Δx[i], Δy[i], ix[i], h[i], XDEB[i], XFIN[i], XGAUCHE[i],
      XDROITE[i]) qui permet de calculer le x du bord gauche et du bord
      droit de chaque arête
    finpour
    i ← 1
    tantque i <= fin TABCOUR
      si DROITE [TABCOUR[i]] < bords de l'écran
        on prend les arêtes deux à deux et on écrit le segment horizontal
        du bord gauche de l'arête de gauche au bord droit de l'arête de
        droite. C'est à ce moment là que l'on tronque le segment si
        celui-ci, à cause des différentes transformations géométriques,
        dépasse le cadre de l'écran :
        - éventuelle troncature de GAUCHE [TABCOUR[i]]
        - écriture du segment horizontal
          SEGMENT (Y, GAUCHE [TABCOUR[i]], DROITE [TABCOUR [i + 1]])
      finsi
      i ← i + 2
    fin tantque
  fin pour
FIN
```

5.4.3.3. Analyse des résultats obtenus

L'avantage de cet algorithme est qu'il traite tous les cas en n'effectuant qu'une numérisation des arêtes pour les points se trouvant à l'intérieur d'un trapèze ou d'un triangle. Mais la procédure ainsi déterminée est tout de même relativement longue d'exécution. Une amélioration logicielle de ce temps de réponse ne semble guère possible.

Par contre, si un microprocesseur était affecté au remplissage des trapèzes et des triangles, le logiciel devrait simplement décomposer le polygône en une suite de trapèzes et de triangles (figure 77), puis les communiquer au microprocesseur.

Avec un algorithme simple, le temps de remplissage d'un objet d'une cinquantaine de faces, est de l'ordre de 2 minutes 20 secondes. Avec notre algorith il se situe à environ 23 secondes, ce qui représente un rapport de 10. Il semble qu'avec le microprocesseur, en utilisant le parallélisme, décomposition logicielle - remplissage microprocesseur, on pourrait approcher "les quelques secondes", ce qui est acceptable pour les opérations de conception assistée par ordinateur.

Notons que la décomposition logicielle en trapèzes et triangles demande seulement quelques petites modifications de la procédure REMPFACE.

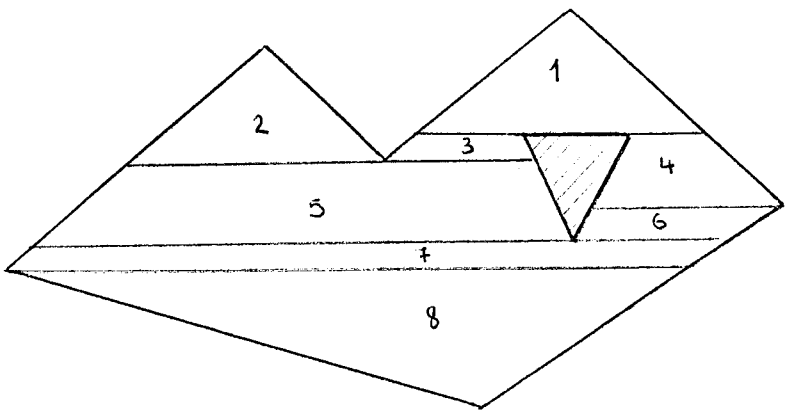


Figure 77 - Décomposition d'un polygône en trapèzes et triangles

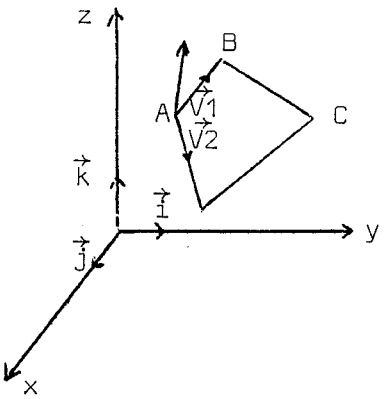
5.5. Transmission des repères tri-dimensionnels

Du repère associé à une face, nous devons déduire :

. la normale,

. la matrice de projection.

La connaissance de trois points de cette face nous permet de calculer, par convention, les vecteurs de base de la face $\vec{V1}$ et $\vec{V2}$ (figure 78).



$$A = \begin{pmatrix} x_A \\ y_A \\ z_A \end{pmatrix} \qquad B = \begin{pmatrix} x_B \\ y_B \\ z_B \end{pmatrix} \qquad C = \begin{pmatrix} x_C \\ y_C \\ z_C \end{pmatrix}$$

$$\vec{V1} = \begin{pmatrix} x_B - x_A \\ y_B - y_A \\ z_B - z_A \end{pmatrix} \qquad \vec{V2} = \begin{pmatrix} x_D - x_A \\ y_D - y_A \\ z_D - z_A \end{pmatrix}$$

Figure 78 - Les vecteurs de base $\vec{V1}$ et $\vec{V2}$

1) Calcul de la normale

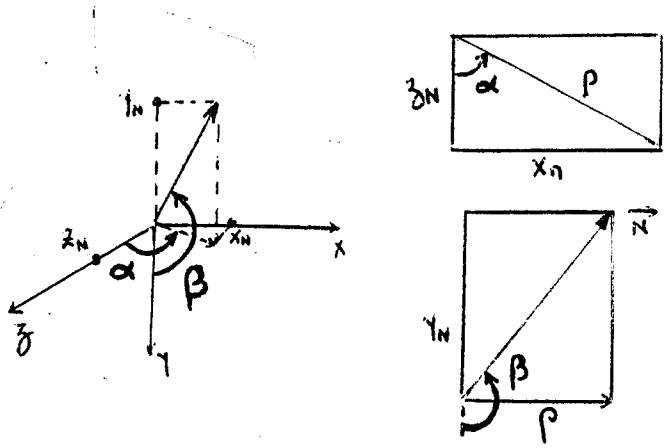
La normale $\vec{N}$ est le produit vectoriel des vecteurs de base V1 et V2.

$$\vec{N} = \begin{pmatrix} XN \\ YN \\ ZN \end{pmatrix} = \vec{V1} \wedge \vec{V2}$$

On a donc :

$$XN = (y_B - y_A)(z_C - z_A) - (z_B - z_A)(y_C - y_A)$$
$$YN = (x_B - x_A)(z_C - z_A) - (z_B - z_A)(x_C - x_A)$$
$$ZN = (x_B - x_A)(y_C - y_A) - (y_B - y_A)(x_C - x_A)$$

La transformation des coordonnées cartésiennes en coordonnées sphériques revient, comme le montre la figure 79, au calcul des angles alpha et beta.



$$\rho = \sqrt{x_N^2 + z_N^2}$$

Figure 79 - Transformation des coordonnées cartésiennes en coordonnées sphériques

```
PROCEDURE NORMALE (XN, YN, ZN, alpha, beta)
DEBUT
   si ZN < 0 alors changement de signe XN, YN, ZN
   ALPHA ← ANGLE (XN, ZN)
   ρ = √(XN² + YN²)
   BETA ← ANGLE (ρ, YN)
FIN
```

```
FONCTION ANGLE (XN, ZN)
DEBUT
   si ZN = 0 alors ANGLE ← π/2
   sinon angle   ARCTG (|XN| / |ZN|) finsi
   si (XN = 0) et ZN < 0 alors ANGLE ← π  finsi
   si XN . ZN < 0 alors angle ← π - angle finsi
   si XN < 0 alors angle ← angle - π
FIN
```

ceci est dû au fait qu'ARCTAN nous donne l'angle à π près

2) La matrice de projection

Elle est représentée par les vecteurs de base $\vec{V1}$ et $\vec{V2}$:

$$M = \begin{vmatrix} X_{V1} & X_{V2} \\ Y_{V1} & Y_{V2} \end{vmatrix} = \begin{vmatrix} x_D - x_A & x_B - x_A \\ y_D - y_A & y_B - y_A \end{vmatrix}$$

Cette matrice est transmise à HELIOS par la procédure PROJECTION qui l'inverse et se charge de la symétrie du repère si les coefficients diagonaux sont négatifs (cf. § 4.5.1.).

5.6. Représentation de fonctions à deux variables z = f(x, y) dans R^3 par l'approximation de surfaces polygonales planes

L'objectif de ce paragraphe est de montrer que des éléments plus complexes peuvent être visualisés par décomposition en éléments simples, compris par le système.

Nous avons choisi de représenter ces courbes (figure 71) avec des éléments "pleins" et non avec des éléments "fil de fer". Nous allons donc décrire un algorithme simple de décomposition en faces polygonales planes, pour l'exemple.

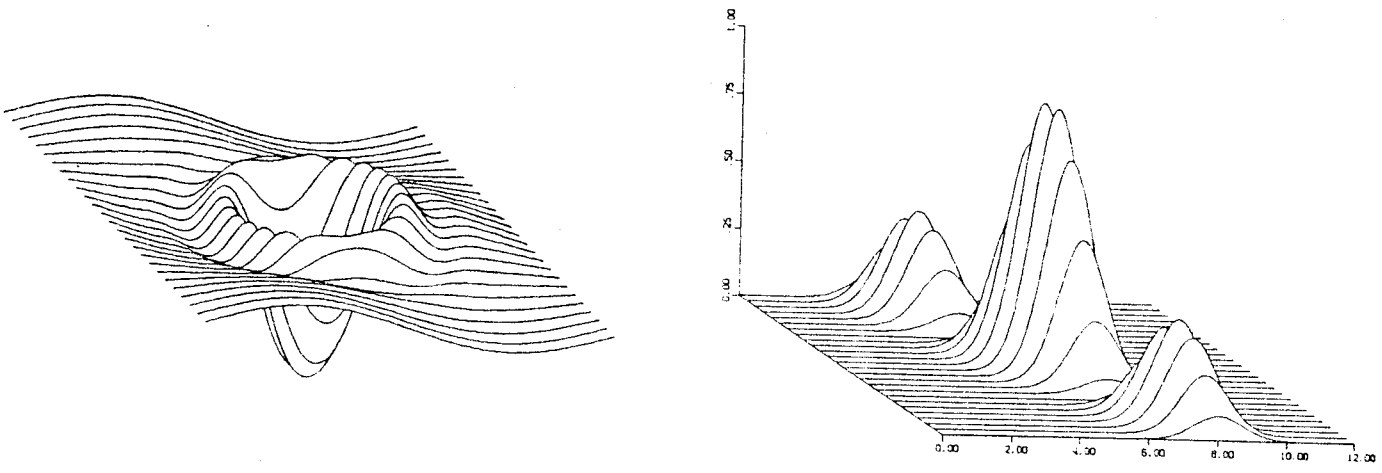
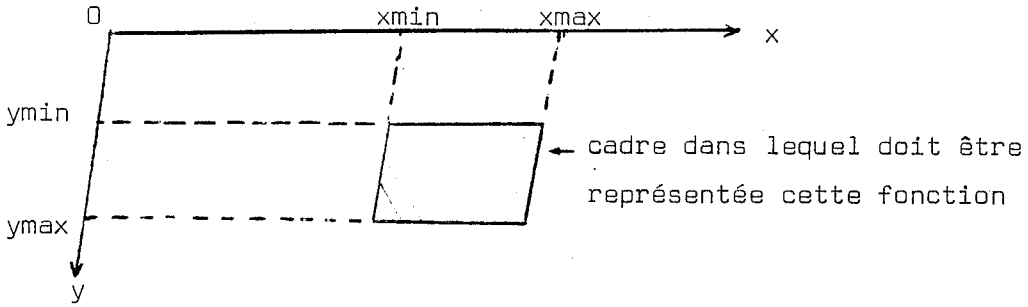


Figure 71 - Représentation "fil de fer" de fonctions à deux variables

La fonction à analyser est représentée par son équation (par exemple z = ax + by + c) et son domaine de visualisation en x (xmin, xmax) et en y (ymin, ymax).

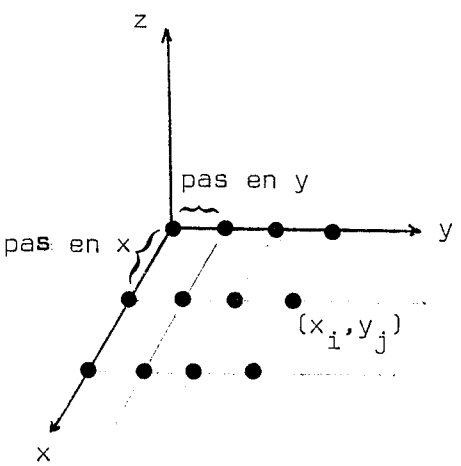


Dans tous les calculs, nous utiliserons les coordonnées homogènes, de façon à pouvoir représenter par une matrice l'ensemble des transformations géométriques appliquées. Comme nous travaillons dans R^3, M sera une matrice 4x4. Les étapes de la visualisation de cette fonction sont les suivantes :

1) Translation du cadre à l'origine

$$M = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -xmin & -ymin & 0 & 1 \end{matrix}$$

2) Calcul du pas que l'on veut appliquer en x et en y pour faire le maillage



HELIOS admet au plus 1024 faces. Le maillage pourrait être défini avec 32 pas en x et 32 pas en y ; la valeur d'un pas en x serait de $\frac{xmax - xmin}{32}$ et en y de $\frac{ymax - ymin}{32}$.

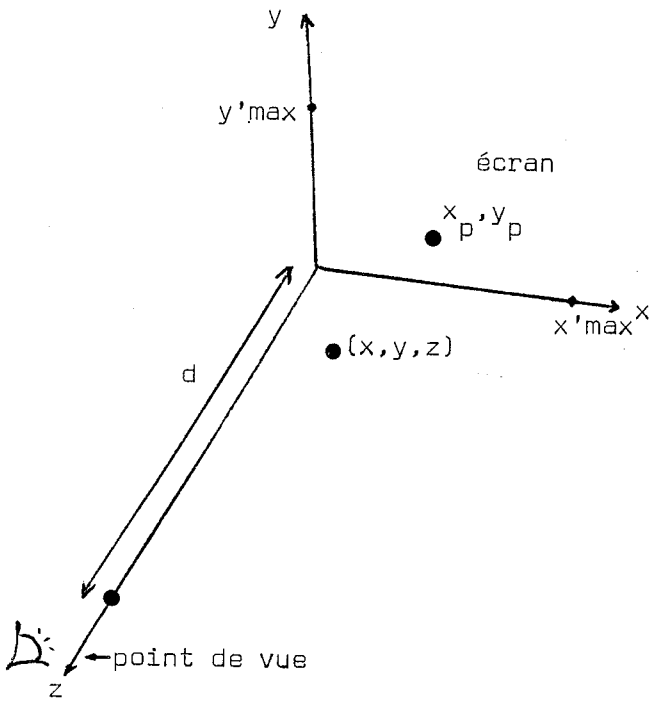
Le pas en x et en y pourrait aussi être demandé à l'utilisateur, ou mieux, calculé par étude de la fonction (mais de toutes façons : nombre de pas en x * nombre de pas en y <= 1024).

3) Calcul de z en fonction de chaque point (x_i, y_j) du maillage

Tz (i, j) = f(x_i, y_i) avec x_i = origine en x + i . pas en x

y_i = origine en y + j . pas en y

4) Projection en perspective sur l'écran



La matrice de transformation devient :

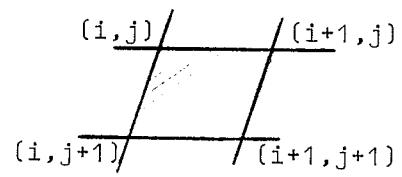
$$M = M \times \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1/d \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\text{matrice de projection}}$$

Quand on applique cette matrice de transformation au point (x, y, z), on a :

$$x_p = \frac{x - xmin}{1 - \frac{z}{d}}$$

$$y_p = \frac{y - ymin}{1 - \frac{z}{d}}$$

5) Affichage des quadrilatères ainsi définis



POUR j ← 0 à nb de pas max en y - 2 faire

  pour i ← 0 à nb de pas max en x - 2 faire

    . affectation des coordonnées en x (dans TX) en y (dans TY) pour les quatre sommets (i, j) (i+1, j) (i, j+1) (i+1, j+1)

    . numéro face ← j . (nb de pas max en y - 1) + i

    . appel procédure de remplissage

      REMPFACE (no-plan, numéro-face, 4, TX, TY)

                                  ↑ nb de sommets

    . calcul normale et (éventuellement matrice de projection) avec 3 points (cf. § 5.4.) et communication au matériel pour la face numéro-face

  fin pour

fin pour

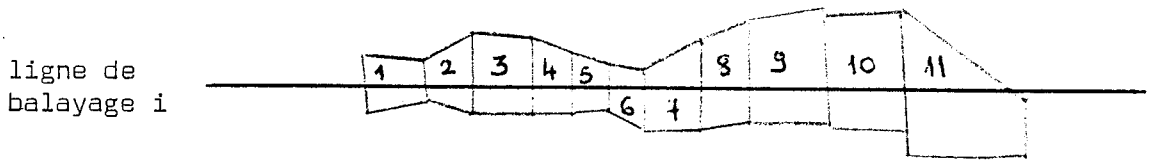
Pour la réalisation de ce programme, nous ne suivrons pas ces différentes étapes qui demanderaient une mémorisation très lourde des données.

Nous conservons la procédure d'affichage et les coordonnées (x_p, y_p) des sommets concernés seront calculées avant l'affectation. Des variables tampons conserveront les sommets (i+1, j) et (i+1, j+1) pour éviter des doubles calculs.

Ce programme fonctionne localement.

La génération des faces polygonales n'est nécessaire que pour le calcul de la normale qui permet à l'oeil d'appréhender le relief, à travers les effets d'éclairage. Si les plans d'identification ne comprenaient pas non plus les numéros de faces mais la normale en chaque point (extension prévue) l'algorithme de visualisation serait tout autre et permettrait d'avoir des représentations plus lisses et donc plus réalistes.

