Etude et réalisation d'un synthétiseur d'images basé sur une architecture banalisée

Victor Hugo Zarate Silva

THESE

présentée à

l' Institut National Polytechnique de Grenoble

pour obtenir le grade de
DOCTEUR INGENIEUR
«Informatique»

par

Victor Hugo ZARATE SILVA

ETUDE ET REALISATION D'UN SYNTHETISEUR D'IMAGES BASE SUR UNE ARCHITECTURE BANALISEE.

Thèse soutenue le 4 novembre 1985 devant la commission d'examen.

G. VEILLON	Président
J. MERMET	Examinateurs
F. MARTINEZ	
M. MERIAUX	
J.F. MIRIBEL	

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1982-1983

Président de l'Université : D. BLOCH

Vice-Président : René CARRE
Hervé CHERADAME
Marcel IVANES

PROFESSEURS DES UNIVERSITES :

ANCEAU François	E.N.S.I.M.A.G.
BARRAUD Alain	E.N.S.I.E.G.
BAUDELET Bernard	E.N.S.I.E.G.
BESSON Jean	E.N.S.E.E.G.
BLIMAN Samuel	E.N.S.E.R.G.
BLOCH Daniel	E.N.S.I.E.G.
BOIS Philippe	E.N.S.H.G.
BONNETAIN Lucien	E.N.S.E.E.G.
BONNIER Etienne	E.N.S.E.E.G.
BOUVARD Maurice	E.N.S.H.G.
BRISSONNEAU Pierre	E.N.S.I.E.G.
BUYLE BODIN Maurice	E.N.S.E.R.G.
CAVAIGNAC Jean-François	E.N.S.I.E.G.
CHARTIER Germain	E.N.S.I.E.G.
CHENEVIER Pierre	E.N.S.E.R.G.
CHERADAME Hervé	U.E.R.M.C.P.P.
CHERUY Arlette	E.N.S.I.E.G.
CHIAVERINA Jean	U.E.R.M.C.P.P.
COHEN Joseph	E.N.S.E.R.G.
COUMES André	E.N.S.E.R.G.
DURAND Francis	E.N.S.E.E.G.
DURAND Jean-Louis	E.N.S.I.E.G.
FELICI Noël	E.N.S.I.E.G.
FOULARD Claude	E.N.S.I.E.G.
GENTIL Pierre	E.N.S.E.R.G.
GUERIN Bernard	E.N.S.E.R.G.
GUYOT Pierre	E.N.S.E.E.G.
IVANES Marcel	E.N.S.I.E.G.
JAUSSAUD Pierre	E.N.S.I.E.G.
JOUBERT Jean-Claude	E.N.S.I.E.G.
JOURDAIN Geneviève	E.N.S.I.E.G.
LACOUME Jean-Louis	E.N.S.I.E.G.
LATOMBE Jean-Claude	E.N.S.I.M.A.G.

.../...

LESSIEUR Marcel	E.N.S.H.G.
LESPINARD Georges	E.N.S.H.G.
LONGEQUEUE Jean-Pierre	E.N.S.I.E.G.
MAZARE Guy	E.N.S.I.M.A.G.
MOREAU René	E.N.S.H.G.
MORET Roger	E.N.S.I.E.G.
MOSSIERE Jacques	E.N.S.I.M.A.G.
PARIAUD Jean-Charles	E.N.S.E.E.G.
PAUTHENET René	E.N.S.I.E.G.
PERRET René	E.N.S.I.E.G.
PERRET Robert	E.N.S.I.E.G.
PIAU Jean-Michel	E.N.S.H.G.
POLOUJADOFF Michel	E.N.S.I.E.G.
POUPOT Christian	E.N.S.E.R.G.
RAMEAU Jean-Jacques	E.N.S.E.E.G.
RENAUD Maurice	U.E.R.M.C.P.P.
ROBERT André	U.E.R.M.C.P.P.
ROBERT François	E.N.S.I.M.A.G.
SABONNADIERE Jean-Claude	E.N.S.I.E.G.
SAUCIER Gabrielle	E.N.S.I.M.A.G.
SCHLENKER Claire	E.N.S.I.E.G.
SCHLENKER Michel	E.N.S.I.E.G.
SERMET Pierre	E.N.S.E.R.G.
SILVY Jacques	U.E.R.M.C.P.P.
SOHM Jean-Claude	E.N.S.E.E.G.
SOUQUET Jean-Louis	E.N.S.E.E.G.
VEILLON Gérard	E.N.S.I.M.A.G.
ZADWORNY François	E.N.S.E.R.G.

PROFESSEURS ASSOCIES

BASTIN Georges	E.N.S.H.G.
BERRIL John	E.N.S.H.G.
CARREAU Pierre	E.N.S.H.G.
GANDINI Alessandro	U.E.R.M.C.P.P.
HAYASHI Hirashi	E.N.S.I.E.G.

PROFESSEURS UNIVERSITE DES SCIENCES SOCIALES (Grenoble II)

BOLLIET Louis
Chatelin Françoise

PROFESSEURS E.N.S. Mines de Saint-Etienne

RIEU Jean
SOUSTELLE Michel

CHERCHEURS DU C.N.R.S.

FRUCHART Robert	Directeur de Recherche
VACHAUD Georges	Directeur de Recherche

.../...

ALLIBERT Michel	Maître de Recherche
ANSARA Ibrahim	Maître de Recherche
ARMAND Michel	Maître de Recherche
BINDER Gilbert	
CARRE René	Maître de Recherche
DAVID René	Maître de Recherche
DEPORTES Jacques	
DRIOLE Jean	Maître de Recherche
GIGNOUX Damien	
GIVORD Dominique	
GUELIN Pierre	
HOPFINGER Emil	Maître de Recherche
JOUD Jean-Charles	Maître de Recherche
KAMARINOS Georges	Maître de Recherche
KLEITZ Michel	Maître de Recherche
LANDAU Ioan-Dore	Maître de Recherche
LASJAUNIAS J.C.	
MERMET Jean	Maître de Recherche
MUNIER Jacques	Maître de Recherche
PIAU Monique	
PORTESEIL Jean-Louis	
THOLENCE Jean-Louis	
VERDILLON André	

CHERCHEURS du MINISTERE de la RECHERCHE et de la TECHNOLOGIRE (Directeurs et Maîtres de Recherches, ENS Mines de St. Etienne)

LESBATS Pierre	Directeur de Recherche
BISCONDI Michel	Maître de Recherche
KOBYLANSKI André	Maître de Recherche
LE COZE Jean	Maître de Recherche
LALAUZE René	Maître de Recherche
LANCELOT Francis	Maître de Recherche
THEVENOT François	Maître de Recherche
TRAN MINH Canh	Maître de Recherche

PERSONNALITES HABILITEES à DIRIGER des TRAVAUX de RECHERCHE (Décision du Conseil Scientifique)

ALLIBERT Colette	E.N.S.E.E.G.
BERNARD Claude	E.N.S.E.E.G.
BONNET Rolland	E.N.S.E.E.G.
CAILLET Marcel	E.N.S.E.E.G.
CHATILLON Catherine	E.N.S.E.E.G.
CHATILLON Christian	E.N.S.E.E.G.
COULON Michel	E.N.S.E.E.G.
DIARD Jean-Paul	E.N.S.E.E.G.
EUSTAPOPOULOS Nicolas	E.N.S.E.E.G.
FOSTER Panayotis	E.N.S.E.E.G.

.../...

GALERIE Alain	E.N.S.E.E.G.
HAMMOU Abdelkader	E.N.S.E.E.G.
MALMEJAC Yves	E.N.S.E.E.G. (CENG)
MARTIN GARIN Régina	E.N.S.E.E.G.
NGUYEN TRUONG Bernadette	E.N.S.E.E.G.
RAVAINE Denis	E.N.S.E.E.G.
SAINFORT	E.N.S.E.E.G. (CENG)
SARRAZIN Pierre	E.N.S.E.E.G.
SIMON Jean-Paul	E.N.S.E.E.G.
TOUZAIN Philippe	E.N.S.E.E.G.
URBAIN Georges	E.N.S.E.E.G. (Laboratoire des ultra-réfractaires ODEILLON)
GUILHOT Bernard	E.N.S. Mines Saint Etienne
THOMAS Gérard	E.N.S. Mines Saint Etienne
DRIVER Julien	E.N.S. Mines Saint Etienne
BARIBAUD Michel	E.N.S.E.R.G.
BOREL Joseph	E.N.S.E.R.G.
CHOVET Alain	E.N.S.E.R.G.
CHEHIKIAN Alain	E.N.S.E.R.G.
DOLMAZON Jean-Marc	E.N.S.E.R.G.
HERAULT Jeanny	E.N.S.E.R.G.
MONLLOR Christian	E.N.S.E.R.G.
BORNARD Guy	E.N.S.I.E.G.
DESCHIZEAU Pierre	E.N.S.I.E.G.
GLANGEAUD François	E.N.S.I.E.G.
KOFMAN Walter	E.N.S.I.E.G.
LEJEUNE Gérard	E.N.S.I.E.G.
MAZUER Jean	E.N.S.I.E.G.
PERARD Jacques	E.N.S.I.E.G.
REINISCH Raymond	E.N.S.I.E.G.
ALEMANY Antoine	E.N.S.H.G.
BOIS Daniel	E.N.S.H.G.
DARVE Félix	E.N.S.H.G.
MICHEL Jean-Marie	E.N.S.H.G.
OBLED Charles	E.N.S.H.G.
ROWE Alain	E.N.S.H.G.
VAUCLIN Michel	E.N.S.H.G.
WACK Bernard	E.N.S.H.G.
BERT Didier	E.N.S.I.M.A.G.
CALMET Jacques	E.N.S.I.M.A.G.
COURTIN Jacques	E.N.S.I.M.A.G.
COURTOIS Bernard	E.N.S.I.M.A.G.
DELLA DORA Jean	E.N.S.I.M.A.G.
FONLUPT Jean	E.N.S.I.M.A.G.
SIFAKIS Joseph	E.N.S.I.M.A.G.
CHARUEL Robert	U.E.R.M.C.P.P.
CADET Jean	C.E.N.G.
COEURE Philippe	C.E.N.G. (LETI)

.../...

DELHAYE Jean-Marc	C.E.N.G. (STT)
DUPUY Michel	C.E.N.G. (LETI)
JOUVE Hubert	C.E.N.G. (LETI)
NICOLAU Yvan	C.E.N.G. (LETI)
NIFENECKER Hervé	C.E.N.G.
PERROUD Paul	C.E.N.G.
PEUZIN Jean-Claude	C.E.N.G. (LETI)
TAIEB Maurice	C.E.N.G.
VINCENDON Marc	C.E.N.G.

LABORATOIRES EXTERIEURS

DEMOULIN Eric	C.N.E.T.
DEVINE	C.N.E.T. (R.A.B.)
GERBER Roland	C.N.E.T.
MERCKEL Gérard	C.N.E.T.
PAULEAU Yves	C.N.E.T.
GAUBERT C.	I.N.S.A. Lyon

ECOLE NATIONALE SUPERIEURE DES MINES DE SAINT-ETIENNE

Directeur : **Monsieur M. MERMET**
Directeur des Etudes et de la formation : **Monsieur J. LEVASSEUR**
Directeur des recherches : **Monsieur J. LEVY**
Secrétaire Général : **Mademoiselle M. CLERGUE**

Professeurs de 1ère Catégorie

COINDE	Alexandre	Gestion
GOUX	Claude	Métallurgie
LEVY	Jacques	Métallurgie
LOWYS	Jean-Pierre	Physique
MATHON	Albert	Gestion
RIEU	Jean	Mécanique - Résistance des matériaux
SOUSTELLE	Michel	Chimie
FORMERY	Philippe	Mathématiques Appliquées

Professeurs de 2ème catégorie

HABIB	Michel	Informatique
PERRIN	Michel	Géologie
VERCHERY	Georges	Matériaux
TOUCHARD	Bernard	Physique Industrielle

Directeur de recherche

LESBATS	Pierre	Métallurgie

Maîtres de recherche

BISCONDI	Michel	Métallurgie
DAVOINE	Philippe	Géologie
FOURDEUX	Angeline	Métallurgie
KOBYLANSKI	André	Métallurgie
LALAUZE	René	Chimie
LANCELOT	Francis	Chimie
LE COZE	Jean	Métallurgie
THEVENOT	François	Chimie
TRAN MINH	Canh	Chimie

Personnalités habilitées à diriger des travaux de recherche

DRIVER	Julian	Métallurgie
GUILHOT	Bernard	Chimie
THOMAS	Gérard	Chimie

Professeur à l'UER de Sciences de Saint-Etienne

VERGNAUD	Jean-Maurice	Chimie des Matériaux & chimie industrielle

****

A mis padres y hermanos.

AVANT PROPOS

Ce travail a été réalisé grâce à l'aide du gouvernement mexicain à travers l'organisme CONACYT ("Consejo Nacional de Ciencia y Tecnologia"), dans le cadre d'une formation de ressources humaines du Mexique.

Ces quelques lignes sont pour exprimer ma reconnaissance :

à tous mes amis français et latinoamaricains qui m'ont toujours encouragé au long de mon séjour en France, tout particulièrement YO qui a été un soutien pour moi.

aux copains de l'equipe de travail avec lesquels j'ai trouvé l'aide malgré les barrières de la langue.

à Mr. Martinez pour m'avoir accuillit dans son equipe.

aux membres du jury pour avoir accepté de juger ce travail. A Mr. Meriaux pour ses suggestions dans la redaction de la thèse.

aux membres de la societé Getris-Images.

à l'equipe de reprographie de l'institut IMAG pour son excellent travail.

TABLE DE MATIERES

CHAPITRE II : L'ARCHITECTURE BANALISEE

CHAPITRE III : CONCEPTION DES MODULES DU POST PROCESSUS

INTRODUCTION

L'informatique graphique ou infographie a subit des développements importants dans ces dernières années. En effet, on sait d'une part que l'interprétation des informations est plus facile à réaliser lorsqu'on les présente sous une forme graphique (visuelle) que sous une forme purement textuelle. D'autre part, l'accroissement de champs d'application spécialisés tels que la CAO, FAO, l'architecture, la conception des dessins animés, etc. nécessite des interfaces homme-machine faciles à interpréter telles que des images.

Il est donc nécessaire de pouvoir synthétiser ces images.

On trouve dans la littérature des images synthétiques très saisissantes, ce qui fait penser que tous les problèmes relatifs à la synthèse d'images sont déjà résolus. Cependant, une étude plus approfondie sur ce sujet nous montre que le matériel et le temps de calcul utilisés pour arriver à ces images sont très importants.

Par ailleurs, des applications spécialisés nécessitent des structures particulières, ce qui demande des systèmes graphiques aussi spécialisés.

Le processus de génération d'une image se divise en plusieurs étapes de synthèse. Ces étapes sont caractérisées par des opérateurs de synthèse qui différents par leur construction (matérielle, logicielle) et leur fonction (description, modélisation, géométrique, aspect...).

On peut dire que c'est l'ordonnancement des opérateurs internes de

synthèse et leur puissance qui définissent les caractéristiques du système (processus de visualisation...) et son champ d'application.

Cela signifie que les systèmes graphiques, conçus sur une base matérielle fixe, ne sont utilisables que dans un cadre précis et assez rigide.

Cette rigidité matérielle impose une restructuration logicielle lourde, dès que l'on désire changer le champ d'application du système (ou le faire évoluer).

Le but d'une architecture banalisée, est d'assouplir l'ordonnancement des opérateurs pour permettre l'émulation de systèmes de puissance différents (bas de gamme, jusqu'au temps réel) et de donner ainsi au système une plus grande souplesse.

Le travail présenté dans cette thèse consiste à l'étude et la réalisation des opérateurs nécessaires pour faire évoluer un système basé sur une architecture banalisée. Un tel système a été réalisé par l'équipe de recherche sur la synthèse d'image du laboratoire ARTEMIS, au sein de l'institut IMAG. Nous nous sommes plus précisément intéressés aux opérateurs de synthèse de l'aspect final, notamment ceux concernant la couleurs, la texture des objets et l'éclairage de la scène. En particulier, le choix d'un haut degré d'intéractivité pour ceux-ci soulève des problèmes de temps de réponse très courts. Aussi une réalisation câblée a été retenue. Ce choix d'implémentation est original car à l'heure actuelle aucun système sur le marché ne propose d'opérateurs câblés sur l'aspect final.
Le travail que nous présentons est divisé en 6 chapitres.

Le **premier chapitre** propose les concepts de base liés à la synthèse d'images ainsi que l'évolution et l'état de l'art des systèmes de synthèse d'images.

Dans le **deuxième chapitre**, nous faisons un survol historique des travaux matériels réalisés par l'équipe de systèmes graphiques de l'Institut IMAG. Par la suite, nous présentons le système d'architecture banalisée qui est l'aboutissement des premiers travaux sur les terminaux graphiques.

Le **troisième chapitre** est consacré à la mise en évidence des différents opérateurs du post-processus. Nous utiliserons des exemples réels pour dégager les principaux opérateurs nécessaires dans ce post-processus.

Dans le **chapitre quatre**, nous présentons la réalisation des opérateurs intermédiaires correspondants au calcul des couleurs et des textures dans le post-processus. Les problèmes d'ordonnancement des différents modules sont également étudiés.

En ce qui concerne le **cinquième chapitre**, nous nous consacrons à l'étude de l'opérateur final de calcul dans le post-processus : le calcul de l'éclairage de la scène. Nous présenterons les différentes techniques et critères de réalisation d'un tel opérateur.

Finalement, dans le **chapitre six**, nous présenterons nos conclusions ainsi que les travaux futurs prévus pour l'amélioration du système.

CHAPITRE I : GENERALITES

1 - La synthèse d'Images

Depuis l'origine, le but des travaux de synthèse d'image sur ordinateur a été d'améliorer la communication homme-machine [Sut 63] [BIN 77]. L'affichage graphique est un moyen de présenter les résultats (d'un calcul ou d'un processus) que l'opérateur assimile plus facilement que des résultats présentés sous forme exclusivement textuelle. Il est vrai, donc, qu'une image vaut mieux que mille mots, néanmoins pour les informaticiens ceci est un défi de capacité de calcul : si on veut mieux afficher, on doit aussi plus calculer.

Le processus de synthèse d'une image peut-être représenté d'un point de vue fonctionnel comme une série de "sous-processus" et de représentations de données organisés d'une façon séquentielle ("pipe-line") [Mye 84], [Hui 84], comme on le voit sur la figure I.1.

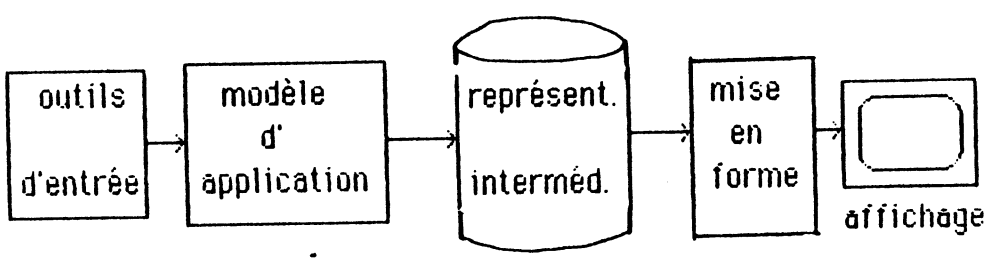


Figure I. 1 Modélisation du Processus de Synthèse

A partir du modèle d'application qui donne aux utilisateurs les dispositifs ad-hoc pour modéliser et caractériser les objets à synthésiser ainsi que pour les représenter d'une façon structurée dans des bases de données intermédiaires, le processus arrive finalement à les mettre sous forme convenable pour les afficher sur le moniteur de visualisation.

Une grande partie de ce chapitre va être consacrée à l'étude du processus de synthèse d'images (partant des informations élémentaires et passant par la modélisation d'objets pour arriver à l'affichage de l'image). Ensuite, nous traiterons l'état de l'art et des exemples de systèmes de synthèse trouvés sur le marché mondial.

1.1. Les Attributs de Base de la Synthèse d'Image.

Les attributs de base sont les informations les plus élémentaires qui caractérisent les objets à synthétiser. D'après [Mar 82], il existe six types différents d'attributs :

- Identité (I) : Cet attribut correspond à l'identification ou à la nomination des objets dans la scène à synthétiser.
- Morphologie (M) : correspond à la forme intrensèque des objets indépendamment de leur couleur, position, etc...
- Aspect (A) : Exprime l'apparence finale des objets.
- Géométrie (G) : regroupe les données permettant de situer les objets les uns par rapport aux autres et par conséquent agit sur leur taille, leur profondeur par rapport à l'écran, ainsi que sur la prise de vue de l'ensemble d'objets.
- Eclairage (E) : Cet attribut détermine les paramètres d'illumination de l'ensemble des objets de la scène et par conséquent prend en compte les sources lumineuses et les conditions de visibilité.
- Structure (S) : Détermine les différentes relations entre les objets, telles que l'appartenance, la proximité, le contact, etc...

Les caractéristiques d'un système de synthèse d'images dépendent de la façon de composer les attributs.

1.2. Le processus de Synthèse.

Nous diviserons en trois étapes fondamentales la modélisation logique du processus de synthèse : la description, la construction et finalement la visualisation de la maquette.

1.2.1. La description de la maquette.

Cette étape comprend le modèle d'application (figure I.1.) et par conséquent elle contient les dispositifs de saisie de données élémentaires (I, M, A, G) appliqués à chacun des objets, ainsi que les attributs jouant un rôle global dans la scène (E,S)

A partir de ces attributs, la maquette est structurée dans des bases de données intermédiaires.

Cette structuration peut-être formée d'une façon hiérarchique, ce qui permet d'appliquer plus facilement des transformations aux différents portions de la structure de l'image, surtout au moment de la modélisation de la maquette.

1.2.2. La construction de la maquette

Il s'agit ici de construire les objets à partir de la description de la maquette, à l'aide d'algorithmes de modélisation et de génération. Pour cela, on s'appuie sur la structure de la maquette dans les bases de données intermédiaires.

La modélisation des objets est généralement réalisée de trois façons différentes : à l'aide de "fil de fer", au travers de faces planes et/ou de faces gauches.

1.2.2.1. Les éléments Fil de Fer.

Cette technique représente les objets à l'aide de lignes ou arêtes. Ici la notion de taches pleines n'existe pas. Le coloriage des objets n'est possible que pour les contours. Malgré le manque de réalisme, cette technique, plus facile à mettre en oeuvre de point de vue calcul, est beaucoup utilisée. Mais si on veut améliorer le réalisme des objets, les autres techniques sont plus adaptées.

1.2.2.2. Les Faces Planes.

La face plane est l'élément de modélisation des objets (d'un point de vue réaliste) le plus simple et par conséquent le plus répandu. La technique consiste à approximer les surfaces courbes par des faces polygonales planes. La modélisation sera d'autant plus réaliste que le nombre de faces est élevé. Toutefois cela nécessite plus de mémoire de stockage et un temps de calcul plus long. De cette façon, les objets sont formés par de réseaux de polygones. (Fig. I.2.)

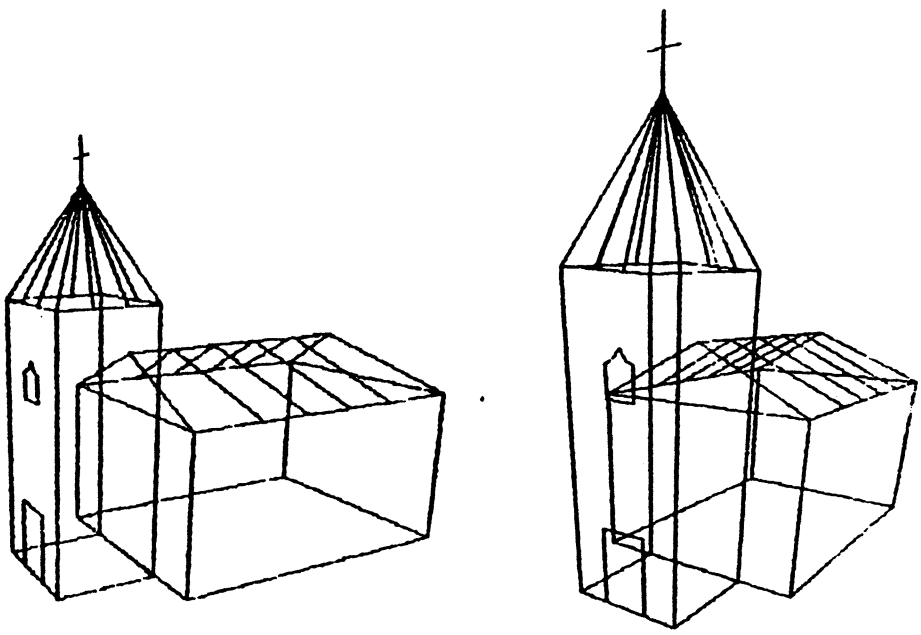


Fig I.2. Représentation d'un objet à l'aide de faces polygonales planes.

1.2.2.3. Les Faces Gauches.

C'est la façon la plus réaliste de représenter des surfaces courbes. Une surface gauche peut-être divisée en parties que l'on peut représenter à l'aide de fonctions paramètriques. La surface est décrite pour trois fonctions de "s" et "t" qui donnent les coordonnées spatiales (x,y,z), [Fo V 82], (voir figure I.3).

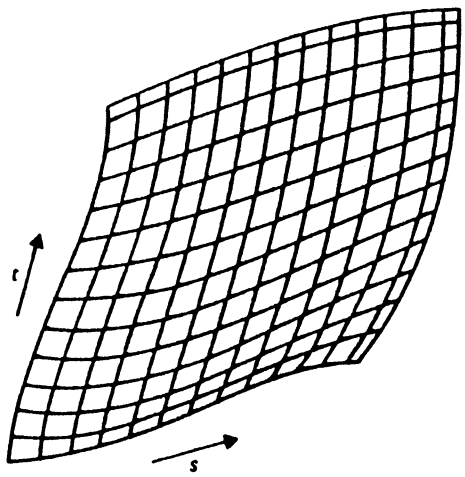


Figure I. 3 Surface représentée par une surface gauche

Le grand inconvénient de cette méthode est l'utilisation d'algorithmes plus compliqués et plus difficiles à mettre en oeuvre que dans le cas précédent. Toutefois des études pour la structuration de la scène visant à simplifier la complexité du calcul ont été réalisés : [Cla 76].

1.2.3. La visualisation de la maquette

Cette étape réalise la mise en forme des objets dans la maquette, pour finalement les afficher sur le moniteur de visualisation. Cette mise en forme comprend toutes les transformations géométriques appliquées aux objets telles que la rotation, la translation, l'homothétie, etc.

Une fois ces transformations faites, la prise de vue et l'affichage proprement dit sont réalisés.

1.2.3.1. La prise de Vue.

Il s'agit ici d'une information géométrique qui s'applique à l'ensemble de la maquette : [Gv], et qui est différente des attributs géométriques propres à chaque élément de la maquette [G].

La prise de vue effectue le passage de la structure 3D de la maquette (dans les coordonnées de l'utilisateur) à une représentation 2D dans les coordonnées propres au moniteur utilisé en guise de support de visualisation. Cette transformation nécessite un point de vue et une direction de visée. (figure I.4)

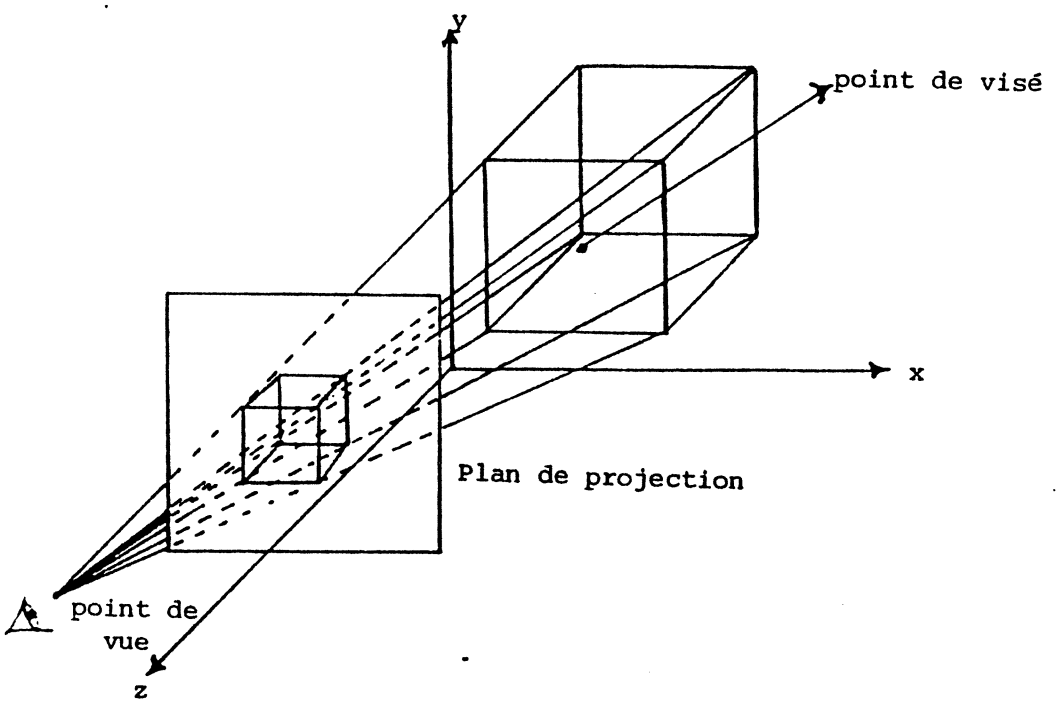


Figure I. 4. La prise de vue

Le plus grand inconvénient de cette transformation est la perte d'informations sur les objets qui peut donner lieu à certaines ambiguités au moment de leur interprétation. Pour résoudre ce problème on utilise cette étape pour éliminer des lignes ou des parties cachées.

1.2.3.2. L'affichage.

Cette dernière étape du processus de visualisation n'est plus vraiment une étape de synthèse ; c'est uniquement le cadrage de l'image à afficher par rapport au résultat de la prise de vue (fenêtre). Par conséquent l'affichage utilise aussi des informations géométriques (Ga), notamment les repères géométriques exprimées en 2D dans les coordonnées du

moniteur (clôture), comme on voit sur la figure I.5.

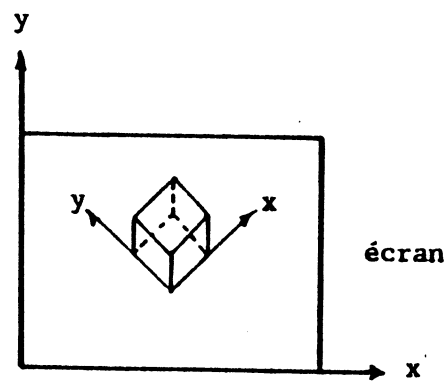


Figure I.5. Affichage final de l'image.

1.3. Les facteurs qui influent sur les systèmes de synthèse d'images.

A partir des informations de base, l'information finale à afficher sur le moniteur, subit plusieurs opérateurs de synthèse et de transformation. Ces opérateurs peuvent être conditionnés par les performanes finales souhaitées du système. On peut considérer trois facteurs importants dans la réalisation d'un système de synthèse d'images : le temps de réponse, le niveau de réalisme et la complexité de la scène à synthétiser.