Le principal inconvénient de la méthode décrite plus haut est que le remplissage se fait quadrilatère par quadrilatère et non ligne de balayage après ligne de balayage :



La ligne de balayage i est appelée 11 fois, alors qu'il semble plus rapide d'appeler une fois la ligne i et de remplir les plans d'identification avec les 11 segments comportant un numéro de face différent. Cette optimisation demanderait :

. le calcul des points du maillage et la projection de ceux-ci sur l'écran dans un ordre qui permette en même temps le calcul et l'affectation des normales des faces ;

. le calcul des intersections des arêtes des faces avec la ligne de balayage

Il semble néanmoins que l'ensemble des calculs avec ce procédé, demande une place mémoire importante pour conserver les données, même si les segments utilisés à la ligne i+1 sont déduits de ceux utilisés à la ligne i, en fonction des arêtes qui débutent ou qui se terminent.

5.7. Conclusion

La réalisation de l'ensemble du logiciel s'est faite module après module, de façon aisée. Par contre, l'assemblage de ces modules s'est souvent heurté à des problèmes de place mémoire, et certains modules ont dû être abandonnés, ou ne seront développés que localement.

Nous avons réalisé un système minimum où l'ajout d'un élément de base ou d'un type d'information, ne remet pas en cause l'architecture générale, mais se traduit par l'ajout d'un module de modélisation et d'un module de visualisation.

CONCLUSION

Les modules de base du logiciel pilote pour le prototype HELIOS, sont ajourd'hui terminés. Tout en réalisant des fonctions de synthèse non intégrées au matériel, nous avons essayé d'utiliser au maximum les possibilités du matériel.

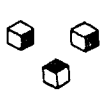
Nous avons noté dans les différents chapitres, les améliorations qui devraient être apportées au matériel pour une interactivité en temps réel plus efficace (utilisation de microprocesseurs, de générateur de caractères, etc ...).

Les différents modes de fonctionnement du terminal (local, relié au HB 68) permettent une certaine souplesse et la définition des interfaces des outils de dialogue devrait faciliter la description. En effet, le principal problème qui limite le nombre de scènes créées, est la description de la scène (description des objets, des relations entre ces objets). Un logiciel de description et de modélisation s'appuyant sur le logiciel pilote pour la visualisation et sur la tablette pour la collection des coordonnées, devrait être réalisé pour que ce terminal assure plus complètement son rôle de conception assistée par ordinateur.

Notons par ailleurs que la visualisation d'images synthétiques sur d'autres consoles, demande la mise en place d'autres logiciels pilotes. Il semblerait intéressant de définir, de façon automatique, le chemin à suivre parmi les différentes opérations de synthèse, en fonction des possibilités du dispositif d'affichage.

Le logiciel pilote réalisé sur HELIOS est une preuve de plus du bien fondé de la séparation des informations de synthèse le plus longtemps possible. Une augmentation de la capacité mémoire permettrait d'envisager de traiter toutes les informations de synthèse au niveau du logiciel pilote, (meilleure interactivité).

Nous pensons que cette réalisation est un pas de plus dans le développement de nature logicielle, pour une synthèse interactive en temps réel, mais de nombreux axes de recherche sont encore à explorer.



ANNEXE 1

MODÉLISATION DE L'ASPECT INTRINSÈQUE DES OBJETS

ET DES PARAMÈTRES D'ÉCLAIRAGE

I - LA MODELISATION DE L'ASPECT INTRINSEQUE DES OBJETS

Après avoir bâti une scène, il faut la colorier tout en tenant compte de la texture des objets, de la transparence ou de l'opacité de ceux-ci. Nous allons donc essayer de déterminer le comportement des objets vis-à-vis de la lumière qui détermine l'aspect de ceux-ci (couleur,...).

1.1. La modélisation d'une couleur quelconque [MAR 79]

La couleur rend plus intelligente les images tridimensionnelles complexes ; elle permet de distinguer la surface interne de la surface externe d'un volume ; elle sert à l'interprétation rapide des informations graphiques. Bien utilisée, elle améliore la productivité, réduit la fatigue, diminue le nombre d'erreurs. Le monde que nous percevons est coloré, mais "modéliser cette couleur" implique des logiciels plus complexes et des moyens plus importants. C'est ce qui explique sans doute que la coloration est une technique assez récente.

1.1.1. Les principes de la trichromie

Tous les procédés de reproduction des couleurs s'appuient sur les principes de la trichromie, énoncés par YOUNG, MAXWELL et HELMHOLTZ, d'après lesquels on peut recréer les sensations colorées par mélange de trois couleurs primaires judicieusement choisies.

La colorimétrie ou science de la couleur définit trois teintes fondamentales, le rouge, le vert et le bleu, qui mélangées donnent le blanc.

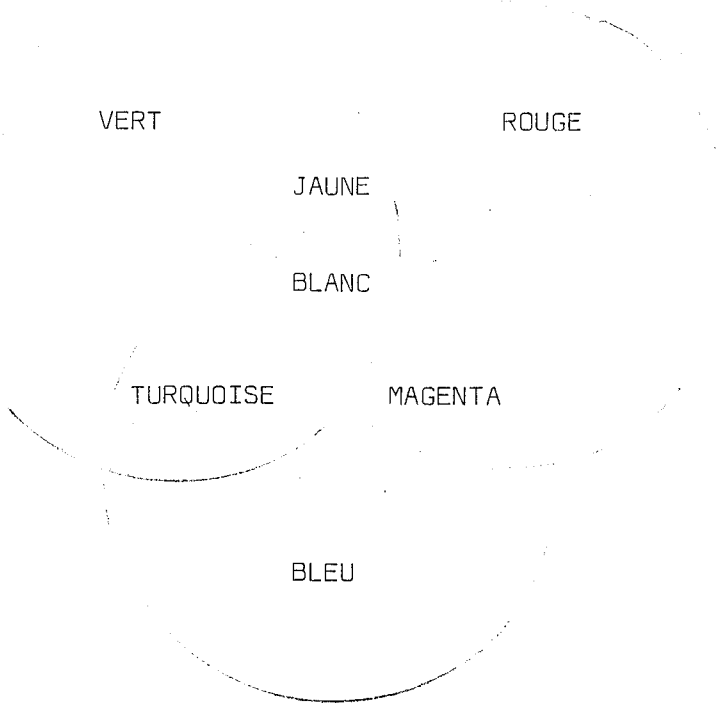


Schéma 1 - L'association des 3 couleurs additionnée deux à deux donne le magenta, le turquoise et le jaune. Les trois donnent normalement du blanc. En fait, à luminance égale, ce n'est pas vrai. Pour créer artificellement du blanc à partir des trois fondamentales, il faut respecter la proportion 23,75% rouge + 65% vert + 11,25% bleu

Ainsi une couleur quelconque C est représentée à l'aide des trois coefficients représentant l'intensité des trois primaires rouge, vert et bleu :

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

↑ intensité de la couleur c

On utilise ainsi l'additivité des couleurs ; or, la synthèse des couleurs utilisées en peinture par exemple est une synthèse coustractive ; il serait donc plus judicieux d'utiliser l'espace inverse de (R,V,B) qui est (C,M,J) (cyan, magenta et jaune).
La transformation des coefficients normalisés étant immédiate :

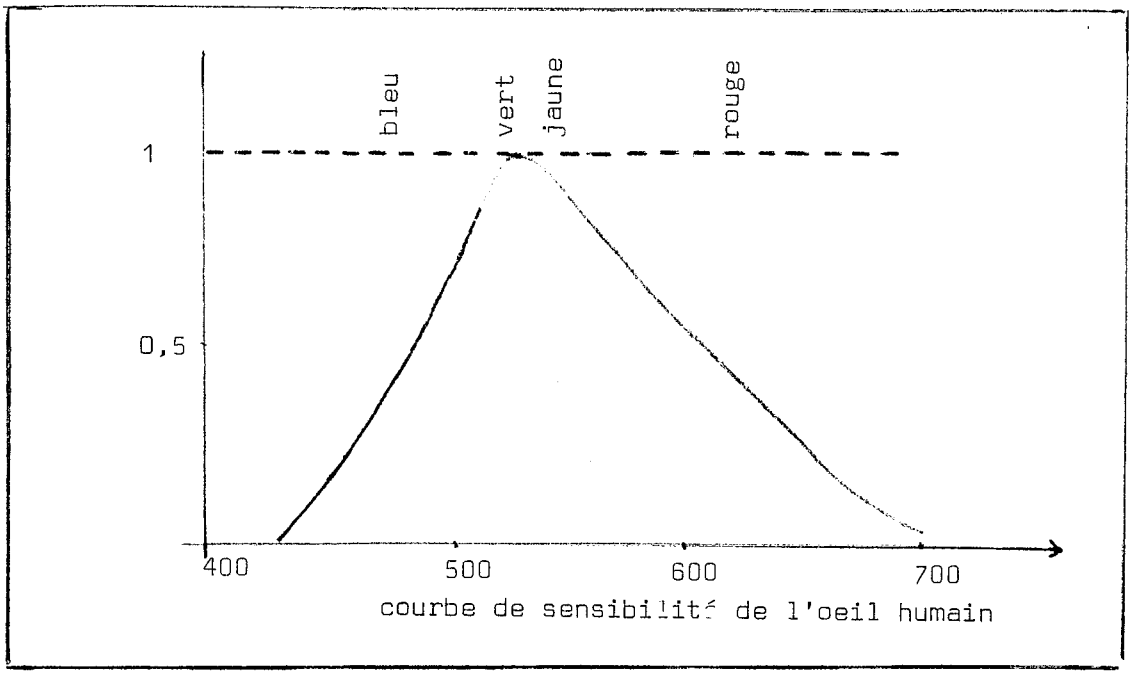
$$(r,v,b) = (1,1,1) - (c,m,j).$$

1.1.2. Teinte, saturation et luminance [MAC 81]

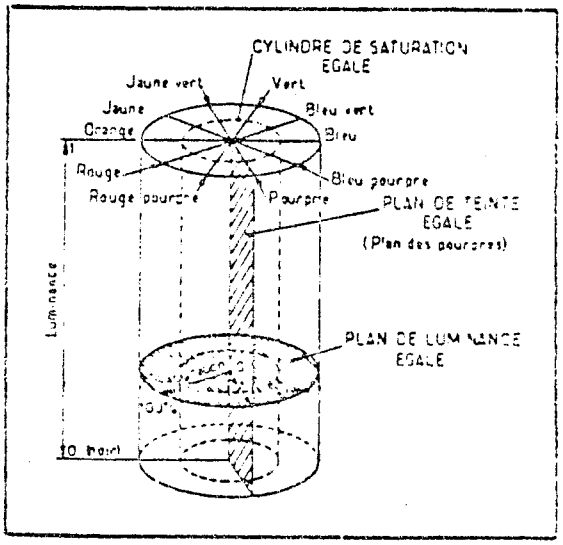
La teinte associe la couleur à une longueur d'onde déterminée (ou fréquence) Dans le domaine du visible, elle va de 0,75 micron pour le rouge à 0,39 micron pour le violet.

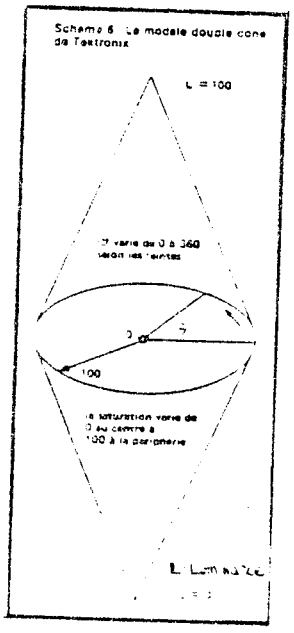
La saturation est le degré de pureté qui permet de distinguer les couleurs "vives" des couleurs "pastels" ; c'est-à-dire à partir de quand la teinte apparaît comme mélangée avec du blanc. Le nombre de couleurs différentiables en fonction de leur teinte et de leur saturation est d'environ 20 000.

La luminance ou luminosité correspond à l'énergie lumineuse perçue. Ainsi des couleurs apparaissent plus brillantes que d'autres à éclairement égal. On distingue environ mille niveaux de luminance.



On peut donc représenter l'espace des couleurs en fonction de ces trois paramètres, teinte, saturation et luminance ; suivant les constructeurs d'écrans graphiques, ce sera à l'aide d'un double cône, d'un cube ou d'un cylindre.





1.1.3. Traitement numérique des couleurs

La représentation spatiale des trois composantes, teinte, saturation et luminance a l'avantage de faciliter le traitement numérique.
Par exemple, l'effet d'ombrage ou de vision crépusculaire d'un objet coloré se traduit par une interpolation linéaire sur le paramètre de luminance (entre sa couleur et le noir) ; des effets de brume, fumées lors de visions lointaines peuvent être une interpolation linéaire de la saturation (entre la couleur initiale et le blanc).

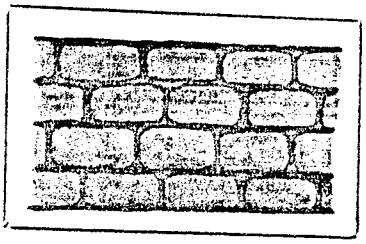
1.2. La modélisation des textures

Maintenant que nous savons présenter des surfaces colorées, il faut ajouter des détails réalistes. En effet les objets ne se présentent pas sous forme de tâches de couleur uniforme. La couleur varie en fonction de la nature de la surface, de sa texture. Ainsi, il est aberrant de représenter une colline boisée par un ensemble de facettes vertes... mais il est impensable de simuler par des facettes différentes tous les arbres se trouvant sur cette colline.

1.2.1. Définition d'une micro-texture [MAR 79]

Une forêt ou un champ labouré vu sur une photo font apparaître que le grain de la surface peut être découpé en un ensemble de micro-textures. Nous allons donc chercher à extraire un ensemble de points comportant des informations de couleur et de luminance d'une partie limitée de la texture. Cet ensemble de points pourrait par exemple être stocké dans une matrice, et l'ensemble des matrices dans une base de données des textures. Il faudrait alors un module de calcul qui permettrait, en pavant la zone à remplir, d'obtenir une texture complète.

micro-texture représentant des briques



texture complète du mur

1.2.2. Modèles obtenus grâce à des images digitalisées

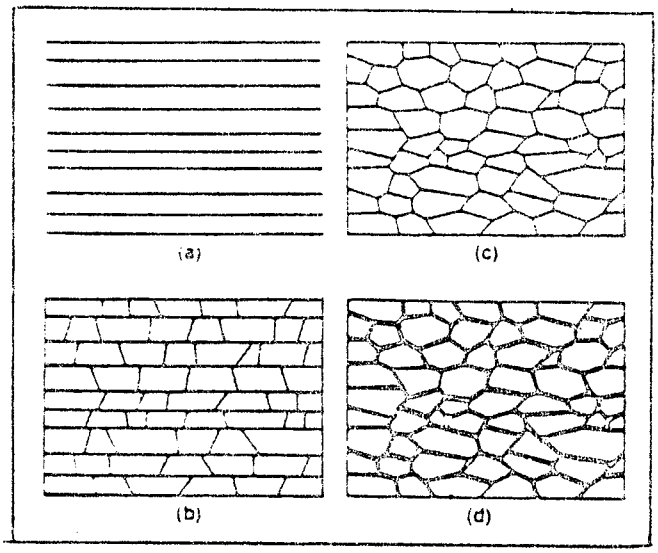
Une caméra numérique peut nous permettre d'obtenir une image digitalisée, soit à l'aide d'une photographie, soit à l'aide d'un dessin de modèle de texture.

L'aspect réaliste de la micro-texture obtenue, dû à des détails très fins est un gros avantage mais le "pavage" de celle-ci engendre souvent des discontinuités.

1.2.3. Modèles calculés [MYE 81]

Un ensemble de courbes sur lesquelles on applique des perturbations peut aussi donner un modèle de texture :

Quatre étapes pour présenter un mur de pierre :



(a) - structure horizontale de lignes parallèles avec espacements verticaux variant suivant un rang prédéfini.

(b) - éléments réguliers : lignes verticales représentant la fin de chaque brique. Puis de façon aléatoire, deux modifications peuvent intervenir :
- changement de la distance horizontale
- changement de l'angle par rapport à l'horizontale.

(c) - puis les points de jonction sont déplacés de haut en bas par des calculs de sommes aléatoires.

(d) - l'espace entre les briques est agrandi et rempli pour simuler des joints au mortier.

On peut aussi essayer de générer des micro-textures telles que la juxtaposition de modèles quelconques ne provoque pas de discontinuités nuisibles. Ceci peut par exemple être obtenu par générations de modèles "cohérents", c'est-à-dire dont les frontières supérieures et inférieures coïncident parfaitement.

1.2.4. Modèles obtenus par perturbation ou transformation d'images

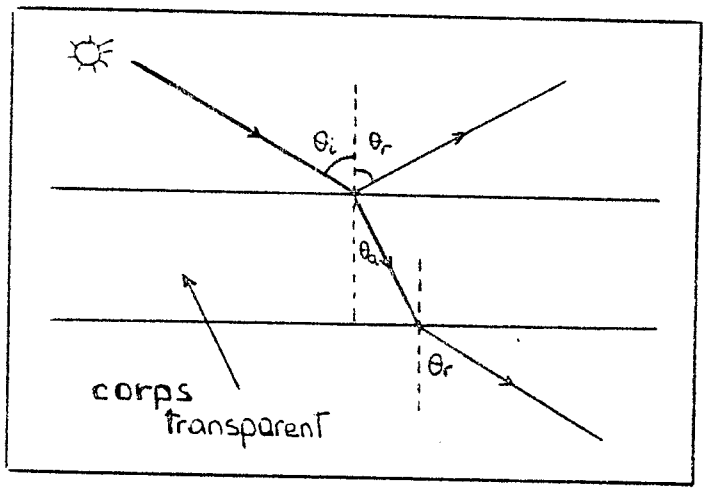
Des modèles de textures peuvent aussi être obtenus par des méthodes de traitement d'images. Ce sont des techniques qui font appel aux transformées de FOURIER, qui calculent des perturbations aléatoires ou qui agissent par "filtrage" ou sommation de régions de l'image initiale.