1.3.1. La vitesse de réponse du système.

La réponse du système influe directement sur le niveau d'intéractivité. Ainsi, l'identification aisée d'un objet, à l'aide d'un moyen extérieur d'identification sur l'écran (photostyle, réticule), nécessite la sauvegarde des attributs d'identité et de structure (I,S) des objets au niveau le plus proche possible du matériel dans les structures de données intermédiaires.
De même, la vitesse de réponse nécessite des structures matérielles qui dépendent des opérateurs et de leur performances : deux systèmes ayant des temps de réponse de 40 msec en (temps réel) et d'une minute sont très différents. Ceci peut même influer sur l'architecture du système : séquentielle si la vitesse de réponse n'est pas prioritaire, parallèle ou en

pipe-line pour le cas de temps de réponse assez court ou même en temps réel.

De la même manière, chaque opérateur peut être réalisé avec différentes techniques : la technique logicielle microprogrammée pour des réponses lentes et la technique de logique câblée ou des circuits VLSI spécialisés pour le traitement à grande vitesse.

1.3.2. Le niveau de Réalisme de la scène.

Ce facteur influe directement sur l'ordonnancement des opérateurs, car plus on fait un traitement de l'information "au début" du processus de synthèse (qui est à l'origine d'un traitement fait par logiciel), plus réaliste est l'image obtenue. En effet, un traitement prématuré de l'information est d'autant plus réaliste qu'il est fait par logiciel, car cette technique peut prendre en compte tous les facteurs réels qui interviennent dans la synthèse d'une image (par exemple la réfraction de la lumière).

1.3.3. La complexité de la scène.

Ce facteur influe directement sur le nombre de données à traiter. Il est évident que la masse des calculs augmente dans la plupart des algorithmes de synthèse, notamment dans ceux de l'élimination de parties cachées. (cf. III. 2. 1. 3.).

D'autre part, la complexité d'une scène fait intervenir des objets modélisés différemment (c'est le cas d'un paysage où les nuages, arbres et décor en général peuvent mieux être modélisés à l'aide des surfaces gauches ; alors que une maison peut-être modélisée à l'aide des surfaces planes). Cette situation pose le problème de convergence, des éléments différents vers un seul processus d'affichage sur l'écran, surtout au niveau de l'architecture et de la réalisation des opérateurs au détriment de la vitesse de réponse du système.

1.4. L'organisation Hiérarchique [Mar 82].

Aprés la construction de la maquette, plusieurs éléments de base sont considérés (surfaces gauches, faces planes), pour lesquels il faudra des processus de visualisation différents.

Le traitement simultané de plusieurs éléments de base différents pose le problème de la convergence des différents processus à l'aide d'un opérateur de composition ; surtout au niveau de la synchronisation et de la compatibilité des éléments résultants, ainsi que par la multiplicité des opérateurs identiques qui peuvent se répéter dans les différents processus (figure I.6.)

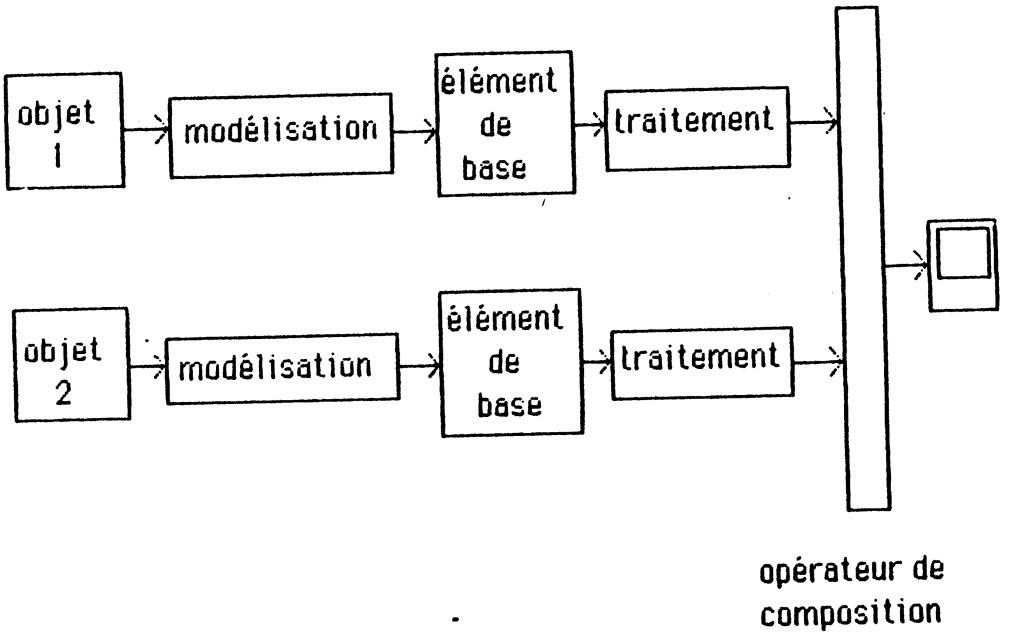


Figure I. 6. Convergence de différents processus.

On peut imaginer un systéme comportant une unité de contrôle et une banque d'opérateurs. L'unité de contrôle servant à ordonnancer les opérateurs, pour réaliser le processus de visualisation (Fig. I. 7).

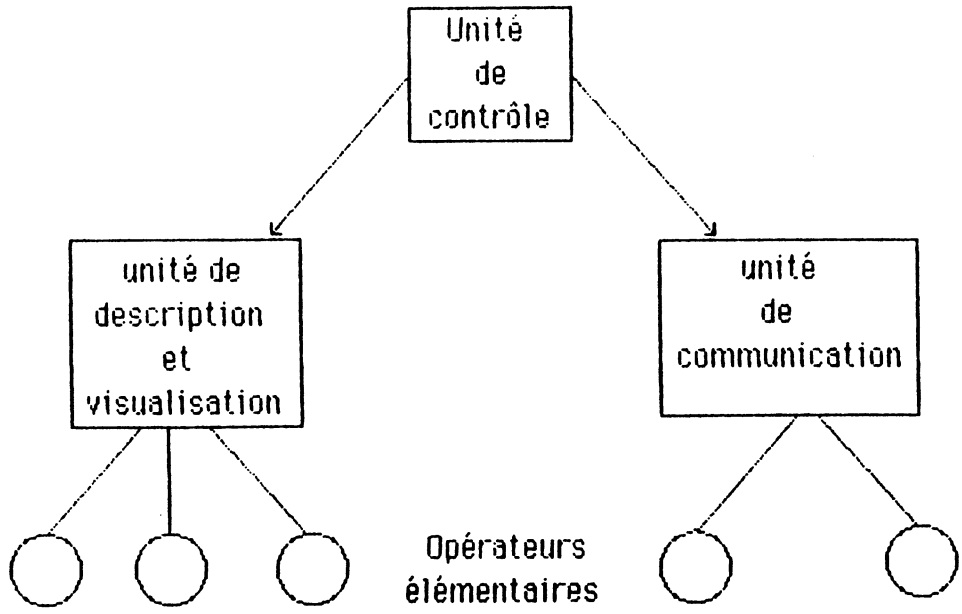


Figure I.7 L'organisation Hiérarchique. [Mar 82]

Ce type d'organisation est dit hiérarchique et il comprend un noyau de synthèse relié par des unités différentes, dont la plus importante est l'unité de contrôle.

1.4.1. L'unité de contrôle.

Elle est chargée de la communication entre les unités et surtout de la gestion de l'ordonnancement des différents opérateurs pour arriver à un processus de visualisation valide. En plus de la gestion des opérateurs, elle assure l'indépendance de l'application vis à vis des transports d'informations entre les différentes unités.

1.4.2. L'Unité de description et visualisation.

Cette unité assure l'indépendance vis à vis des dispositifs de saisie et d'affichage. Elle comprend les opérateurs de synthèse, de construction et de compostion. Ces opérateurs sont simulés par logiciel dans le cas d'un matériel bas de gamme.

1.4.3. L'unité de Communication.

De même qu'il peut y avoir duplication d'opérateurs dans des processus différents, il y a possibilité de duplication des données intermédiaires. Le but de l'unité de communication est la génération et la gestion d'une base de données intermédiaire unique. Par conséquent les opérateurs appartenant à cette unité concernent l'affectation, la structuration, la recherche ou la destruction dans cette base de données, des données graphiques.

L'approche suivie dans l'organisation hiérarchique soulève plusieurs problè mes ; au niveau logiciel, pour la gestion de ressources, mais surtout au niveau matériel pour lequel un réordonnancement des opérateurs doit être possible. Il va de soi que ce dernier est le plus difficile à mettre en oeuvre.

2. L'Etat de l'Art et l'Evolution de la Synthèse d'Images.

L'infographie apparaît dans les années 50, au moment où l'on relie le premier tube cathodique avec un ordinateur [Eve 52]. Etant donné son évolution, à l'heure actuelle elle peut être considérée comme une technologie de pointe.

Les premiers problèmes qui se posaient, consistaient à donner un réalisme aux objets à partir de quelques lignes. Pour les résoudre, des techniques de création d'images 3D en fil de fer ont été imaginées. Dans les années 60, le développement de l'industrie des simulateurs a augmenté le niveau d'intéractivité des systèmes de synthèse d'images, au prix d'un gros support matériel [Bun 82]. C'est dans d'autres domaines, où la rapidité de réponse n'est pas très pénalisante (CAO, FAO) que l'effort vers une amélioration du réalisme a été réalisé.
Il existe plusieurs raisons qui ont longtemps empêché le développement des systèmes graphiques [Maw 84] :

- Le coût élevé de calcul sur ordinateur.
- La complexité du logiciel pour la génération des objets.

- Les contraintes de temps de réponse.

Cependant, le développement des miniordinateurs, et plus récemment celui des microprocesseurs, a accru la complexité des systèmes ainsi que leurs performances [ChM 83].

2.1. L'Architecture des Systèmes de Synthèse d'Images.

2.1.1. L'Organisation.

Une représentation assez simple de l'organisation d'un système de synthèse est représenté par la figure I.8.

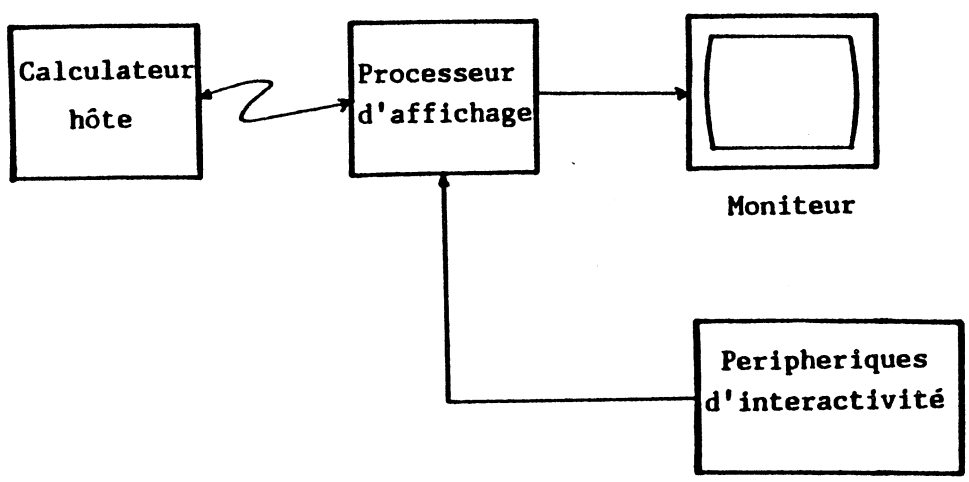


Figure I.8. Organisation simple d'un système de synthèse.

A l'origine, le calculateur hôte était un grand ordinateur, mais le développement des miniordinateurs au début des années 70 ont provoqué une évolution quasi-automatique. Maintenant avec les microprocesseurs les systèmes de synthèse sont devenus de plus en plus courants et autonomes, de sorte qu'au début des années 80, la notion de station de travail ("work station") a été développée.

Le processeur d'affichage reçoit les données du calculateur hôte pour les délivrer sous forme convenable au moniteur d'affichage. Toutefois c'est le moniteur d'affichage qui influe sur l'architecture du système. A l'heure actuelle, il existe 4 types de moniteurs différents appliqués à la synthèse d'images : les moniteurs à balayage cavalier, les moniteurs à tube mémoire, les moniteurs à balayage de trame et les moniteurs à écran plat. Tous sauf les derniers sont basés sur les tubes cathodiques (CRT).

Pour l'intéraction, il existe différents dispositifs d'entrée manipulables par l'utilisateur. Parmi les plus courants on peut mentionner : le clavier, la tablette à numériser, le manche à balai, le réticule, la souris, et même à l'heure actuelle la reconnaissance de la parole [Bih 84].

2.1.2. L'architecture avec moniteur à balayage cavalier.

Historiquement les premiers terminaux de synthèse furent associés aux moniteurs à balayage cavalier. (figure I.9.).

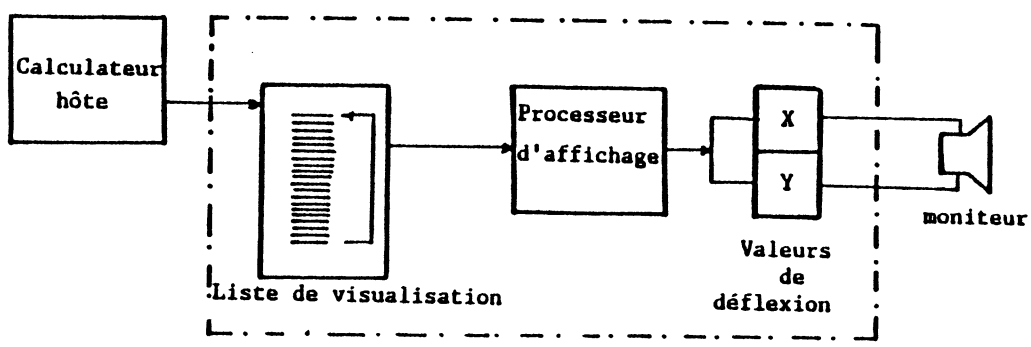


Figure I.9 Système avec moniteur à balayage cavalier.

L'image formée par des lignes, des points et du texte, est mémorisée par le calculateur hôte dans un buffer ou liste de visualisation sous forme de commandes.

Ces commandes sont interprétées par le processeur d'affichage qui transforme les valeurs numériques en valeurs analogiques pour contrôler les tensions de déflections du faisceau d'électrons dans le tube cathodique.

Le faisceau d'électrons trace directement les lignes sur la surface de phosphore; après quelques microsecondes, l'exitation disparaît et l'image aussi.

Le processeur d'affichage lit cycliquement l'information dans la liste de visualisation ce qui évite l'effacement de l'image. Toutefois, des problèmes de clignotage se posent lors des images trop chargées.
Pour les couleurs, on obtient au maximum quatre avec la technique de deux couches de pénétration de phosphore. (voir figure I.10).

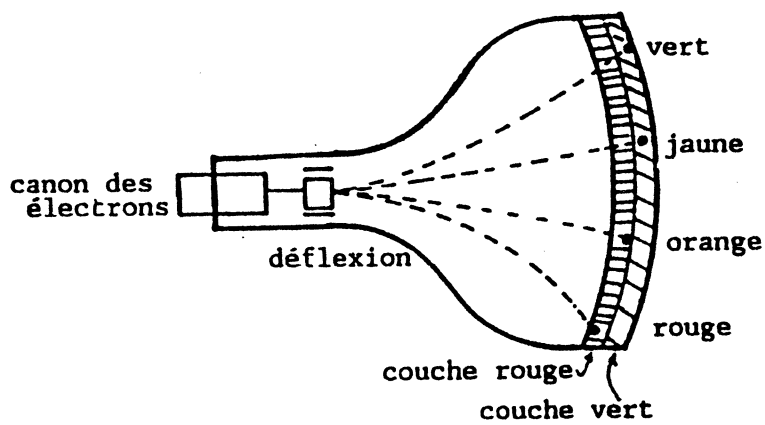


Figure I. 10. Tube à pénétration.

Les plus grands avantages de ce type de moniteurs sont : une grande définition d'écran (jusqu'à 4096x4096 points), et une grande vitesse d'affichage de lignes ; par contre les images sont toujours formées par des fils de fer avec peu de possibilités de taches pleines et avec un nombre toujours limité de couleurs.

2. 1. 3. L'architecture avec moniteur à tube mémoire

A la fin des années 60, Tektronix a présenté les premiers terminaux avec tubes à mémoire. Ce type de tube, basé sur le principe du

balayage cavalier, n'a besoin d'aucun système de rafraîchissement pour conserver l'image, un système basé sur un tel tube n'a besoin que d'une interface série pour communiquer avec le calculateur hôte. (figure I. 11)

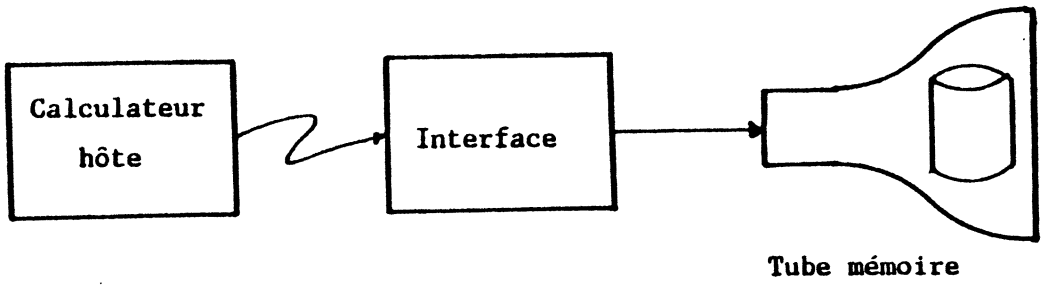


Figure I. 11. Système minimum à tube mémoire.

Après un seul balayage de l'image sur l'écran, ces tubes la conservent pour un long temps : ils comportent des canons auxiliaires qui arrosent tout l'écran avec des électrons de faible vitesse afin d'entretenir l'image sur le phosphore. (voir figure I. 12).

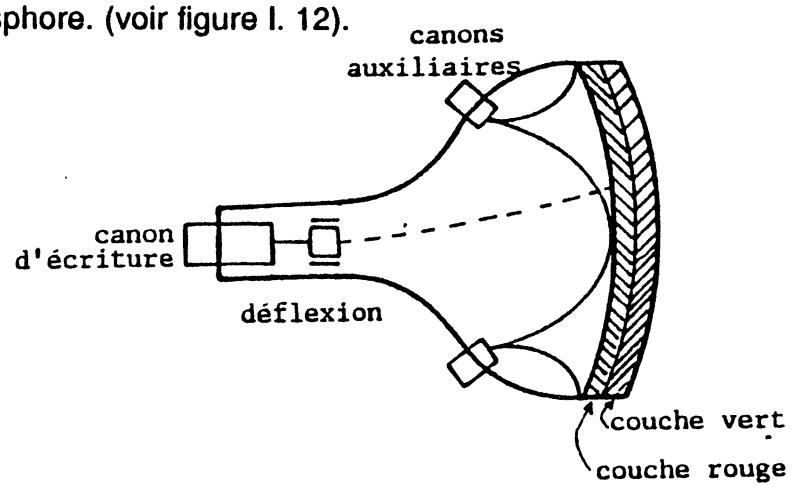


Figure I. 12. Tube à mémorisation à plusieurs couleurs.

Des problèmes se posent au moment de la régénération des images ainsi que pour obtenir plusieurs couleurs. Toutefois, une mémoire locale de régénération peut être ajoutée pour des générations sans faire intervenir le calculateur hôte. Pour les couleurs, une technique silimaire à celle des tubes à balayage cavalier est utilisée. [McC 84].

2. 1. 4. L'achitecture avec moniteur à balayage trame.

Les systèmes à balayage de trame commencent à prédominer au début des années 70. Ils ont le même principe de fonctionnement que les moniteurs T.V. habituels. Autrement dit, ils balayent l'écran de haut en bas et de gauche à droite cycliquement toutes les 40 msec au minimum. (voir figure I. 13).

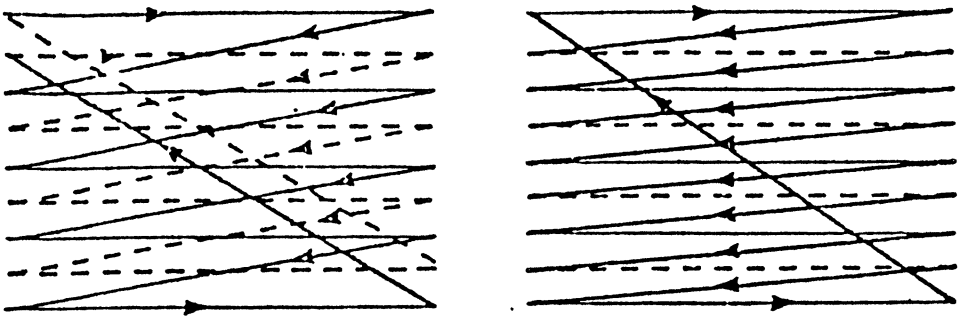


Figure I. 13. Balayage dans un moniteur trame.

Le grand inconvénient des systèmes à base de moniteurs à balayage de trame est la nécessité d'une mémoire d'image (ou mémoire de trame) additionnelle de la même taille et forme que la résolution de l'écran, pour des raisons de rafraîchissement de l'image. Pour une image de 512x512 points, chaque point ayant une définition de 8 bits, on a besoin de 256 K octets de mémoire. On comprend facilement pourquoi ce type de systèmes ne s'est vraiment développé qu'à partir de la sortie commerciale des mémoires RAMs permettant un rapport densité/prix favorable. (figure I. 14).

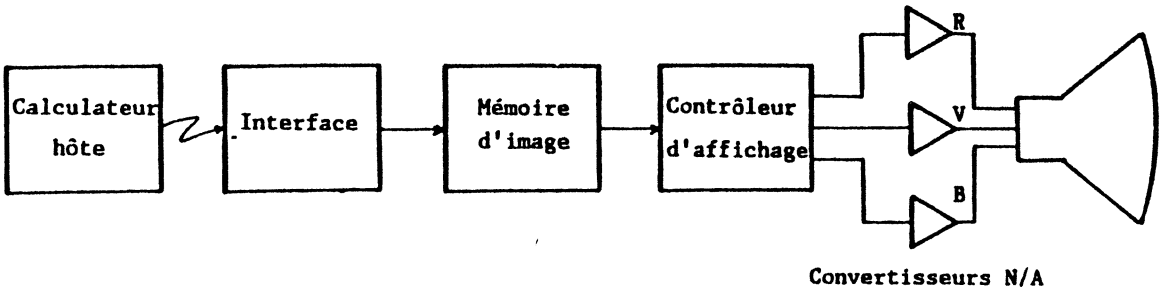


Figure 1.14 Système avec Moniteur à Balayage Trame.

Des couleurs sont produites avec la technique du tube à masque, qui dirige 3 faisceaux d'électrons pour chacune des couleurs primaires (voir figure I.15).

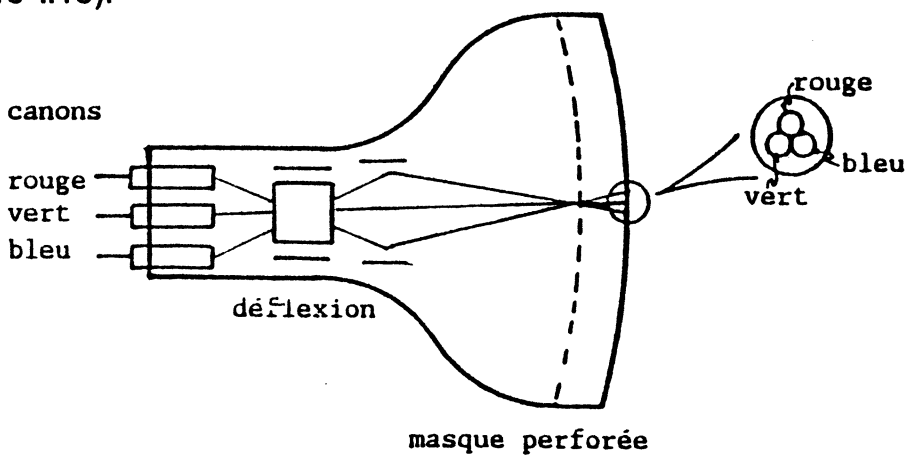


Figure 1.15. Tube couleur à masque.

Un avantage du système de balayage trame par rapport aux autres techniques étudiées auparavant est l'indépendance de la complexité de l'image à afficher, par rapport au rafraîchissement. Cependant, la production des images nécessite la conversion des différents formes (lignes, taches, cercles, etc...) ou pixels dans la mémoire d'image.

Pour l'intéractivité, le temps nécessaire au changement d'une image sur l'écran est déterminé par la vitesse de conversion et remplissage de la mémoire de trame, ainsi que la vitesse de transmission de données sur la ligne de connexion série entre le calculateur hôte et l'interface de remplissage de mémoire.

La définition de l'image, est lié directement à la résolution de l'écran du moniteur et pendant longtemps a été un handicap de ce type de système [Thi 83]. Des développements technologiques récents dans l'intégration de mémoires d'état solide d'une part [PNG 83], [MRS 84], et l'augmentation de la résolution des écrans d'autre part ont atténué ce handicap. A l'heure actuelle des moniteurs couleurs 2048x2048 pixels ont déjà été annoncés [Sha 85], et même des techniques pour augmenter la définition de l'image sans augmenter la résolution de l'écran sont déjà utilisées [Ase 84], [OSP 84].

2.1.5. L'architecture avec moniteurs à écran plat.

Ce type de moniteurs est caractérisé par une matrice d'éléments lumineux. Cette matrice est placée entre deux couches de matériaux transparent. L'allumage d'un pixel ou élément nécessite l'excitation de la colonne et de la ligne où il se trouve. La sélection du pixel nécessite en plus un grand nombre de connexions, (par exemple une matrice 512x512 points nécessite 1 024 connexions différentes).

Un grand avantage de ces types d'écran est sa capacité d'affichage permanente et par conséquent, des circuits ou des mémoires supplémentaires ne sont pas nécessaires. Un système de synthèse d'image à base d'un moniteur de ce style est similaire au système à base de tubes cathodiques à mémoire. Les moniteurs plats comme ceux à tube mémoire ont jusqu'à présent des vitesses d'écriture assez faibles. Nous présenterons les deux types les plus répandus et les plus appropriés pour la synthèse d'images : les écrans à plasma et les écrans à cristaux liquides.

Les moniteurs à plasma: Les moniteurs à plasma sont des matrices de petits tubes néon. Ces tubes se trouvent au milieu de deux couches de verre où se trouvent aussi collées les électrodes pour les colonnes et les lignes de la matrice.

Pour allumer un pixel (c'est-à-dire un tube) on applique une excitation à une tension élevée aux bornes du tube grâce à la ligne et à la colonne correspondante. De la même façon, une extinction du pixel nécessite une nouvelle excitation à un potentiel différent, ce qui rend les processus d'écriture et d'effaçage fort lents. Pour arriver à faire des couleurs, une technique de rafraîchissement est nécessaire.

Les moniteurs à cristaux liquides: Une matrice des cristaux liquides est déposée entre deux couches de polariseurs et d'électrodes. Lorsque la lumière traverse un polariseur, elle est réfractée et parvient jusqu'au second polariseur. Si on applique un

champ électrique au matériau liquide, cette réfraction ne se produit plus, et la lumière est totalement réfléchie. On voit donc une surface obscure sur un fond blanc. Le grand avantage de ce type de moniteurs est sa faible consommation d'énergie et son pouvoir de mémorisation, mais au prix d'une circuiterie de commande et d'adressage de pixels complexe et d'une faible résolution : 220 * 240 pixels seulement.

2.1.6. Evaluation des architectures de systèmes.

Il existe plusieurs critères de sélection d'un système qui se ramènent au choix du moniteur d'affichage utilisé comme support de visualisation. Ce sont : la résolution, l'intéractivité, la couleur, les taches pleines et le coût [Gro 84], [McC 84], [Bon 83], [TaG 78].

La résolution. La résolution, qui conditionne la finesse de l'affichage, dépend de l'application. Mais, en général une haute résolution signifie moins de clignotement, une bonne convergence des faisceaux d'électrons, moins de problèmes "d'aliasing", mais aussi un coût plus élevé. Les meilleures résolutions sont obtenues par balayage cavalier. En effet, pour le balayage cavalier, la résolution dépend uniquement du diamètre du faisceau d'électrons ; cependant, des clignotements peuvent se produire lorsque des images sont trop chargées.Les moniteurs à balayage trame et à écran plat ont une résolution faible, mais elle est indépendante de la complexité de l'image. Ceux à tube mémoire ont une bonne résolution d'image et ne posent pas de problèmes de clignotement.

L'intéractivité. Ce concept est lié au degré de dynamisme de l'image et à la facilité avec laquelle un utilisateur peut la changer. De par leur structure, seuls les moniteurs à balayage trame et à balayage cavalier avec rafraîchissement ont cette facilité.
La couleur. La couleur peut contribuer à diminuer la fatigue et augmenter la productivité ; elle est plus ou moins présente dans tous les types de moniteurs. Néanmoins, celui qui reste toujours en tête est le moniteur à balayage trame, pour lequel toutes les nuances de couleurs

peuvent être obtenues.

La capacité d'avoir des taches pleines. Ce facteur mesure la capacité de remplissage de surfaces et conditionne le réalisme de l'image. Dans ce cas, les moniteurs à écran plat et balayage trame prédominent largement.

Le coût. Il existe une relation directe entre le niveau de performances et le coût des moniteurs. Largement diffusés auprès du public, les moniteurs à balayage trame ont le meilleur rapport performances/prix.

Un moniteur idéal pour la synthèse d'images réalistes doit être peu encombrant (à écran plat), doit permettre d'obtenir toutes les nuances de couleurs et des taches pleines (balayage trame). Il doit en plus offrir une bonne résolution (balayage cavalier ou tube mémoire), avoir un prix intéressant (écran plat ou balayage trame), et permettre un bon dynamisme (balayage trame et balayage cavalier).

A l'heure actuelle, ce sont les moniteurs à tube cathodique qui prédominent sur le marché et parmi ceux-là, les tubes à balayage trame ont le meilleur rapport performances/prix pour la synthèse d'images.

Les paragraphes qui suivent tiendront compte de ce fait et nous parlerons surtout des systèmes à balayage trame.

2.2. L'évolution des systèmes de synthèse d'images.

2.2.1. Les systèmes bas de gamme et leur évolution.

Dans les années précédentes, d'importants développements technologiques ont amélioré les performances des systèmes de synthèse d'images à balayage trame ; partant d'une architecture assez rudimentaire, comportant une mémoire de trame, la logique cablée de rafraîchissement du moniteur et comme liaison au calculateur hôte un port série (fig. I.16).

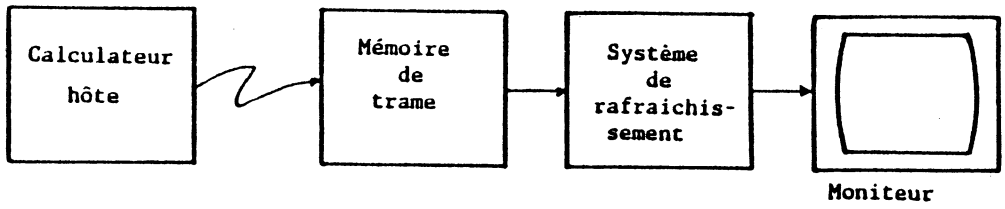


Figure I.16. Système bas de gamme.

Sur la figure I.16 ci-dessus, le calculateur hôte est chargé de manipuler les informations jusqu'à obtenir l'information finale du pixel. Cette information, niveau de gris ou de couleur, est mémorisée dans la mémoire de trame. Quelques fois, cette mémoire ne contient pas directement l'information de couleur, mais une identification qui est "post-traitée" à travers une table de couleur pour être finalement affichée sur le moniteur. La figure I.17 nous montre cette solution.