1.3. La modélisation de la transparence

Certains objets se laissent traverser par la lumière, ce qui permet de distinguer d'autres objets à travers leur épaisseur. Deux phénomènes physiques entrent en jeu quand un rayon lumineux rencontre un corps transparent (voir schéma) :

- celui de la réfraction : le rayon lumineux incident est dévié selon un angle de réfraction qui varie en fonction de l'angle d'incidence et de l'indice de réfraction n du matériau.

- celui de la transmission : le rayon lumineux qui traverse la lame transparente reparaît de l'autre côté, en subissant une nouvelle réfraction.



avec $\sin\theta_a = \frac{\sin\theta_i}{n}$

$n = \frac{a}{\lambda^2} + b$

λ longueur d'onde

Schéma des deux phénomènes définis ci-dessus

Pour modéliser la transparence, il faut lors de la détermination de parties visibles, garder toutes les informations des objets situés dans une partie déterminée.
L'algorithme d'élimination de surfaces cachées de NEWELL, NEWELL & SANCHA [NNS 72] semble bien adapté à la présentation d'objets transparents. En effet à tout moment on dispose de l'aspect actuel de la scène et des paramètres attachés à une face : on peut alors, quand une face transparente est rencontrée, combiner pour chaque point les intensités.

Pour savoir quelle sera la nouvelle intensité I, on combine des fonctions de l'intensité du matériau et des fonctions de l'intensité de ce qu'il obscurcirait.

$$I < I_o \qquad I = W \cdot I_1 + (1 \cdot W) \cdot I_o$$

$$I \geq I_o \qquad I = I_1$$

I_o, représentant l'intensité précédente ; I_1, celle de la face traitée et w, le facteur de réflexion du matériau.

Ces fonctions ne simulent pas tout à fait le monde réel mais une combinaison judicieuse permet de bons effets.

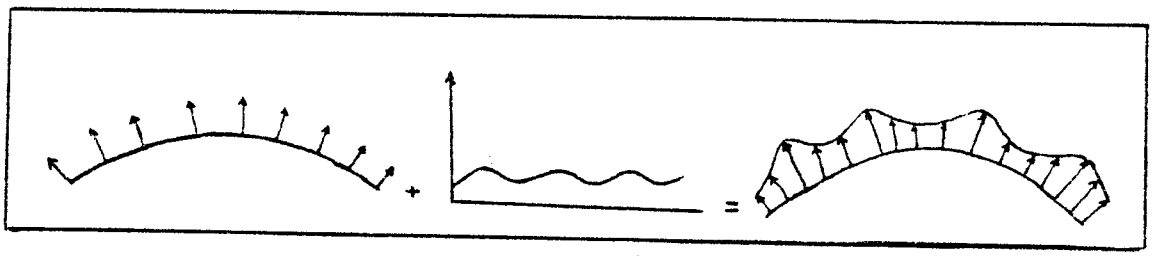
Il semblerait intéressant, pour simuler par exemple des façades d'immeubles vitrées, d'étendre la notion de texture aux textures transparentes où l'on pourrait faire varier transmittance et réfraction.

1.4. La modélisation du relief

Les éléments décrits jusqu'ici confèrent déjà un important degré de réalisme aux images. Mais celles-ci semblent souvent lisses et donc artificielles. On a donc besoin de simuler des surfaces irrégulières telles qu'elles existent.
Le relief est perçu par des variations de luminance provoquées par la perturbation de la normale sur une surface. Pour donner l'illusion de relief et donc présenter ces surfaces irrégulières, il faut moduler la luminance.

La présentation d'une telle technique a été exposée par BLINN [BLI 78] (peu d'articles ont paru, essayant d'évoquer et de résoudre ce problème).

Il utilise une texture, lui applique une petite perturbation avant de faire les calculs d'intensité.



surface lisse + perturbation surface irrégulière

Le calcul du nouveau vecteur normal N' en chaque point de la surface irrégulière est :

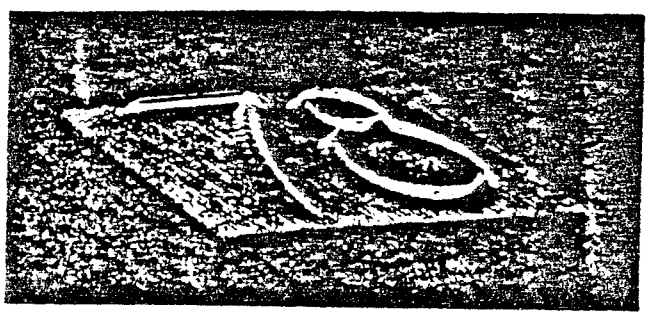
$$\vec{N'} = \vec{N} + \vec{D} \quad \text{avec} \quad \vec{N} \text{ vecteur normal initial}$$

$$\vec{D} = \frac{Fu\ (N.Pv) + Fv\cdot(N.Pu)}{|N|}$$

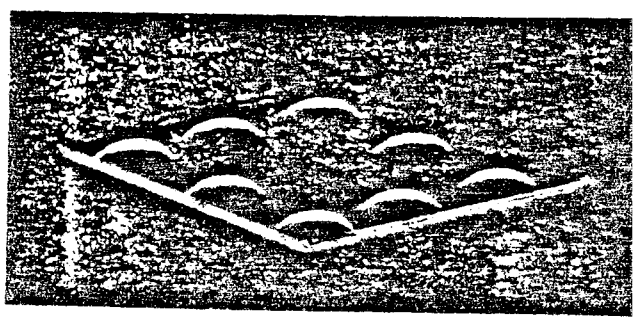
Pour la génération des textures, on emploie, soit des images digitalisées, soit des modèles géométriques simples générés de façon algorithmique, soit des lettres.

De plus, il faut au départ avoir défini une fonction de perturbation simple pour que le calcul soit rapide... Les polygones à n variables ou les transformées de FOURIER sont beaucoup trop complexes et BLINN fait appel à une fonction d'interpolation [BLI 78].

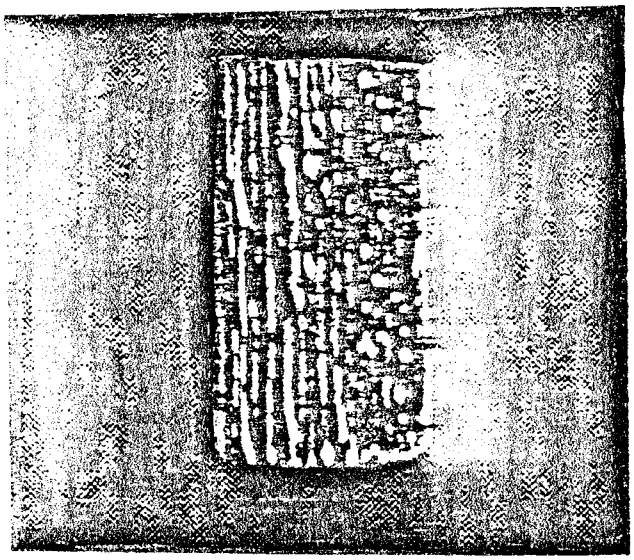
Cette méthode permet d'obtenir des effets d'un réalisme satisfaisant :



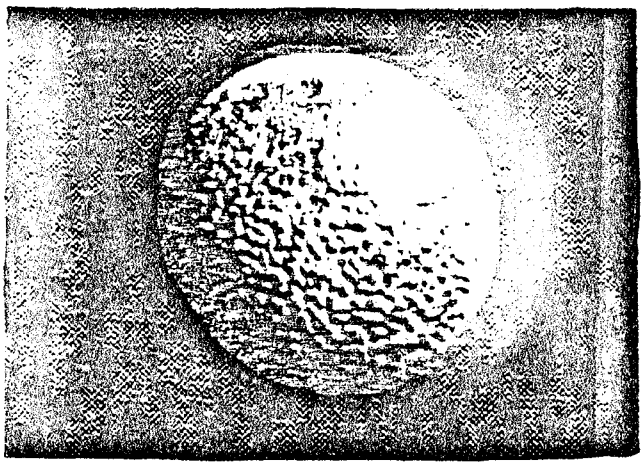
obtenu grâce à des caractères tridimensionnels



obtenu grâce à des sphères



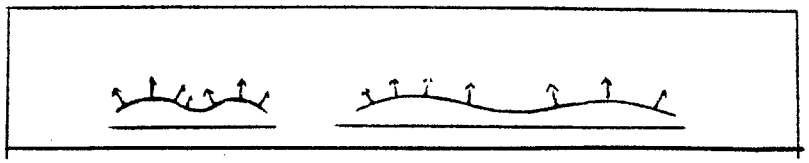
Obtenu par une fonction de perturbation sur un dessin fait à la main



Après passage dans filtres passe-bas pour supprimer l'"ALIASING".

Mais :

1 - la perturbation engendrée n'est pas indépendante de la taille de l'objet. Quand un objet est agrandi, la fonction F de perturbation ne l'est pas, on a alors l'effet suivant :



normal agrandi

Il faut alors non plus considérer la fonction F comme l'altitude du relief, mais comme l'expression d'un angle de rotation appliqué au vecteur normal.

2 - Des "artifices" indésirables peuvent entrer dans l'image. Connu sous le nom d'"aliasing", cet inconvénient peut être supprimé par de simples filtres passe-bas.

3 - Cette méthode ne s'applique qu'aux surfaces gauches.

4 - Le temps de calcul nécessité par cet algorithme est important, environ 2 à 3 fois plus important que pour un objet sans relief.

Les surfaces rugueuses sont modélisables et elles peuvent être calculées en même temps que les reflets et les paramètres d'éclairage. L'illusion provoquée par une surface irrégulière est convaincante et même si le contour du bord de l'objet est régulier, l'impression qui en découle n'est pas trop embarrassante.

II - LA MODELISATION DES PARAMETRES D'ECLAIRAGE

Un paysage nous apparaît différent suivant que nous le regardons avec un soleil éclatant, dans la brume ou le soir. Il va sans dire que chaque scène présentée par ordinateur devra tenir compte de la position de la (ou des) source(s) lumineuse(s), de leur(s) couleur(s), de leur(s) intensité(s), de la lumière ambiante et des ombres et pénombres que provoquent ces sources sur les objets de l'image.

2.1. La position, l'intensité, la chrominance d'une source lumineuse

On peut caractériser une source lumineuse en fonction d'un certain nombre de paramètres.

- la forme et la surface de la source : une source est rarement ponctuelle : le soleil représente un ensemble de points lumineux recouvrant un cercle.
- le flux lumineux qui représente la quantité d'énergie rayonnée pendant une unité de temps.
- l'intensité lumineuse qui mesure dans une direction donnée la quantité de lumière émise.

Pour la majorité des applications, l'approximation d'une source réelle par une source ponctuelle dont l'intensité est identique dans toutes les directions donne une précision suffisante.

- l'éloignement de la source : la plupart du temps on la considère située à l'infini, ce qui simplifie les calculs car les rayons lumineux sont alors parallèles.

- la chrominance de la source qui regroupe les paramètres associés à la couleur de la source.
La modélisation de cette chrominance sera la même que celle d'un point d'un objet : un coefficient sera donné à chacune des composantes : rouge, vert, bleu.

2.2. Les effets de la lumière ambiante

En plus d'une source lumineuse principale, les surfaces reçoivent une somme de petits rayons de lumière. L'intégrale de ces lumières ambiantes venant de toutes les directions, produit une valeur supposée constante pour chaque direction normale.
Il faudra donc tenir compte d'un coefficient mesurant la proportion de réflexion ambiante.

Ainsi BLINN [BLI 77] en tient compte quand il définit l'intensité perçue

$$i = pa + dpd \qquad d = \max(0, \vec{N}.\vec{L})$$

où i est l'intensité perçue, pa la production de réflexion ambiante, d : la somme de réflexion diffusée, pd : la proportion de réflexion diffuse, $\vec{N}$: le vecteur normal à la surface, L : le vecteur de direction de la lumière.

De même B.T. PHONG [PHO 75] calcule l'éclairement d'un point en sommant 3 termes :

$$E = L + F(i) + g(d)$$

où E représente l'éclairement, L : la lumière ambiante, F(i) : la fonction simulant la diffusion du matériau en fonction de l'angle d'incidence, g(d) : la fonction simulant la réflexion en fonction de la normale et de la direction médiane entre la source et l'observateur.

2.3. La réaction des objets à la lumière

Examinons le phénomène de réflexion spéculaire engendré par la réaction des objets à la lumière.
L'ombre Sp à chaque point d'un polygone peut être déterminée par la fonction Sp = Cp cos(i), Cp étant le coefficient de réflexion du matériau constituant le polygone p et i l'angle d'incidence du rayon lumineux.

Cette approximation est pauvre car elle ne fait apparaître :
- ni les propriétés de réverbération du matériau, c'est-à-dire la capacité du matériau de générer des reflets,
- ni la position de l'observateur.

Quelques bons résultats ont cependant été obtenus en réduisant, en plus, les discontinuités de l'ombrage de deux polygones adjacents.

1. NEWELL, NEWELL & SANCHA [NNS 72] ont modelé les aspects de la transparence et des reflets.
Les reflets ne sont pas dus seulement au rayon incident mais aussi à la réflexion de la lumière des autres objets de la scène. Ce modèle permet-tant de générer des reflets est malheureusement limité car on ne peut faire varier l'intensité de la lumière.

2. L'algorithme proposé par B.T. PHONG est fondé sur le fait que pour modeler les propriétés spéculaires des objets, il nous faut un modèle permettant de déterminer la normale à la surface de chaque point [PHO 75].

La fonction d'ombrage dépend des caractéristiques de la source, de l'objet et de la position de l'observateur.

L'ombre $Sp = Cp\ [\cos(i)\ (1-d) + d] + W(i)\ [\cos(s)]^n$

où Cp = coefficient de réflexion de l'objet

i = angle incident

d = coefficient de réflexion de la diffusion du milieu

W(i) = une fonction donnant le pourcentage de lumière miroitante reflétée et de lumière incidente comme une fonction de l'angle incident (varie entre 10% et 80%)

s = angle entre la direction de la lumière reflétée et la position de l'observateur

n = puissance qui modèle la lumière reflétée pour chaque matériau (1 à 10).

Ce modèle est plus rapide que d'autres modèles publiés par WARNOCK, NEWELL ou GOUROUD, car trois interpolations se font en parallèle. Mais s'il y a un fort changement d'orientation de deux facettes adjacentes, il y a toujours l'effet de MACH qui fait apparaître une brillance subjective sur les bords. Celle-ci est toutefois moins visible que dans les modèles précédents.

2.4. Les ombres portées

La qualité d'une image tridimensionnelle est notablement augmentée par la prise en compte du problème d'ombres portées. Des objets dans une scène s'ombrent mutuellement et porter ces ombres sur l'image permet de mieux appréhender la taille ou la position des objets.

Un certain nombre d'algorithmes d'élimination de parties cachées éludent ce problème en mettant la source de lumière au point d'observation ; d'une part, ils simplifient le calcul de la luminance, d'autre part, ils suppriment les ombres.

La technique d'ombres portées dépend, pour une bonne part, de l'algorithme d'élimination de parties cachées, qui lui-même dépend de la modélisation des objets. Aussi les algorithmes d'ombrages sont parfois classés par groupes :

- ceux calculant les ombres portées pour des objets à faces planes,
- ceux calculant les ombres portées pour des objets à surfaces gauches.

Algorithme de BOOKNIGHT & KELLEY

BOOKNIGHT & KELLEY [BOK 70] utilisent un algorithme d'élimination de parties cachées s'appuyant sur un balayage horizontal de l'image.
Il effectue un balayage original qui fait passer de la structure en trois dimensions, à la structure de l'image finale.
Un second balayage fait de la même manière passer de la structure tridimensionnelle des ombres à une information qui sera combinée à la précédente pour donner l'intensité à chaque point de la ligne de balayage, ceci uniquement pour les polygones visibles et ombrés.

Cet algorithme est notoirement accéléré par le fait que le nombre d'ombres à projeter a été réduit au minimum ; en effet on ne considère que les paires de polygones pouvant se masquer mutuellement la lumière. Cette détection est effectuée en projetant tous les polygones sur une sphère, dont le centre est la source lumineuse et en comparant les coordonnées des points ainsi transformés, ce qui permet de vérifier leur recouvrement éventuel. Un certain nombre d'autres contrôles allègent les calculs.

ANNEXE 2

COPROCESSEURS GRAPHIQUES D'EFCIS
EF 9365/9366

Coprocesseurs graphiques EF 9365/9366 pour terminaux : une solution économique

Ces coprocesseurs, présentés par Efcis à Electronica (voir « minis et micros », numéro 131, page 23), ont retenu l'attention des spécialistes pour leurs caractéristiques de vitesse et de simplicité d'implantation dans la conception des terminaux graphiques. Qu'apportent-ils à l'utilisateur par rapport aux solutions classiques, du type microprocesseurs en tranches ?