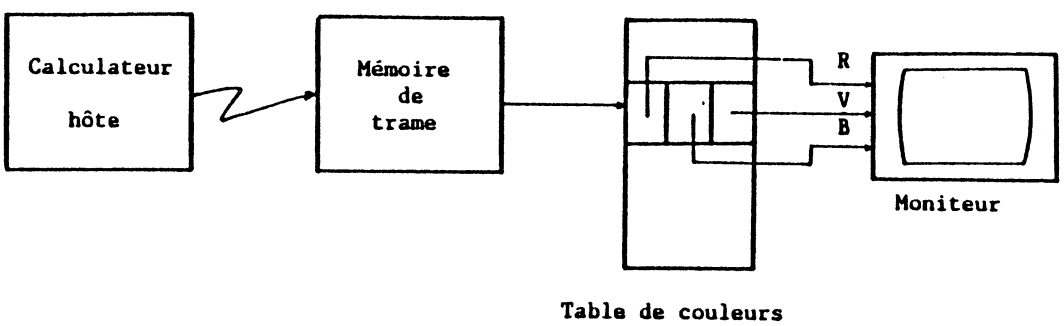


Figure I.17 Système bas de gamme avec transcodage de couleur.

Dans une image synthétique, on n'utilise pas simultanément toutes les couleurs, mais il faut des nuances. La mémoire de trame permet d'attribuer une couleur à chaque pixel : elle fait le lien d'un pixel avec une couleur de la table intermédiaire. Dans la mémoire de trame, chaque pixel correspond à un nombre fixe de bits. Plus il y a de bits plus le choix des couleurs est grand. Il est donc clair que plus ce choix est vaste, plus il faut de mémoire. La table de couleurs permet d'augmenter les nuances, mais pas le nombre de couleurs simultanément affichables qui est fonction du nombre de bits de chaque pixel.

Le gros problème se pose au niveau de l'interactivité. En effet, pour avoir une nouvelle image affichée, le calculateur hôte doit calculer à nouveau toute la nouvelle image et la transmettre à la mémoire de trame (ou mémoire d'image). Une solution adoptée couramment est l'addition d'un processeur graphique associé à une liste de visualisation. Cette liste contient les adresses des commandes graphiques transmises par le calculateur hôte ; le processeur graphique transforme ces ordres et génére l'image à remplir dans la mémoire de trame. (fig I. 18).

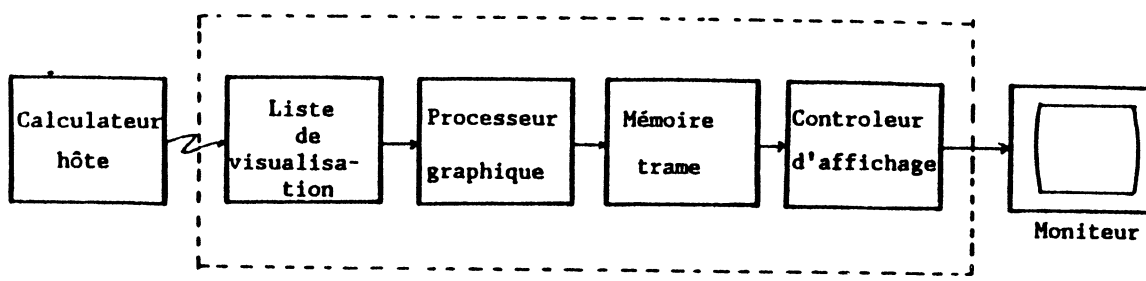


Figure I. 18. Système avec liste de visualisation.

Ici la vitesse de génération des images est partagée entre le calculateur hôte et la puissance du processeur graphique, toutefois il est essentiel que le calculateur hôte ait un accés direct à la mémoire de trame pour pouvoir la remplir en cas d'images déjà calculées. (figure I. 19).

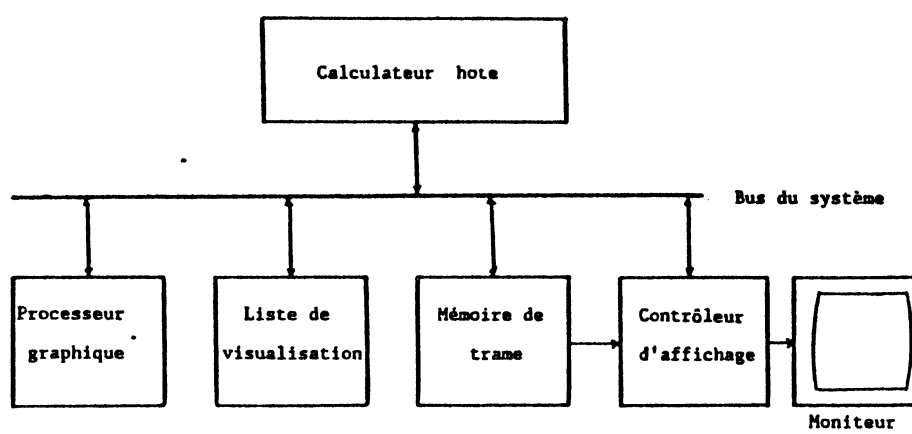


Figure I. 19. Système d'accés direct par le calculateur hôte.

D'autres problèmes se posent dans ce type de systèmes, par exemple

l'accés simultané de la mémoire de trame par le processeur graphique qui génére les images et par le contrôleur graphique qui fait le rafraichissement de l'écran. Une solution est l'addition d'une porte multi-entrées pour que les mémoires puissent être accédées simultanément par plusieurs unités. (Figure I. 20).

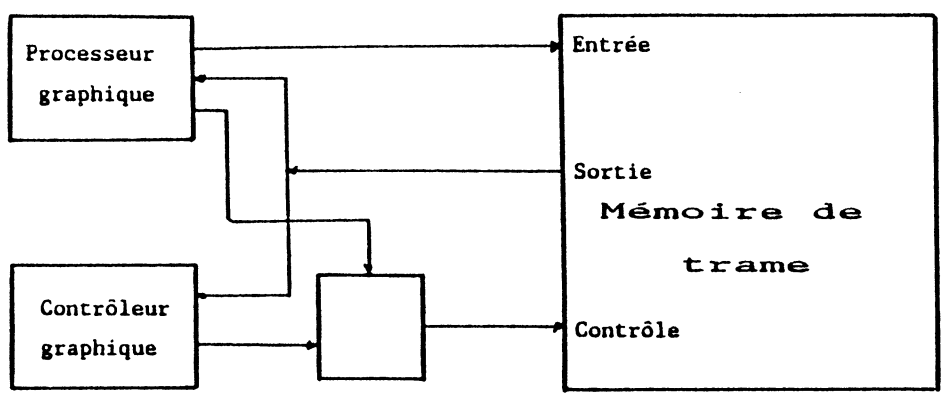


Figure I. 20. Système à plusieurs entrées dans la mémoire de trame.

Une autre technique pour résoudre ce problème est l'utilisation d'une double mémoire de trame : le processus graphique écrit dans la première tandis que le contrôleur lit la deuxième. Lorsque le contrôleur finit de parcourir toute la deuxième mémoire, il commence à lire la première et le processeur graphique commence à écrire dans la deuxième et ainsi de suite ; on arrive ainsi à un système d'un assez haut niveau de dynanisme. Dans le cas de remplissage de la mémoire de trame, il faut engendrer des images selon le format de trame (pixels), et par conséquent, des algorithmes de transformation sont nécessaires. Des points sont engendrés pour donner une approximation d'une ligne droite. Des arcs, des lignes brisées ou courbes, etc... sont engendrées de la même façon. [Bre 65], [NeS 79], [FoV 82], comme on le voit sur la figure I. 21.

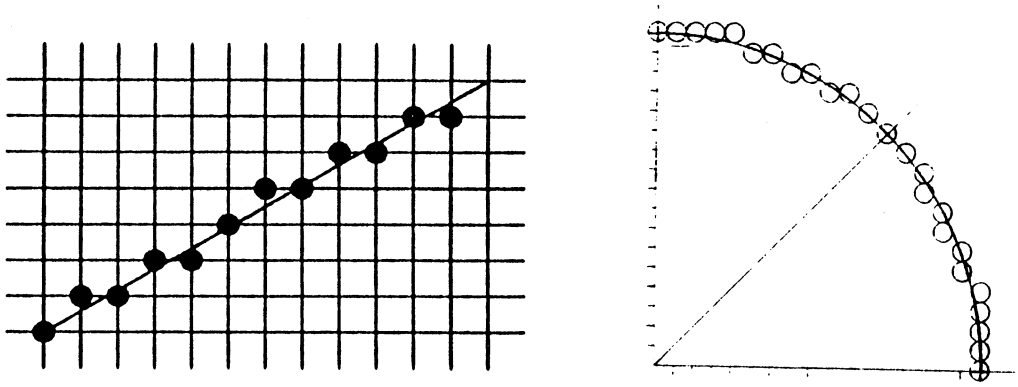


Figure I. 21. Génération d'une ligne droite et d'un quart de cercle.

De la même manière, la génération de taches pleines polygonales dans l'espace 3D nécessite l'utilisation d'algorithmes spéciaux de génération. (Figure I. 22).

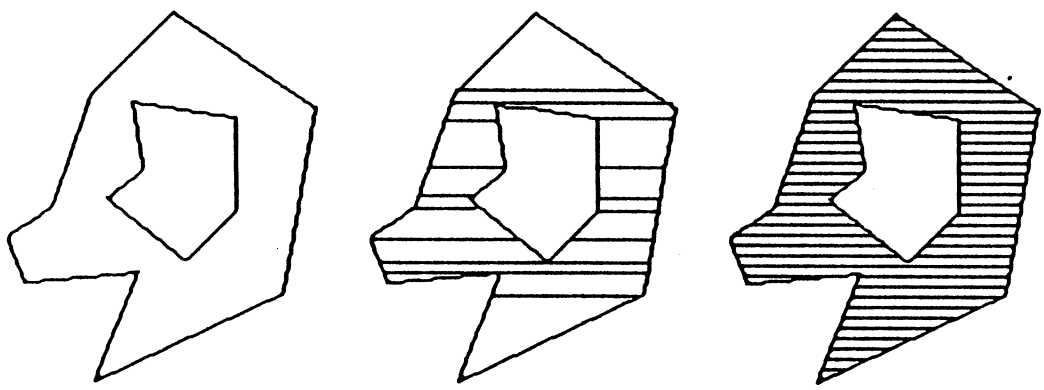


Figure I. 22. Génération et remplissage de tâches.

2. 2. 2. Les Systèmes évolués de Synthèse d'image

Plusieurs facteurs ont permis d'améliorer les architectures des systèmes de synthèse [Hub 84], [ChM83] :

- Le développement des microprocesseurs.

- Le développement et l'intégration des mémoires de masse (disquettes, disque dur) et de mémoires d'état solide RAMs.

- Le développement des circuits VLSI à la demande (custom, semicustom), de processeurs graphiques spécialisés pour la manipulation des images 3D en un temps proche du temps réel.

Dans le cas des images réalistes, les projections dans l'espace 3D sont nécessaires. Les transformations 3D, comme par exemple la translation, la rotation, l'homothétie des objets, prend normalement beaucoup de temps de calcul, car la méthode la plus simple est basée sur l'utilisation de coordonnées homogénes. [FoV 82].

Des matrices 4x4 doivent multiplier chaque chaque sommet du polygone (pour le cas de faces planes) ou tous les points (pour le cas de surfaces gauches).

Ce type de calculs (multiplications) est très pénalisant du point de vue logiciel, c'est pourquoi des modules spéciaux au niveau matériel sont ajoutés pour des systèmes à haut degré d'intéractivité (géométry engine, Ramtek, etc). Ces modules spécialisés sont intégrés au niveau du processeur graphique.

La liste de visualisation contient la structure 3D de l'image à afficher. Cette structure est traitée par les différents opérateurs de synthèse du processeur graphique pour obtenir la structure 2D à mémoriser dans la mémoire de trame (figure I.23). Les opérateurs concernent directement les attributs de base pour la synthèse : I, M, A, G, E (c.f. I.I.I.).

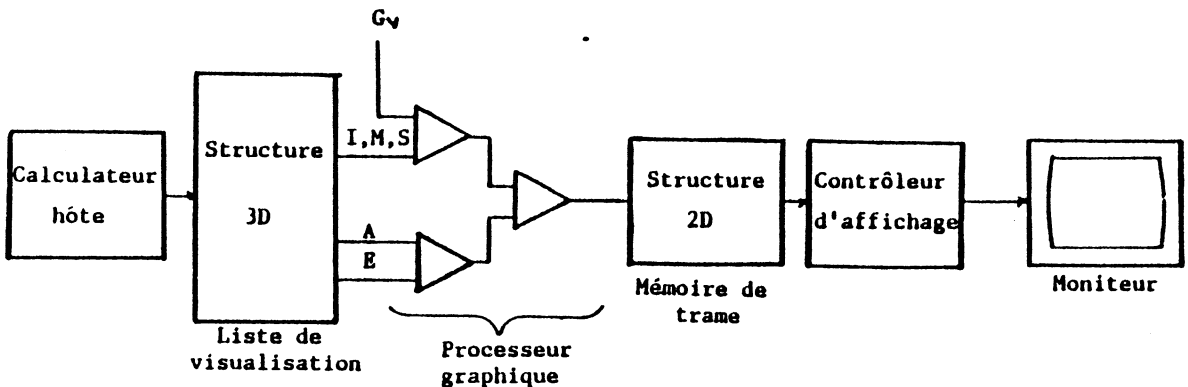


Figure I.23. Transformation de la structure de l'image à afficher.

Le transdocage de couleurs peut se faire au niveau d'un "post-traitement" après la mémoire de trame comme nous le montre la figure I.24.

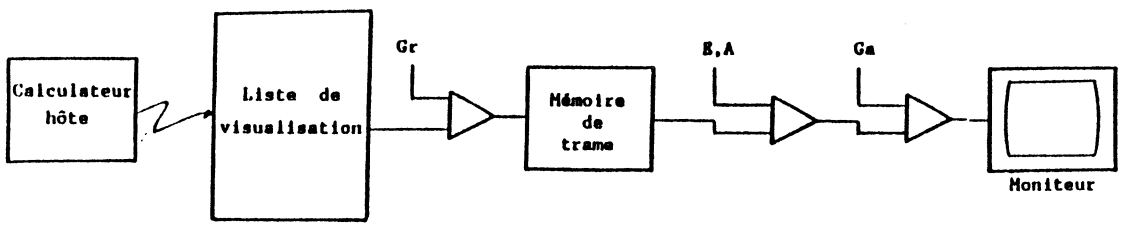


Figure I.24. Post-traitement de couleurs

De la même façon, le traitement de l'élimination de parties cachées par Z-buffer [cf. III.1.1.3], nécessite l'utilisation de plusieurs mémoires de trame. (figure I.25).

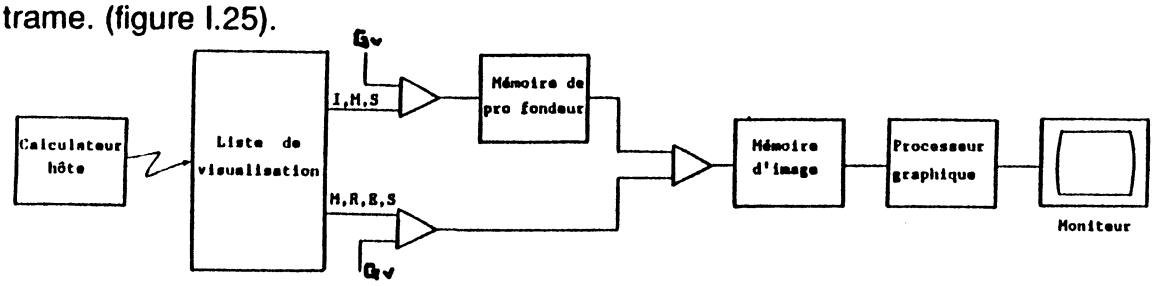


Figure I.25 Traitement du Z-buffer.

On peut simplifier notre étude de l'architecture de systèmes de synthèse d'images, en les modélisant (d'un point de vue matériel) en trois étapes différentes :

- Préprocessus
- Mémoires de trame
- Post-processus

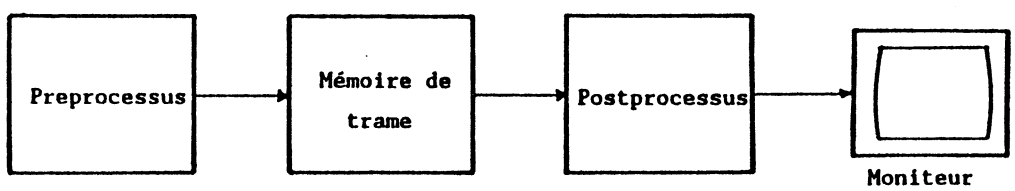


Figure I.26. Modélisation matérielle d'un système de synthèse.

La partie préprocessus comprend la description et modélisation de la maquette et par conséquent elle contient les dispositifs d'entrée nécessaires, ainsi que l'implantation des algorithmes pour engendrer des images au format trame. Par ailleurs, cette partie réalise les transformations géométriques de mise en forme aux coordonnées de l'écran pour pouvoir les enregistrer dans la mémoire de trame.

La partie de mémoire de trame reçoit : soit l'image 2D synthétisée par le préprocessus, soit une structure de cette image. Dans ce dernier cas, il faut un post-traitement. C'est ici que se présentent des conflits d'accès multiples, car le post-processus lit le contenu de cette mémoire en continu, pendant que le préprocessus met en forme cette information. Ce problème est résolu avec des techniques déjà citées auparavant (cf. I.2.2.1.)

La partie post-processus est chargée principalement de l'affichage de l'image et possède donc les moyens pour réaliser le rafraîchissement de l'écran à partir des informations mémorisées dans la mémoire de trame. Par ailleurs, elle réalise des traitements ou des effets spéciaux concernant la mise en forme de l'image pour l'afficher. Le post-processus dépend, donc, du type de données mémorisées dans la mémoire de trame.

2.2.3. Les stations de travail.

Une Station de Travail consiste en un ensemble complet de dispositifs pour la génération et la gestion d'une application graphique. Les stations de travail sont des systèmes autonomes qui prennent en charge toutes les étapes de la synthèse d'images.
Elle comportent :

- des processeurs puissants au niveau du calcul, tels que des microprocesseurs, à 16 ou 32 bits, des microprocesseurs en tranches et des coprocesseurs arithmétiques associés.
- des supports de mémoire de masse à haute capacité et à grande

vitesse tels que les disques dur winchester.

- des dispositifs de saisie d'informations comme les tablettes, les réticules, etc, et aussi des dispositifs de désignation et d'identification comme le photostyle.

- des processeurs câblés ou utilisant des circuits VLSI ("custom" ou "semi custom"), qui traitent les informations à haute vitesse comme : "geometry engine" ou "solid view" de lexidata, etc.

- des dispositifs de sortie en "hard copy" tel que le copieur ou les tables à tracer.

- des moniteurs à haute définition sur lesquels on visualise le travail en cours de réalisation.

Du point de vue logiciel, ces stations doivent comporter aussi :

- des modèles concernant le problème à traiter.

- des bases de données : bibliothèque de pièces, de projets, etc...

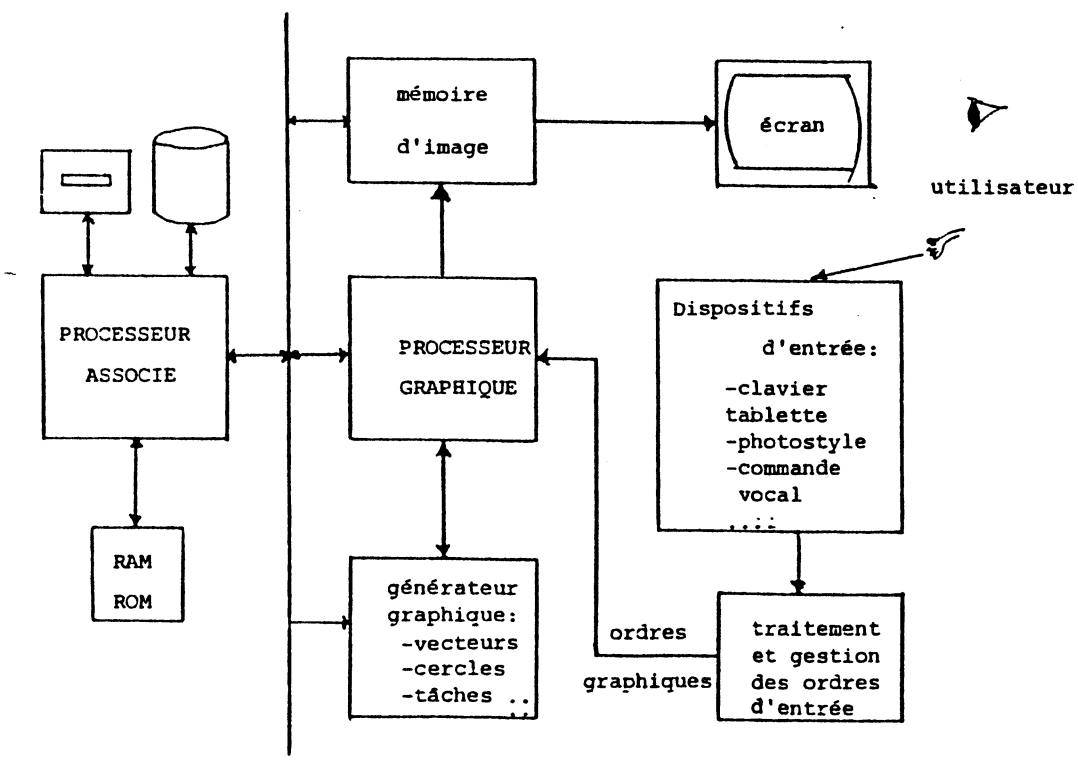


Figure I.27. Organisation d'une Station de Travail.

L'utilisateur fait partie de la boucle d'intéractivité d'un système. En effet, les données affichées sur l'écran sont assimilées par l'utilisateur qui en réponse manipule les dispositifs d'entrée : tablette, photostyle, etc. Les ordres graphiques ainsi donnés au poste de travail sont traités et gérés puis transmis au processeur graphique. Ce dernier décode des ordres et engendre à l'aide du processeur spécialisé les images à mémoriser dans la mémoire de trame qu'il affiche finalement sur l'écran du moniteur.

2.3 Exemples de systèmes.

2.3.1. Systèmes bas de gamme.

Dans ce domaine, on trouve les terminaux graphiques qui comportent un circuit VLSI comme processeur graphique ou contrôleur d'affichage, tel que le EF 9367 de Thomson ou le 7220 de Nec.

Ces circuits VLSI ont des facilités de fonctionnement qui permettent le tracé des points, cercles etc. ainsi que la génération de caractères alphamériques dans le même boitier.

Un système à base de ces circuits comporte :

- un ou deux processeurs (un micro-processeur et le contrôleur lui même).
- une mémoire d'image généralement de 512 x 480 ou de 640 x 512 points.
- une table réduite de couleurs en sortie.
- un convertisseur numérique/analogique.
- des dispositifs d'intéractivité tel que les claviers, des portes d'entrée/sortie, etc.
- 'un moniteur T.V. pour l'affichage.

La figure ci-après nous montre un système minimal à base d'un circuit contrôleur VLSI.

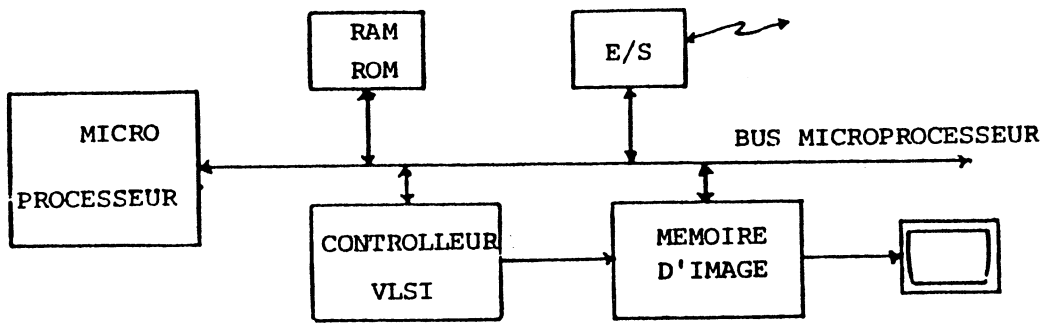


Figure I. 28. Système minimal à base d'un contrôleur graphique VLSI.

Par ailleurs, des nouveaux circuits du même style mais plus performants sont développés à l'heure actuelle, notamment la famille Am 815x d'Advance Micro Device qui assure l'interfaçage entre un microprocesseur et le moniteur de visualisation.

2.3.2. Systèmes haut de gamme.

Dans ce domaine, on trouve des véritables systèmes informatiques qui sont basés soit sur des circuits VLSI spécialisés ("custom" ou "semi-custom"), soit sur des cartes de processeurs graphiques assez évolués. Les caractéristiques générales de ce type de systèmes sont :

- des microprocesseurs 16 ou 32 bits comme processeurs généraux et graphiques chargés de la gestion des différentes parties de de l'ensemble du système et des échanges avec un calculateur hôte.

  Les commandes et la liste de visualisation sont aussi traitées par ces microprocesseurs.

- des microprocesseurs en tranche pour des tâches bien définies telles que le remplissage de la mémoire de trame.

- des circuits VLSI spécialisés pour la modélisation 3D des objets ou les transformations géométriques.

- des processeurs spécialisés câblés en logique rapide pour l'ombrage, l'élimination des parties cachées, l'antiallisage, etc.

- des mémoires de trame allant jusqu'à 2048 x 2048 pixels.
- un choix de couleurs allant jusqu'à 16 M.
-des moniteurs d'affichage avec une résolution d'écran de 1224 x 1024 points ou même plus.

Le cas le plus classique est celui du système **IRIS de Silicon Graphics Inc.** [Nic 84], [CID 83]. D'un point de vue conceptuel, le système IRIS est divisé en 3 unités qui font partie d'un processus en pipe-line : l'unité centrale de traitement, des VLSI "geometry engine" et l'unité de contrôle de l'affichage (mémoire d'image).

L'unité centrale de traitement est basée sur un microprocesseur motorola 68010. Ce microprocesseur est chargé de gérer la liste de visualisation, d'exécuter des programmes d'application, de contrôler les "geometry engine"et l'unité de contrôle d'affichage. Par ailleurs, toutes les extensions du MULTIBUS peuvent être associées au système.

La partie des circuits VLSI "geometry engine" réalise les transformations géométriques 2D et 3D. Un contrôlateur adjoint (un microprocesseur en tranche de 16 bits) réalise la génération de lignes tâches, etc. Ainsi, on peut arriver jusqu'à 65 000 transformations/sec. Finalement, l'unité d'affichage contrôle les plans mémoire qui peuvent contenir jusqu'à 34 bits de définition par pixel (fig I.29).

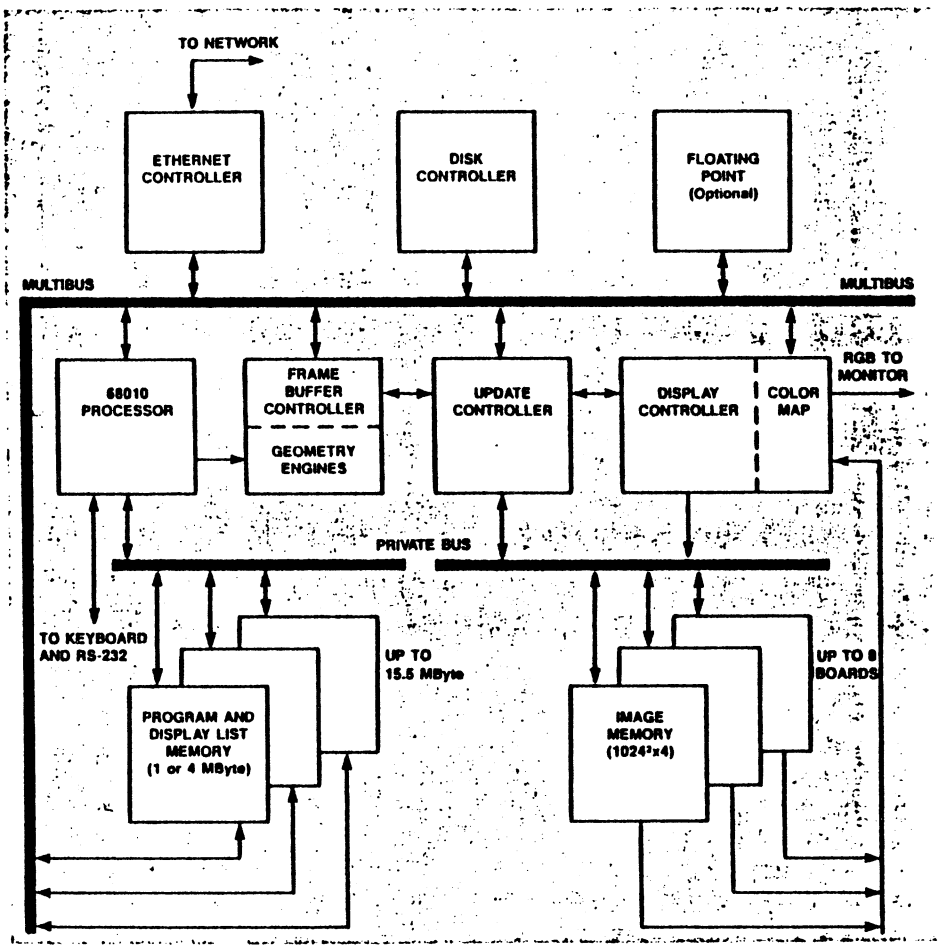


Figure I.29. Le système IRIS

Un autre exemple qui fait une utilisation intensive des processeurs et des circuits VLSI spécialisés est celui du système **SEILLAC-7** de Seillac Co. [IKE 84].

En effet, le système contient (figure I.30) entre autres :

- deux microordinateurs 16 bits
- un microprocesseur en tranche de 32 bits
- deux microprocesseurs en tranche de 16 bits.
- un multiplieur de matrices 4 x 4.

- cinquante circuits LSI "custom".

Toutes les unités sont rangées dans une configuration pipe-line avec une vitesse uniforme de traitement pour conserver une cadence constante dans la ligne de pipe-line de traitement.

Par ailleurs, des processus parallèles, notamment dans la partie de transformation géométrique assurent une vitesse de réponse du système élevée .