L'introduction récente par Efcis des coprocesseurs graphiques EF 9365 et EF 9366 marque une étape importante qui devrait permettre de réaliser maintenant des terminaux graphiques élaborés pour un coût ne dépassant que de peu celui des terminaux alphanumériques classiques.

Quatre générations de contrôleurs d'écrans

L'utilisation généralisée de l'écran cathodique comme interface entre l'homme et la machine a stimulé le développement de contrôleurs d'écrans de plus en plus performants.

Le **contrôleur alphanumérique** a tout d'abord permis la visualisation de caractères sur un écran. A ce niveau, l'unité de travail est une matrice de points (5 x 7 ou plus) avec un certain nombre de configurations permises à l'intérieur de ce bloc. Cette première famille a vu des circuits permettant soit de visualiser des caractères situés dans une mémoire gérée par un processeur (mémoire partagée), soit de reporter la gestion de la mémoire d'écran au niveau du contrôleur alphanumérique (c'est le cas du processeur alphanumérique EF 9364 dont un exemple d'application est donné en **figure 1**).

Le **controleur semi-graphique** est directement dérivé de la famille précédente. Il suffit de définir à l'intérieur de la matrice, certaines combinaisons de points, pour obtenir des traits prédéfinis permettant alors d'approcher les tracés de courbes. Un circuit tel que le contrôleur EF 6845 offre cette possibilité et de nombreux ordinateurs individuels utilisent ce principe. Les figures ainsi tracées sont grossières et insuffisamment précises pour beaucoup d'applications.

Le **contrôleur d'écran graphique** ne travaille plus au niveau d'une matrice, mais au niveau du point. On peut éteindre et allumer séparément chaque point de l'écran. La mémoire d'écran est gérée par le microprocesseur central ainsi que les différents calculs de points (**fig. 2**). Le contrôleur se contente de visualiser le contenu de la mémoire d'écran.

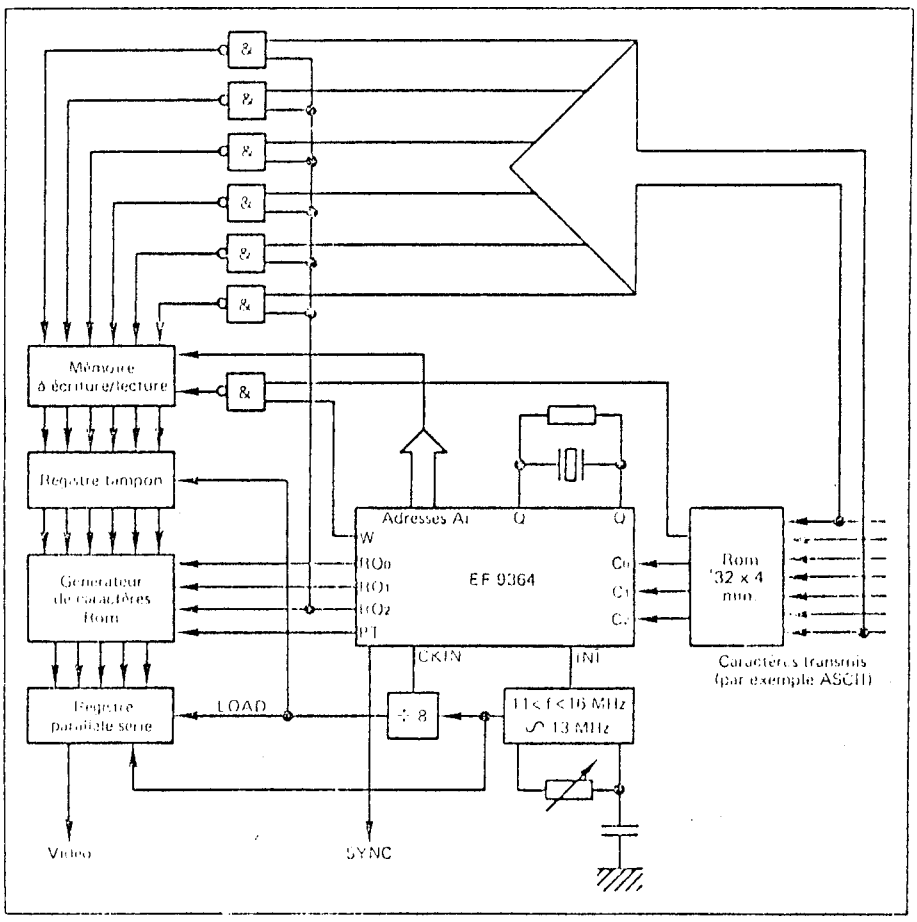
Il est évident que cette méthode possède de gros inconvénients. La mémoire d'image occupe une bonne partie de l'espace d'adressage du microprocesseur, réduisant ainsi sa mémoire de programme (sauf utilisation de techniques de pagination). Le calcul des points par logiciel sur un microprocesseur est lent et limite les possibilités d'animation.

On démontre (1) ainsi qu'un tracé de vecteur exécuté à l'aide de l'algorithme de Bresenham (2) (voir encadré) par n'importe quel microprocesseur 8 bits ne peut se faire qu'à raison de un point toutes les cent micro- ▶

(1) « A high performance integrated true graphic processor », Ph. Matherat, N. Forget, J. Lebrun, J.-P. Moreau (Efcis). — Proceedings Esscirc 80 (Grenoble), p. 271.

(2) « Algorithm for computer control of a digital plotter », IBM system Journal, n° 4 (1965), p. 25 à 30.

Fig. 1 - Schéma général d'une application utilisant le processeur alphanumérique EF 9364



AU CŒUR DES PROCESSEURS GRAPHIQUES

Les EF 9365 et 66 sont architecturés suivant le schéma-bloc ci-dessous. On remarque qu'ils intègrent onze registres dont les fonctions principales sont les suivantes :

— **CMD** : registre de commande 8 bits qui déclenche différentes actions à chaque écriture (signal R/W = 0) telles que traçage des vecteurs ou des symboles, effacement, séquences photostyle, accès mémoire externe, écriture dans les registres internes ;

— **STATUS** : registre d'état 8 bits, actif lorsque R/W = 1, qui doit être testé, avant d'envoyer une nouvelle instruction pour s'assurer que celle en cours est bien exécutée ;

— **CTRL 1 et CTRL 2** : registres de commande 7 et 4 bits utilisés pour contrôler l'activité du processeur (inhibition de la mémoire d'affichage, modes écriture affichage ou effacement, masque d'interruption, positions de caractère, ligne continue, pointillée ou à traits interrompus, etc.).

— **CSIZE** : registre 8 bits donnant le facteur d'échelle des symboles et caractères ;

— **DELTAX et DELTAY** : registres 8 bits donnant la projection sur les axes X et Y du vecteur à tracer ;

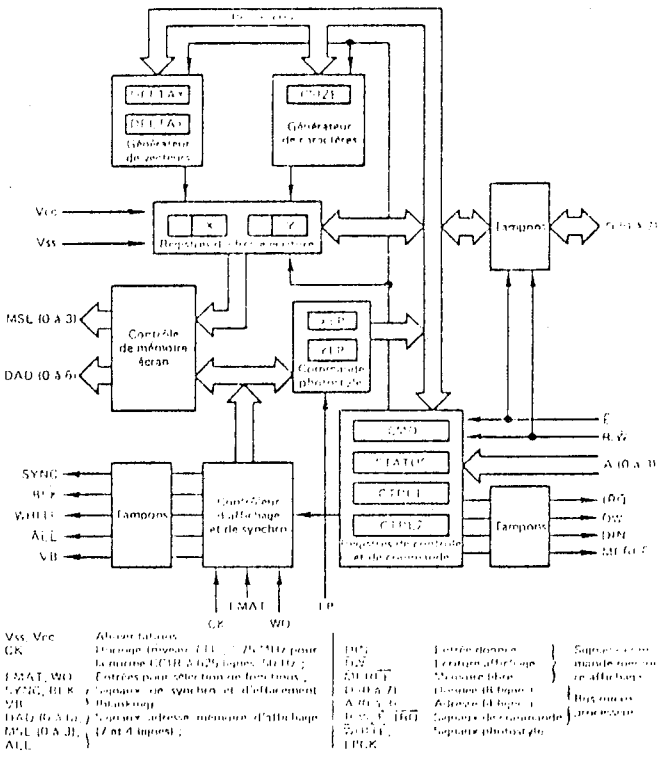
— **X et Y** : registres 12 bits indiquant la position du point à écrire dans la mémoire d'affichage. Ils définissent un espace de travail de 4096 × 4096, mais seul l'espace d'adressage 512 × 512 est mémorisé de sorte que seuls les 9 LSB de chaque registre sont utilisés habituellement ; cependant, on peut écrire directement la position du point dans ces registres, qui normalement sont incrémentés ou décrémentés par les générateurs internes de vecteur et de symbole avant chaque écriture dans la mémoire d'affichage ;

— **XLP et YLP** : registres 7 + 1 bits à lecture seule contenant l'adresse du photostyle.

Le générateur de vecteurs dans lequel est implémenté, en logique câblée, l'algorithme de Bresenham, a été conçu pour minimiser le nombre de cycles d'horloge nécessaire pour tracer un vecteur. Théoriquement, la vitesse du tracé pourrait atteindre 560 ns/pixel, mais comme l'accès à la mémoire n'est pas autorisé durant le temps d'affichage, la vitesse est de 1,36 µs/pixel, conduisant à l'écriture d'une diagonale sur tout l'écran en 700 µs.

Le générateur de symboles peut générer 96 types de symboles sous forme de matrice 5 × 8 à partir d'une Rom interne (code Ascii). Les symboles (ou caractères) peuvent être agrandis en X et/ou en Y jusqu'à seize fois et positionnés dans toutes les directions de la verticale à l'horizontal.

Les différences entre les deux modèles portent essentiellement sur la capacité de résolution :
EF 9365 : 512 × 512 (en balayage entrelacé) ; 256 × 256, 128 × 128 ou 64 × 64 (en balayage non entrelacé) ;
EF 9366 : 256 × 512 (en balayage non entrelacé).



secondes (10 000 points par seconde).

Les coprocesseurs graphiques : deux solutions envisageables

Pour résoudre ce dernier modèle, deux solutions sont envisageables. La première consiste à utiliser des calculateurs rapides, en général des microprocesseurs « en tranche ». C'est la solution actuellement la plus répandue sur le marché du graphique professionnel. Elle permet d'améliorer dans une échelle importante les performances de la famille précédente. Son inconvénient essentiel est le prix, ce qui limite cette solution à des matériels hauts de gamme professionnels.

La deuxième voie consiste à tirer profit de l'état d'avancement technologique actuel pour intégrer une couche de logiciel de base, dans le maté-

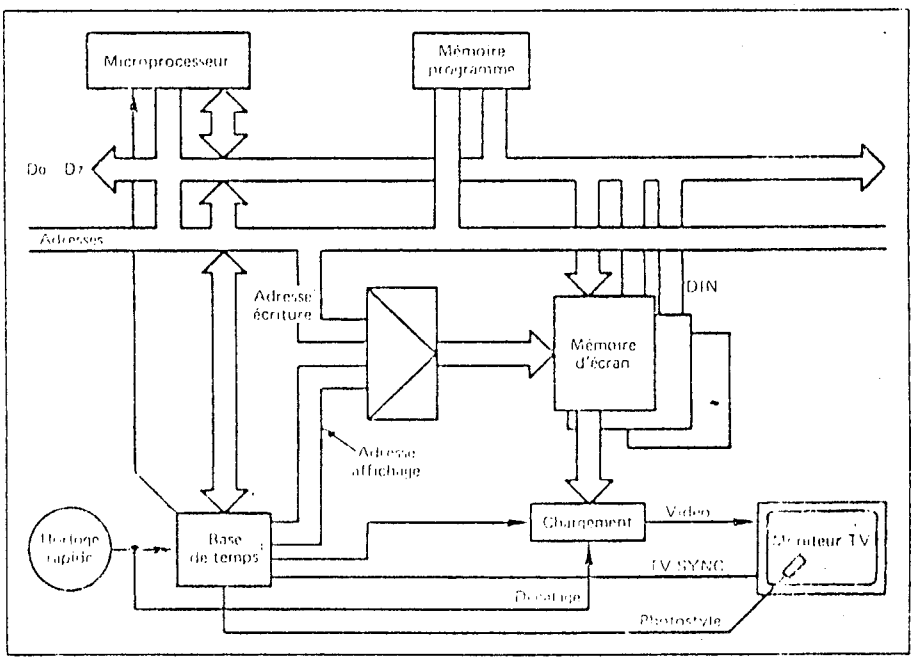


Fig. 2 - Architecture d'un système classique d'affichage graphique.

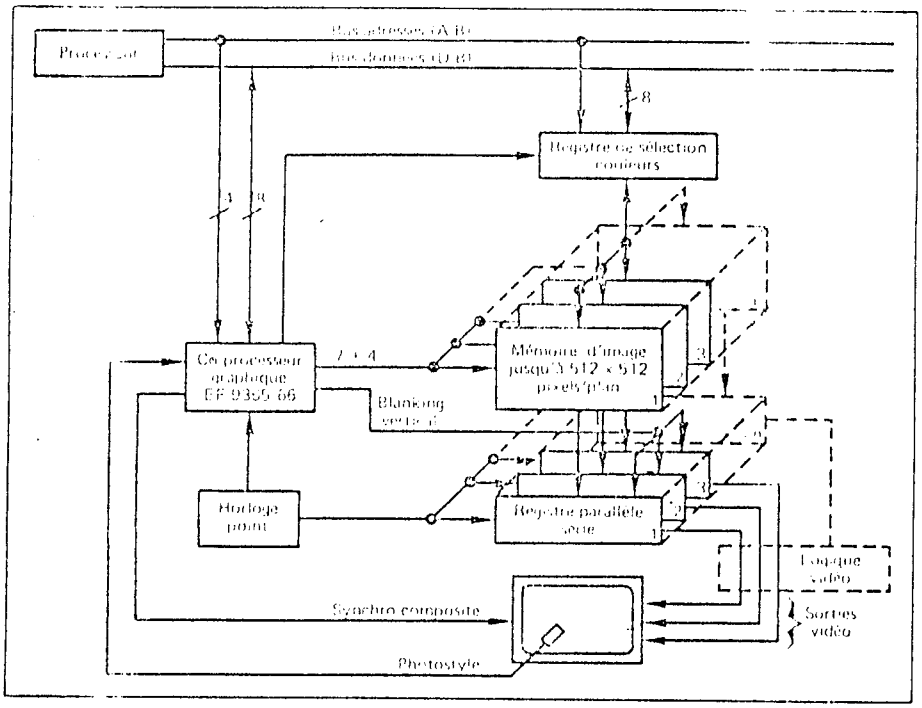


Fig. 3 - L'utilisation du processeur graphique n'est pas plus compliquée que celle du 9364 suivant le nombre de plan-mémoire requis

L'ALGORITHME DE BRESENHAM

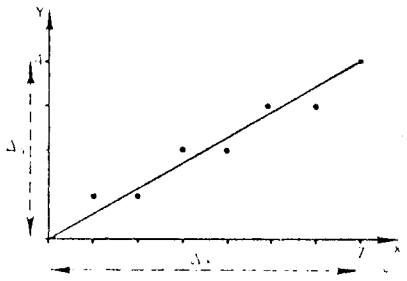
Cet algorithme, décrit en 1965 par Bresenham, est conçu pour optimiser le tracé d'un vecteur point par point compte tenu des quantifications nécessaires. Voici résumé son fonctionnement.

Soit un segment à tracer défini par son origine Xo, Yo et ses projections sur les axes x et y, ΔX et ΔY.

Pour simplifier supposons Xo = Yo = 0 et $\Delta X \geqslant \Delta Y \geqslant 0$

```
begin  X = 0 ;        Y = 0 ;  S = -ΔX/2
       While                X < ΔX        do
              begin           black (X, Y)
                              X = X + 1
                              S : S + ΔY
                              if S ⩾ 0 then
                                    begin  Y = Y + 1
                                           S = S - ΔX
                                    end
              end
end
```

Black (X, Y) est la fonction qui écrit sur l'écran le point de coordonnées X et Y.
Exemple : Xo = Yo = 0, $\Delta X = 7$, $\Delta Y = 4$



Point initial :	X = 0,	S = − 3,5,	Y = 0,	
boucle 1 :	X = 1,	S = + 0,5,	Y = 1,	S = − 6,5
boucle 2 :	X = 2,	S = − 2,5,	Y = 1,	
boucle 3 :	X = 3,	S = 1,5,	Y = 2,	S = − 5,5
boucle 4 :	X = 4,	S = − 1,5,	Y = 2	
boucle 5 :	X = 5,	S = + 2,5,	Y = 3,	S = − 4,5
boucle 6 :	X = 6,	S = − 0,5,	Y = 3	
boucle 7	X = 7,	S = + 3,5,	Y = 4,	S = − 3,5

riel. Pour améliorer la vitesse des calculs, certains algorithmes répétitifs sont câblés à l'intérieur du processeur graphique.

Issus de cette réflexion, les coprocesseurs graphiques EF 9365 et EF 9366 intègrent sur 30 mm² de silicium, les 8 000 transistors nécessaires aux fonctions suivantes :

— gestion de la mémoire d'écran jusqu'à 512 × 512 points (rafraîchissement des Ram dynamiques, écriture et lecture, visualisation de la mémoire d'écran) ;

— génération des signaux de synchronisation d'écran ;

— générateur de caractères programmables en taille et en orientation ;

— générateur de vecteurs capable de générer jusqu'à 1 500 000 points par seconde ;

— interface direct avec n'importe quel microprocesseur 8 bits ;

— commande par instructions ou par adresses (registres chargeables) ;

— utilisation possible d'un photostyle (light-pen).

Avantages et performances de cette solution

La solution proposée par les coprocesseurs graphiques EF 9365 et EF 9366, dont un exemple d'application est donné en figure 3, s'inscrit bien dans la tendance actuelle de décharger le calculateur (répartition des fonctions) et de câbler les fonctions logicielles les plus répandues dans « le silicium » du périphérique.

La densité d'intégration élevée des technologies Mos actuelles (N-Mos, H-Mos 1) permet de réaliser certaines fonctions à un coût très compétitif face aux solutions du type microprocesseurs en tranches.

Les processeurs graphiques EF 9365 et EF 9366 permettent d'améliorer dans une proportion très importante le rapport coût/performance d'un grand nombre d'applications :

— en réduisant le coût des applications haut de gamme, professionnelles, dont il possède les performances ;

— en offrant des performances inégalées (jusqu'à 1 500 000 points/seconde) dans des applications à coût raisonnable.

Les marchés concernés sont nombreux. Le graphique professionnel, le contrôle de processus, les terminaux informatiques, les jeux en sont quelques exemples et il y a fort à parier que ce nouveau composant suscitera de nouveaux types de produits.

Christian Roy
Phillppe Lambinet

Coprocesseurs graphiques EF 9365/9366 : les modalités d'exploitation

Récemment introduits par Efcis, les co-processeurs EF 9365 et EF 9366, de par leur structure et leur principe de fonctionnement, devraient permettre l'avènement d'une nouvelle génération de terminaux graphiques performants et néanmoins économiques axés autour de microprocesseurs 8 bits. Dans notre numéro 136, nous avons montré leur intérêt par rapport aux techniques classiques ; aujourd'hui, nous examinerons les modalités d'exploitation

La difficulté de réaliser une réelle animation à partir des contrôleurs graphiques classiques, est liée au nombre important de points à recalculer par trame. Si on laisse au processeur central le soin d'effectuer seul ce calcul, un maximum d'environ 10 000 points par seconde peut être généré par un microprocesseur 8 bits courant. C'est une vitesse insuffisante pour faire de la réelle animation.

Une autre solution consiste à décentraliser, au niveau du contrôleur graphique, certains traitements de base. C'est la solution retenue pour les coprocesseurs EF 9365 et 9366 qui exécutent « en câblé » des routines courantes : génération de caractères bien sûr, mais aussi génération automatique de vecteurs. Le processeur central se contente alors de donner à son périphérique des directives de « haut niveau » : « tracer vecteur » ou « tracer caractère », par exemple.

Génération des caractères et des vecteurs

Comme nous l'avons vu, le 9365 assure une résolution de 512 par 512 avec balayage entrelacé, ou 256 × 256, 128 × 128, 64 × 64 avec balayage non entrelacé. Le 9366 autorise, lui, une résolution de 256 × 512 en balayage non entrelacé. Dans ce qui suit, les différentes possibilités du 9365 vont être étudiées sous l'angle de la programmation avec comme hypothèse une résolution de 512 × 512.

Tout dessin est un ensemble de vecteurs, de courbes, de points et bien entendu de lettres. Aussi, nous étudierons tout d'abord l'écriture des caractères puis la génération des vecteurs.

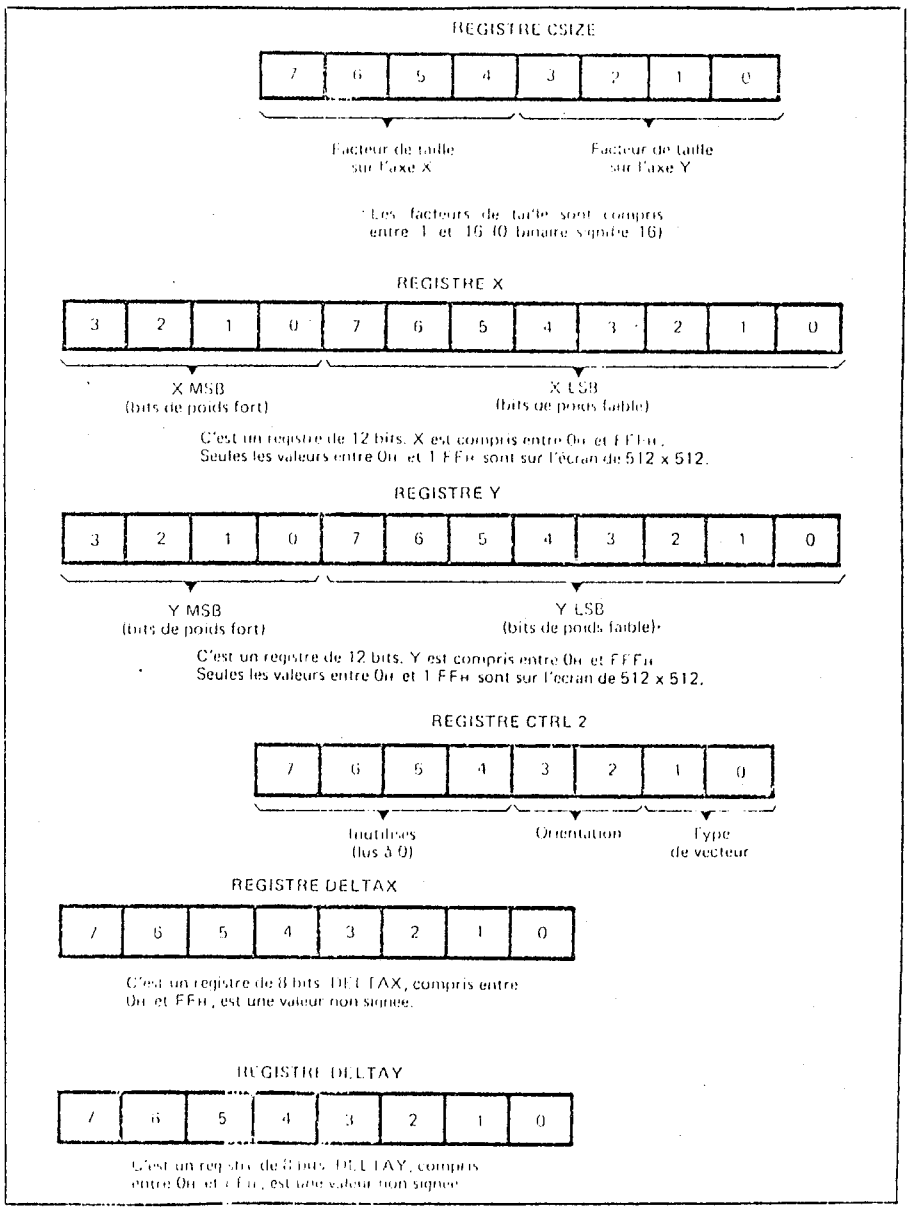
Le 9365 se comporte vis-à-vis du microprocesseur central comme onze registres disposés dans l'espace d'adressage et permettant de programmer le fonctionnement du co-processeur.

Ecriture des caractères

Un caractère se définit à l'aide de quatre paramètres : sa taille (registre CSIZE) ; son orientation (registre CTRL 2) ; sa place sur l'écran (registres X et Y) ; son code Ascii (registre CMD).

Pour écrire un caractère sur l'écran, il suffit de programmer correctement les registres CSIZE, X, Y et CTRL 2 (voir **fig. 1**) et d'écrire le code ▶

Fig. 1 - Constitution des différents registres pour l'écriture des caractères (taille, orientation et emplacement) et des vecteurs (registres DELTAX et DELTAY donnant leur projection)



▶ Ascii du caractère souhaité dans le registre de commande CMD. Le caractère apparaît alors sur l'écran (en supposant que le montage soit correct !). Le 9365 met lui-même à jour les registres X et Y pour pouvoir immédiatement écrire un nouveau caractère sans recalculer sa position sur l'écran.

Seul le premier caractère d'une ligne doit être programmé en taille, orientation et position. L'exemple de la figure 2 montre l'effet de CTRL 2

b1 b0	Orientation du caractère	b1 b0	Type de vecteur
0 0		0 0	Trait continu
0 1		0 1	Pointillé
1 0		1 0	Trait interrompu
1 1		1 1	Mixte

Fig. 2 : Le registre CMD permet d'orienter le caractère ou de choisir le type de vecteur à tracer

dont les bits 3 et 2 définissent l'orientation du caractère (les bits 1 et 0 définissent le type de vecteur à tracer, dans le cas d'écriture de vecteur).

Le registre CMD ne doit être chargé que si l'exécution de la commande en cours est terminée. Le registre d'état (status) permet de tester l'état du processeur (voir fig. 3).

Tracé des vecteurs

Un vecteur se définit à l'aide de quatre paramètres : sa taille (projection sur les axes X et Y) ; son origine sur l'écran ; son type de tracé (continu, pointillé, tireté, mixte) ; son code. Ces paramètres sont fixés en programmant les registres DELTAX et DELTAY pour la projection, CTRL 2 pour le tracé, et X, Y pour l'origine. Un vecteur peut être codé dans le registre de commande (CMD) de trois façons différentes :

— **Codage des vecteurs longs :** le code a alors le format :

0	0	0	1	0	b_2	b_1	b_0

avec b_2 b_1 b_0 prenant les valeurs 000 à 111 en binaire (fig. 4 a). Ces trois bits de poids faible déterminent la direction du vecteur (codes compris entre 10 et 17 hexa). La direction est prioritaire sur Δ X et Δ Y en cas de conflit.

— **Codage des vecteurs de direction « spéciale » :** un vecteur de direction spéciale est un vecteur incliné à 45°. Le code a alors le format :

0	0	0	1	1	b_2	b_1	b_0

avec b_0 b_1 b_2 prenant les valeurs binaires 001, 011, 101 ou 111 (fig. 4 b). Ces trois bits de poids faible déterminent la direction du vecteur. La longueur du vecteur est déterminée par la plus grande des valeurs (DELTAX et DELTAY) qui devient la projection sur les deux axes X et Y.

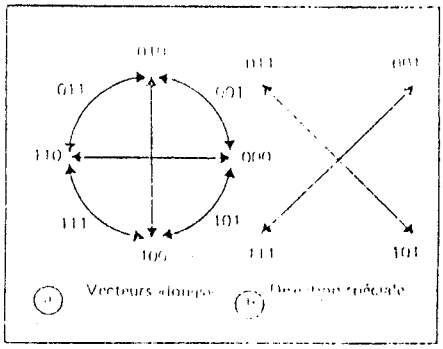
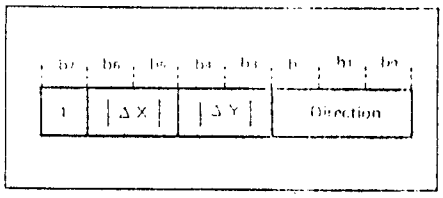


Fig. 4 : Codage des vecteurs « longs » et « direction spéciale » dans le registre commande

— **Codage des vecteurs courts :** le code a le format :



La direction est déterminée par b_2, b_1, b_0 de la même manière que la direction des vecteurs longs. La longueur est déterminée par les projections sur les axes X et Y définies par Δ X et Δ Y, suivant les valeurs indiquées dans le tableau ci-dessous :

Δ Y		Δ X		Longueur (sur l'axe X)
b_4	b_3	b_6	b_5	
0	0	0	0	0 pas
0	1	0	1	1 pas
1	0	1	0	2 pas
1	1	1	1	3 pas

On entend par « pas », dans ce tableau, l'incrémentation que subit X ou

Fig. 3 : Soustraction des bits des registres d'état et CTRL 1

Registre d'état (lecture seulement)

0	Haut	séquence du photostyle terminée
1	Haut	retour de trame en cours
2	Haut	prêt à recevoir une nouvelle commande
		bits ci-verrouillés ni masqués
3	Haut =	Plume hors de l'écran
4	Haut	indicateur d'IRQ (*) par fin de séquence photostyle
5	Haut =	indicateur d'IRQ par retour de trame
6	Haut	indicateur d'IRQ par circuit libre pour une nouvelle commande
		bits repositionnés après cycle de lecture du registre d'état
7	Haut =	indicateur d'IRQ

Registre de contrôle 1 (lecture/écriture)

0	Haut = crayon (ou effaceur) baissé ; bas = crayon levé
1	Haut = crayon sélectionné ; bas = effaceur sélectionné
2	Haut = écriture à haute vitesse (pas de visualisation)
3	Haut = écran cyclique
4	Haut = autorisation d'IRQ par fin de séquence photostyle
5	Haut = autorisation d'IRQ par retour de trame
6	Haut = autorisation d'IRQ par circuit libre pour une nouvelle commande
7	Non utilisé (lu à 0)

(*) IRQ = demande d'interruption

b_7 b_6 b_5 b_4 / b_3 b_2 b_1 b_0		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0 0 0 0	0	Met à 1 le bit 1 de CTRL1 : « Choix de la plume qui écrit ».	Vecteurs standards	Espace	0	@	P	`	p								
0 0 0 1	1	Met à 0 le bit 1 de CTRL1 : « Choix de la plume qui efface ».		!	1	A	Q	a	q								
0 0 1 0	2	Met à 1 le bit 0 de CTRL1 : « Passage en plume baissée ».		"	2	B	R	b	r								
0 0 1 1	3	Met à 0 le bit 0 de CTRL1 : « Passage en plume levée ».		#	3	C	S	c	s								
0 1 0 0	4	Effacement de l'écran.		$	4	D	T	d	t								
0 1 0 1	5	Remise à 0 des registres X et Y.		%	5	E	U	e	u								
0 1 1 0	6	Effac. de l'écran et remise à 0 reg. X et Y		&	6	F	V	f	v								
0 1 1 1	7	Effacement de l'écran, positionnement à S11 du registre CSIZE, remise à 0 des autres registres (sauf XLP, YLP).		'	7	G	W	g	w								
1 0 0 0	8	Initialisation du photostyle (forçage sortie WHITE niveau bas).	Vecteurs de directions privilégiées	(	8	H	X	h	x								
1 0 0 1	9	Initialisation du photostyle		)	9	I	Y	i	y								
1 0 1 0	A	Lancement du tracé du pavé 5 × 8		*	:	J	Z	j	z								
1 0 1 1	B	Lancement du pavé 4 × 4		+	;	K	[	k	{								
1 1 0 0	C	Effacement de l'écran avec la plume indiquée dans CTRL1.		,	<	L	\	l	¦								
1 1 0 1	D	Remise à 0 du registre X.		—	=	M	]	m	}								
1 1 1 0	E	Remise à 0 du registre Y.		.	>	N	↑	n	~								
1 1 1 1	F	Demande d'accès direct à la mémoire d'image pour le premier cycle libre.		/	?	O	←	o									

PETITS VECTEURS :

b_7	b_6 b_5	b_4 b_3	b_2 b_1 b_0
1	ΔX	ΔY	Direction

Dimension

ΔX or ΔY		Long. du vecteur
0	0	0 pas
0	1	1 pas
1	0	2 pas
1	1	3 pas

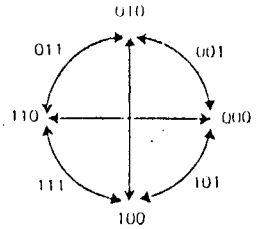
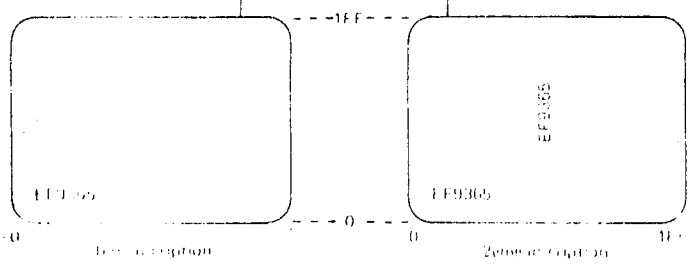


Tableau I
Résumé des codes de commande

```
00102                   *
00103                   *
00104                   *       EXEMPLE
00105                   *
00106                   *  code SOURCE EF96800 POUR ECRIRE
00107                   *
00108                   *  "EF9365" SUR L ECRAN
00109                   *
00110                   *  VERTICALEMENT ET HORIZONTALEMENT
00111                   *
00112                   **
00113 0000 86 33    A EXEMPL LDAA   #$33     TAILLE CARACTERES
00114P 0002 B7 0000 A        STAA   CSIZE    TAILLEX=TAILLEY=3
00115P 0005 CE 0020 A        LDX    #$0020   ORIGINE SUR L ECRAN
00116P 0008 FF 0000 A        STX    XMSB
00117P 000A FF 0000 A        STX    YMSB     X=Y=$0020
00118P 000D 7F 0000 A        CLR    CTRL2    CARACTERES HORIZONTAUX
00119P 0010 BD 0085 P        JSR    ECRIT    AFFICHAGE DE "EF9365"
00120P 0013 CE 0100 A        LDX    #$0100   NOUVELLE ORIGINE
00121P 0016 FF 0000 A        STX    XMSB
00122P 0019 FF 0000 A        STX    YMSB     X Y=$0100
00123P 001C 86 00   A        LDAA   #$00     NOUVELLE ORIENTATION
00124P 001E B7 0000 A        STAA   CTRL2    CARACTERES VERTICAUX
00125P 0021 BD 0085 P        JSR    ECRIT    AFFICHAGE DE "EF9365"
00126P 0024 39               RTS
```




```
00127                   **
00128P 0025 86 45   A ECRIT LDAA   #$45     AFFICHE "E"
00129P 0027 B7 0000 A        STAA   CMD
00130P 002A BD 0086 P        JSR    PRET     ATTENTE
00131P 002D 86 46   A        LDAA   #$46     AFFICHE "F"
00132P 002F B7 0000 A        STAA   CMD
00133P 0032 BD 0086 P        JSR    PRET     ATTENTE
00134P 0035 86 39   A        LDAA   #$39     AFFICHE "9"
00135P 0037 B7 0000 A        STAA   CMD
00136P 003A BD 0086 P        JSR    PRET     ATTENTE
00137P 003D 86 33   A        LDAA   #$33     AFFICHE "3"
00138P 003F B7 0000 A        STAA   CMD
00139P 0042 BD 0086 P        JSR    PRET     ATTENTE
00140P 0045 86 36   A        LDAA   #$36     AFFICHE "6"
00141P 0047 B7 0000 A        STAA   CMD
00142P 004A BD 0086 P        JSR    PRET     ATTENTE
00143P 004D 86 35   A        LDAA   #$35     AFFICHE "5"
00144P 004F B7 0000 A        STAA   CMD
00145P 0052 BD 0086 P        JSR    PRET     ATTENTE
00146P 0055 39               RTS             RETOUR A L ENVOYEUR
00147                   **
00148P 0056 B6 0000 A PRET   LDAA   STATPS    LECTURE REGISTRE D ETAT
00149P 0059 85 04   A        BITA   #$04      BOUCLE SI OCCUPE
00150P 005B 27 F9 0056       BEQ    PRET     EXIT SI PRET
00151P 005D 39               RTS
00152                   **
```

Ce programme permet d'écrire, en deux temps, EF 9365 horizontalement en bas et à gauche de l'écran, puis verticalement au milieu de l'écran, la première inscription restant apparente. La première partie du programme (lignes 113 à 126) spécifie les caractéristiques des caractères (taille, orientation et origine) et la deuxième partie ci-dessus donne les ordres d'affichage

(ou Y). Par exemple le code 10000 xxx indique 0 pas sur X et 0 pas sur Y ; cela signifie que l'on écrit 1 point et que l'on reste à l'endroit du point (X et Y ne changent pas).