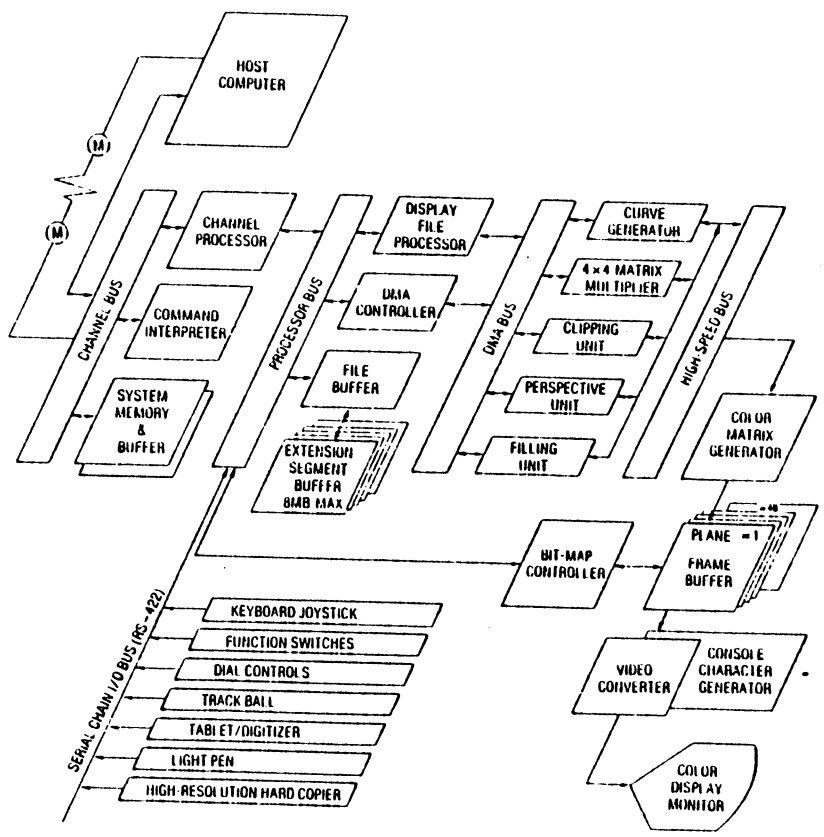


Figure I. 30. Système SEILLAC -7

Une caractéristique importante de ce système est l'utilisation d'une liste de visualisation hiérarchisée. Les informations concernant les transformations géométriques ainsi que celles de la modélisation et l'interrelation des objets sont ainsi structurées et enregistrées dans la liste. Cette solution permet, donc, des changements dynamiques sur les attributs des objets sans nécessiter des appels aux processeurs spécialisés de calcul.

CHAPITRE II : L'ARCHITECTURE BANALISEE

1 - Historique du projet Hélios

Les travaux réalisés au sein de l'équipe communication graphique du laboratoire ARTEMIS de l'Institut IMAG concernent principalement la mise en oeuvre d'outils graphiques tant logiciels (Clovis) que matériels (Hélios). Le logiciel Clovis, s'intéresse à une structure arborescente et banalisée des données graphiques [Mar 82], [GeG 85a], [GeG 85b].

Le projet Hélios qui a démarré au début des années 80, consiste en une mise en oeuvre matérielle d'un système de synthèse d'images réalistes. Ce projet a abouti à un premier prototype développé dans le cadre d'une thèse de Docteur- Ingénieur [Fer 81] : Hélios I.

Par la suite, ce travail a été repris dans le cadre d'une deuxième thèse de Docteur-Ingénieur [Chi 85] concernant l'amélioration des communications inter-processus : Hélios II.

Notre travail est consacré à la conception et l'implémentation d'une nouvelle version basée sur une architecture banalisée et qui vient d'être commercialisée par la Société Gétris-Images sous le nom de GETRIS.

1.1. Le prototype Hélios. I. [Fer 81] [MaF 82] [Mar 82].

Ce premier prototype considère comme élément de base la face plane définie dans l'espace 3D. 1024 faces maximum peuvent être synthétisées simultanément.

Ce système est basé sur une architecture à balayage trame, avec comme support de visualisation un moniteur T.V à la norme RVB (figure II. 1) sur écran 512 x 512 points.

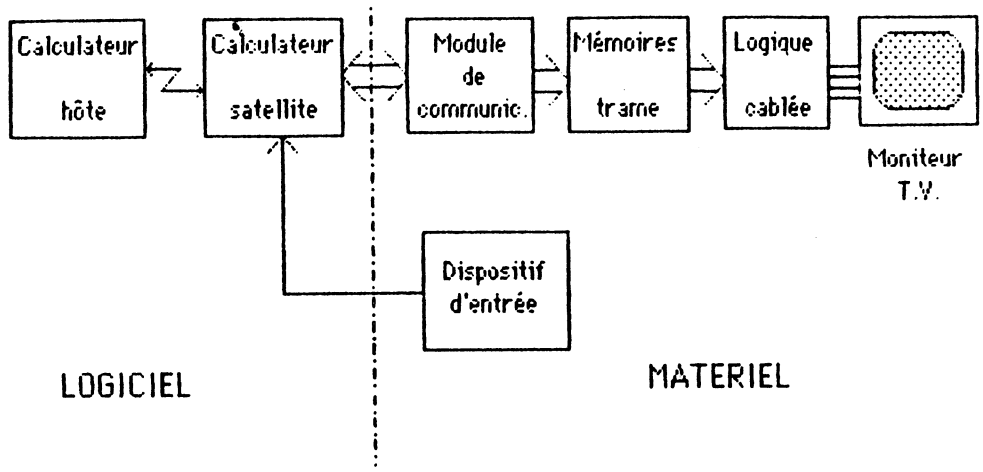


Figure II.1. Système Hélios-1.

Les informations morphologiques, géométriques et d'identification sont traitées par logiciel dans la partie du pré-processeur, (calculateur hôte et calculateur satellite) avec pour but final le remplissage des mémoires de trame. Ces mémoires ne contiennent pas l'information de couleur proprement dite, mais une identification de chaque face présynthétisée, on les appelle plans d'identification.

D'autre part, les informations d'aspect et d'éclairage et une partie des informations géométriques sont traitées par la logique câblée pour chaque point visible au rythme du signal vidéo. (voir figure II.2.)

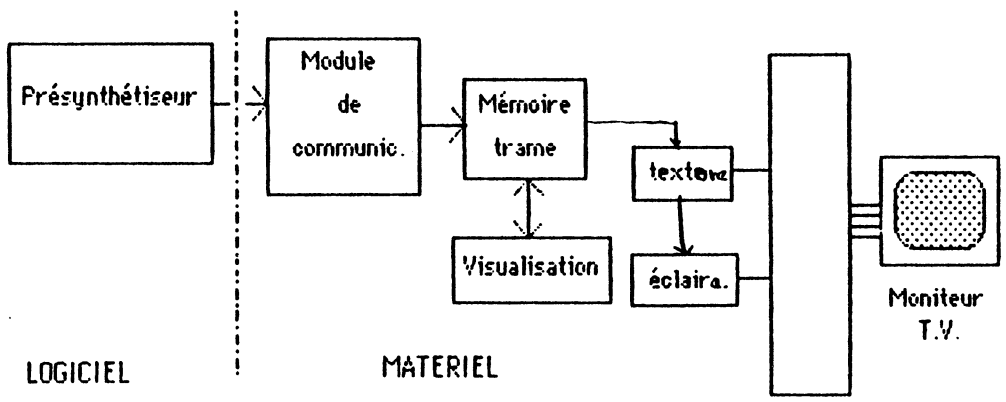


Figure II.2. Structure général d'Hélios-1.

On peut reconnaître deux modes de fonctionnement dans ce système :

1.1.1. Le mode de rafraîchissement de l'écran.

Cette partie correspond au post-processus du système. Elle calcule (au rythme du balayage vidéo) la couleur finale en chaque point à partir des données d'identification issues des mémoires de trame. Ce mode comporte des opérateurs qui effectuent en temps réel les taches suivantes :

-calcul visibilité/invisibilité des faces
-calcul des fenêtres de tailles distinctes sur chaque plan d'identification.
-calcul du pavage de texture sur les différentes faces.
-calcul de l'éclairage de la scène.

1. 1. 2. Le mode d'accès externe.

Cette partie correspond aux accès du présynthétiseur vers les mémoires du post-processeur (mémoires de trame, tables de correspondance) et en plus elle résoud les conflits d'accès externe en lecture et en écriture entre ces deux unités. Cette liaison est faite à travers un module spécial de communication.

La présynthèse composée du calculateur hôte (HB 68) et du calculateur satellite (micro engine 16 bits), est chargée de faire la synthèse des attributs M, G, I pour arriver à obtenir l'identification de toutes les faces visibles dans la scène, ainsi que les paramètres jouant un rôle complémentaire dans le calcul de la projection de textures et le calcul de l'éclairage.

Ces informations sont transmises via un port parallèle vers le module de communication qui les distribue aux différentes mémoires.

1.1.3. Evaluation du système

Bien que la partie du post-processus soit très poussée au niveau du matériel (tous les calculs de pavage de textures et de l'éclairage sont faits

en temps réel), la partie de communication avec l'extérieur est très pénalisée.

Cette pénalisation est le produit de deux facteurs :

- La transmission entre le calculateur hôte et le module de communication passe par un port parallèle qui doit être programmé à chaque fois qu'on accède à la partie câblée du post-processus, (même en utilisant la facilité de compactage de données du module de communication). Un transfert nécessite 3 accès au port parallèle :
  1) adresse, sens de transfert.
  2) acquitement de la logique cablée.
  3) lecture/écriture d'une donnée 16 bits.

- L'utilisation d'un langage de haut niveau (pascal) pour la réalisation des algorithmes de génération (lignes, taches, etc) [Sar 82], diminue le débit de transmission des données vers le post-processus.

A l'heure actuelle ces deux contraintes font que la vitesse de transmission de données est de 1.2. M bits/sec, et que la génération de vecteurs soit faite à la vitesse de 0.7 M points/sec et celle des tâches au rythme moyen de 2 tâches/sec. [Sar 82].

1. 2. Le prototype Hélios II .

Pour pallier à la faible efficacité des communications du "calculateur satellite", une seconde version d'Hélios a été construite autour d'un présynthétiseur "micro-programmé". Dans un premier temps il a été réalisé à partir d'un microprocesseur 8 bits : (6809 Motorola), puis étendu à un microprocesseur 16 bits (68000 Motorola ou 8086 Intel).
Ces améliorations ont conduit à une version de "Console évolué" d'Hélios réalisant :

- Une optimisation des échanges entre le pré-synthétiseur et le post-synthétiseur en supprimant la transmission d'informations redondantes. Par exemple, deux points ayant la même coordonnée en x ou y n'auront besoin que la transmission une seule fois de cette coordonnée si le points sont

adjacents.

- Une normalisation du dialogue entre le calculateur hôte et le microprocesseur.

La figure II. 3. nous montre l'organisation générale de cette deuxième version.

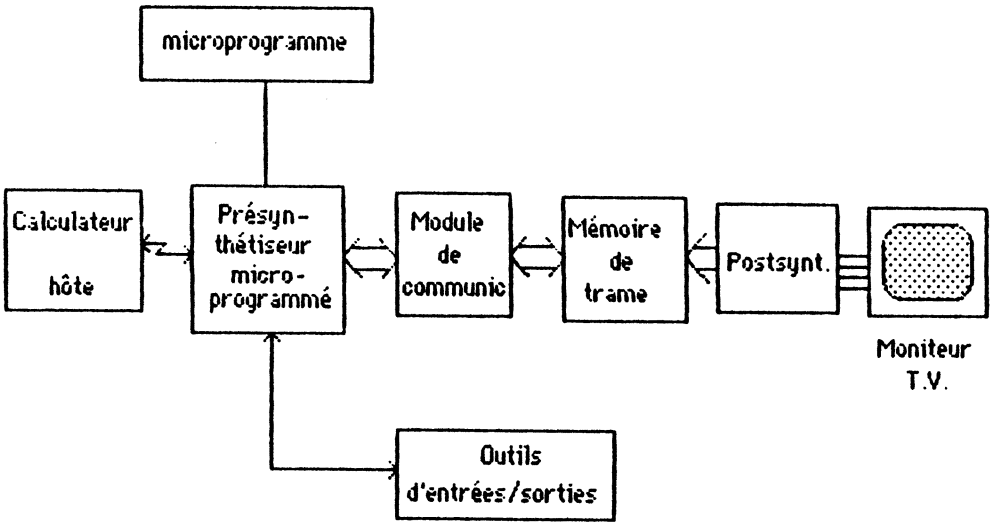


Figure II.3. Système Hélios-2.

On peur noter aussi comme dans la version précédente deux modes de fonctionnement :

I. 2. 1. Le mode rafraichissement

Ce mode correspond toujours au post-processus et réalise en temps réel (à partir des identifications issues des mémoires de trame) :

- Le calcul de visibilité/invisibilité de faces,
- Le calcul de fenêtres de différentes tailles,
- Le calcul de la couleur de la face (la notion de texture n'existe plus).

1.2.2. Le mode d'accès externe.

Ce mode concerne les exchanges entre le présynthétiseur microprogrammé, les mémoires de Trame et les mémoires du post-processeur. Ces exchanges sont effectués à l'aide d'un module de communication. Ce dernier est vu par le présynthétiseur comme un ensemble de 8 registres de 16 bits dans l'espace d'adressage du microprocesseur (permettent un accès direct). Un certain nombre de registres spéciaux ont été mis en oeuvre tel que :

- le registre "répétition" : pour faire une compression de données transmises.
- le registre "incrémentation" pour faire des incrémentations automatiques des adresses en x ou y au moment des lectures/écritures dans les mémoires.
- le registre "masque" : permet de définir des pseudotextures spéciales de tracé : vecteurs, continu, pointillé, etc.

1. 2. 3. Evaluation du système.

En ce qui concerne la partie du post-processus, il y a des changements appréciables par rapport à la version Hélios-I, notamment pour les communications externes.

En effet, la mise en oeuvre des nouveaux registres et surtout l'exclusion du port parallèle pour l'interface du module de communications a permis une amélioration des transferts de 80% (évaluation pour la version microprocesseur 8 bits).

L'écriture des algorithmes de génération (lignes, taches, etc...) en langage assembleur et l'utilisation des registres "incrémentation", "répétition" et "masque" ont permis une amélioration significative des vitesses de géneration.

Pour les lignes droites le gain par rapport à la première version est de l'ordre de 50% et pour les tâches pleines à 4 sommets de plus de 100 % (actuellement on arrive à générer de l'ordre de 5.25 Ktâches/sec [chi 85].

2. L'architecture banalisée d'Hélios : GETRIS [Mar 84] [Zar 84]

2. 1. Caractéristiques des opérateurs de synthèses

Le processus de synthèse d'une image fait appel à plusieurs opérateurs pour combiner les attributs initiaux. [cf.I.1.1.]. Ces opérateurs conditionnent les performances du système.

D'une façon générale, on peut classer selon trois types les opérateurs élémentaires de synthèse :

a) Les opérateurs de calcul proprement dits, qui fournissent pour chaque entrée des sorties synchrones. Par exemple les opérateurs de transformation géométrique (rotation, translation, etc...), qui reçoivent en entrée des données morphologiques qui se retrouvent aussi en sortie. (fig. II. 4). M'=f(M,G) :

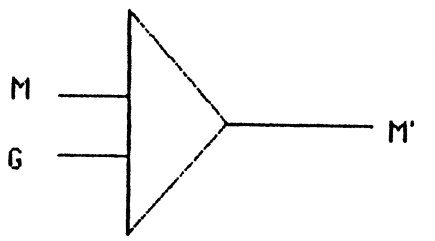


Figure II.4. Exemple d'opérateurs de calcul: application d'une transformation géométrique G à une forme M.

Un autre exemple est la modulation de l'aspect (A) d'un objet par rapport aux conditions d'éclairage de la scène (E) et qui donne comme résultat un autre aspect (A'). (Fig. II. 5)

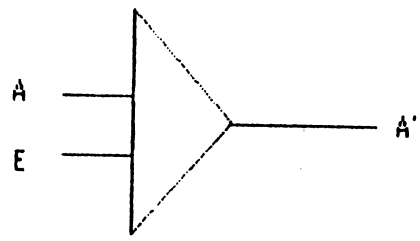


Figure II.5. Exemple de modulation d'aspect par rapport a l'éclairage.

b) Les opérateurs de mémorisation synchrone qui fournissent pour chaque entrée un résultat en sortie. Dans ce cas il existe une relation associative entre l'information d'entrée et l'information de sortie. L'exemple le plus commun est l'opérateur de correspondance entre deux informations. Ainsi, la normale d'une face polygonale plane peut être trouvée à partir de l'identification de cette face : N=f(I). (Fig. II. 6).

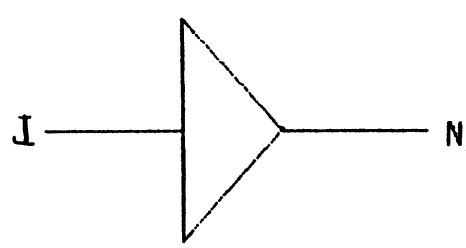


Figure II.6. Opérateur de correspondence N=f(I).

Ces types d'opérateurs sont de simples mémoires vectorielles dites " tables de correspondance" et leur taille est fonction du nombre d'informations fournies en entrée. (Fig. II. 7).

c) Les opérateurs de mémorisation asynchrone qui jouent le rôle de mémoires "tampon" entre deux processeurs différents. L'exemple typique est l'utilisation des mémoires d'image entre le processeur graphique et le processeur d'affichage. Ce sont des mémoires de trame qui contiennent

les informations projetées point par point dans le plan de l'écran et pour lesquelles, la position géométrique X,Y, établit la correspondance avec les coordonnées écran. (Fig. II. 8).

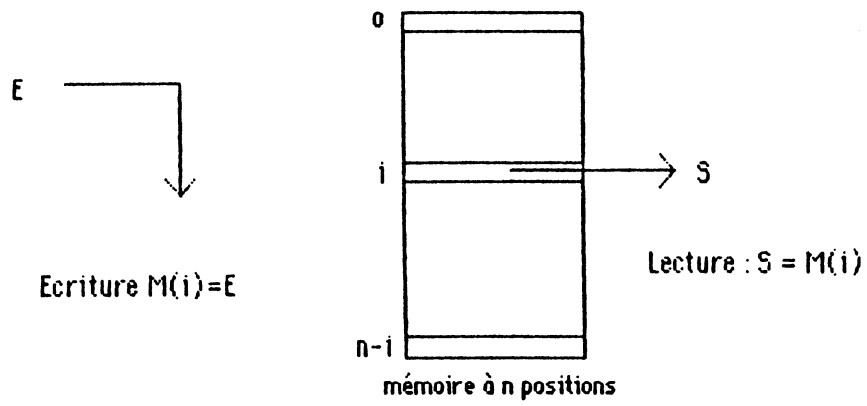


Figure II.7 Représentation d'une table de correspondance.

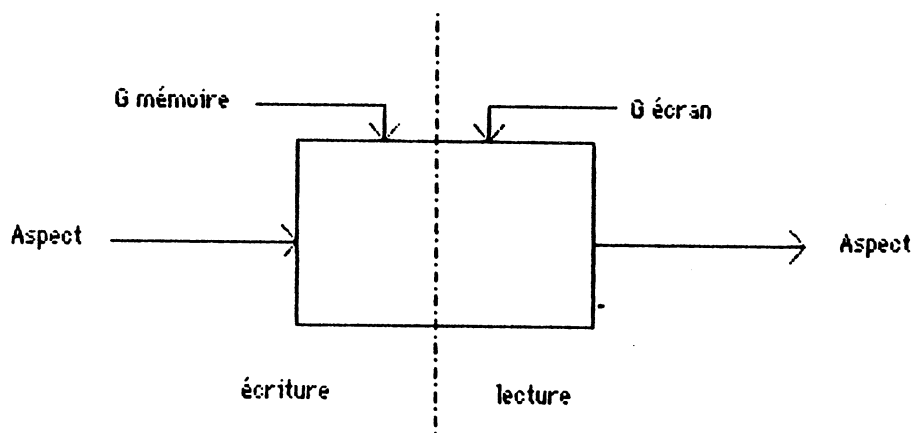


Figure II.8. Exemple de mémoire tampon asynchrone:
la mémoire d'image.

Ce type d'opérateur de mémorisation dit "Plan mémoire" est indispensable pour tout système à balayage trame.

On peut en déduire facilement que les opérateurs type "table de correspondance" et "plan mémoire", en tant qu'entités de mémorisation,

peuvent constituer deux modules uniques pour émuler différents opérateurs.

Ainsi, par exemple, une table de correspondance peut devenir soit une table de couleurs, soit une table de coefficients de réflexion de la lumière, etc..., en modifiant le contenu de la mémoire et sa place dans l'ordonnancement du processus de visualisation.

Dans le cas des plans mémoire, leur contenu peut être assimilé à des normales pour les surfaces gauches, à la profondeur des pixels pour l'algorithme d'élimination de parties cachées par Z-buffer, etc...

Cette approche nous amène à considérer ces deux types d'opérateurs comme "Opérateurs banalisés" d'un point de vue logique.

2. 2. Modularité du système

Aux trois types d'opérateurs logiques composant un système de synthèse, correspond un module indépendant pouvant être banalisé dans le cas des opérateurs de mémorisation (synchrone ou asynchrone). Cette approche nous suggère que la construction d'un processus de visualisation correspond à un ordonnancement de ces différents modules. Les performances du système seront donc déterminées par les performances des modules.

L'ensemble des modules banalisés, plus les différents modules de calcul, constitue le coeur de ce que que nous appelons le "Système d'architecture banalisée". La modularité de ce système est l'une de ses caractéristiques la plus remarquable. La conception et la mise au point de l'architecture banalisée ont été les points les plus délicats lors de la réalisation de ce système.

Mon travail de recherche a concerné principalement la définition et l'ordonnancement des différents modules du post-processeur de synthèse.

L'architecture banalisée en tant que système de synthèse d'images est organisée autour des modules suivants :

- Le module du Processeur graphique.

- Le module de Communication.

- Le module Vidéo.

- Les modules Plan Mémoire.

- Les modules du Post-processus.

Ces différents modules sont placés dans un environnement spécial qui a comme support des bus banalisés (c'est-à-dire qu'ils servent à plusieurs types d'information (c.f. II.2.3). La banalisation de l'architecture consiste alors à réordonner les différents modules à l'aide des bus banalisés. La figure de l'annexe A nous montre le schéma global de l'architecture banalisée.

Par la suite les paragraphes qui suivent traitent de ces modules.

2. 2. 1. Le Module du Processeur Graphique.

Grâce à son architecture banalisée, le système GETRIS peut émuler des systèmes à performances différentes; ainsi le simple terminal bas de gamme peut devenir une puissante station de travail avec des fonctions temps réel.

Le module du processeur graphique est chargé de faire l'émulation des différents opérateurs de présynthèse à l'aide (éventuellement) du calculateur hôte pour former la partie du préprocessus de synthèse. (Fig. II. 9.).

Cette pré-synthèse correspond à la description et modélisation de la maquette à visualiser (cf. Chapitre I), et elle a comme but final le remplissage des "Plans Mémoire" avec le modèle résultant.

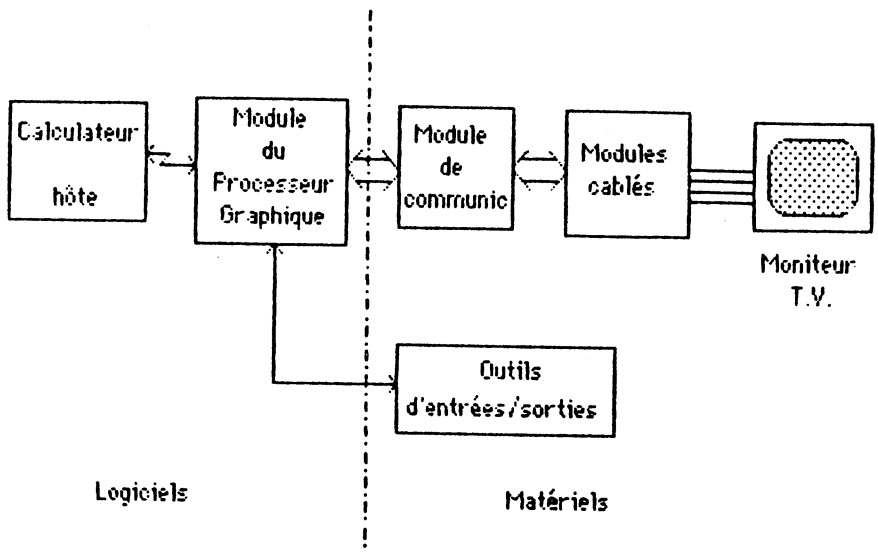


Figure II.9. Le système GETRIS.

Par ailleurs, ce module gère l'ordonnancement du processus de visualisation en programmant les différents modules ainsi que les opérateurs émulés par le logiciel.

Le processeur graphique est constitué d'un puissant microprocesseur 16 bits : Intel 8086 avec un co-processeur 8087 de la même famille pour les calculs arithmétiques. Le microprocesseur, en plus de son environnement classique, possède des options matérielles et logicielles permettant d'évaluer la puissance finale du système. La figure II.10. nous montre le module du processeur graphique.

ıL'unité de disquette et le disque dur permettent d'avoir une capacité supérieure à 10 M octets de mémoire de masse (cette mémoire a une capacité optional de 52 M).

En ce qui concerne les différents modules de l'architecture banalisée, ils sont vus comme un ensemble de registres par le processeur graphique. Ces registres sont placés dans l'espace d'adressage du microprocesseur pour permettre un accès direct.

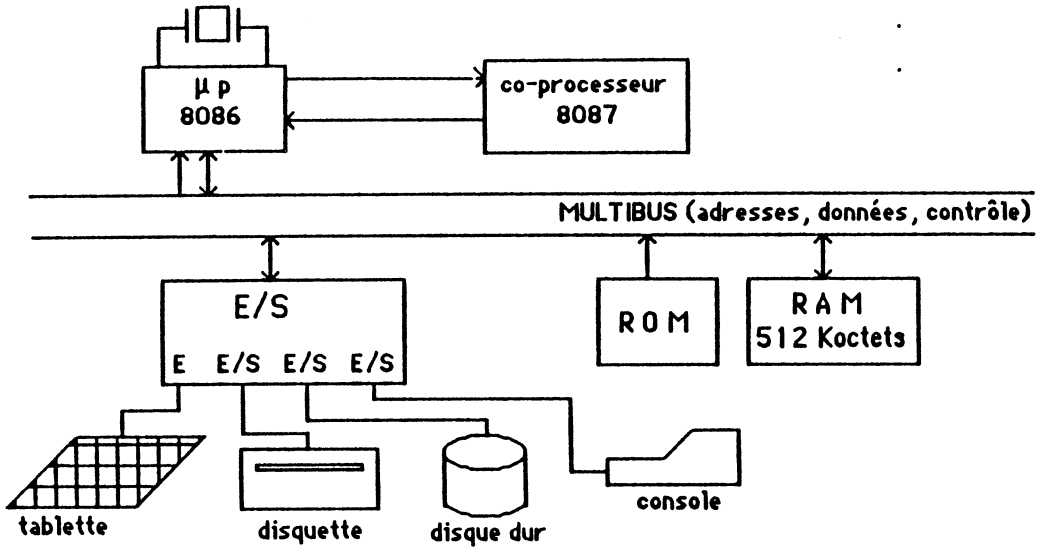


Figure II.10. Processeur graphique de GETRIS.

2. 2. 2. Le module de Communication.

Le module de communication est chargé de synchroniser les accès du processeur graphique aux différents modules de l'architecture banalisée (Fig. II. 11). Le processeur graphique accède un registre spécial du module de communication pour vérifier que le transfert est possible.

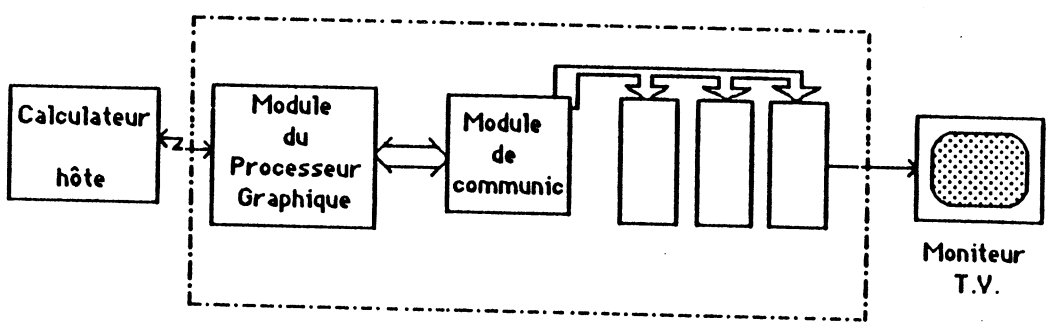


Figure II.11. Rôle du module de communication.

Le module de communication prend en charge le bus de données, le bus d'adresses et le bus de contrôle vers tous les autres modules. Il comporte trois parties correspondant au contrôle du module lui-même, à la lecture et à l'écriture dans les différents modules. La figure de l'annexe B nous

montre le schéma de ce module. On peut distinguer trois parties principales:

- **La partie contrôle**. Chaque module possède sa propre partie contrôle (celle-ci étant identique pour tous les modules). Elle est construite autour d'une logique cablée qui lit en permanence le bus d'adresses internes du microprocesseur et compare son contenu avec l'adresse interne du module. Si les adresses coïncident, elle déclenche un accès au module. De son côté, le microprocesseur (processeur graphique) accède au module à travers le bus d'adresses en sortant :

- le numéro du module accédé

- la fonction à exécuter.

La partie contrôle possède un registre de "fonctions" qui mémorise la fonction à exécuter et sert à sélectionner le registre qui recevra la donnée issue du bus de données du microprocesseur. Les registres varient selon les modules et parmi eux on peut signaler :

Répétition :	répéter n fois une même action.
Fenêtr :	déterminer la taille de la fenêtre dans le module plan mémoire.
Adresse X :	valider l'adresse en X.
Adresse Y :	valider l'adresse en Y.
Incrémentation :	incrémenter n fois l'adresse X et Y.
Donnée :	valider la donnée du bus de données vers tous les modules.

- **La partie écriture**. Elle permet au processeur graphique d'écrire dans les mémoires des différents modules. Elle contient :

- les compteur à partir de l'horloge maître "CK" (correspondant au cycle de pipe-line de 75 nsec.) pour générer l'adresse courante en X et Y : ADRX, ADRY.

- le registre de données valides : BD.

De même, elle contient les mécanismes pour l'autoincrémentation, de

l'adresse en X ou en Y lors de l'appel des fonctions répétitives (avec l'aide de la fonction répétition).

- **La partie lecture.** Elle gère les mécanismes de lecture dans les différents modules; elle a accès aux différents bus de l'architecture banalisée (cf. II.2.3.). cette partie est formée principalement par une fonction de sélection de bus SELBUS et un multiplexeur d'entrée pour sélectionner un bus parmi tous ceux qui sont accédés en lecture. (voir annexe A).

2. 2. 3. Le module vidéo.

Ce module effectue les transformations nécessaires pour afficher l'information sur le moniteur de visualisation. Il reçoit au travers d'un bus spécialisé (le bus vidéo) les informations de couleur, pixel par pixel, au rythme vidéo. (Fig. II. 12)

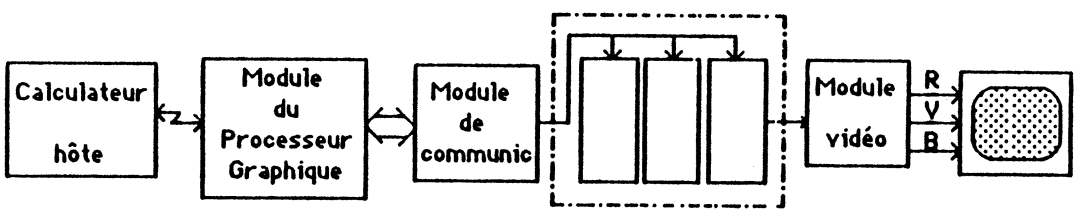


Figure II.12. Rôle du module vidéo.