Pour ces vecteurs, il n'est pas utile de programmer les registres DELTAX et DELTAY.

Une fois le vecteur tracé, les registres du 9365 sont inchangés, excepté les registres X et Y qui ont été incrémentés (ou décrémentés suivant la direction du vecteur) respectivement de DELTAX et DELTAY (Δ X et Δ Y pour les petits vecteurs).

Autres utilisations du registre de commande

Le registre de commande CMD est non seulement utilisé pour déclencher l'écriture de vecteurs ou de symboles, mais aussi pour initialiser les modes de fonctionnement particulier des 9365/66. Le **tableau I** de la page précédente résume tous les différents cas d'utilisation du registre.

Le dessin et l'animation avec le 9365

Nous ne prétendons pas ici aborder tous les problèmes posés par l'écriture de programmes d'animation utilisant les possibilités du EF 9365. Nous nous contentons seulement de donner quelques conseils aux utilisateurs non familiarisés avec le « graphique ».

Comment créer un dessin fixe ?

Il existe deux solutions : la solution manuelle, la solution assistée (par ordinateur bien sûr).

Pour créer un dessin sur l'écran, il est possible de le dessiner d'abord sur papier, en tenant compte de la définition de 512 points par 512 points, de le décomposer en vecteurs, puis de le traduire en instructions EF 9365. C'est la solution manuelle...

La solution « professionnelle » consiste bien sûr à utiliser un éditeur graphique. Ce programme est un outil permettant de définir des figures complexes en termes de primitives graphiques de haut niveau. Une figure peut en effet être définie à très haut niveau par des cercles, courbes, rectangles, triangles... paramétrés. L'éditeur graphique interprète cette description « source » et la transforme en une description « objet » exécutable par le 9365.

Dans la page précédente, nous donnons un exemple de programme source pour écrire l'EF 9365 sur l'écran horizontalement et verticalement.

Comment animer une image ?

Pour animer un dessin, il suffit de l'effacer totalement ou partiellement et de le réécrire avec de légères différences puis de répéter ce cycle d'effacement-écriture jusqu'à la fin du mouvement.

Précautions à prendre

Puisque l'on travaille sur des écrans à balayage de trame, il convient de prendre quelques précautions.

D'abord, il est nécessaire d'écrire et d'effacer très rapidement, pour donner une impression d'animation avec un grand nombre de points en mouvement.

Il faut s'assurer que le dessin est présent en mémoire au moment où on visualise la mémoire d'image. Pour cela, on pourra, par exemple, se synchroniser, comme le montre la **figure 5**, sur le retour trame et éviter

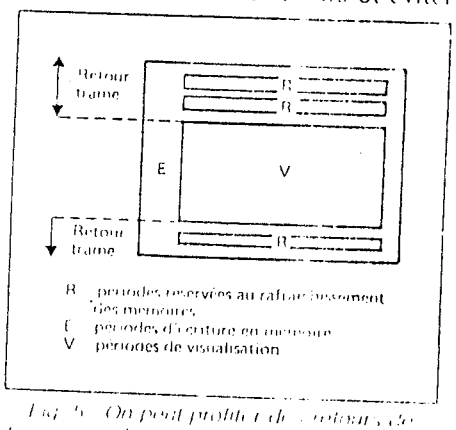


Fig. 5 : On peut profiter du retour trame pour le rafraîchissement des mémoires et du temps de visualisation pour l'écriture des données.

d'effacer le dessin pendant la période de visualisation (utilisation du bit 1 du registre d'état).

D'autres méthodes peuvent consister à travailler sur un plan pendant que l'on visualise un autre plan (le passage d'un plan à l'autre se faisant grâce au signal BLK).

Des figures vraiment animées

Durant une période de 20 ms, plus de 10 ms sont disponibles pour les cycles d'écriture. Puisque les automates internes peuvent écrire un point toutes les 571 ns, on voit qu'en une période, il est possible d'écrire plus de 17 000 points (plus de 6 000 rien que pendant le retour trame). En une seconde, on aura pu ainsi modifier plus de 900 000 points.

Ce chiffre seul, permet d'imaginer les possibilités du EF 9365.

Philippe Lambinet
et Christian Rey

ANNEXE 3

IMPLÉMENTATION D'ÉLÉMENTS CONSTITUÉS DE FACES 3D SUPERPOSÉES

. Les types d'éléments que nous aurions aimé avoir en entrée étaient constitués par des faces planes 3D, superposées ou non. Pourquoi ? La plupart des objets peuvent être modélisés comme un ensemble de faces planes 3D. La communication d'une scène, en vue de l'affichage, provient souvent d'un algorithme d'élimination de parties cachées. Si notre logiciel admet des faces superposées, la suppression d'une face superposée devrait nous permettre de voir la face qui se trouve dessous. Le matériel ne possède pas cette possibilité de mémorisation de faces superposées c'est-à-dire que sa gestion et sa mémorisation reviennent au calculateur pilote. Sa réalisation peut se faire comme suit (fig. 1) :

pour chaque face communiquée :

* conservation de ses sommets et de son rectangle englobant dans un tableau,

* Mise à jour d'un pointeur de face.

* N° de face	Pointeurs	nb. de sommets	Liste des sommets	Rectangle englogant
	- précédant - suivant		(x,y,z).....	g(x,y), d(x,y)

* DEBUT ← pointeur sur le n° de la première face à afficher.

Fig.1 : mémorisation de faces superposées

. L'affichage de la scène se fait par parcours des pointeurs de face.
. La modification du contour d'une face implique la comparaison du rectangle englobant de l'ancien et du nouveau contour.

A. Ancien contour inclu dans nouveau contour : affichage du nouveau contour et des faces superposées.

B. Nouveau contour inclu dans ancien contour : affichage de toutes les faces ayant une intersection avec l'ancien contour, du nouveau contour, et des faces superposées.

. La superposition d'une face se fait comme l'action B ci-dessus, de même que l'attribut d'aspect : invisibilité d'une face.

Exemple : ce qu'on doit re-afficher quand suppression de la face A qui cachait la face B (fig. 2).

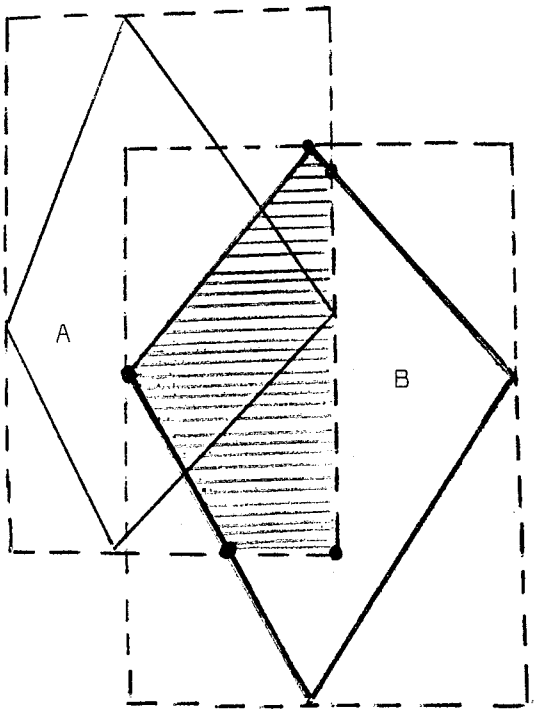


Fig. 2 : AFFICHAGE de la partie hachurée quand suppression de la face A qui se trouvait devant la face B.

. La modification des attributs géométriques (translation d'une face ... etc.) impose ces mêmes remarques.

Nous avons implémenté ce type d'élément mais il est apparu que les temps de calcul pour l'affichage étaient considérablement augmentés, que des restrictions dans le nombre de faces devaient être faites pour que ces éléments tiennent dans la mémoire du calculateur. Des essais ont montré que la mémorisation sur disquette ne donnait plus des temps de calculs acceptables. Une autre solution consistait à ne pas mémoriser les informations morphologiques soit

- en ayant accès à la base de données graphiques,

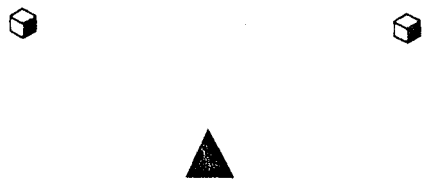
soit

- en demandant au calculateur principal de nous communiquer toutes les informations nécessaires à chaque modification.

Dans le premier cas, le logiciel pilote devait donc connaître la représentation graphique sous forme arborescente et être capable de l'exploiter. Ceci détruisait la cohérence et allourdissait considérablement les programmes. Dans le deuxième cas, la communication était si longue que ce type d'éléments se révélait être un piètre outil.
Néanmoins ce type d'élément pourrait être implémenté sur un gros calculateur.

ANNEXE 4

LE MANUEL D'UTILISATION



L'utilisateur peut travailler soit à partir de MULTICS, soit en local. Les primitives sont de toutes façons identiques, c'est le logiciel qui gère la transmission.

N.B. . Le clavier (AZELTINE 1520 par exemple) sera connecté au port A
. MULTICS ou la tablette au port B.

La mise en marche du système comprend le branchement

* des alimentations
* du moniteur
* de l'azeltine
* de la pascaline
* du lecteur de disquette.

Introduire seulement après, disquette de programmes (4) et base de données des textures (5).

Appuyer sur RESET

l'utilisation d'un programme se fait en frappant X Nom du programme

→ les programmes sont nombreux parmi les principaux :

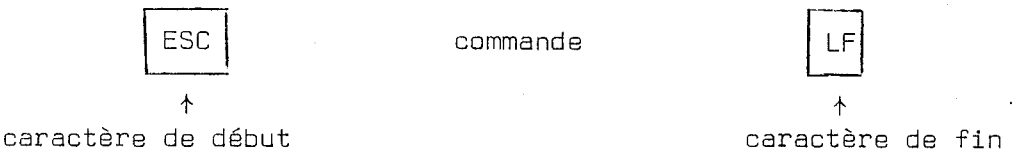
. le logiciel PILOTE : LOGVISU
. le logiciel de création de textures TEXTURES
. le logiciel de récupération d'images : IMAGDISK
. la création d'objets de REVOLUTION : REVOLUT
. la création de courbes du type z = f(x, y) : COURBXY

} progr. locaux

Nous ne parlerons ici que du logiciel PILOTE, pour qui a été défini un véritable langage de transmission, ce langage se trouve sous forme de primitives codées avec toujours le minimum de caractères pour réduire le temps de transmission.

1. TRANSMISSION DE COMMANDES

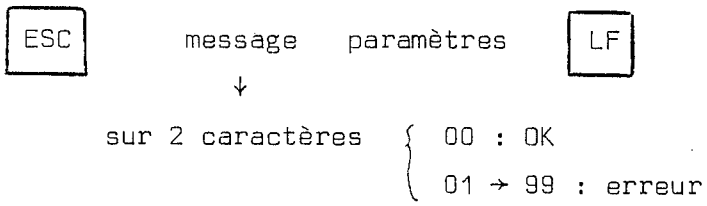
* UTILISATEUR → LOGICIEL



une commande comprend une primitive suivie d'un certain nombre de paramètres.

* LOGICIEL → UTILISATEUR

Quand une commande s'est exécutée, le logiciel renvoie en écho un message (d'erreur éventuellement) et des paramètres



N.B. : La séparation entre les entités se fait par un blanc quand il y a besoin d'un séparateur.

* Définition générale des primitives

Au début d'un programme, il faut INITIALISER le matériel par la primitive I.

Elle comprend 2 paramètres :

. Le nom du fichier du modèle d'éclairage (8 caractères maximum)

Si le fichier s'appelle MODELE.TEXT, donner MODELE (.TEXT est rajouté systématiquement).
Voici par exemple le contenu du fichier :

modele.text

```
 0  0  0  1  1  1  1  2  2  2  3  3  3  4  4
 5  5  6  7  7  8  8  9 10 11 12 12 13 14 14 13
 0  0  0  1  1  1  1  2  2  2  3  3  3  4  4
 5  5  6  7  7  8  8  9 10 11 44 76 139 126 156 191
15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
 3  3  3  3  3  3  3  3  3  3  3  3  3  3  3
22 33 33 54  7  7 33 132 113 116 134 157 166 166 172 134
```

. Le nom du fichier des textures (8 caractères maximum)

(.TEXT est aussi rajouté systématiquement au nom fourni).
Voici par exemple le contenu du fichier :

textnorm.text

doit exister à la fin de chaque fichier ↓

or brique bonbon or drapeau ciel jaune rouge vert bleu blanc rose fin

↑ texture appartenant à la base de donnée des textures

Ces deux fichiers doivent se trouver sur # 4 , la base de données sur # 5.

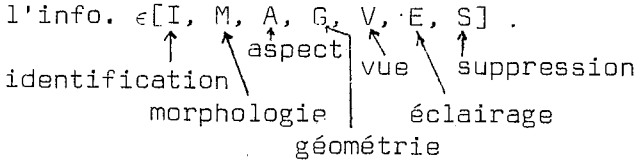
Cette primitive initialise le matériel (mémoires de la logique cablée : module d'éclairage, mémoire de trame ...), et les tables et paramètres qui se trouvent au niveau logiciel (table des noms de textures, indice table des faces ... etc ...).

a. L'attribution des informations se fait par la primitive

A X paramètres

↑

caractéristique de l'info. ∈[I, M, A, G, V, E, S] .



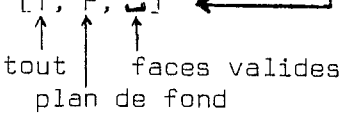
les informations de vue ou d'éclairage concernent toute la scène alors que les autres informations ne concernent qu'une (ou plusieurs) face(s). L'attribution de l'identification permet de dire quelles sont les éléments valides pour la communication d'informations qui suit (Si ces éléments sont nouveaux dans le système, on leur attribue une face d'HELIOS), il faut donc

→ attribuer l'identification avant la morphologie, l'aspect ou la géométrie.

les paramètres dépendent du type d'information et sont explicités plus loin.

b. La VISUALISATION se fait par la primitive V X

caractéristique de l'affichage ∈ [T, F, ب]



l'affichage complet se fait dans l'ordre de communication des nouvelles faces alors que l'affichage des faces valides se fait dans l'ordre de communication des faces valides.

c. La description de coordonnées se fait par la primitive

D X

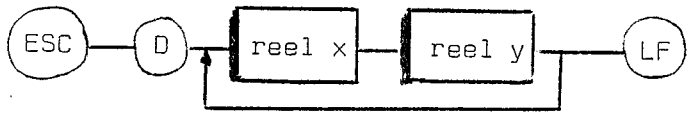
↑

caractéristique du dispositif de description ∈ [T, R, C, ?]

↑ tablette, ↖ réticule, ↖ clavier

. Quand le dispositif est défini par "?", lors de l'exécution de la commande, il est demandé au clavier.

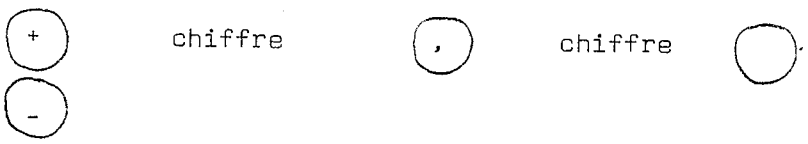
. Cette primitive renvoie à MULTICS ou à la console-opérateurs, (si LOCAL) les coordonnées récupérées (max : 20) sous la forme



N.B. * la communication des points de la tablette pourra être interrompue par le bouton 4

* la communication de chaque point du réticule sera validée par l'effacement du réticule et pourra être interrompue par ESC

* la communication au clavier se terminera par ESC ; chaque coordonnée sera saisie comme suit :



exemple : - 1205 ␣ -4231.5 ␣ 12.05 ␣ 2 ␣ +12.0567 ␣

d. La consultation se fait par la primitive

C X paramètres

↑

caractèristique du type d'info : [I, A, G]

↑ identification, ↖ aspect, ↖ géométrie

exemple : Identification de faces

C I champ1 champ2 champ3 champ4

champ1, champ2, champ3, champ4 seront ou non présents. Ils permettent d'identifier des faces ayant des caractéristiques communes.

champ1 := R (ident. grâce au réticule)

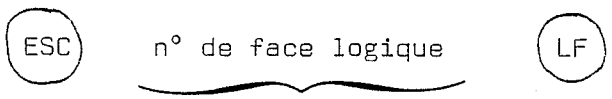
champ2 := N x y z (ident. de toutes les faces ayant cette normale)
3 réels

champ3 := T nom-texture (ident. des faces ayant la texture non-texture)
8 car.max

champ4 := P no-plan (ident. des faces appartenant au plan noplan)
car:Ø ou1

n.B. Si plusieurs champs sont présents c'est l' qui donnera les faces.

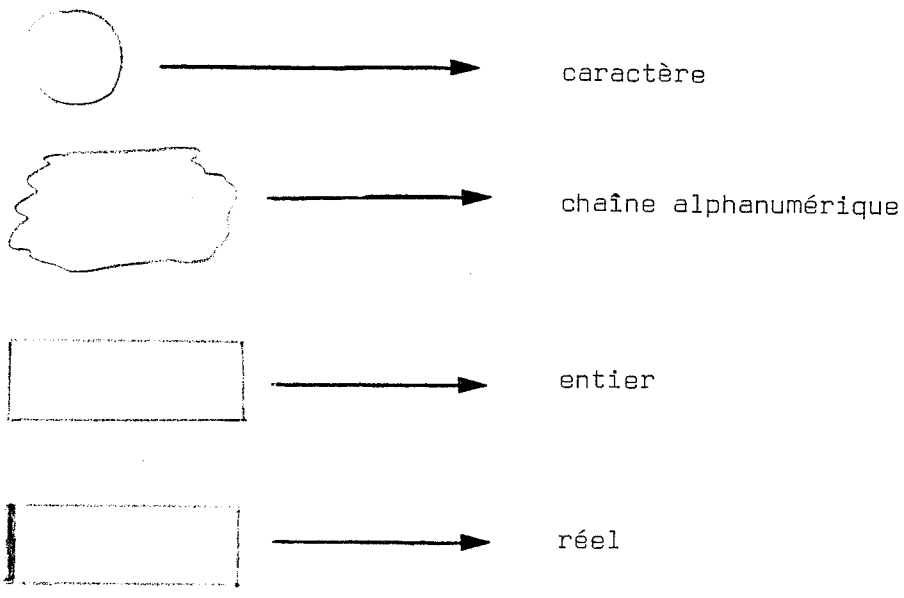
Ces faces seront communiquées de la façon suivante :

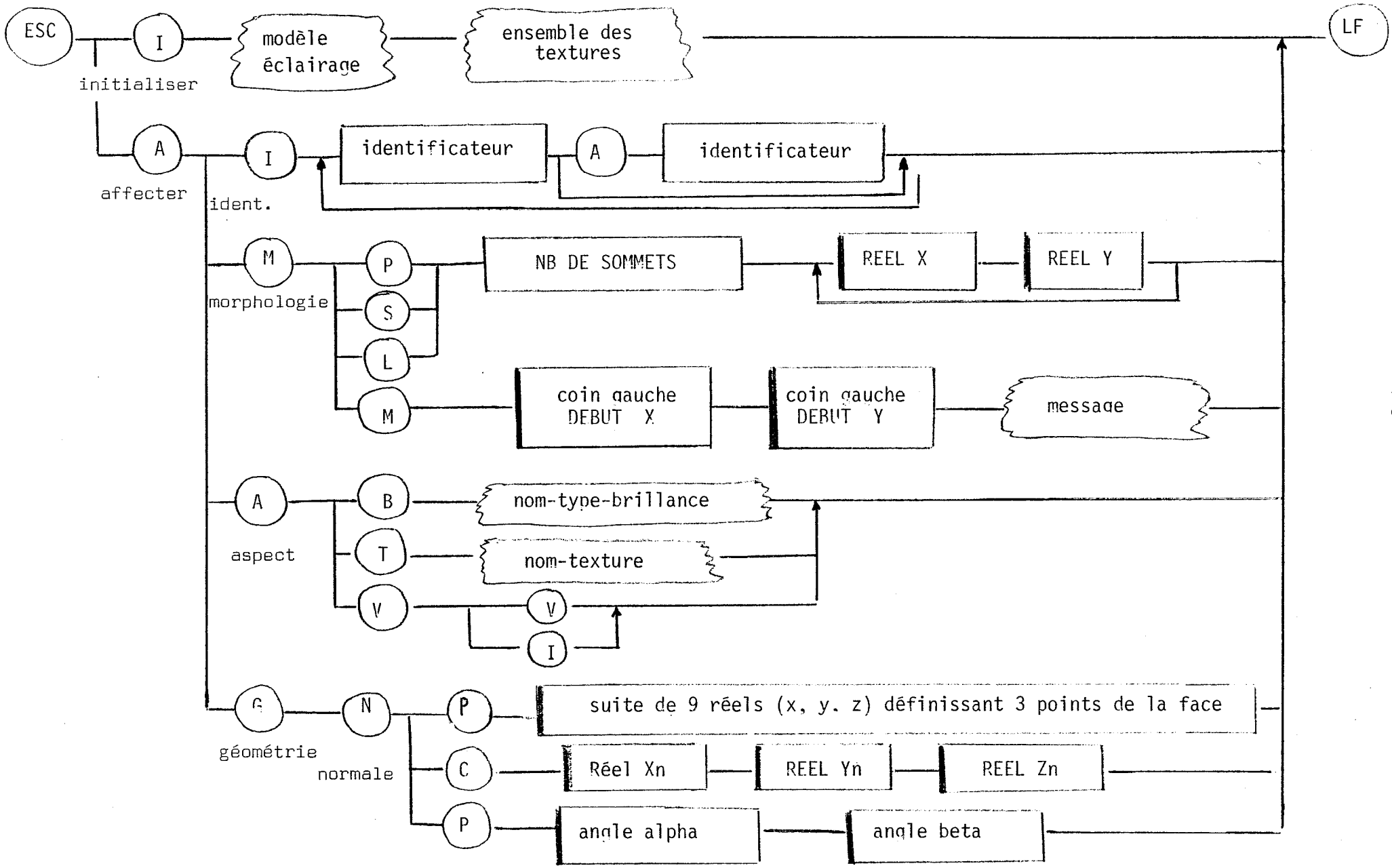


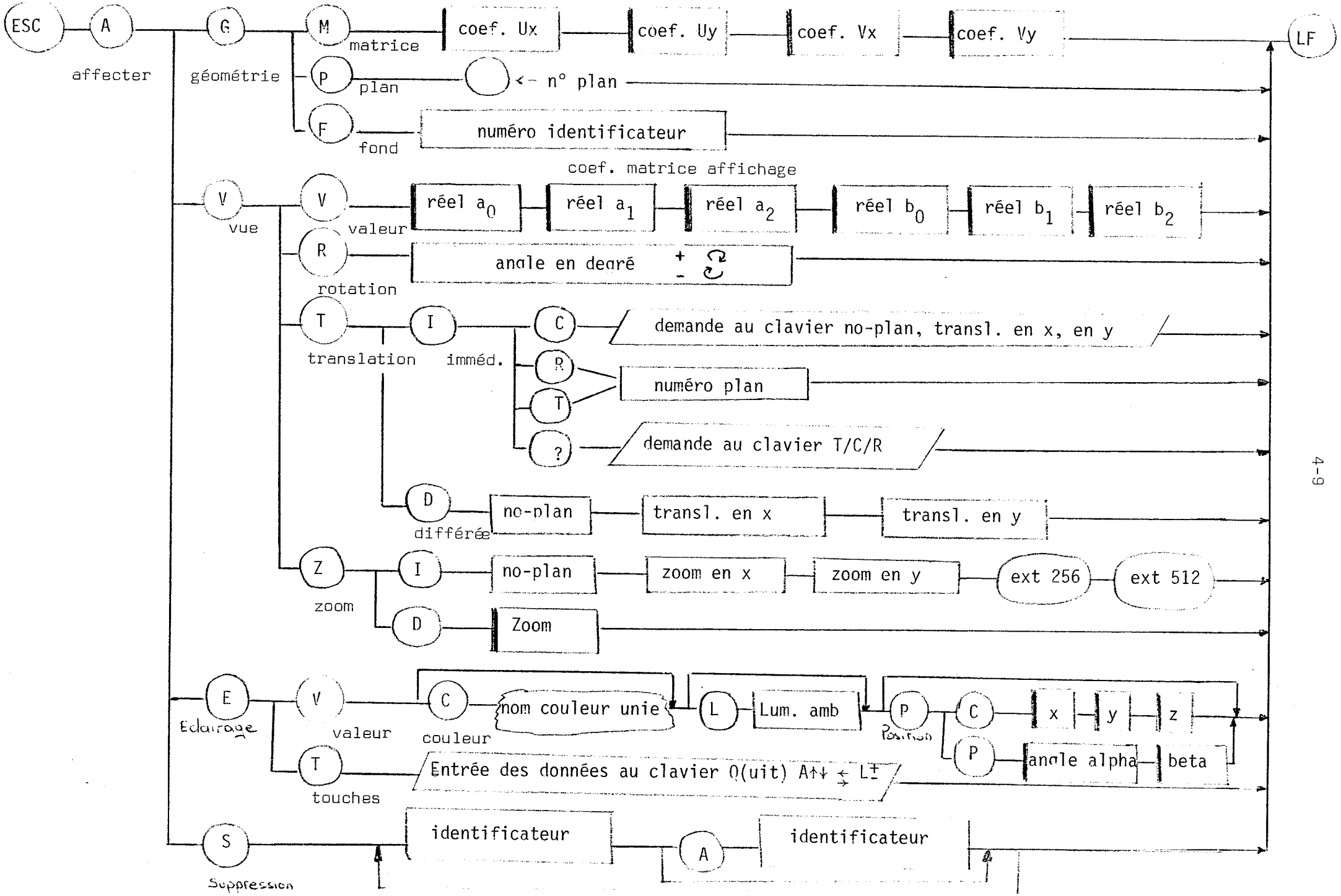
car. de début n entiers car. de fin

2. UTILISATEUR → LOGICIEL

Les paramètres sont représentés respectivement par les symboles suivants :




ESC
I
initialiser
modèle éclairage
ensemble des textures
LF
A
affecter
I
ident.
identificateur
A
identificateur
M
morphologie
P
S
L
M
NB DE SOMMETS
REEL X
REEL Y
coin gauche DEBUT X
coin gauche DEBUT Y
message
A
aspect
B
T
V
nom-type-brillance
nom-texture
V
I
G
géométrie
N
normale
P
C
P
suite de 9 réels (x, y, z) définissant 3 points de la face
Réel Xn
REEL Yn
REEL Zn
angle alpha
angle beta


ESC
A
affecter
G
géométrie
M
matrice
coef. Ux
coef. Uy
coef. Vx
coef. Vy
LF
P
plan
n° plan
F
fond
numéro identificateur
coef. matrice affichage
V
vue
V
valeur
réel a_0
réel a_1
réel a_2
réel b_0
réel b_1
réel b_2
R
rotation
angle en degré
T
translation
I
imméd.
C
demande au clavier no-plan, transl. en x, en y
R
T
numéro plan
?
demande au clavier T/C/R
D
différée
no-plan
transl. en x
transl. en y
Z
zoom
I
no-plan
zoom en x
zoom en y
ext 256
ext 512
D
Zoom
E
Eclairage
V
valeur
C
couleur
nom couleur unie
L
Lum. amb
P
Position
C
x
y
z
P
angle alpha
beta
T
touches
Entrée des données au clavier Q(uit) A↑↓ ← → L±
S
Suppression
identificateur
A
identificateur

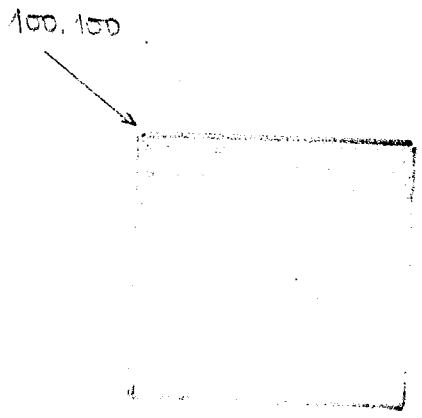
5 - Exemples de programmes locaux générant des fichiers "macro-commandes"

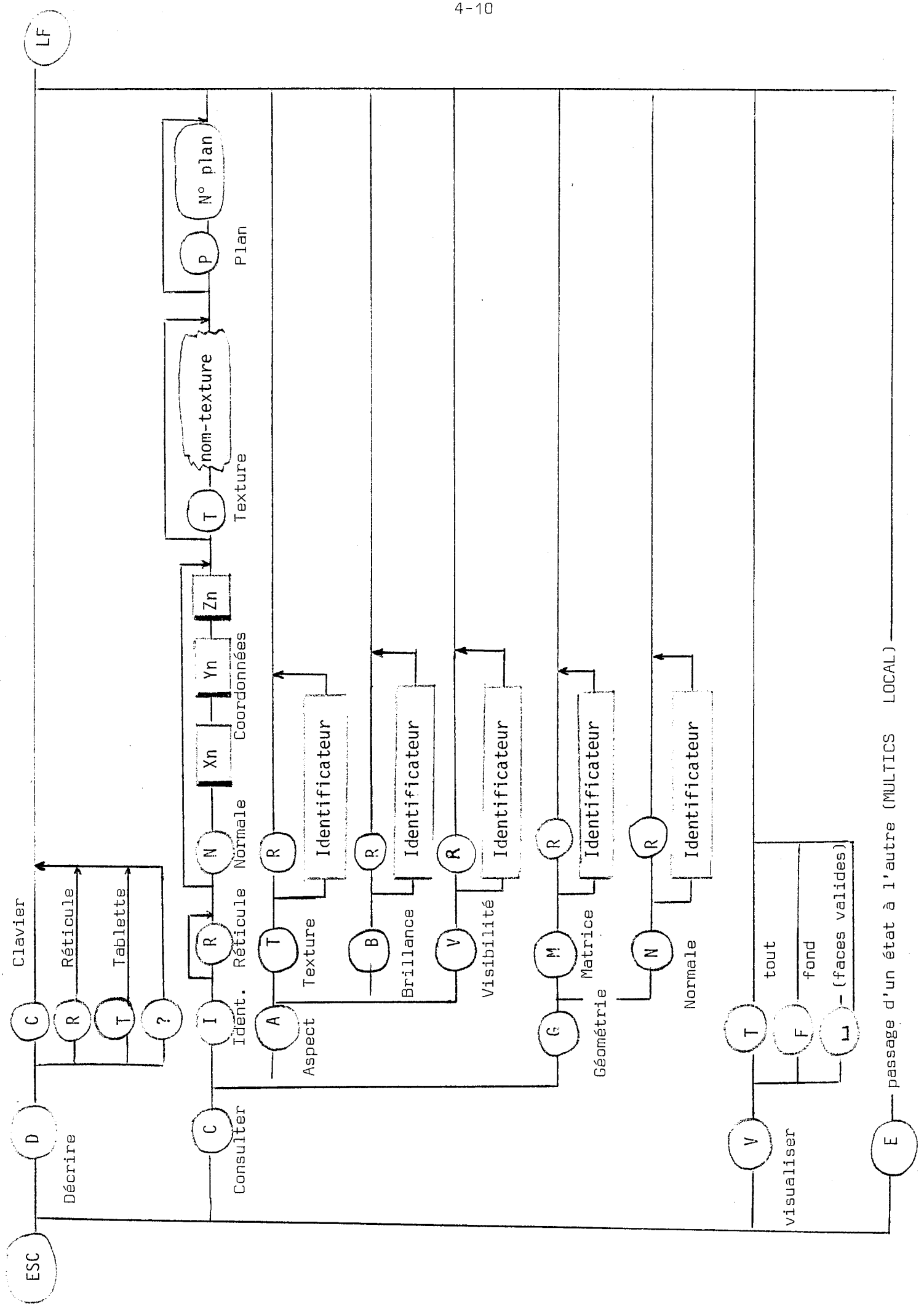
```
program commande;

var f:text; nomfich:string[12];
begin
  writeln('nom fichier de commande'); readln(nomfich);
  rewrite(f,nomfich);
  writeln(f,'i modele texture',chr(13));   initialisation modèle éclairage et mémoire de texture
  writeln(f,'a i 2000 5000 4000',chr(13));   attribution des paramètres
  writeln(f,'a a p 4 100 100 100 200 200 200 200 100',chr(13));
  writeln(f,'a o a',chr(13));   attribution de brillance
  writeln(f,'a a c bleu',chr(13));   attribution d'une texture
  writeln(f,'v',chr(13));   visualisation
  writeln(f,'e',chr(13));   fin de programme
  close(f,lock);
end.