ıLe module vidéo inclut les convertisseurs numériques/analogiques pour les trois couleurs primaires : rouge,vert, bleu, ainsi que la génération du synchronisme pour le balayage du moniteur. Ce moniteur utilise la norme RVB de TV et peut avoir plusieurs niveaux de résolution, allant de 512 x 512 jusqu'à 720 x 576 points.

D'autre part, il génère aussi l'horloge maître "CK" à 13,5 MHz qui donne la cadence de génération des pixels en pipe-line dans tout le post-processus de synthèse (à partir de la lecture des plans mémoire), comme on peut le voir dans la figure II.13.

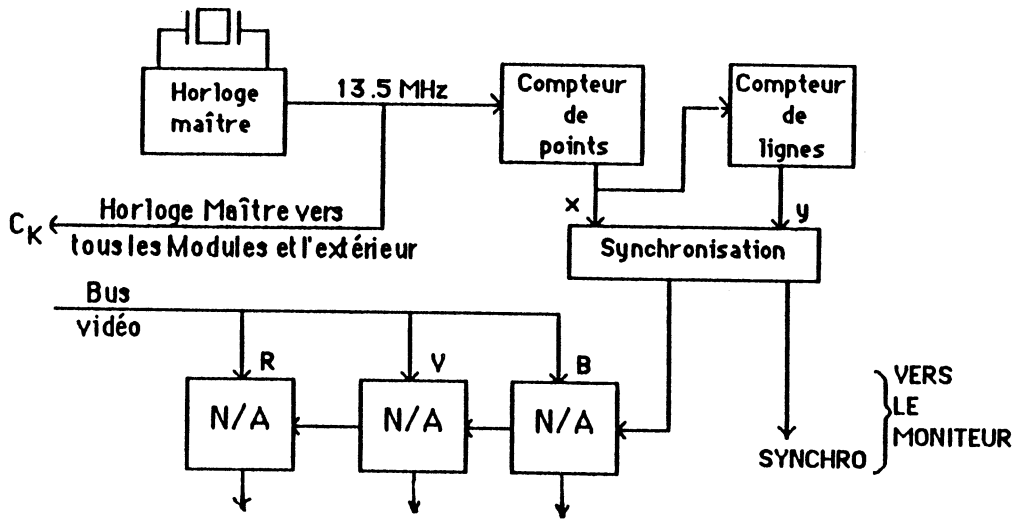


Figure II.13. Le module vidéo.

122. 2. 4. Le Module Plan Mémoire.

Lors de la conception d'un système avec moniteur à balayage trame, c'est la mémoire trame qui joue le rôle de "mémoire tampon" entre le générateur d'images et le contrôleur d'affichage pour le rafraîchissement de l'écran. Figure II. 14.

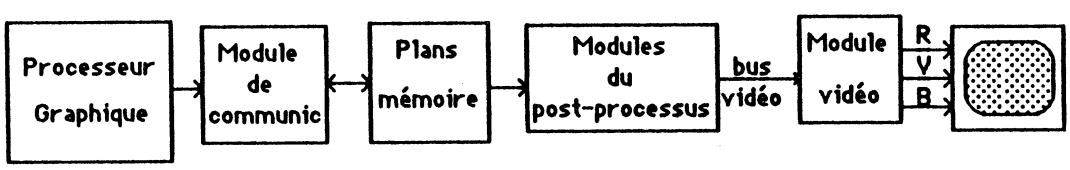


Figure II.14. Rôle des modules "plan mémoire".

Si le système a un degré assez important d'intéractivité, (voir temps réel), le contenu de cette mémoire doit pouvoir être renouvelé complétement à chaque nouvelle trame du moniteur et par conséquent, des conflits de lecture écriture se posent. En effet, cette mémoire doit être capable de supporter d'une part des accès en écriture pour la mise à jour de l'image,

d'autre part des accès en lecture pour les besoins de rafraîchissement de l'écran.

Dans le cas de l'architecture banalisée, le post-processus dépend du contenu des plans mémoires. Ce contenu peut être différent de la couleur finale à afficher, d'où le principe de banalisation du module.

Des fonctions temps réel sont intégrés dans ce module pour dégager le processeur graphique de tâches fastidieuses et compliquées, surtout au niveau des informations géométriques (par exemple la gestion d'une fenêtre). Les principales caractéristiques des plans mémoire sont :

- capacité de 1024 x 1024 points chacun, avec 12 bits de définition par point,

- l'enregistrement de 4096 informations différentes,

- la lecture et l'écriture de la mémoire à la vitesse vidéo,

- la translation en temps réel du plan image dans un espace de 4096x 4096 points,

- la création et modification de fenêtres en temps réel,

- le calcul de visibilité dans le cas de plusieurs plans mémoires,

- la visibilité/invisibilité des objets sur un plan.

La figure de l'annexe C nous montre le schéma du module plan mémoire.
Figure II.16. Module plan mémoire.

Outre la partie contrôle qui est similaire à celle du module de communications, le module plan mémoire possède des mécanismes de lecture/écriture dans les différentes mémoires qui le composent, notamment la mémoire de trame.

Le module nécessite 1 M x12 bits de mémoire de trame. Pour composer

cette matrice de 1024 x 1024 points, il utilise 48 boîtiers de mémoires RAMS dynamiques haute intégration de 256k x 1 bits, organisés eux-mêmes en quatre colonnes de 256k x 12. Chaque colonne est accédée tous les 4 cycles de pipeline. Cette solution évite l'attente de réponse dans les boîtiers RAM mais nécessite une technique spéciale d'adressage en alternance pour chaque colonne.

Vu de l'extérieur la mémoire de trame fournit ou accepte une valeur de pixel tous les 4 cycles de pipe-line. (entre deux accès à la même colonne, il s'écoule 4 cycles). (voir figure II.15).

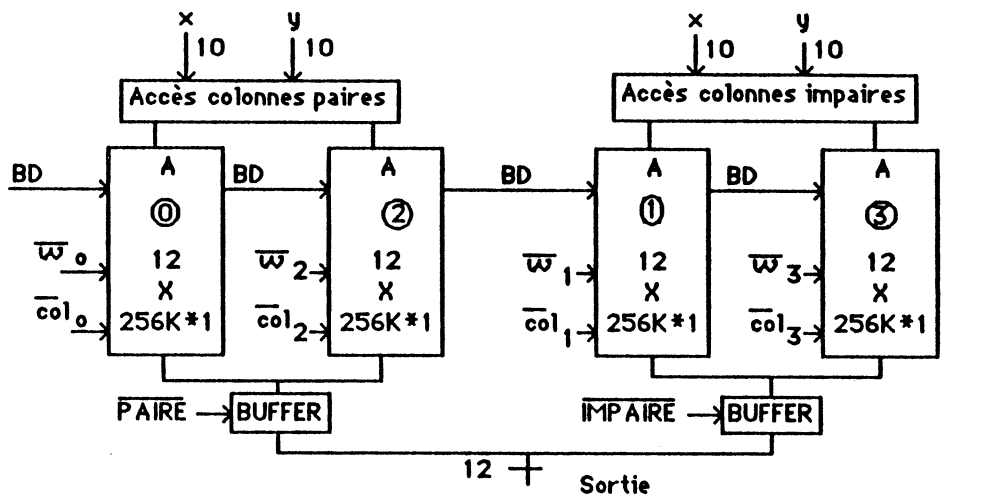


Figure II.15. Accés à la mémoire de trame.

iA l'écriture, le module accède aux mémoires de Trame au travers d'une ligne externe "EXT" générée par la partie contrôle, et à travers le bus d'adresse BAX et BAY provenant du module de communication. La facilité des répétitions d'écriture est souvent utilisée pour améliorer le temps d'accès et arriver à écrire à la vitesse vidéo.

Pour la lecture externe, une adresse externe est générée par le processeur graphique au même temps que le signal d'écriture à la colonne respective et au bout de 7 cycles de pipeline (6 cycles de lecture et 1 cycle de sélection de bus de sortie), la donnée est récupérée.

Dans le mode normal de lecture pour le rafraîchissement de l'écran, une

adresse de balayage est générée en continu par les compteurs de ligne et de colonne pour désigner les emplacements à lire en X et Y dans la mémoire de trame. L'initialisation de ces compteurs à chaque début de trame permet une translation du plan en X ou Y en temps réel.

Une fenêtre de taille variable est gérée grâce à deux tables à mémoire rapides (RAM statiques). Ces tables contiennent pour chaque pixel son appartenance ou non-appartenance à la fenêtre. Toute lecture ou écriture en dehors de limites de la fenêtre peut être inhibée par le module.

A la sortie des mémoires, un dernier test pour la visibilité/invisibilité des pixels est faite. Ce test permet (en accord avec le choix du bus de sortie), de valider ou de ne pas valider la sortie dans les registres de sortie.

2.2.5. Les modules de Post-Processus

Les hautes performances des systèmes évolués sont obtenues au prix d'une forte intégration de fonctions au niveau matériel.
L'architecture banalisée, permet de faire évoluer ces modules existants et en même temps d'en ajouter de nouveaux pour améliorer les performances. Les modules peuvent être réalisés à l'aide de puissants processeurs spécialisés à base de microprocesseurs, circuits VLSI ou de logique câblée. Cependant ces modules ont des caractéristiques différentes selon qu'ils se trouvent intégrés au préprocessus ou au post-processus de synthèse.

Au niveau du préprocessus, ces modules doivent être capables de calculer et fournir le débit d'information nécessaire pour remplir les "plans mémoire" à la vitesse vidéo (dans le cas du temps réel). Les types d'opérateurs adaptés à cette partie sont : la modélisation, les transformations géométriques, l'élimination de parties cachées ou la prise de vue, c'est-à-dire , des manipulations des objets en 3D.

Pour le post-processus, les modules doivent être intégrés entre les "plans mémoire" et le "module vidéo", et par conséquent ils doivent lire (à la vitesse vidéo), les informations des plans mémoire pour les fournir au

même débit au module vidéo. Dans ce cas les opérateurs doivent agir sur la structure 2D de la maquette mémorisée dans les mémoires de trame. Le fait d'avoir une structure 2D implique que les attributs morphologiques et géométriques relatifs à chaque objet sont déjà synthétisés. Les manipulations possibles seront celles appliquées à l'ensemble de la maquette : l'affichage (Ga) ou de simples transcodages de couleurs.

Cependant le cas de l'architecture banalisée est particulier car le contenu des mémoires de trame peut être quelconque, on peut ainsi mémoriser des informations géométriques relatives à chaque face ou à chaque pixel de la mémoire (normale, profondeur, couleur), ou bien l'identification des objets.

Ces informations serviront à un post-traitement pour obtenir la couleur des objets, l'illumination de la scène, et subir en même temps les manipulations relatives à l'affichage (Ga). Les modules du post-processus peuvent nous servir dans ce but.

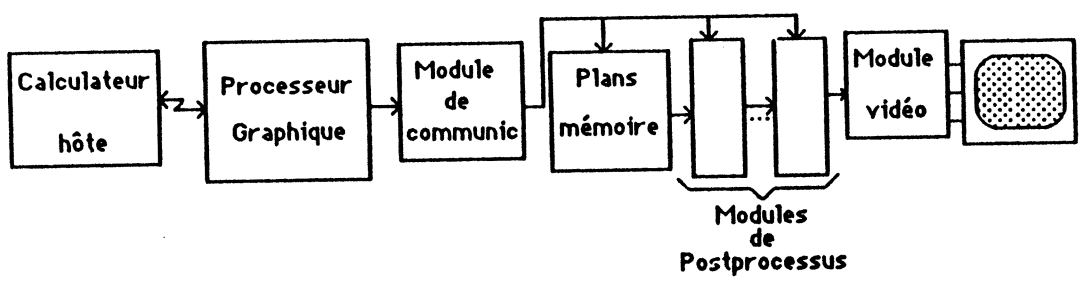


Figure II.16. Rôle des modules du Post-processus.

Un exemple simple de module de post-processus est celui du transcodage de couleurs ; à partir de l'identification de l'objet dans le plan mémoire, un module de post-processus est chargé de sa réalisation à l'aide d'une table de couleurs (voir figure II.17).

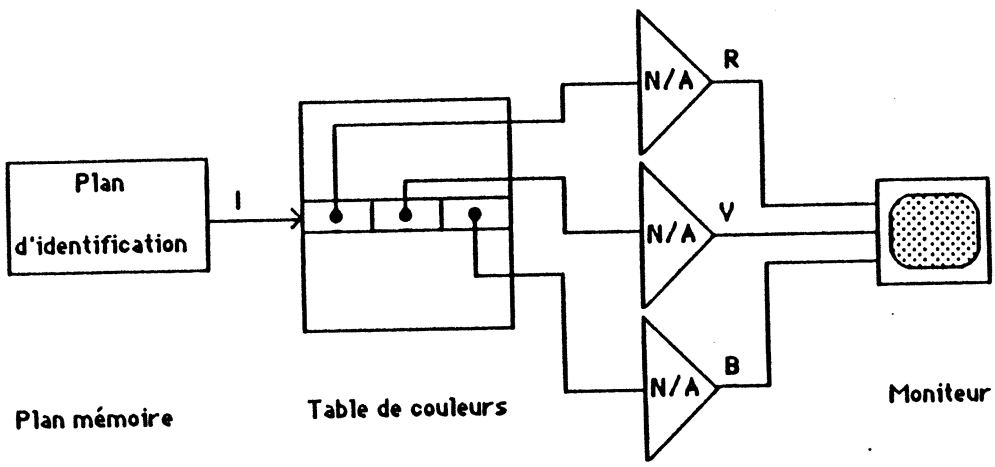


Figure II.17. Exemple de module de post-processus.

A l'heure actuelle, peu de systèmes (voire même aucun) ne proposent des fonctions de post-processeur autres qu'une table de couleurs. Continuant la ligne d'évolution d'Hélios, nous nous attacherons aux modules de post-processus pour rendre aux images un réalisme plus saisissant.
Dans le cas de l'architecture banalisée, les modules du post-processus peuvent être constitués par deux types de modules distincts :

- les modules banalisés
- les modules de calcul spécifique.

Dans le chapitre III, nous montrerons la démarche suivie pour la conception de ces modules.

2. 3. Les bus banalisés.

Dans le cadre de l'architecture banalisée, le fait d'avoir des modules banalisés entraîne aussi l'utilisation des différents bus au momende la configuration du système. L'ordonnancement du processus de visualisation (grâce aux différentes possibilités des modules), nous amène à considérer la possibilité de plusieurs entrées et plusieurs sorties

dans chacun d'eux. En effet, plus de possibilités de réordonnancement auront les modules, plus de combinaisons de chemins différents auront à parcourir les informations à manipuler. L'exemple le plus simple est celui pour lequel un plan mémoire est configuré en mémoire d'image : son information doit sortir directement sur le bus vidéo pour arriver au module vidéo. Par contre si le même plan est reconfiguré, en plan d'identification de faces, l'information contenue doit parcourir un bus différent pour arriver à un module du post-processus qui traitera cette information.

2. 3. 1. Classification des Informations.

Ici, nous nous intéressons aux informations issues des plans mémoires. Nous considérons que la partie préprocessus a réalisé la description, modélisation et prise de vue de la maquette, c'est-à-dire la synthèse des informations morphologiques, géométriques et structurelles (M + G + S + Gv) ; ces informations plus l'identification des objets "I", constituent la maquette à visualiser. Mis à part les problèmes de réalisme résultant de ces modélisations (cf. I.1.2.2.), ces informations peuvent évoluer de plusieurs manières dépendant du processus de visualisation choisi. Ainsi, si on choisit de donner priorité à l'aspect final de la scène (A), la synthèse de cet attribut doit s'effectuer à la fin du processus ; mais si par ailleurs le degré d'interactivité est assez important, un tampon de mémorisation est nécessaire avant cette synthèse [Mar 82]. La figure II.18 nous montre le processus de synthèse.

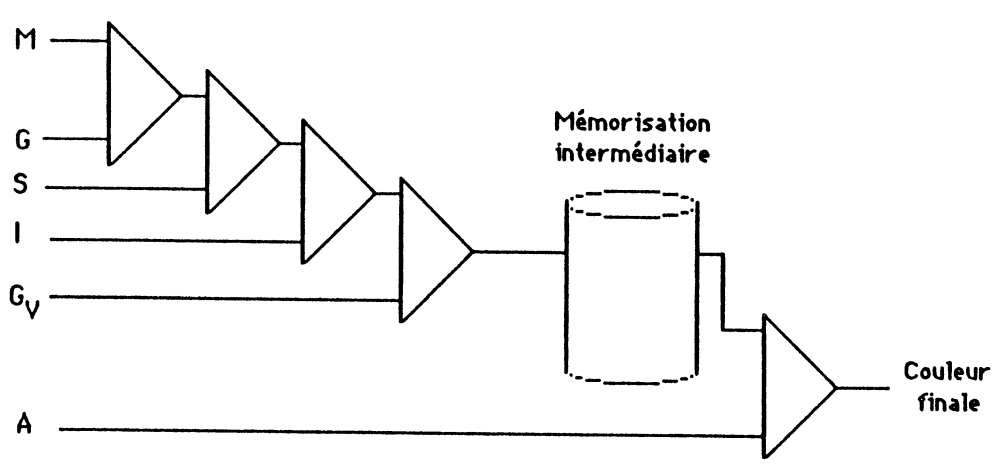


Figure II.18. Processus de synthèse en donnant priorité à l'aspect .

Cette mémoire tampon peut être constituée par un module plan mémoire dans le système d'architecture banalisée.

D'un point de vue fonctionnel, la couleur finale d'un objet est la modulation de l'aspect de l'objet par rapport aux facteurs d'éclairage de toute la scène. Ces deux types d'informations sont synthétisés par le préprocessus d'une façon structurée et mémorisée dans les plans mémoires. Le processus de post-synthèse est, donc, chargé de lire et synthétiser ces deux informations pour obtenir la couleur finale à afficher sur le moniteur. Dans les deux paragraphes qui suivent, nous analyserons les informations d'aspects des objets ou leur chrominance et celles d'éclairage ou leur luminance.

2. 3. 1. 1. La Chrominance.

Dans le sens strict du terme, la chrominance est la couleur d'un objet. Nous utiliserons ce terme un peu plus particulièrement, car il dénomme, à part la couleur de l'objet, toutes les autres informations qui ont des rapports avec cette couleur, pour aider (avec un traitement particulier) à trouver cette couleur.

Par exemple, à partir de l'identification I de l'objet, on peut lui assigner une texture particulière pour arriver à la couleur finale de l'objet, comme on le voit sur la figure II.19.

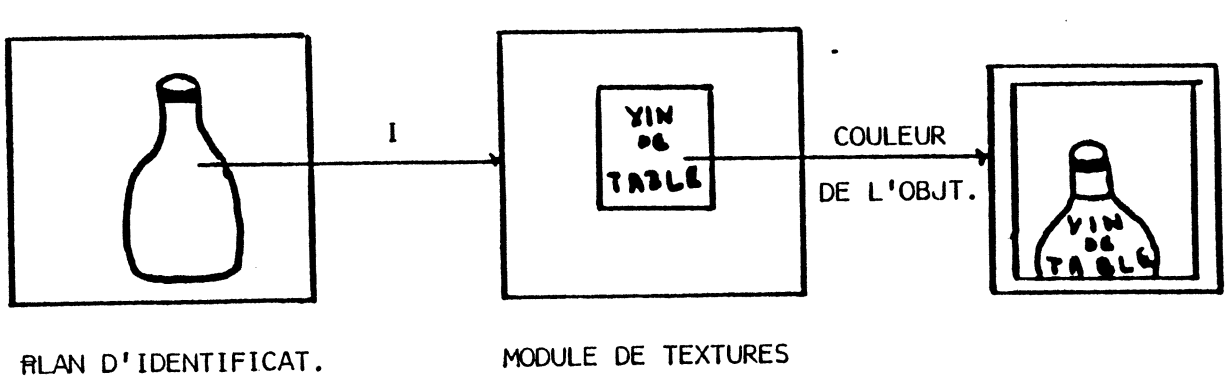


Figure II.19. Exemple d'utilisation de la chrominance.

Dans ce cas la chrominance est associée à l'identification pour devenir après le module de textures la couleur elle-même.
Le chemin que parcourt l'information de chrominance est indépendant de celui des facteurs d'illumination des objets.
Nous préfixerons par "C" les informations de chrominance.

2. 3. 1. 2. La luminance.

Pour la luminance aussi, il existe une définition particulière en accord avec nos besoins.

On définit la luminance comme toute information qui a des rapports avec l'illumination des objets, indépendamment de l'aspect (couleur, matériau, texture qui les compose). Evidemment, la luminance peut être directement les valeurs d'illumination en chaque pixel de l'objet. Il faut souligner que cette information de luminance est liée aux informations géométriques de chaque objet (par exemple la normale d'une face plane).

Ainsi par exemple, à partir de la normale d'une face plane, on peut trouver les coefficients de réflexion de la lumière par rapport à une source lumineuse située dans l'espace (figure II.20)

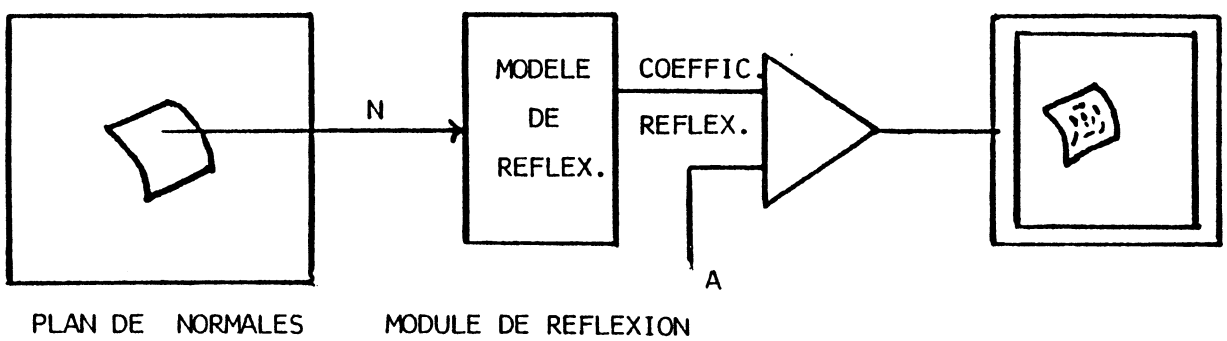


Figure II.20. Exemple d'utilisation de la luminance.

L'information de la normale contient, donc, une information liée à la luminance et qui est transformée à l'aide d'un modèle de lumière et de l'aspect de l'objet en la couleur finale sur le moniteur. Nous utiliserons le préfixe "L" pour identifier les informations du type luminance dans le processus de synthèse.

2. 3. 2. Le nombre de Bus banalisés.

Le nombre de bus ainsi que leur largeur (capacités de bits de transmission), dépend généralement de l'application et de son processus de visualisation. Il est évident qu'un nombre élevé de bus avec une grande largeur de bits est l'idéal pour un système qui veut être banalisé du point de vue fonctionnel. Cependant, cette solution n'est pas réalisable du point de vue matériel, car plus on a de bus, plus on a de lignes qui parcourent le système (le nombre devient prohibitif pour un processus assez compliqué), et plus coûteuse et difficile est sa mise en oeuvre.

La mise en évidence d'un nombre minimum de bus sera faite au travers de l'analyse du parcours de l'information dans les processus de visualisation.

D'un point de vue général, les informations issues des plans mémoire sont de deux types : la chrominance et la luminance. Ces deux informations vont se "mélanger" à la fin du processus de visualisation pour donner la couleur finale.

La chrominance C(P) et la luminance L(P) sortent des plans mémoires ; chaque module banalisé "plan mémoire" doit, alors, sortir sur deux bus minimum : un bus pour la chrominance est autre pour la luminance, par exemple BUS1 et BUS2. Bien entendu, cette sortie doit être programmable par logiciel pour pouvoir choisir un des bus en fonction de l'ordonnancement des modules.

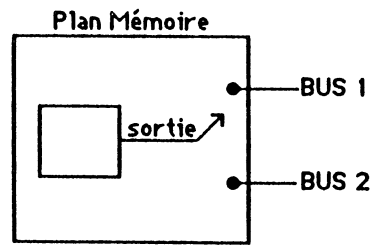


Figure II.21. Interface sortie des modules plans mémoire.

Si les informations issues des plans mémoire C(P) et L(P) ne sont ni la couleur des objets ni les paramètres d'illumination des objets, elles doivent traverser des modules de post-processus pour une mise en forme. Les sorties de ces derniers modules doivent s'effectuer encore sur deux bus différents de ceux utilisés pour les plans mémoires. Par exemple BUS3 et BUS4. (Figure II.22)

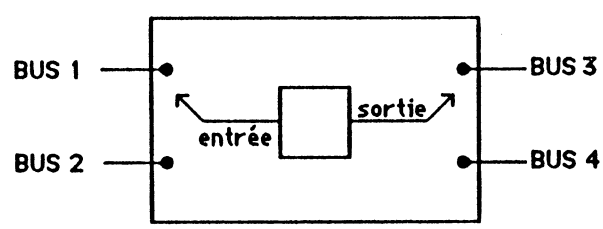


Figure II.22. Interface entrée/sortie des modules de post-traitement.

Finalement la chrominance C(T) et la luminance L(T) mises déjà en forme sont mélangées pour sortir directement la couleur sur le bus vidéo, comme on le voit sur la figure II. 23.

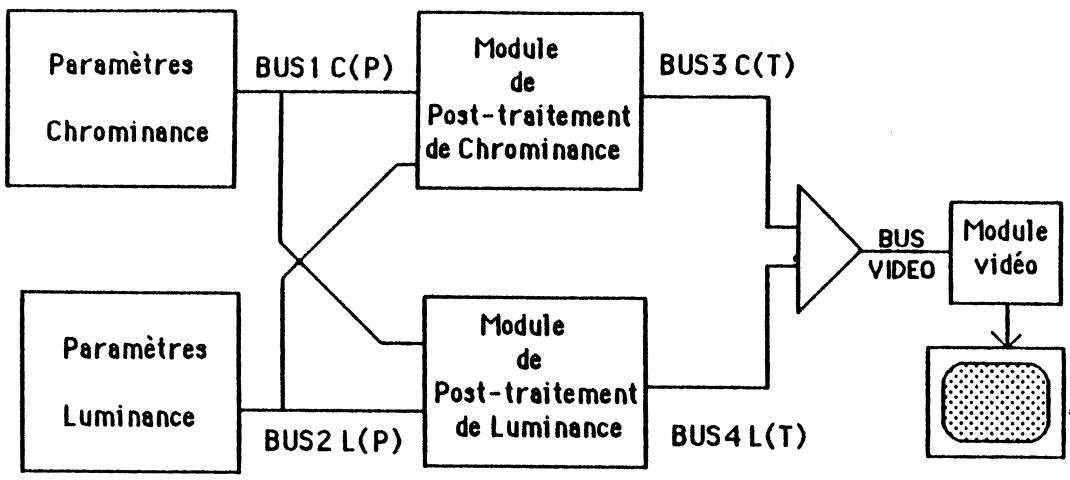


Figure II.23. Mise en évidence des bus banalisés.

L'exemple antérieur met en évidence le nombre minimum de bus qui doivent parcourir le fond panier de l'architecture banalisée. Ces bus contiennent des informations soit de chrominance soit de luminance :

- 4 bus banalisés minimum,

- 1 bus vidéo de sortie.

En ce qui concerne la largeur de bus, un compromis a été trouvé entre une information très précise avec une grande largeur de bus et une information suffisament compléte qui ait un nombre minimum de bits.

Finalement, l'expérience nous a montré qu'une largeur de 12 bits pour chaque bus s'avère suffisante pour toutes les informations présentes dans le post-processus.

Nous utilisons aussi un cinquième bus banalisé pour les lectures externes dans les différents modules, ainsi que pour d'éventuels ajustements..

Le bus vidéo est sur 24 bits pour donner une plus grande souplesse dans le nombre de couleurs synthétisées (16 millions de couleurs

maximum). (voir figure II. 24).

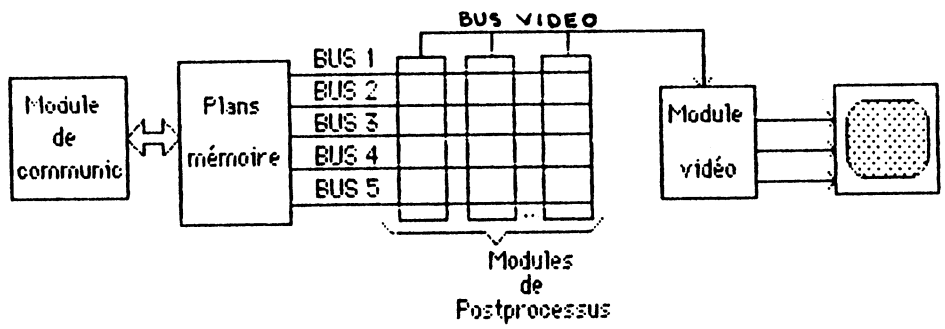


Figure II.24. Parcours des différents bus banalisés.

2. 4. Bus de profondeur.

Une autre caractéristique de l'architecture banalisée est sa souplesse et son adaptabilité. Ainsi à tout instant on peut reprogrammer les modules pour configurer un nouveau processus de visualisation.

Pour une application donnée, on peut avoir besoin de plusieurs processus de visualisation se mêlant au fur et à mesure de la visualisation. (voir Fig. II. 25).

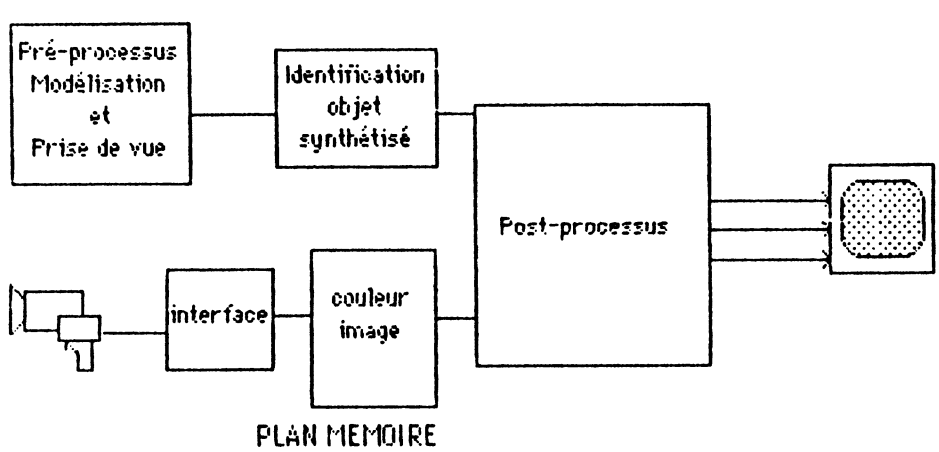


Figure II.25. Mélange de processus différents.

Pour pallier à ce problème on a introduit la notion de bus de profondeur.

Ce bus assigne à chaque instant une information de 4 bits sur la nature du pixel à traiter. Evidemment ce bus est géré par les modules plan mémoire et suit l'étude de visibilité de chacun. (fig. II. 26).

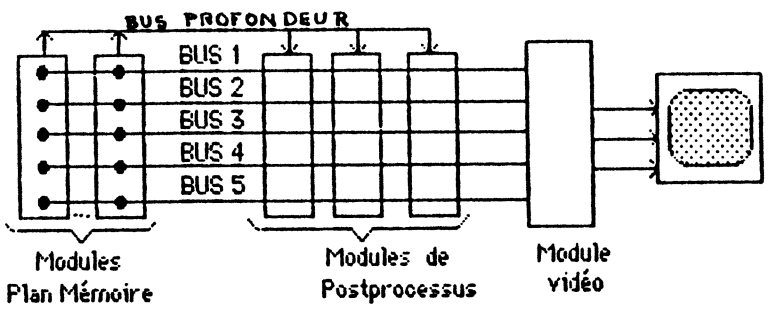


Figure II.26. Le bus de profondeur.