```

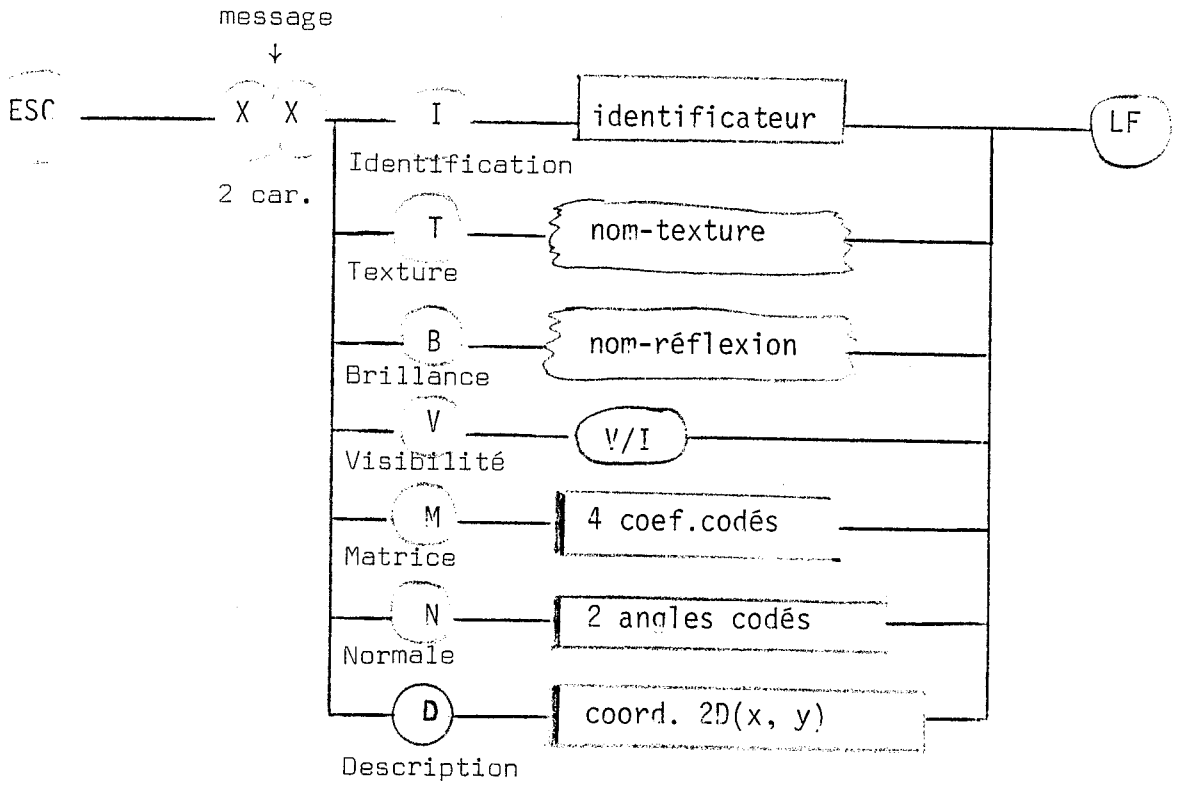
Contenu du fichier "macro-commande"

ESC i modele texture LF
ESC o i 2000 5000 4000 LF
ESC a m p 4 100 100 100 200 200 200 200 100 LF
ESC o a b 2 LF
ESC a a t bleu LF
ESC v LF
ESC e LF




LF
ESC
D
Décrire
C
Clavier
R
Réticule
T
Tablette
?
C
Consulter
I
Ident. Réticule
R
N
Normale
Xn
Yn
Zn
Coordonnées
T
nom-texture
Texture
P
N° plan
Plan
A
Aspect
T
Texture
R
Identificateur
B
Brillance
R
Identificateur
V
Visibilité
R
Identificateur
G
Géométrie
M
Matrice
R
Identificateur
N
Normale
R
Identificateur
V
visualiser
T
tout
F
fond
L
(faces valides)
E
passage d'un état à l'autre (MULTICS LOCAL)

3. LOGICIEL → UTILISATEUR



← caractère ASCII (= 1 octet)

← entier codé sur 1, 2, ou 3 octets

← réel codé sur 1 ou 2 octets pour l'exposant
\+ 1 ou 2 octets pour la montisse.

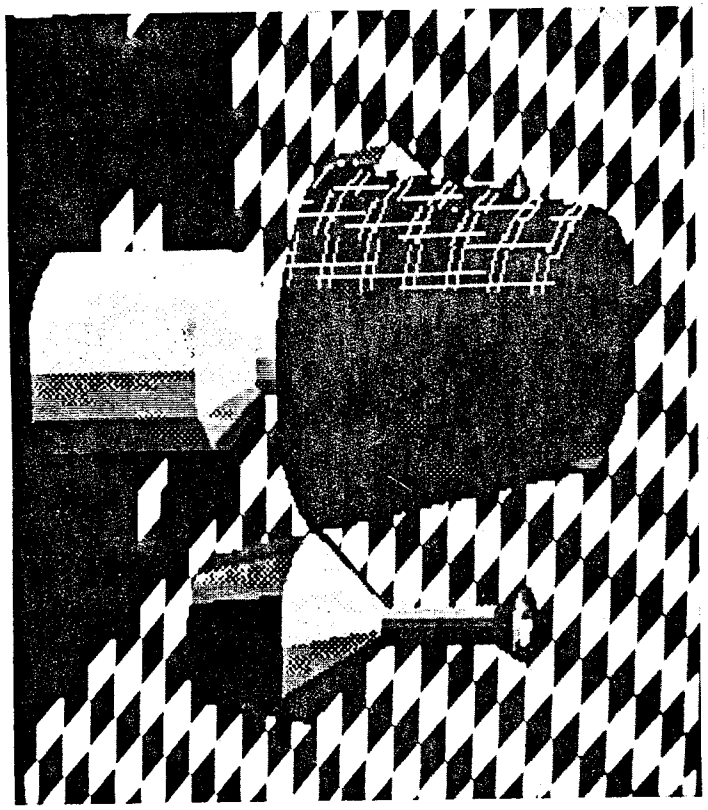
```
procedure [illegible];

var [illegible]: [illegible];
procedure [illegible];
begin
  writeln('Ooh Richter 10 commands');  read(ch);
  repeat
   write(e,ch);
   write(e,enc(27),'E Double Command',enc(0));
   write(e,enc(27),'4 1 ///',enc(1));
   write(e,enc(27),'4 4 [illegible]',enc(1));
   write(e,enc(27),'4 5 1 ///',enc(1));
   write(e,enc(27),'4 6 [illegible]',enc(1));
   write(e,enc(27),'4 7 1///',enc(1));
   write(e,enc(27),'4 8 [illegible]',enc(1));
   write(e,enc(27),'4 9 1///',enc(1));
   write(e,enc(27),'4 10 [illegible]',enc(1));
   write(e,enc(27),'4 11 1///',enc(1));
   write(e,enc(27),'4 12 [illegible]',enc(1));
   write(e,enc(27),'4 13 1///',enc(1));
   write(e,enc(27),'4 14 [illegible]',enc(1));
   write(e,enc(27),'4 15 1///',enc(1));
   write(e,enc(27),'4 16 [illegible]',enc(1));
   write(e,enc(27),'4 17 1///',enc(1));
   write(e,enc(27),'4 18 [illegible]',enc(1));
   write(e,enc(27),'4 19 1///',enc(1));
   write(e,enc(27),'4 20 [illegible]',enc(1));
   write(e,enc(27),'4 21 1///',enc(1));
   write(e,enc(27),'4 22 [illegible]',enc(1));
   write(e,enc(27),'4 23 1///',enc(1));
   write(e,enc(27),'4 24 [illegible]',enc(1));
   write(e,enc(27),'4 25 1///',enc(1));
   write(e,enc(27),'4 26 [illegible]',enc(1));
   write(e,enc(27),'4 27 1///',enc(1));
   write(e,enc(27),'4 28 [illegible]',enc(1));
   write(e,enc(27),'4 29 1///',enc(1));
   write(e,enc(27),'4 30 [illegible]',enc(1));
   write(e,enc(27),'4 31 1///',enc(1));
   write(e,enc(27),'4 32 [illegible]',enc(1));
   write(e,enc(27),'4 33 1///',enc(1));
   write(e,enc(27),'4 34 [illegible]',enc(1));
   write(e,enc(27),'4 35 1///',enc(1));
   write(e,enc(27),'4 36 [illegible]',enc(1));
   write(e,enc(27),'4 37 1///',enc(1));
   write(e,enc(27),'4 38 [illegible]',enc(1));
   write(e,enc(27),'4 39 1///',enc(1));
   write(e,enc(27),'4 40 [illegible]',enc(1));
   write(e,enc(27),'4 41 1///',enc(1));
   write(e,enc(27),'4 42 [illegible]',enc(1));
   write(e,enc(27),'4 43 1///',enc(1));
   write(e,enc(27),'4 44 [illegible]',enc(1));
   write(e,enc(27),'4 45 1///',enc(1));
   write(e,enc(27),'4 46 [illegible]',enc(1));
   write(e,enc(27),'4 47 1///',enc(1));
   write(e,enc(27),'4 48 [illegible]',enc(1));
   write(e,enc(27),'4 49 1///',enc(1));
   write(e,enc(27),'4 50 [illegible]',enc(1));
   write(e,enc(27),'4 51 1///',enc(1));
   write(e,enc(27),'4 52 [illegible]',enc(1));
   write(e,enc(27),'4 53 1///',enc(1));
   write(e,enc(27),'4 54 [illegible]',enc(1));
   write(e,enc(27),'4 55 1///',enc(1));
   write(e,enc(27),'4 56 [illegible]',enc(1));
   write(e,enc(27),'4 57 1///',enc(1));
   write(e,enc(27),'4 58 [illegible]',enc(1));
   write(e,enc(27),'4 59 1///',enc(1));
   write(e,enc(27),'4 60 [illegible]',enc(1));
   write(e,enc(27),'4 61 1///',enc(1));
   write(e,enc(27),'4 62 [illegible]',enc(1));
   write(e,enc(27),'4 63 1///',enc(1));
   write(e,enc(27),'4 64 [illegible]',enc(1));
  until;

procedure [illegible];
begin
   write(e,enc(27),'4 1 1/// 13/// 15/// 21/// 3/// 4///',enc(1));
   write(e,enc(27),'4 4 [illegible]',enc(1));
   write(e,enc(27),'4 5 11/// 1///// 15/// 2///',enc(1));
   write(e,enc(27),'4 7 [illegible]',enc(1));
   write(e,enc(27),'4 9 1/// 13/1',enc(1));
   write(e,enc(27),'4 11 [illegible]',enc(1));
   write(e,enc(27),'4 13 1/// 1/// 11/// 12/// 13/// 14/// 15/// 16/// 17/// 18/// 19/// 20/// 21/// 22/// 23/// 24/// 25/// 26/// 27/// 28/// 29/// 30/// 31/// 32/// 33/// 34/// 35/// 36/// 37/// 38/// 39/// 40/// 41/// 42/// 43/// 44/// 45/// 46/// 47/// 48/// 49/// 50/// 51/// 52/// 53/// 54/// 55/// 56/// 57/// 58/// 59/// 60/// 61/// 62/// 63/// 64///',enc(1));
   write(e,enc(27),'4 15 [illegible]',enc(1));
   write(e,enc(27),'V',enc(1));
   write(e,enc(27),'@',enc(1));
   close(e,false);
end;

begin
   [illegible]; [illegible];
end.
```

```
program commande;
var fichier: text; condition: string[10];
    i,j,k,l,n,dx,dy,nfois,pfois:integer;
    a,b,c,x,xm,yx,bb,ch,co,alpha,beta:real;
    ax,ay:array[1..15] of integer;
    xbon,ybon:integer; nombon:string; fich:text;
begin
  write('nom fichier de commande'); readln(nomfich);
  reset(f,nomfich);
  write(f,chr(27),'l droite centrée',chr(13));
  assign(fich,'bonc.text'); rewrite(fich,nfois);
  write('dessin ?');readln(dx,dy);
  write('couleur ? ');readln(xm);
  write('nombre fois fois toujours ? ');readln(pfois);
  i:=pfois; write(pfois,' ',nfois);
  repeat
    i:=1; readln(fich,j,k);
    while j<> 9999 do
    begin
      i:=i+1;
      ax[i]:=round(xm*j+dx); ay[i]:=round(-xm*k+dy);
      readln(fich,j,k);
    end;
    write(' ',i); write(f,chr(27),'a 1 ',i,chr(13));
    write(f,chr(27),'a a 2 ',i,' ');
    for n:=1 to i do write(f,ax[n],' ',ay[n],' ');
    write(f,chr(13));
    readln(fich,a,b);
    write(f,'a 3 a 2',a,' ',b,chr(13));
    readln(fich,a,b,c,d);
    write(f,'a 3 a',a,' ',b,' ','c',' ',d,chr(13));
    i:=i+1;
  until i=pfois+nfois;
  close(fich);
  for i:=pfois to pfois+nfois do
  begin
    write(f,chr(27),'a 1 ',pfois,'a',pfois + nfois,chr(13));
    write(f,chr(27),'a a c bleu',chr(13));
    write(f,chr(27),'v',i,chr(13));
  end;
  write(f,chr(27),'e',chr(13)); close(f,lock);
end.
```



BIBLIOGRAPHIE

[MAR 81] MARTINEZ F., *HELIOS : terminal interactif pour la synthèse d'images réalistes,* Congrès AFCET TTI, Gif sur Yvette, Novembre 1981, paru dans Le Nouvel Automatisme, n° 30, Mai 1982.

[MAR 82] MARTINEZ F., *Vers une approche systématique de la synthèse d'images. Aspects logiciel et matériel,* Thèse de Doctorat d'Etat, Grenoble, à paraître.

[MAT 78] MATHERAT Ph., *A chip for low cost raster scan graphic display,* SIGGRAPH, ACM Computer Graphics, 1978.

[MER 79] MERIAUX M., *Etude et réalisation d'un terminal graphique couleur tri-dimensionnel fonctionnant par tâches,* Thèse de Docteur-Ingénieur, Lille, Janvier 1979.

[MYE 81] MYERS W., *Reaching the user,* Computer, pp. 7-17, Mars 1981.

[NNS 72] NEWELL M.E., NEWELL R.G., SANCHA T.L., *A new approach to the shaded picture problem,* Proceedings ACM National Conference, 1972.

[PAV 81] PAVLIDIS T., *Filling algorithms for raster graphics,* SIGGRAPH, ACM Computer Graphics, vol 15, n° 3, Août 1981.

[PAY 82] PAYAN J.L., *Textures : description et modélisation,* Rapport de D.E.A., Grenoble, 1982, à paraître.

[PHO 75] PHONG THONG Bui, *Illumination for computer generated pictures,* Communications of the ACM, vol 18, n° 6, Juin 1975.

[SAR 81] SARRASIN M.T., *La synthèse d'images réalistes,* Probatoire CNAM, Grenoble, Mai 1981.

[WAT 70] WATKINS G.S., *A real time visible surface algorithm,* University of Utah, UTEC, CCS 70101, Juin 1970

[BLI 77] BLINN J.F., *Models of light reflection for computer synthetized pictures,* SIGGRAPH, ACM Computer Graphics, vol 11, n° 2, Juillet 1977.

[BLI 78] BLINN J.F., *Simulation of wrinkled surfaces,* SIGGRAPH, ACM Computer Graphics, vol 12, n° 3, Août 1978.

[BOU 70] BOUKNIGHT J. & KELLEY K., *An algorithm for producing half tone computer graphics. Presentation with shadow and movable light sources,* Spring Joint Computer Conference, 1970.

[BOU 80] BOULLE Ph., *Etude et réalisation d'algorithmes pour la visualisation de scènes composées de facettes planes,* Thèse de Docteur-Ingénieur, Grenoble, 1980.

[BRE 65] BRESENHAM J.E., *Algorithm for computer control of a digital plotter,* IBM System Journal, vol 4, n° 1, 1965.

[BRE 82] BRESENHAM J.E., *Incremental line compaction,* The Computer Journal, vol 25, n° 1, 1982.

[EAR 77] EARNSHAW R.A., *Line-tracking for incremental and raster devices,* SIGGRAPH, ACM Computer Graphics, vol 11, n° 2, 1977.

[ENC 80] ENCARNACAO J. & ECKERT R., *Concepts in workstation in G.K.S.,* Computer Graphics, ACM, vol 14, n° 3, 1980.

[FER 81] FERREIRA NUNES Fernando, *Conception et réalisation d'un système INTERACTIF pour la synthèse d'images,* Thèse de Docteur-Ingénieur, Grenoble, Octobre 1981.

[IGL 78] Note technique TEKTRONIX 1978.

[LER 80] LERAY P., *Réalisation d'un système de génération synthétique d'images en temps réel (G.S.I.)*, Congrès AFCET TTI, Nancy, Novembre 1980.

[LIE 78] LIEBERMAN H., *How to color in a coloring book*, SIGGRAPH, ACM Computer Graphics, vol 12, n° 3, Août 1978.

[LLM 77] LEDUC LEBALLEUR A., LUCAS M., MARTINEZ F., *Conception et réalisation d'un logiciel graphique interactif indépendant du contexte d'utilisation : le logiciel de base GRIGRI*, Séminaire d'Informatique, Laboratoire IMAG, Juin 1977, et Congrès AFCET TTI, Novembre 1977.

[LUC 74] LUCAS M., *Technologie, programmation et utilisation des consoles de visualisation. Etat des recherches actuelles*, Rapport DRME, Laboratoire IMAG, Mai 1974.

[LUC 77] LUCAS M., *Contribution à l'étude des techniques de communication graphique avec un ordinateur. Elément de base des logiciels graphiques interactifs*, Thèse de Doctorat d'Etat ès-Sciences, Grenoble, Décembre 1977.

[MAC 81] MARSON C., *Terminaux : la couleur en plus*, 01 Informatique, Février 1981.

[MAR 79] MARTINEZ F., *La synthèse d'images*, Convention LMT - MICADO, Grenoble, Janvier 1979.

[MAR 79] MARTINEZ F., *An approach to the modelling and display of landscape* EUROGRAPHICS 79, Bologne, Italie, 1979.

[MAR 80] MARTINEZ F., *CLOVIS : complexe logiciel pour la visualisation interactive structurée*, Congrès AFCET TTI, Nancy, Novembre 1980.