Dans chaque module du post-processus, il existe un décodeur de priorité qui à partir du bus de profondeur identifie la nature du pixel à traiter.

2. 5. Evaluation du Système.

2. 5. 1. Avantages du Système.

De grandes améliorations se sont produites par rapport aux versions précédentes d'Hélios. En effet, grâce à sa conception modulaire, on est en face d'un système complétement extensible. A l'heure actuelle le sytème existe en deux versions différentes : Terminal Graphique et Station de Travail.

Dans le cas d'un Terminal Graphique, le système est connecté à un Calculateur hôte. L'application définit sur le calculateur hôte les éléments de la maquette à synthétiser ainsi que les transformations géométriques dans son repère propre (coordonnées utilisateur). Les ordres reçus par le

terminal sont mémorisés dans une liste de visualisation interne. L'utilisateur peut structurer ses objets grâce à un logiciel associé au terminal. Les opérations de lissage et d'élimination des parties cachées sont traitées par le terminal.

Dans le cas d'une station de travail, le système prend en charge le développement en local d'applications et le stockage d'informations (grâce à sa mémoire de masse). Toutes les possibilités d'un terminal y sont présentes et prises en charge par le système.

D'autre part, les communications entre le processeur graphique, les modules des plans mémoires et les modules du post-processeur se sont améliorées grâce à la mise en oeuvre de nouveaux concepts dans le module de communication. A l'heure actuelle le système permet des vitesses de 13,5 Mpoints/sec. (temps réel en lecture et écriture des modules plans mémoires).

Une autre facilité du système est l'utilisation des dispositifs conventionnels de vidéo pour le stockage et la prise d'images : magnétoscopes, vidéo-disques, caméras, etc. Par ailleurs, on peut utiliser un dispositif de "hard-copy" pour lire tout le contenu de la mémoire trame (ce qui permet une plus grande définition d'image), ou des images dans des plans accolés les uns à coté des autres.

Il faut signaler aussi la facilité d'identification d'objets directement sur l'écran. En effet la structuration de l'image dans les plans mémoire permet de récupérer une identification de l'objet associé au repère de l'écran.

2. 5. 2. Inconvénients du Système.

En ce qui concerne les inconvénients du Système, on peut citer le logiciel de gestion qui est assez lourd. En effet, la gestion des différents processus de visualisation qui se mêlent dans une application, nécessite

la progammation du bus de profondeur ainsi que des mémoires de décodage (mémoires de priorité) dans chaque module. Cette gestion est transparente pour l'utilisateur et assure l'indépendance vis à vis de l'application.

Même si au niveau du post-processus les facilités matérielles sont variées, les opérateurs du pré-processus sont émulés par logiciel ce qui rend cette partie assez lente. En effet, les utilisations du processeur graphique et de son co-processeur arithmétique ne sont pas assez puissantes pour arriver aux fonctions temps réel. A l'heure actuelle les manipulations 3D telles que les transformations géométriques, l'élimination de parties cachées, etc..., ont des vitesses de traitement de 500 faces planes par seconde. Cependant, la modularité du système permet d'envisager un processeur spécialisé pour ces tâches (par exemple un microprocesseur en tranches).

Une dernière remarque du système est sa moyenne résolution d'écran : 720x576 points maximum. Même si à l'heure actuelle cette remarque est une pénalisation du système, des moyens peuvent être mis en oeuvre pour l'augmenter, notament dans le module de communication. La méthode du double buffer de sortie est nécessaire lorsque le temps d'affichage du pixel est inférieur à 75 nsec (temps de cycle de pipe-line actuel). En effet, ces deux buffers seront utilisés en alternance : le premier peut être lu à la vitesse d'affichage du point tandis que le deuxième est rempli à la vitesse actuelle de 75 nsec par point. A la fin de la lecture dans le premier buffer, les rôles doivent s'inverser pour continuer ainsi de suite jusqu'au balayage complet de l'écran.

3. - Retombée sur l'organisation hiérarchique.

Si on considère l'organisation hiérarchique appliquée à l'architecture banalisée on peut en déduire plusieurs choses :

- l'ordonnancement du processus de visualisation peut se faire tant pour le logiciel que pour le matériel.

- l'application est de plus en plus indépendante des supports du système.

Mise à part le réordonnancement fait par logiciel, notre cas est plus ambitieux car on est en face d'un système qui réalise le réordonnancement des opérateurs du post-processus au niveau matériel.

Cependant, cette facilité soulève plusieurs problèmes, surtout au niveau des unités de contrôle et de communication pour lesquelles on a besoin d'un lourd logiciel de gestion.

CHAPITRE III : CONCEPTION DES MODULES DU POST PROCESSUS

I. Présentation

Les avantages de l'architecture banalisée ne sont mis en relief que si on l'utilise de façon adéquate. Cette utilisation nécessite, outre le choix du bon processus de visualisation, des opérateurs performants qui s'adaptent aux variations du processus lui-même.

Les variations sont dues à un changement de la nature du point traité. En effet, les "plans mémoire" (étant des mémoires tampons entre le préprocessus et post-processus) peuvent contenir des informations banalisées qui varient en rapport direct avec l'application.

Nous nous intéressons à la mise en oeuvre des différents opérateurs du post-processus. Ces opérateurs varient selon le contenu des plans mémoire, notamment lorsqu'on privilégie les attributs d'esthétique (aspect des objets et éclairage final de la scène).

Les opérateurs nécessaires seront mis en évidence par l'étude des processus réels de synthèse.

Une grande partie de ce chapitre est consacrée à l'étude de quelques exemples types.

2. La mise en évidence des modules.

Les modules utilisés par le post-processus dépendent de la complexité et des performances demandées mais doivent rester les plus simples possibles pour être facilement implémentés.

La mise en évidence de ces modules suppose la connaissance des opérateurs nécessaires au post-processus. Il y aura autant de modules matériels que d'opérateurs. Mais certains de ces modules pourront être

identiques (s'il est possible d'utiliser un ou plusieurs modules banalisés), ceux ci pouvant servir à réaliser des opérateurs distincts.

il y a trois facteurs dans le processus de visualisation qui déterminent les caractéristiques et la nature des systèmes de synthèse d'images et que nous avons analysés dans le chapitre I. Ici, nous reviendrons sur les facteurs qui influent sur la conception des opérateurs de synthèse : le niveau de réalisme des images, la puissance d'intéractivité et la complexité de la scène.

i) Le niveau de realisme des images.

C'est le facteur qui influe le plus sur le résultat visuel de la scène. Ce facteur nécessite un calcul assez précis pour donner des images de bonne qualité. Il y a deux approches pour produire des bonnes images : l'élimination de parties cachées et la production d'ombres sur les surfaces visibles. Ces deux approches peuvent être utilisées pour tout type d'application : un système de simulation (de vol, de conduite, etc) nécessite des images réalistes en même temps que dynamiques (temps réel) ; de même, la conception d'objets (bijoux, voitures, maisons, vêtements, etc.), nécessite un haut degré de réalisme mais le dynamisme quoique souhaitable n'est pas indispensable. La principale difficulté pour obtenir une bonne qualité des images est sa complexité intrinsèque (les matériaux, les textures, etc), qui doit être prise en compte au moment de la conception. C'est pour cela que la plupart du temps, le logiciel est chargé de rendre ce réalisme, en détriment du temps de réponse.

ii) La puissance d'intéractivité.

La rapidité de prise en compte de nouvelles commandes et la vitesse de réponse du système influent directement sur la puissance d'intéractivité. Par ailleurs, l'identification des objets directement sur l'écran est très importante ; ainsi, par exemple, si le calcul final de l'image est fait complètement par logiciel, l'identification d'un objet nécessite la duplication des informations de Géométrie et d'Identité (G+I) le

concernant, par rapport au repère de l'écran. L'intéractivité s'appuie sur les techniques matérielles et logicielles afin d'offrir à l'utilisateur un contrôle dynamique des objets manipulés et visualisés sur l'écran. Plus l'intéractivité demandée est élevée plus les techniques doivent être rapides. C'est pourquoi il y a une tendance à favoriser les techniques parallèles ou en "pipe line" (dans le cas de l'architecture banalisée).

iii) La complexité de la scène.

La complexité met en évidence le nombre d'informations à traiter par les différents opérateurs dans le processus de visualisation ; dans la modélisation, il est évident que le calcul pour des scènes composées de faces gauches est plus important que celui pour des scènes composées de faces planes. Par ailleurs, la nature des objets (texture, réflexion etc), ainsi que leurs inter-relations (ombres portées, parties cachées, etc), augmentent aussi cette complexité. Il est évident que l'augmentation de la complexité des objets est liée à la puissance de calcul des opérateurs, car plus ils sont performants, plus la quantité d'informations traitées est grande, et plus la scène (et les objets) représentée peut être complexe.

1.1. Exemples de processus de synthèse.

Dans les paragraphes qui suivent, nous étudierons des exemples de processus de visualisation et leur correspondance dans l'architecture banalisée.

1.1.1. Processus pour la modélisation par faces planes.

Le processus de synthèse à l'aide de faces planes peut-être représenté par la figure III.1. [Mar 84].

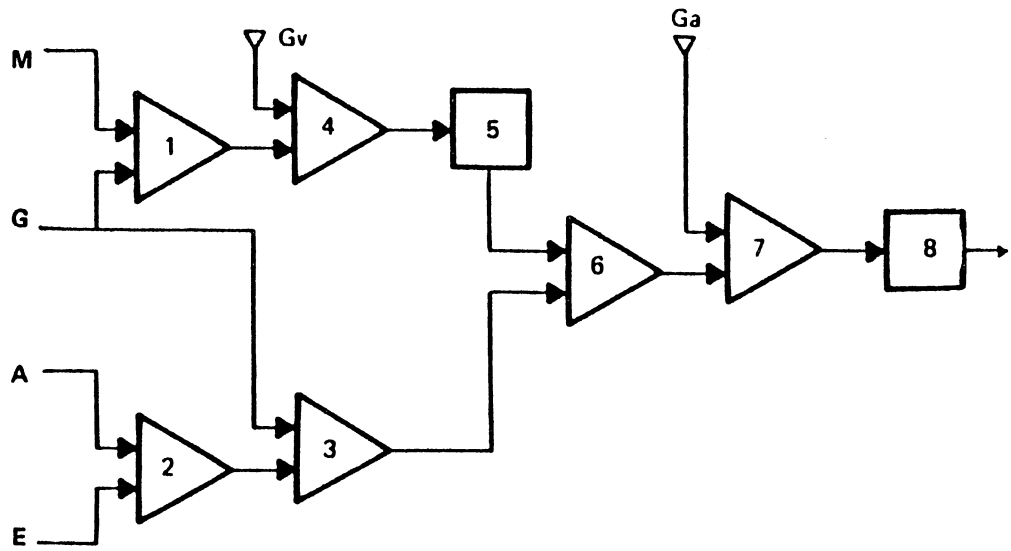


Figure III. 1. Processus pour le traitement par faces planes.

1. Synthèse M+G : les attributs morphologiques M du contour de la face sont mélangés avec les attributs géométriques G pour situer les faces dans l'espace 3D.
2. La synthèse A+E : Pour moduler la couleur A par rapport à l'intensité d'éclairage E.
3. La prise en compte des propriétés géométriques de la lumière (position de la source lumineuse).
4. La projection de la face 3D de l'espace utilisateur à l'espace 2D du support de visualisation (la prise de vue).
5. Le remplissage des faces par les points internes.
6. L'illumination des faces vis à vis des facteurs d'éclairage.
7. Mise en forme de l'image 2D pour l'affichage pris en compte de fenêtres.
8. Eventuelle transformation finale ou transcodage de couleurs.

Un point important à noter est la pénalisation de l'intéractivité, car la modification d'un attribut quelconque entraîne à nouveau le calcul de tout le processus. Il existe deux techniques simples pour éviter ce problème (qui pourront être utilisées séparément ou en même temps).

La première est l'utilisation de mémoires tampon pour les attributs déjà synthétisés. Cette démarche est utilisée normalement pour privilégier la prise de vue ou l'affichage (des opérateurs 6 ou 7 du cas précédent).

La deuxième est la pénétration des attributs non synthétisés (I.M.A.G.E.S.) vers les opérateurs finaux de synthèse.

L'architecture banalisée de GETRIS peut utiliser les modules "plans mémoire" servant comme mémoire tampon pour les attributs synthétisés. Par ailleurs si l'on veut privilégier l'aspect final de la face, cet attribut doit être synthétisé tardivement. Le processus transformé de la figure III.1 est représenté par la figure III.2.

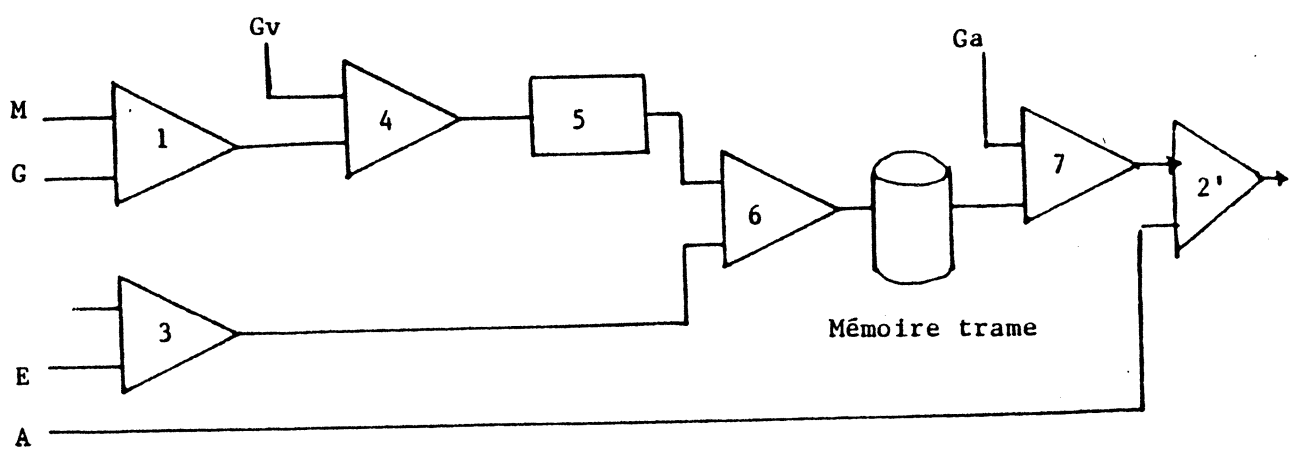


Fig. III.2. Processus privilégiant l'aspect et l'intéractivité de la prise de vue.

Tous les opérateurs restent avec la même fonction qu'auparavant sauf le 2' qui assigne leur aspect aux différentes faces de la scène. Le rôle de la mémoire de trame sera, donc, de permettre aux attributs d'aspect (A) et d'affichage (Ga) d'évoluer indépendamment du reste des attributs.

Si on transporte ce processus sur l'architecture banalisée, on se trouve face à un système qui prend en charge la quasi totalité des opérateurs de synthèse par logiciel.

Ici le post processus ne consiste qu'en la mise en forme de l'image à

afficher (fenêtre), et la prise en charge du coloriage des faces. Le schéma fonctionnel du système sera le suivant (fig III.3.).

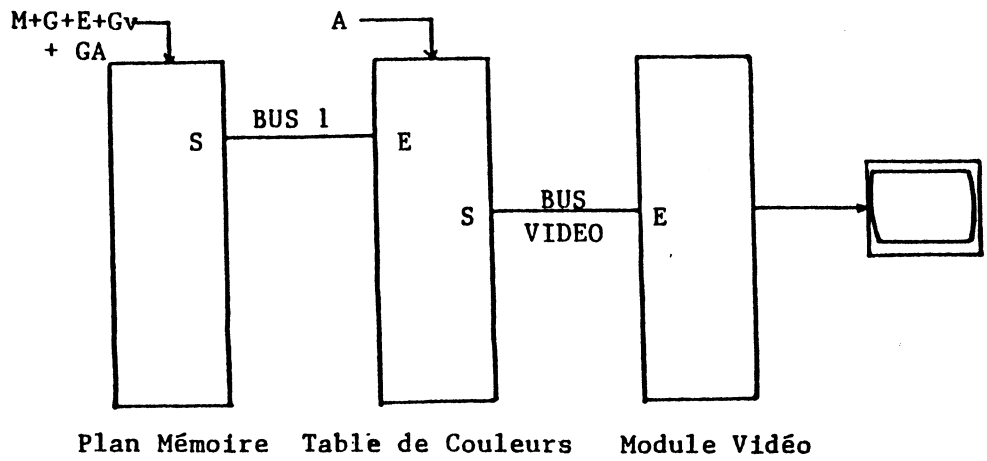


Figure III.3. Rôle du post-processus dans les faces planes privilégiant l'aspect.

Les plans mémoire reçoivent les attributs des faces déjà synthétisés M+G+E+Gv. En plus, la fenêtre Ga est prise en charge implicitement par les modules plan mémoire (c.f. II. 2.2.4.). L'aspect A est transmis par le processeur graphique à un opérateur de mémoire synchrone (c.f. II. 2.1.). L'information d'aspect A est lue à l'aide des valeurs des plans mémoire et transmis convenablement au module vidéo.

1.1.2. Processus pour la modélisation par faces gauches.

Le processus de synthèse pour des faces gauches est très semblable à celui pour des faces planes, sauf que l'opérateur No.5 de remplissage de faces dans les figures III.1 et III.2 n'est plus nécessaire car chaque point peut-être ramené à la taille d'un pixel. En effet, ce processus change au niveau de la modélisation des objets. Maintenant chaque point de l'objet est calculé d'une façon très précise (jusqu'au niveau du pixel). (figure III.4.).

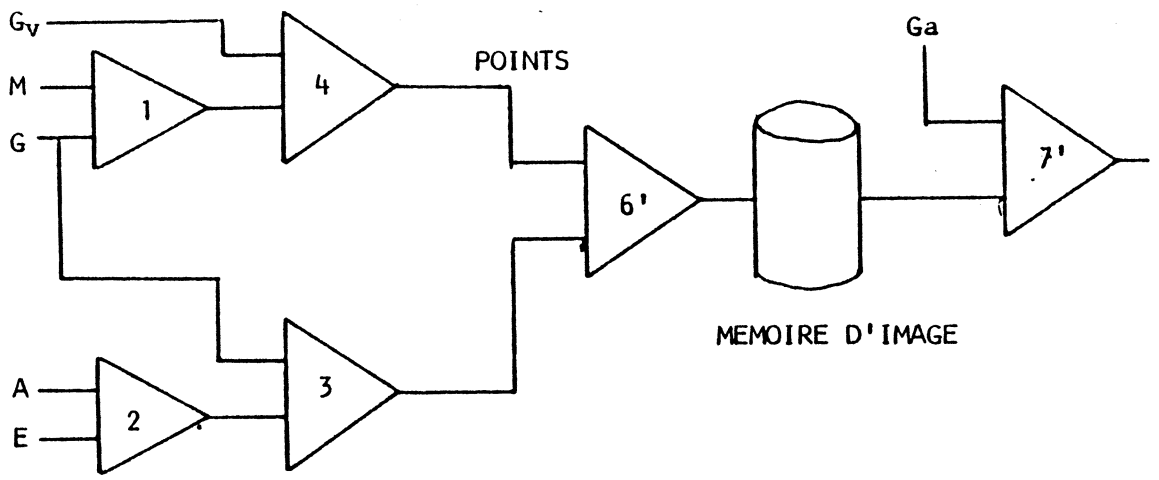


Figure III.4. Processus de synthèse par faces gauches.

Les points issus de l'opérateur 6' sont déjà les points coloriés de l'objet. Cette information est mémorisée dans les plans mémoire qui correspondent aux simples mémoires d'image.

Cependant, les points issus de l'opérateur 1 du processus de la figure III.4 peuvent être identifiés par leur position dans l'espace par exemple la normale à chaque point. De même pour l'opérateur 3 qui peut donner comme résultat les paramètres de l'aspect modulé par l'éclairage de la scène. Ces normales et paramètres d'aspect peuvent être mémorisés dans les plans mémoire. Ils serviront aux opérateurs 7' et 8" de la figure III.5. pour le coloriage final des points.

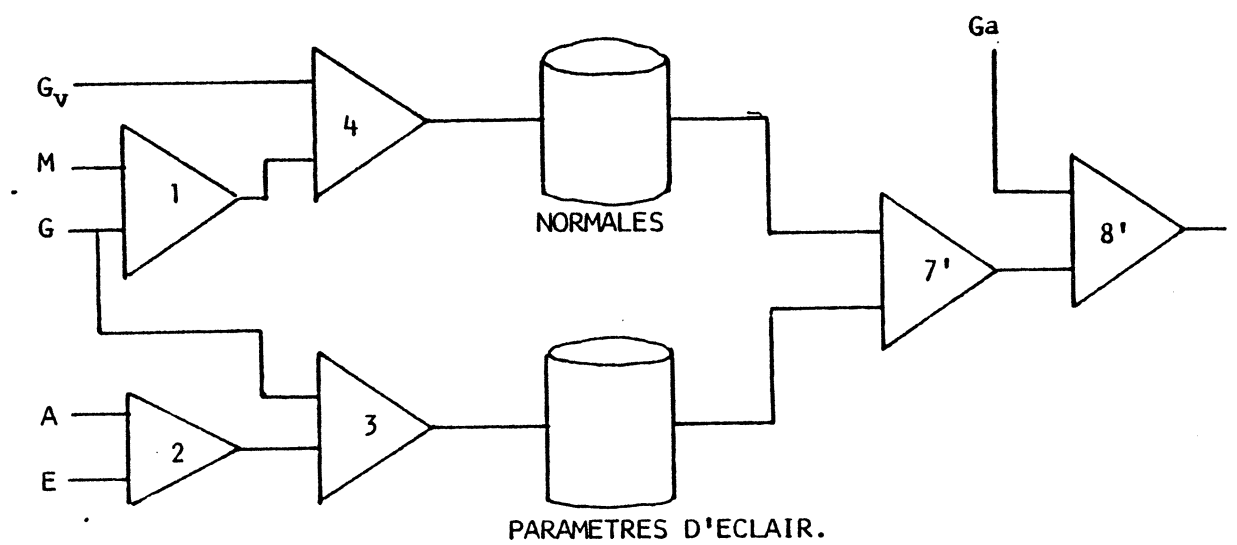


Figure III.5. Processus par faces gauches à l'aide des normales.

Dans ce dernier cas, le préprocesseur peut servir à engendrer les normales à chaque point des surfaces à synthétiser ainsi que leurs paramètres respectifs d'éclairage.

Le post processus sera donc chargé de rendre les caractéristiques de couleurs à chaque point des faces gauches (fig. III.6.).

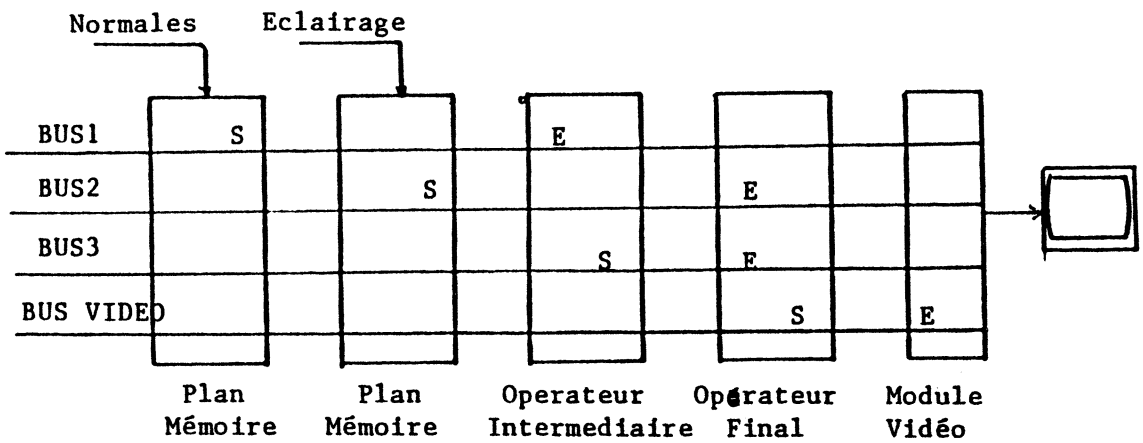


Figure III.6. Rôle du post-processus dans les faces gauches.

Les plans mémoire servent à mémoriser les normales (N) et les paramètres d'éclairage (E) issus d'un calcul logiciel. Un opérateur intermédiaire transforme ces normales en la couleur du point. Finalement un opérateur transforme cette couleur par rapport aux coefficients de l'éclairage.

1.1.3. Processus pour l'élimination de parties cachées.

Au cours du processus de visualisation, chaque élément constituant la maquette peut être synthétisé à l'aide d'un des processus décrits précédemment. Cependant, l'intéraction des différents éléments dans un processus (inévitable lors de la description d'une scène complexe), met en évidence plusieurs problèmes d'intéraction entre éléments : l'élimination de parties cachées, la production d'ombres portées, la

transparence des objets et l'"antialiasing", pour lesquels on trouve plusieurs études dans la littérature : [SSS 74], [Cat 74], [We A 77], [Car 84], [FoV 82].

Ce paragraphe est consacré à l'étude des processus réalisant l'élimination de parties cachées dans le cadre de l'architecture banalisée.

Le problème qui se pose est le suivant : au moment de réaliser la prise de vue (Gv) de l'ensemble de la maquette, l'image 3D est projetée sur un plan 2D, dit plan de projection. Cette projection nécessite le point de vue de l'utilisateur ainsi que le point de visée. Cependant cette transformation géométrique est à l'origine d'ambiguités de perception dues à la perte d'information 3D.

La façon de lever ces ambiguités est l'élimination des parties cachées des objets manipulés, comme on le voit sur la figure III.7.

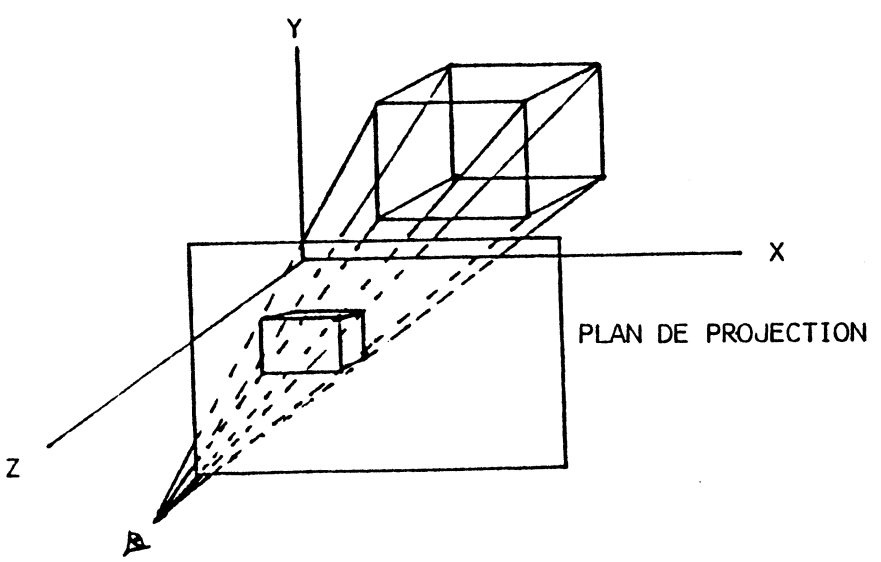


Figure III.7. Projection en perspective et élimination de parties cachées.

Cette projection perspective définit un cône de vision issu du point d'observation. Une grande simplification peut être faite en considérant seulement la direction de visée pour arriver à une projection parallèle. En effet, cette simplification considère que le point de vue se trouve à l'infini

sur l'axe z et par conséquent, il n'y a pas de déformation visuelle de l'objet. La figure III.8. montre cette projection.

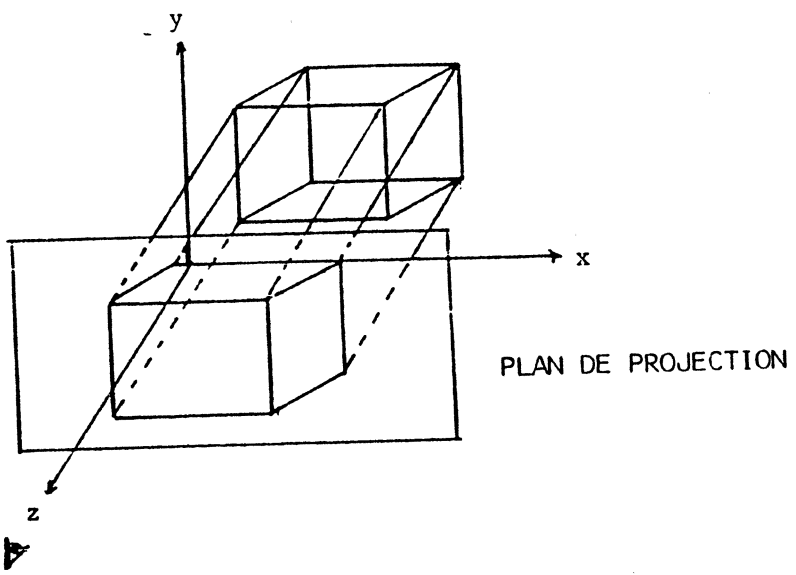


Figure III.8. Projection parallèle et élimination de parties cachées.

On trouve dans la littérature d'excellentes études sur le problème d'élimination de parties cachées ; notamment dans [SSS 74], [Bou 80], les auteurs analysent et classifient différents algorithmes pour le cas des objets à faces planes.

Nous nous consacrons seulement aux plus représentatifs d'entre eux [FoV 82], [Mar 84a] :

- a) algorithme basé sur les faces et profondeurs : Newell, Newell et Sancha [NNS 72].
- b) algorithme basé sur le tri de lignes : Watkins [Wat 70].
- c) algorithme de sous-division d'espaces : Warnock et Atherton -Weiler [War 69] , [WeA 77].
- d) algorithme basé sur le tri de pixels : z-buffer [Cat 74].

1.1.3.1. Algorithme de tri de faces et profondeur [NNS 72].

Cet algorithme est basé sur le tri de profondeur de faces. Chaque face est triée en ordre décroissant de profondeur et stockée au fur et à mesure sur

la mémoire de trame. Les faces les plus proches du point de vue sont traitées à la fin, et peuvent recouvrir les faces déjà mémorisées. Ainsi des objets de la partie arrière peuvent être affichés dans un premier temps, puis recouverts.

Cependant, le tri des faces peut donner lieu à des ambiguités dans la profondeur, ce qui nécessite le découpage de certaines d'entre elles.(figure III.9).

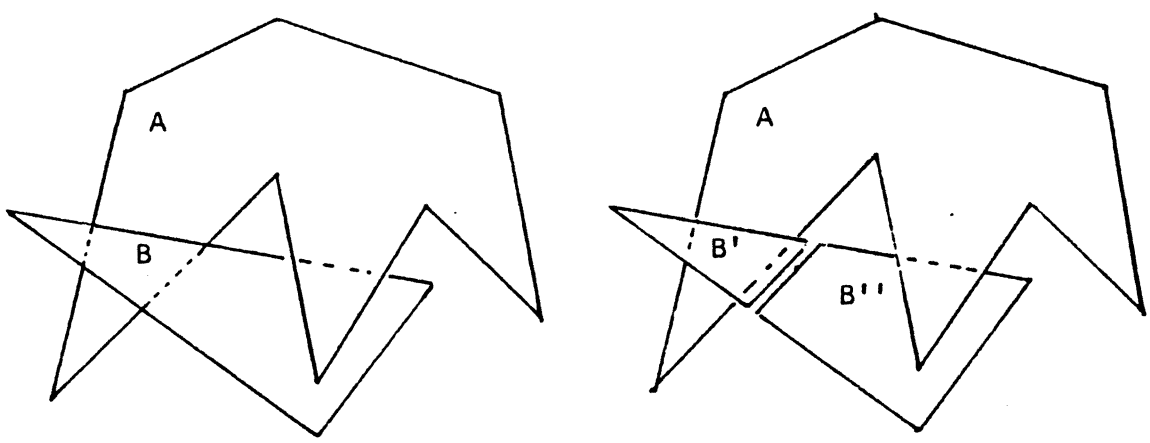


Figure III.9. Ambiguité et découpage des faces.

Après découpage, les nouvelles faces résultantes sont indépendantes entre elles. La figure ci-dessous nous montre le processus suivi:

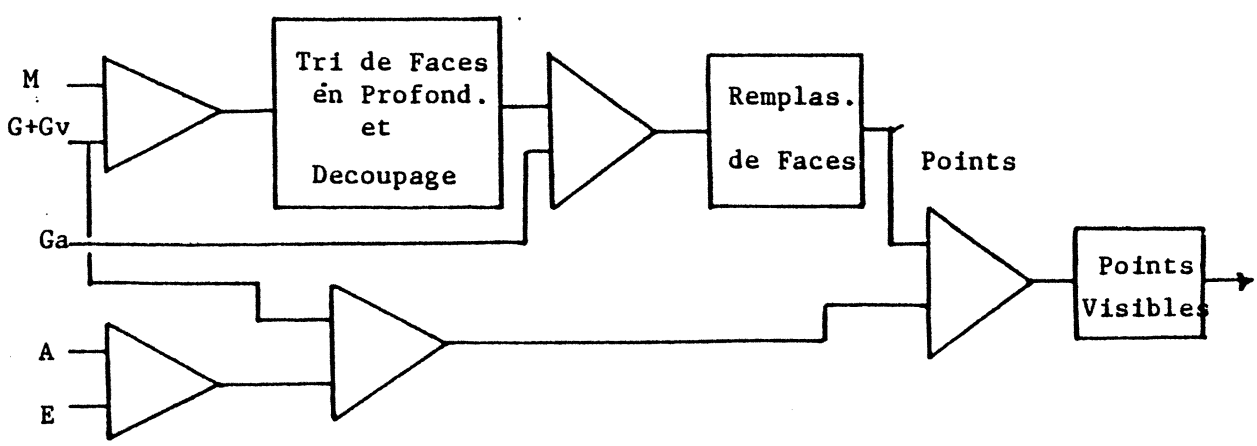


Figure III.10. Algorithme de tri de faces.

1.1.3.2. Algorithme de tri de lignes. [Wat 70].

Cet algorithme travaille sur l'espace de l'image à visualiser, c'est-à-dire sur la trame. Il est basé sur le découpage de plans horizontaux qui peuvent correspondre à chacune des lignes de balayage de la trame.

Après ce découpage, chaque ligne est étudiée tout d'abord pour trouver les segments visibles dans ce plan horizontal et ensuite pour obtenir les points de ces segments.

On se trouve face à un algorithme qui traite l'image ligne par ligne (pour balayer toute la trame). A chaque ligne, on trouve les segments des polygones visibles à l'aide d'un tri de profondeur. Les points de chaque segment visible sont trouvés directement à partir de ces segments. La figure III.10 nous montre le processus de synthèse de cet algorithme.

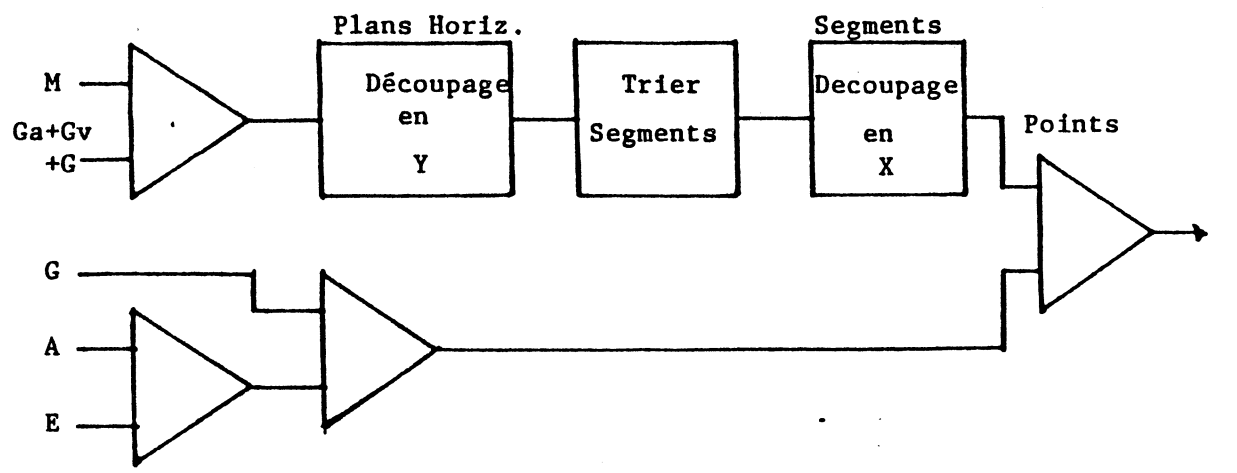


Figure III.11. Algorithme de tri de lignes.

1.1.3.3. Algorithmes de sous-division d'espace [War 69], [WeA-77].

Ces algorithmes sont basés sur la sous-division de l'espace de l'image jusqu'à trouver une surface "cohérente" du point de vue de l'information

contenue pour être traitée. Un espace est cohérent si on peut facilement déterminer le polygone ou les polygones qui sont visibles dans cet espace. Du fait de travailler sur l'espace de la mémoire d'image, la sous-division de cet espace s'arrête à la taille d'un pixel, dans le pire des cas.

Dans le cas de l'algorithme de Warnock [War 69], cette sous-division est faite en quatre carrés chaque fois. Dans chaque carré cohérent, une étude de la profondeur de faces est traitée pour les afficher dans le bon ordre.

Dans le cas de l'algorithme d'Atherton-Weiler [WeA 77], la sous-division est faite par rapport aux frontières de faces ; généralement, c'est la face la plus proche à l'écran qui est utilisée comme référence de départ. Le grand avantage de cette méthode par rapport à Warnock est la réduction du nombre de sous-divisions mais au prix d'une augmentation de la complexité de calcul. La figure ci-dessous nous montre le processus suivi.

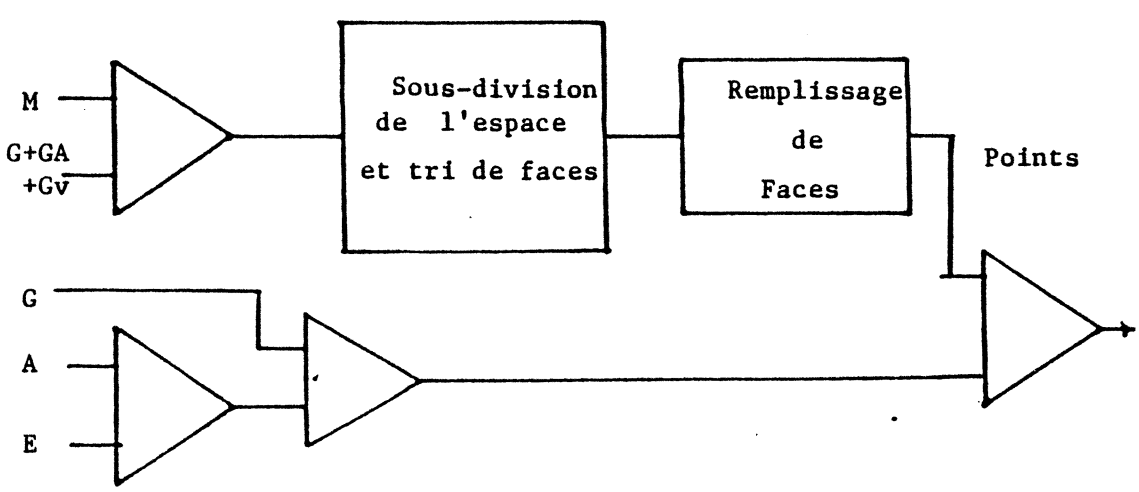


Figure III.12. Algorithme de sous-division d'espaces.

1.1.3.4. Algorithme du z-buffer, (tri de pixels) [Cat 74].

C'est l'algorithme le plus simple pour l'élimination de parties cachées ; il

travaille sur l'espace de l'image et nécessite une mémoire de trame additionnelle, dite "mémoire de profondeur".

Pour chaque point de la face, on calcule sa profondeur et la compare avec celle déjà enregistrée à la même position dans la mémoire de profondeur, il y a deux situations possibles :

i) si le point de la face est plus proche que celui déjà enregistré, on change le contenu des mémoires d'image et de profondeur par les nouvelles valeurs.

ii) si le point de la face est plus éloigné que celui déjà enregistré, on ne fait rien.

La figure III.13 schématise le processus de cet algorithme.

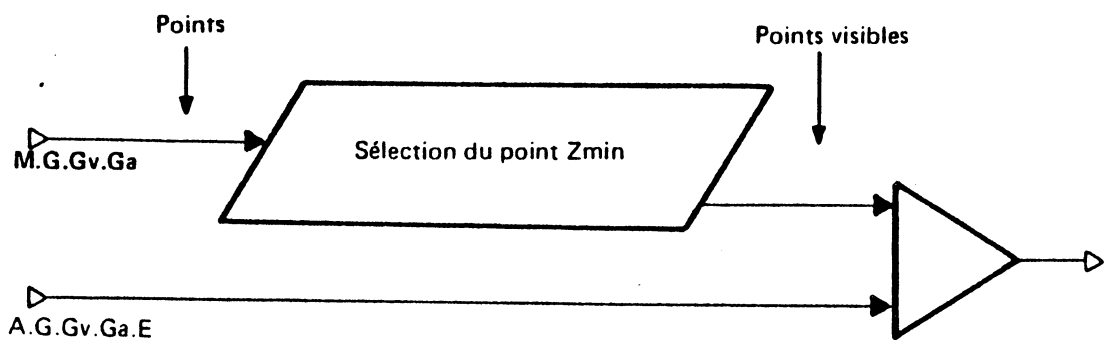


Figure II.13 Algorithme du z-buffer.

1.1.3.5. Evaluation des algorithmes.

En ce qui concerne Newell, Newell et Sancha, le calcul des points est fait tant pour les faces visibles qu'invisibles. Dans le cas de Watkins, le calcul de points est fait uniquement pour les segments invisibles. Ces deux algorithmes sont efficaces pour des scènes ayant un nombre réduit de faces. L'augmentation de la complexité de la scène nécessite plus de tests et de calculs.

Les algorithmes de sous-division d'espace nécessitent des calculs assez

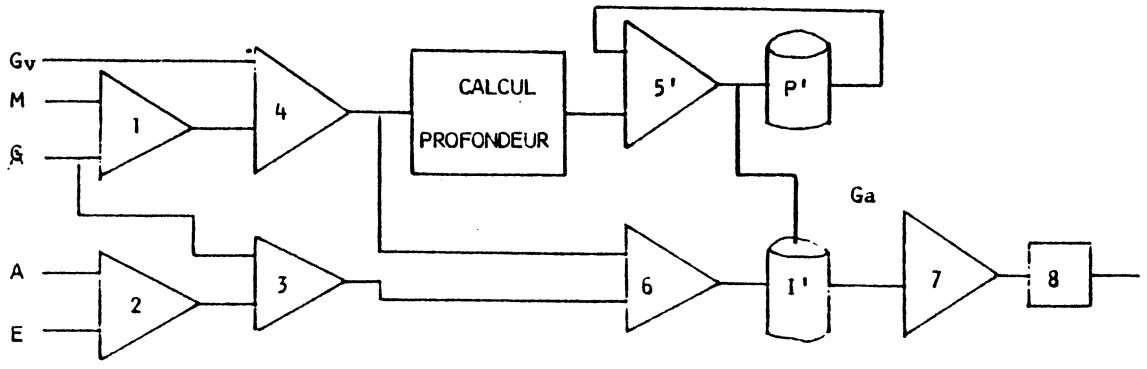
lourds et complexes, malgré la limitation du calcul aux seules faces visibles. Ils sont donc plus lent que les autres.

L'algorithme du z-buffer possède des performances constantes indépendamment de la complexité de la scène, car même si le nombre de faces augmente, le nombre de pixels par face diminue ; mais avec l'inconvénient de la mémoire de profondeur additionnelle. Il va aussi de soi que le traitement pixel par pixel peut correspondre à une portion de faces gauches.

Cet examen rapide des algorithmes met en évidence leur complexité, leurs avantages et leurs inconvénients. Si l'on s'intéresse à tout type de modélisation d'objets (faces planes et faces gauches) intervenant dans la même scène, le niveau pixel semble la seule solution facilement réalisable. Nous considérons donc l'implémentation de l'algorithme de z-buffer.

Dans le cas de l'architecture banalisée, les modules "plan mémoire" effectuent une étude de visibilité câblée, dépendante de la profondeur des plans ; ainsi, plusieurs images peuvent être superposées et incrustées au rythme de balayage vidéo.

Le cas de l'implémentation de l'algorithme du z-buffer est différent car un module additionnel de profondeur est nécessaire, ainsi que des opérateurs de tri et lecture/écriture dans ce module. La figure III.14 nous montre le processus suivi.



P' : mémoire de profondeur

I ' : mémoire d'image.

Figure III.14. Processus pour le z-buffer.

1. - Synthèse M+G : Placer les faces aux points dans l'espace 3D.
2. - synthèse A+E : Modulation de la couleur A par rapport à l'intensité d'éclairage E.
3. - Prise en compte des propriétés géométriques de la lumière (position de la source lumineuse, etc).
4. - La prise de vue :
5. - Opérateur de calcul de la profondeur minimal : z-min.
6. - Illumination des faces par rapport à l'éclairage
7. - Prise en compte des facteurs d'affichage.
8. - Eventuelle transformation finale.

Dans l'architecture banalisée, la mémoire de profondeur peut être assignée à un module plan mémoire pour devenir un plan de profondeur. Par conséquent, l'opérateur 5' de la figure III.14 doit être câblé pour améliorer les performances de lecture/écriture dans le module plan mémoire.

Le préprocessus est chargé donc de la description, modélisation, prise de vue et calcul des points qui composent les objets pour éventuellement les stocker dans la mémoire de trame. De même il calcule la profondeur de chaque point de l'objet pour la comparer avec celle déjà enregistrée dans la mémoire de profondeur.

Le post-processus lit le contenu de la mémoire de profondeur et calcule la profondeur minimale qui détermine l'écriture dans la mémoire de trame. La figure III.15 nous montre la distribution de tâches dans l'architecture banalisée.

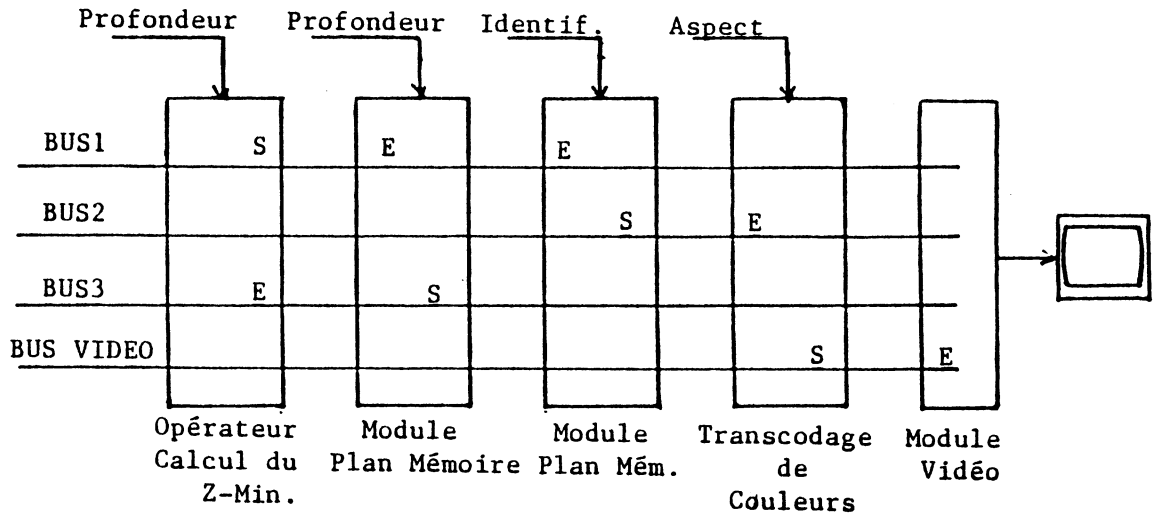


Figure III.15 . Rôle du post-processus pour l'algorithme de z-buffer.

1.1.4. Processus privilégiant l'aspect et l'éclairage final.

Dans tous les processus montrés auparavant, le calcul de l'aspect est laissé au soin du logiciel (la pénétration de cet attribut est très faible).

Cependant, si on veut améliorer le degré d'intéractivité de l'aspect final, c'est-à-dire, donner à l'utilisateur les moyens de faire varier facilement cet attribut, une pénétration de celui-ci est nécessaire et si possible jusqu'au niveau matériel.

Tant pour le cas de faces planes que pour celui de faces gauches, cette pénétration peut être faite au niveau du post-processus (figure III.16).

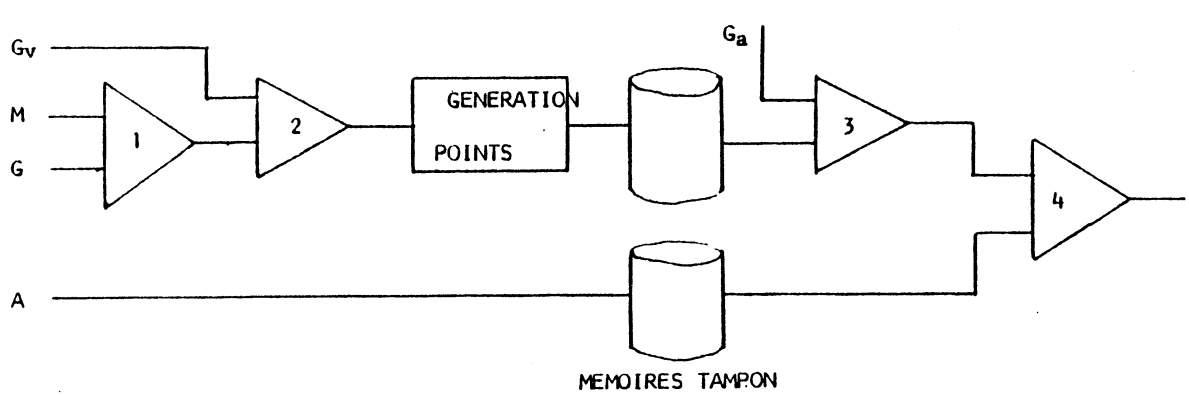


Figure III.16. Processus privilégiant l'aspect.

Dans le processus ci-dessus deux mémoires tampons sont nécessaire : l'une pour mémoriser l'image sans colorier et l'autre pour mémoriser les paramètres de l'aspect de chaque point ou chaque face. L'opérateur 3 prend en compte les paramètres d'affichage et l'opérateur 4 fait le coloriage de la scène.

Une façon d'améliorer le réalisme dans les images est de représenter les objets avec leur aspect naturel. Cette représentation nécessite l'aide de textures coloriées.

Cependant, le problème qui se pose est différent si l'on considère les faces uniformes et les faces avec textures. En effet, dans le premier cas, l'aspect est indépendant de la position des faces, tandis que dans le second, les informations morphologiques et géométriques sont nécessaires surtout au moment du pavage ou de la projection de textures.

Un autre point à souligner concerne l'éclairage de la scène. Si l'attribut d'aspect est très particulier à chaque face, l'éclairage influe sur toutes les faces de l'ensemble de la scène. Et dans le cas de textures colorées, le calcul de l'éclairage doit être fait en chaque point de la face.

1.1.4.1. Le pavage de textures

Nous ne considererons pas ici les problèmes de génération et de niveau de réalisme de textures [Zuc 75], [BIN 76], [DSS 78], [FuL 78], mais nous nous attacherons à leur projection sur les surfaces des objets3D.

Il existe plusieurs études trouvées dans la littérature pour réaliser le pavage [BIN 76], [Sch 83],[Cat 74], [FoV 82], [Cro 77]. Elles sont basées sur la non concordance de frontières (anamorphiques) des textures par rapport au pixels, autrement dit, un pixel de l'écran doit remplacer plusieurs points du modèle de textures (figure III.17).

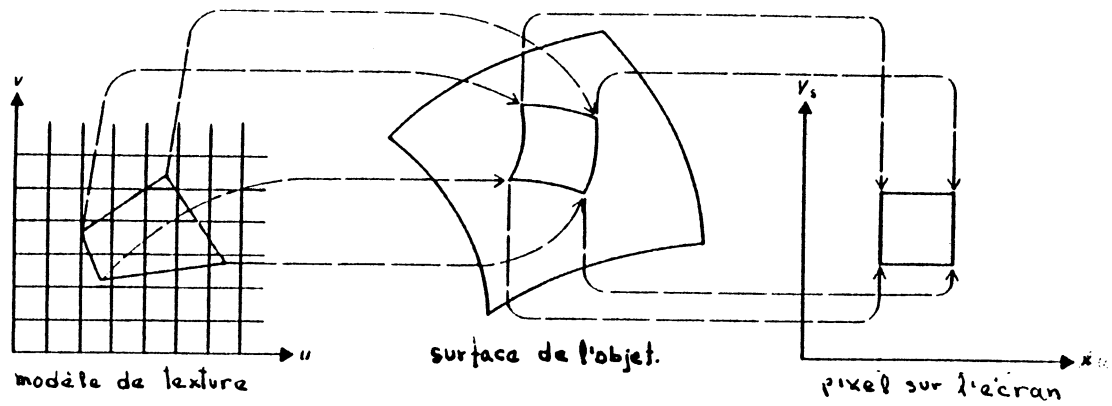


Figure III.17 Pavage des textures avec anamorphisme

Dans la technique ci-dessus on cherche à mettre en correspondance les quatre points extrêmes de la texture à paver, avec les quatre coins du pixel de l'écran. Les points de la texture sont connectés pour former un polygone. Le pavage est fait en pondérant la valeur des points par rapport à la grille interne de la texture comme on le voit sur la figure. La première étape est le pavage sur la surface de l'objet ensuite après la pondération moyenne sur les 4 coins du pixel.

Cette technique a été implémentée au niveau logiciel, notamment avec des images réelles mémorisées et les résultats visuels obtenus sont très saisissants. [FoV82], [BrN 76].

Cependant, le nombre d'opérations et le temps de calcul sont trop délicats pour qu'une implémentation matérielle et en temps réel soit possible.

Nous nous intéressons à des résultats plus modestes mais pouvant être implémentés en temps réel.

1.1.4.2. La modélisation mathématique de l'éclairage

Plusieurs facteurs interviennent dans l'illumination d'un objet dans une image synthétique.

Pour augmenter le réalisme, il est nécessaire de modéliser les propriétés physiques de la lumière dans l'environnement. Néanmoins, dans la plupart des situations, le choix du modèle mathématique d'éclairage se

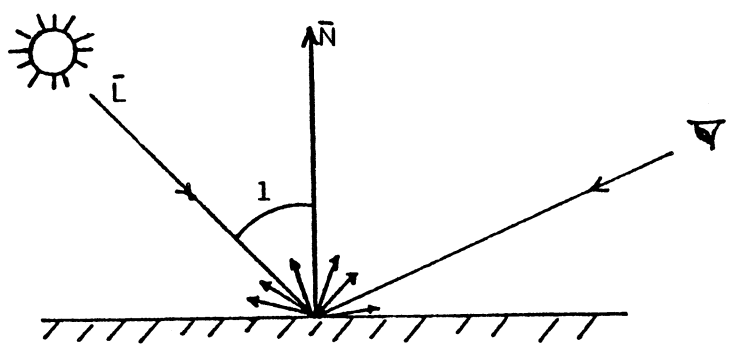
base sur des contraintes de coût, de temps d'exécution et du degré de réalisme souhaité.

Le modèle doit tenir compte aussi de la représentation géométrique des objets, ainsi que des caractéristiques des matériaux et des propriétés de sa surface.

En première approximation la contribution de la lumière dans l'illumination d'un objet est le résultat de la combinaison de deux facteurs : la réflexion diffuse et la réflexion spéculaire.

i) La réflexion diffuse "Rd" est la portion de lumière réflectée uniformément dans toutes les directions. Elle dépend de la position de la source lumineuse par rapport à la normale de la surface illuminée (figure III.18).

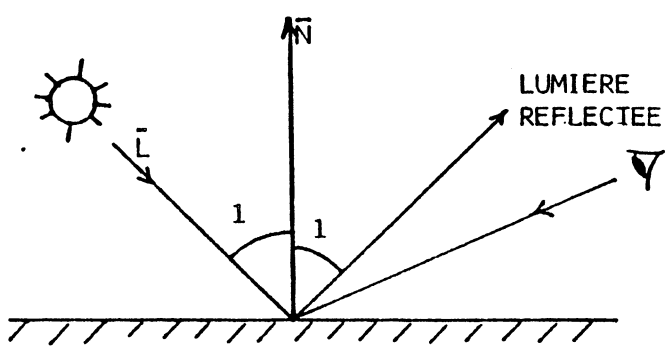
ii) La réflexion spéculaire "Rs" est la lumière réflectée dans la direction formée par l'angle opposé à l'angle d'incidence de la source lumineuse. (Figure III.19)



$\bar{L}$: Direction de la source Lumineuse

$\bar{N}$: Normale à la surface

Figure III.18 Réflexion diffuse.



1: Angle d'incidence

Figure III.19 Réflexion spéculaire.

Les modèles développés jusqu'à présent sont tous le résultat de recherches plus ou moins empiriques [Pho 75], [Bli 77], [Cot 81], [Whi 80].

La partie de lumière réflechie vers l'oeil d'un observateur est déterminée en fonction des propriétés de surface des objets et des caractéristiques de la source lumineuse.

Trois facteurs influencent les caractéristiques de la source lumineuse [VeG 84] :

i) Sa géométrie : forme de la source (ponctuelle, linéaire ou surface).

ii) La distribution spectrale: caractéristiques d'illumination, réflexion, réfraction, etc.

iii) La distribution de la luminosité : intensité de la lumière par unité de surface.

Des simplifications doivent être faites pour alléger les calculs du modèle: ainsi dans le cas d'une source ponctuelle placée à l'infini, les rayons lumineux sont parallèles et avec la même intensité sur toutes les faces de la maquette. (Figure III.20).

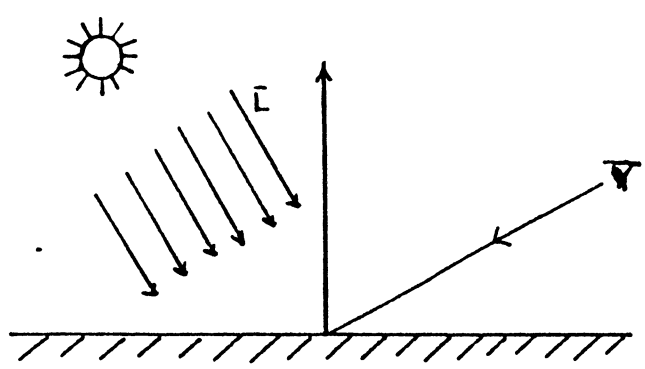


Figure III.20. Direction parallèle des rayons lumineux.

La plupart des modèles considèrent des sources ponctuelles. Des travaux récents [War 83], [VeG 84], modélisent des sources non ponctuelles ce qui entraîne un calcul plus compliqué et difficile à mettre en oeuvre.

Pour notre part nous avons adopté le modèle de Phong [Pho 75], qui prend en compte les attributs de réflexion spéculaire et de réflexion diffuse et l'intensité de la source lumineuse :

Lumière = intensité x (Rd+Rs)

où

Lumière = lumière résultante
intensité = intensité de la source lumineuse
Rd = coefficient de réflexion diffuse
Rs = coefficient de réflexion spéculaire

Ce modèle a subit beaucoup d'améliorations [Bli 77], [CoT 81]. Ces travaux ont abouti aux conclusions suivantes :

i) La réflexion spéculaire dépend normalement de la couleur du matériau et non pas de la source lumineuse.
ii) Les réflexions spéculaires et diffuses peuvent modifier la couleur originale de l'objet.
iii) Tout matériau a ses propres modèles de réflexion diffuse et spéculaire.
iiii) Dans le cas des métaux la réflexion diffuse est presque nulle.

Des réflexions diffuses et spéculaires peuvent être modélisées à l'aide d'une fonction exponentielle de la forme :

$$G = c * e^{-(x/k)2} \qquad (1)$$

où

G = coefficient cherché (Rd ou Rs)
c et k = constantes liées aux matériaux
x = angle de référence par rapport à la normale de la surface.

D'après cette formule on peut modéliser les coefficients à l'aide de la géométrie de réflexion suivante (figure III.21) :

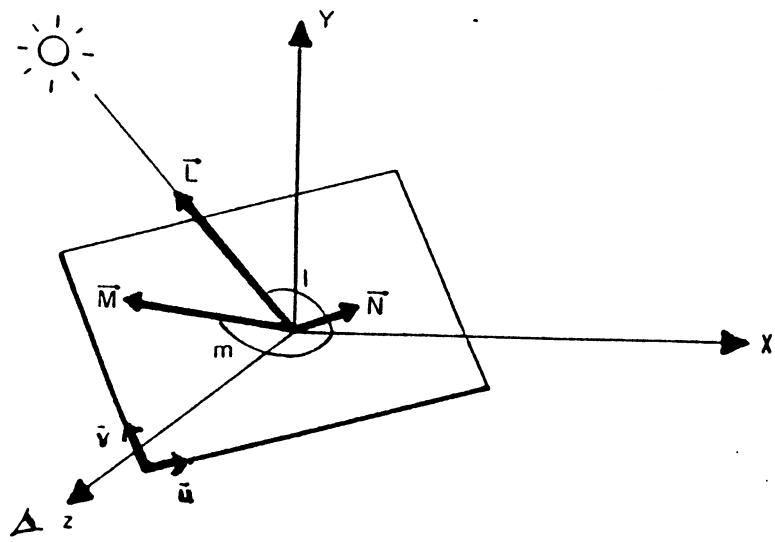


Figure III.21. Géométrie de la réflexion.

On peut considérer que l'observateur est sur l'axe z des coordonnées cartésiennes de référence. La notation suivante s'impose :

$\vec{L}$: direction de la source lumineuse

$\vec{N}$: normale à la face

$\vec{u}$, $\vec{v}$: repère de la face

$\vec{M}$: direction médiane entre la source lumineuse (L) et l'observateur (z)

I : angle entre L et N

m : angle entre M et N

Si on applique l'expression (1) aux coefficients de réflexion, on trouve :

$$Rd = Cd * e -(I/kd)2$$

$$Rs = cs * e - (m//ks)2$$

où

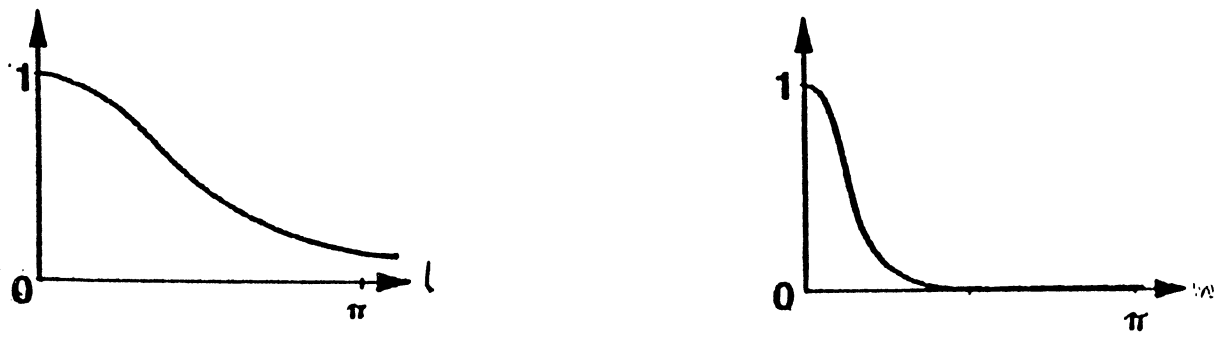
cd = constante relative au coefficient de réflexion maximale du matériau.

cs = constante relative au coefficient de réflexion maximale de la couche superficielle.

kd = constante relative à la rugosité du matériau.

ks = constante relative à la rugosité de la couche superficielle.

Cette modélisation est illustrée par la figure III.22 pour le cas d'un matériau brillant.



Réflexion diffuse

Réflexion spéculaire

Figure III.22 Coefficients de réflexion.

Le choix du modèle mathématique détermine l'architecture du module. Par la suite, nous détaillerons le modèle et les techniques d'implémentation possible pour ce module.

Par ailleurs, l'utilisation de mémoires tampons (pour les attributs synthétisés) et la pénétration pour les attributs non synthétisés, afin de privilégier l'aspect des objets et l'éclairage de la scène nécessitent de mémoriser la couleur des points constituant chaque face et l'identification géométrique de la face. Cette dernière servira au post-processus pour le calcul du pavage des textures.

La figure III.23 ci-dessous représente le processus à suivre pour privilégier l'aspect et l'éclairage de la scène.

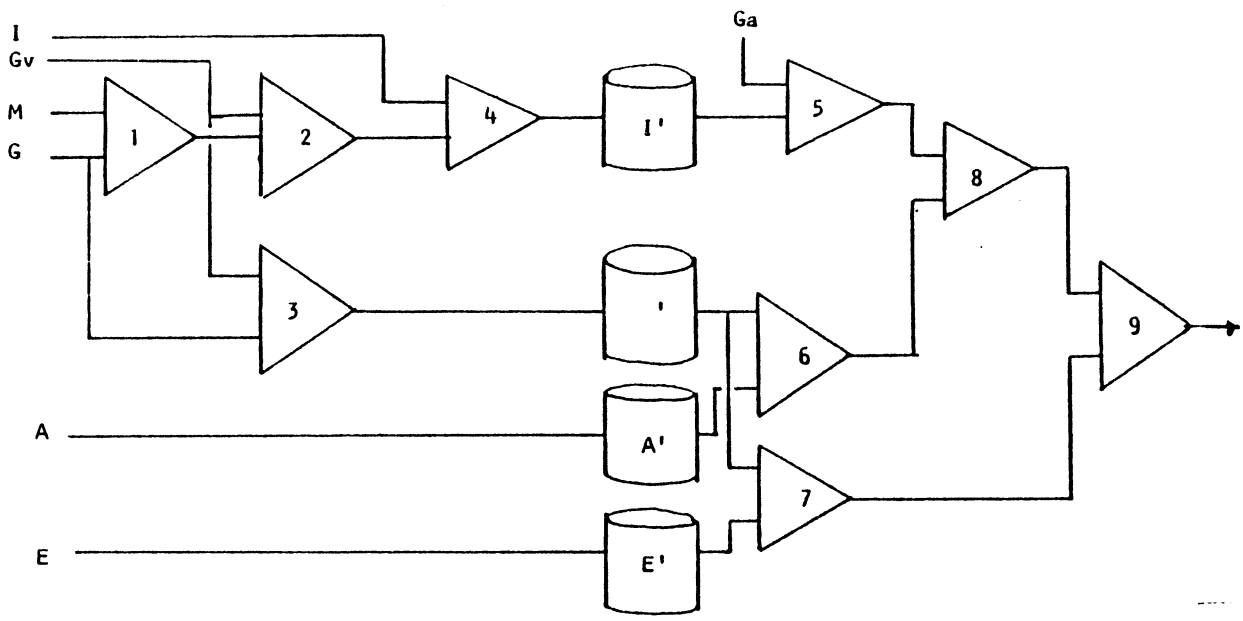


Figure III.23. Processus privilégiant l'aspect et l'éclairage.

1	: M + G	: placement des faces (ou points) dans l'espace 3D.
2	: M + G + Gv	: prise de vue de l'ensemble de faces.
3	: G + Gv	: attributs géométriques des faces.
4	: M + G + Gv + I	: assignation d'une identification aux faces.
I'		: mémoire tampon pour les identifications des faces.
G'	:	: mémoire tampon pour les paramètres géométriques
A'	:	: mémoire tampon pour les paramètres d'aspect
E'	:	: mémoire tampon pour les paramètres d'éclairage.
5	: I + Ga	: cadrage de la scène à afficher.
6	: G' + A	: prise en compte des paramètres géométriques pour le pavage.
7	: G' + E	: éclairage des faces.
8	:	: pavages des textures sur les faces.
9	:	: calcul de la couleur finale.

Le préprocessus de l'architecture banalisée sera chargé alors de la modélisation, de la prise de vue et du calcul des paramètres d'éclairage, d'aspect, et de géométrie pour remplir les différentes mémoires de trame constituant les mémoires tampons. Dans le cas de faces planes, une table de correspondance de la taille du nombre maximum de faces est une mémoire tampon suffisante. (c.f. chapitre IV. 3. 1).

Le post-processus sera chargé alors de la mise en forme des différentes informations pour arriver à la couleur du point final : pavage de textures, calcul d'éclairage , etc.

La figure III.24 nous montre la répartition des fonctions dans les modules de l'architecture banalisée.

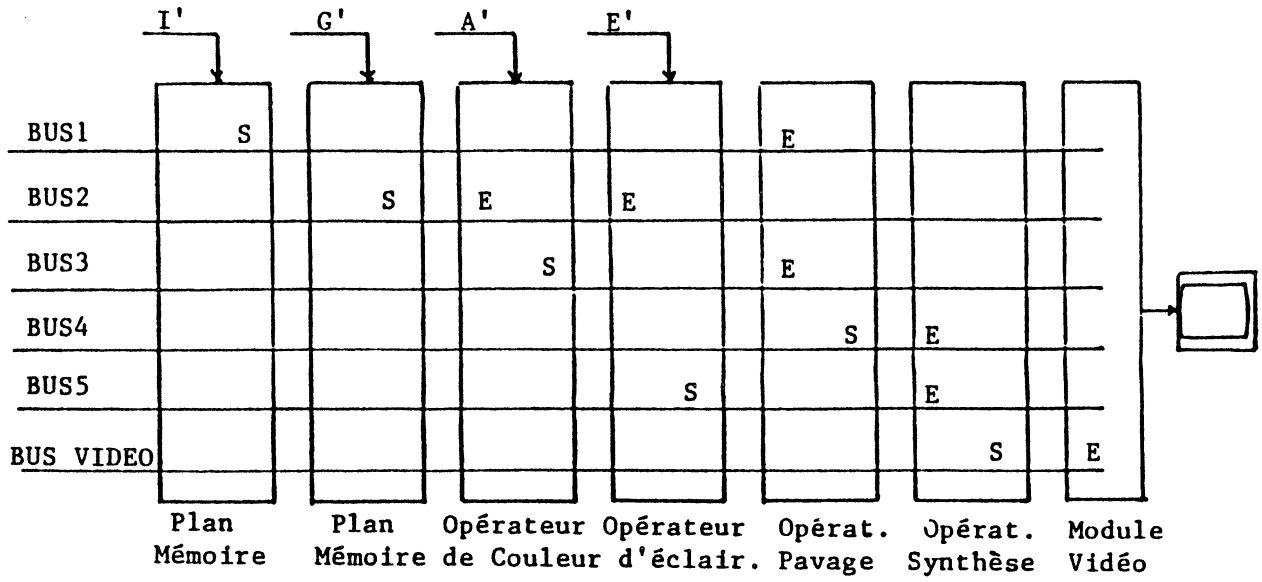


Figure III.24 Rôle du post-processus pour le calcul d'aspect et d'éclairage.

L'identification des faces I' se place dans un plan mémoire, de même que pour l'information géométrique G'. Par contre, celles de l'aspect A' et l'éclairage E' doivent se placer dans des opérateurs de mémorisation synchrone. Ces dernières mémoires sont directionnées par les valeurs de G'. La couleur déjà calculée nécessite un module pour le pavage des textures. Finalement, le pavage de textures plus les paramètres de l'éclairage sont traités par un module final qui calcule la couleur finale et la transmet au bus vidéo.

1.1.5. Processus pour le z-buffer privilégiant l'aspect.

Ce processus est constitué par l'ensemble des deux derniers cas étudiés auparavant. En effet, il prend en compte le calcul de la profondeur de points et en même temps il privilégie l'aspect des objets (textures éventuellement), ainsi que l'éclairage total de la scène.

L'introduction d'une nouvelle mémoire tampon s'avère nécessaire pour la mémoire de profondeur ainsi que l'opérateur de comparaison des profondeurs. La figure III.25 nous montre le schéma de ce processus.

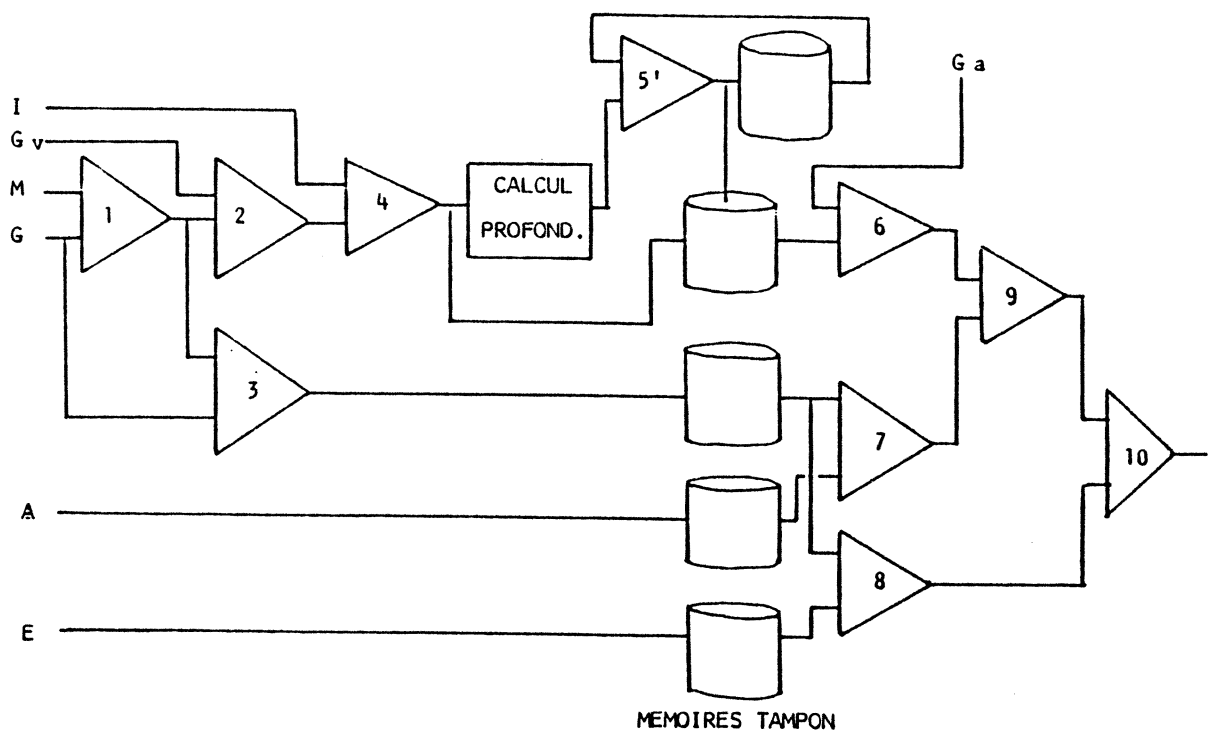


Figure III.25. Processus pour le Z-buffer privilégiant l'aspect.

Pour l'architecture banalisée, le préprocessus doit se charger de la description, la modélisation, la prise de vue, et le calcul de la profondeur des points pour les stocker dans les mémoires tampons.

En ce qui concerne le post-processus, il doit lire la mémoire de profondeur et comparer celle-ci avec la profondeur du point qui arrive. L'information résultante servira pour stocker la bonne valeur du pixel. De même il calcule le pavage des textures ainsi que l'influence de l'éclairage sur la couleur des objets.

La figure ci-dessous nous montre le schéma du processus dans l'architecture banalisée.

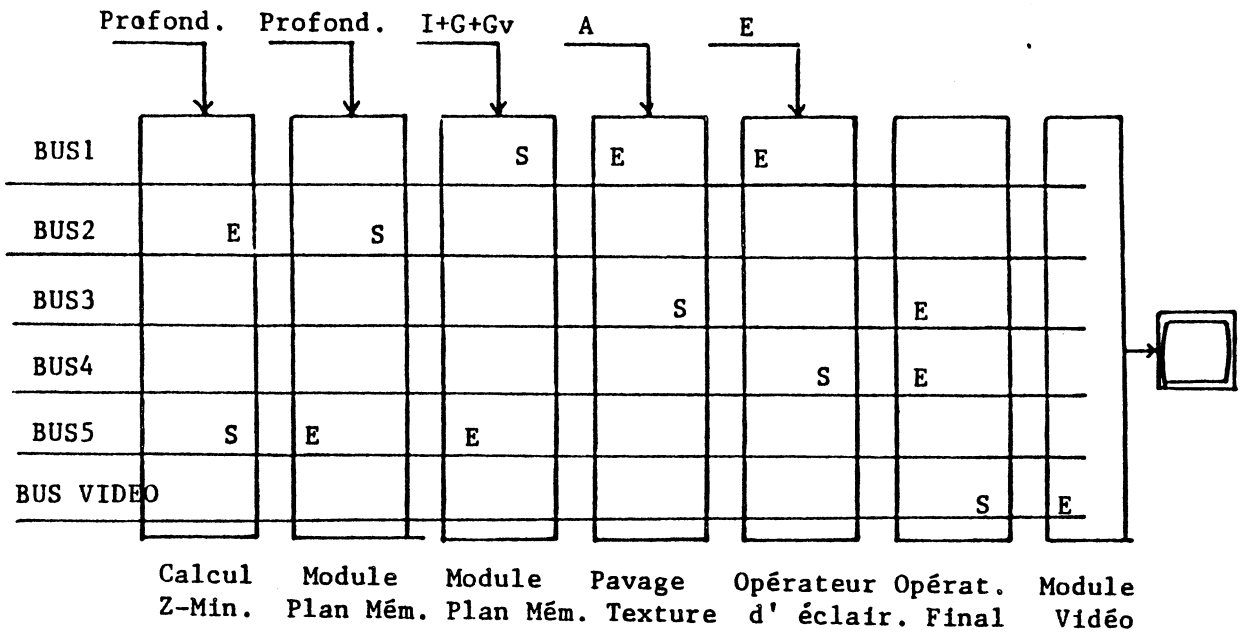


Figure III.26. Rôle du post-processus pour le z-buffer privilégiant l'aspect et l'éclairage.

Un opérateur de calcul de la profondeur minimal (z-min) est nécessaire ainsi que un plan de profondeur. L'opérateur de z-min. détermine la mise à jour de la profondeur (p) et l'identification des faces (I + G+ Gv).

Deux opérateurs de mémorisation synchrone sont encore réquéris (pour les paramètres de couleur et d'éclairage). De la même façon que dans le dernier cas (processus privilégiant l'aspect et l'éclairage final), ûn opérateur de pavage de textures est nécessaire pour augmenter le degré de réalisme, cependant une amélioration à ce processus est faite en intégrant l'opérateur de mémorisation des paramètres de la couleur à celui du pavage de textures.

Finalement, un opérateur est nécessaire pour le calcul final de la couleur à partir des textures et des paramètres d'éclairage.

1.2. Opérateurs nécessaires pour le post-processus.

Les exemples présentés nous amènent à considérer plusieurs types d'opérateurs.

Premièrement : les opérateurs de correspondance entre deux valeurs, le cas typique et le transcodage final de couleurs (information de chrominance), à partir d'un niveau de gris, résultat du processus de synthèse. Une table de correspondance nous donne la couleur rouge, vert, bleu du pixel qui va attaquer directement le moniteur vidéo. (fig. III. 27.).

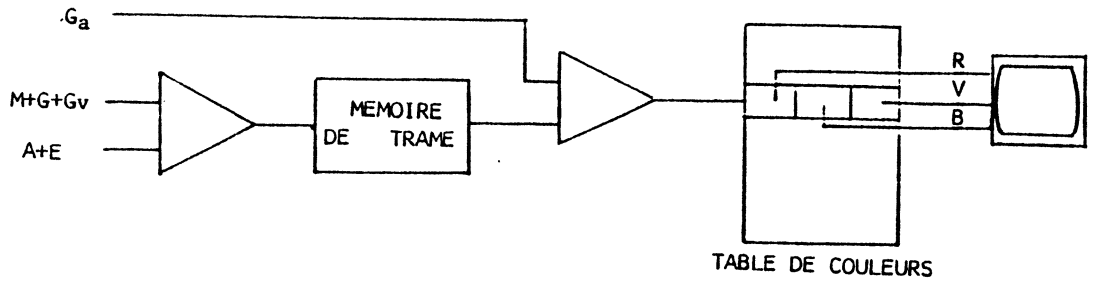


Figure III. 27. Exemple d'utilisation d'un opérateur de correspondance.

Un autre exemple serait l'utilisation de tables de correspondance comme mémoires tampons pour les informations d'éclairage (luminance) et d'aspect (chrominance). (cf. II. 2. 3.).

Deuxièmement : un opérateur de calcul de textures et de pavage sur les faces à partir des informations géomètriques de la position de la face et de ses caractéristiques d'aspect. (Fig. III. 28).

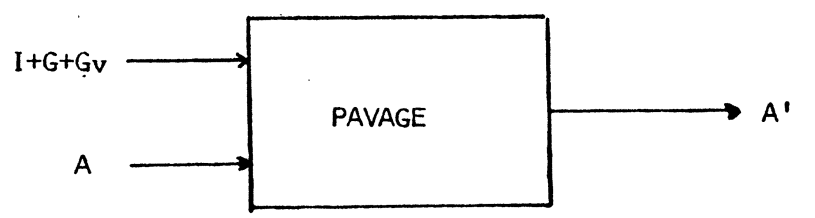


Figure III. 28. Pavage de textures.

Troisièmement : un opérateur pour le calcul du pixel le plus proche de l'observateur dans le cas de l'implémentation de l'algorithme de Z-buffer. Dans un premier temps un opérateur logique peut être implémenté à l'aide du processeur graphique et des facilités de lecture/écriture dans la mémoire de trame grace au module de communication.
Ensuite, un opérateur chargé du calcul de l'éclairage ou de l'illumination de la scène lorsque l'aspect et l'éclairage sont privilégiés. (Fig. III. 29).

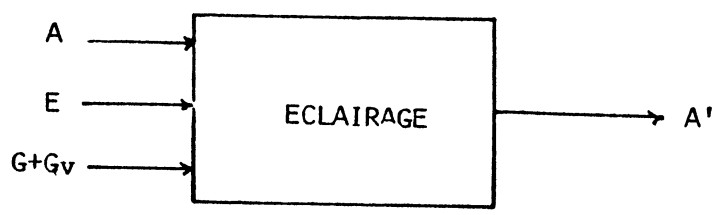


Figure III. 29. Calcul de l'éclairage de la scène.

Finalement: un opérateur pour la mise en forme de l'image à afficher : (Ga, affichage). Cette mise en forme inclue les fenêtres, clotures, translations, etc. Dans les modules "plans mémoires" de l'architecture banalisée existe un opérateur intégré qui réalise ces fonctions et par conséquent il n'est pas pris en compte dans la conception des modules du post-processus. (Cf. II. 2. 2. 2).

2. Première clasification des modules de post-processus.

A partir des exemples antérieurs, on peut classer les modules en deux catégories suivant leurs fonctions et les performances requises :

i) Les modules banalisables
ii) Les modules de calcul spécifique.

2. 1. Les modules banalisables.

D'un point de vue logique, tous les opérateurs du post-processus peuvent être émulés par logiciel. Ce sont les performances qui distinguent un opérateur programmé d'un opérateur cablé.

Il faut trouver un compromis entre la puissance d'interactivité (module cablé) et la simplicité de conception (module programmé).
Les modules banalisés recoivent les calculs fait par logiciel, notament pour émuler les opérateurs de correspondance ou les mémoires tampons. Par conséquent, nous avons deux types de modules banalisables :

i) - Les modules "plan mémoire".
ii) - Les modules "tables de correspondance".

Ainsi, dans le cas de l'algorithme par Z-buffer, le module additionnel de profondeur est simulé par un module plan mémoire.

De la même manière, si le calcul d'éclairage est fait par logiciel, on a besoin d'une table de correspondance. Mais on peut imaginer le bas niveau d'interactivité atteint avec cette méthode (par exemple, lors du changement de position de la source lumineuse).

2. 2. Modules de calcul spécifique.

Dans cette catégorie on trouve les modules qui fournissent un calcul prioritaire aux opérateurs qui en ont besoin, et pour lesquels il n'est pas possible d'utiliser un module banalisé. C'est le cas du calcul de pavage de textures dans les différentes faces qui composent la scène.

De même, le système d'architecture banalisé est perfectible, car on peut améliorer les modules pour émuler un système plus performant. Il est possible de remplacer un module pour améliorer les performances du système entier.

Notre travail donne priorité à l'aspect final de la scène. On peut considérer les modules suivants comme des exemples de modules de calcul spécial :

- Le module de calcul de textures.
- Le module de calcul de l'éclairage.

De la même façon, si on veut améliorer le calcul de l'élimination de parties cachées, un module spécial peut être implémenté pour le calcul

de la profondeur minimale des points, dans la mise en oeuvre de l'algorithme du z-buffer.

3. Seconde classification des modules du post-processus.

A partir des fonctions des modules et de leur performances, une deuxième classification peut être faite, en s'appuyant sur l'ordonnancement des opérateurs dans les différents processus de visualisation du post-processus.

Comme point de départ on prend les modules plan mémoire qui ont la fonction de mémoires tampon intermédiaires entre le pré-processus et le post-processus. Ces mémoires contiennent normalement les informations morphologiques, géomètriques et d'identification nécessaires au calcul des deux informations indispensables dans le processus convergant de visualisation : la chrominance et la luminance (cf. II. 2. 2. 1).

Les informations issues des plans mémoire peuvent être : soit la couleur finale, soit une identification des paramètres luminance-chrominance. Dans le premier cas, l'ordonnancement du processus de visualisation consiste à sortir directement sur le Bus vidéo (cf. II. 2. 2. 1) l'information contenue dans les mémoires. Dans le deuxième cas, l'ordonnancement consiste à trouver les opérateurs de transformation, et de mise en forme des deux informations, et qui servent pour le calcul de la couleur finale à afficher.

Il existe par conséquent deux types de modules : les modules intermédiaires de mise en forme et le module final de calcul de la couleur finale.

3. 1. Les modules intermédiaires.

Ces modules sont principalement les modules qui servent à transformer, à partir de tables de correspondance ou de modules de calcul spécifique, les deux informations originales : la chrominance et la luminance.

Ces modules se placent juste derrière les plans mémoire (du point de vue

fonctionnel uniquement), c'est pourquoi on les nomme "modules intermédiaires".
Nous trouvons dans cette catégorie les modules suivants :

- Les modules tables de correspondances
- Les modules de calcul de textures
- Le module de calcul de la profondeur minimale pour l'algorithme de Z-buffer.

Dans notre travail nous nous consacrons spécialement aux deux premiers modules.
Il faut remarquer que dans le cas des faces lisses, sans textures, le module de calcul de textures n'est plus nécessaire et pourra par conséquent être remplacé par un module banalisé (table de couleurs).
Le chapitre suivant traite de l'implémentation des modules intermédiaires.

3. 2. Le module final.

Ce module est chargé de calculer la couleur finale de l'image à afficher sur le moniteur vidéo.

Cette couleur doit être calculée à partir des données chrominance-luminance provenant des modules intermédiaires, ou des plans de mémorisation.
Il peut se présenter deux cas, dépendants du niveau d'interactivité souhaité :

i) - Si l'application donne priorité aux attributs d'aspect et d'éclairage de la scène, le calcul de la couleur du point final (les couleurs primaires rouge, verte et bleue) doit être fait par un module de calcul spécifique. Ce module existe et on le nomme "module d'éclairage".

ii)- Si l'application n'a pas besoin d'un niveau aussi élevé d'interactivité pourl'aspect et pour l'éclairage, les couleurs primaires rouge, vert, bleue finales peuvent être enregistrées soit directement dans un module de plan mémoire (mémoire d'image), soit à travers une table de couleurs (transcodage de couleurs à l'aide d'une table de correspondance).

La table suivante montre la classification des modules de post-processus.

	MODULE INTERMEDIAI.	MODULE FINAL
MODULE BANALISE	TABLE DE CORRESPOND.	TABLE DE COULEURES
MODULE SPECIFIQUE	CALCUL DE TEXTURES	CALCUL D'ECLAIRAGE

Table de classification des modules du post-processus.

CHAPITRE IV : REALISATION DES OPERATEURS INTERMEDIAIRES.

1. Présentation

Dans le chapitre précédent, nous avons défini les modules banalisés et les modules spécifiques du post-processus, ainsi que l'ensemble des opérateurs intermédiaires dont nous présentons maintenant la réalisation.

Du point de vue fonctionnel les modules intermédiaires nous servent à la mise en forme des deux informations qui parcourent le fond de panier (c.f.II. 2.3.1) de l'architecture banalisée : la luminance (L) et la chrominance (C).

Du point de vue matériel les modules intermédiaires suivent le chemin d'information à partir des modules plans mémoire, cette information est traitée au rythme vidéo, ce qui est à l'origine de l' architecture pipe-line des modules intermediaires.

Les contraintes temporelles et les contraintes sur l'ordre de fonctionnement imposent un ordonnancement des modules intermédiaires. Cet ordonnancement influe sur la réalisation matérielle (vitesses de réponse et synchronismes imposés).

Par ailleurs, ces contraintes ne doivent pas intervenir dans le bon fonctionnement du système, ni empêcher le réordonnancement du processus de visualisation.

De la même manière, ces contraintes influent indirectement sur les performances du système. En effet, un module qui a beaucoup de contraintes nécessite la mise en oeuvre de techniques matérielles pour les satisfaire (voir les annuler), ce qui rend la réalisation encombrante et compliquée .

Les modules intermédiaires que nous étudierons sont le module "table de correspondance" et le "module de textures".

Nous présenterons les contraintes d'ordonnancement avant de passer à la réalisation proprement dite.

2. Contraintes d'ordonnancement.

La nature des informations issues des plans mémoire peut appartenir à l'une des deux catégories suivantes :

- la couleur proprement dite.
- une information banalisée différent de la couleur.

Dans le premier cas le post-processus se limite à trasmettre directement cette information au module vidéo , qui la convertit en signaux analogiques pour le moniteur de visualisation.

Dans le deuxième cas, les informations de "luminance" et de chrominance" en entrée du post-processus ne sont ni triées ni mises en forme convenablement pour être fournies au moniteur de visualisation.

Le but des modules intermédiaires et du module final est cette mise en forme. En particulier, les modules intermédiaires servent à distinguer et traiter les deux informations de chrominance et de luminance, qui sont fournies séparement en entrée du module final de synthèse (module d'éclairage).

Ces informations étant indépendantes l'une de l'autre (pour le traitement aussi), il est nécessaire de les synchroniser avant de les traiter dans le module final.

Cette synchronisation peut se faire de trois manières : soit au niveau des modules intermédiaires, soit au niveau du module final ou soit en distribuant entre ces deux niveaux.

I) Synchronisation au niveau des modules intermédiaires : Cette situation nécessite la mise en oeuvre des techniques spéciales dans les modules intermédiaires que permettront la sortie des deux flots en synchronisme.

Une technique très simple pour implémenter cette synchronisation utilise des registres de pile FIFO ("first in first out"). Chaque module aura un registre de pile rempli avec un retard propre au module (retards différents en général). Les registres seront lus simultanément par le module final.

Cette solution est illustrée sur la figure IV.1. ci-dessous.

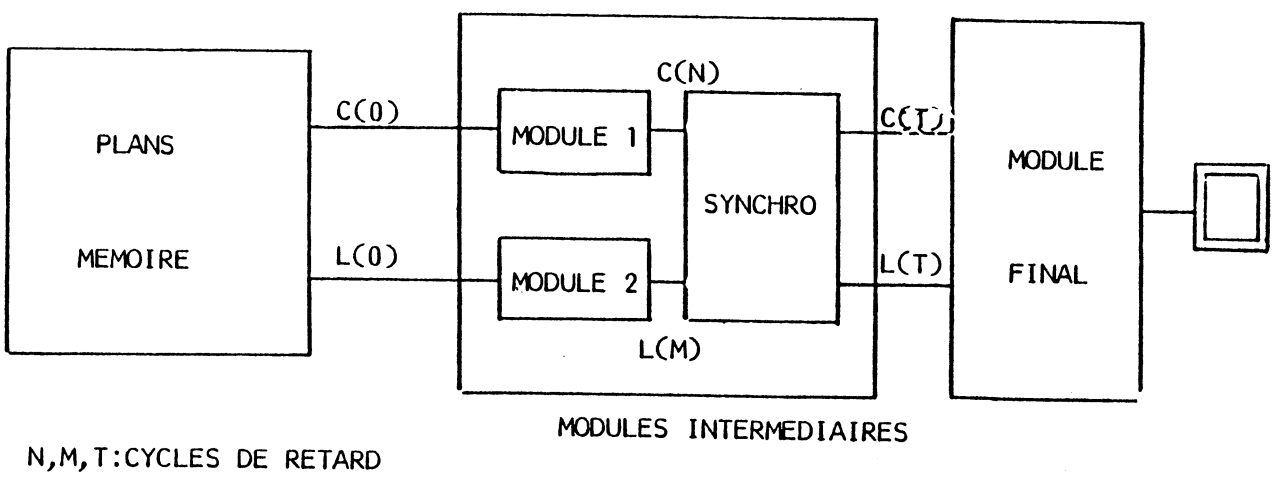


Figure IV.1. Synchronisation au niveau des modules intermédiaires.

i.i.). Synchronisation au niveau du module final : cette solution transporte le problème dans le module final. En effet, les sorties des différents modules intermédiaires pourront arriver avec des temps de retard complètement différents, ce qui oblige à implémenter des techniques servant à la synchronisation au niveau du module final. Par exemple la technique de registre de pile FIFO La figure IV.2. ci-dessous nous montre cette solution.

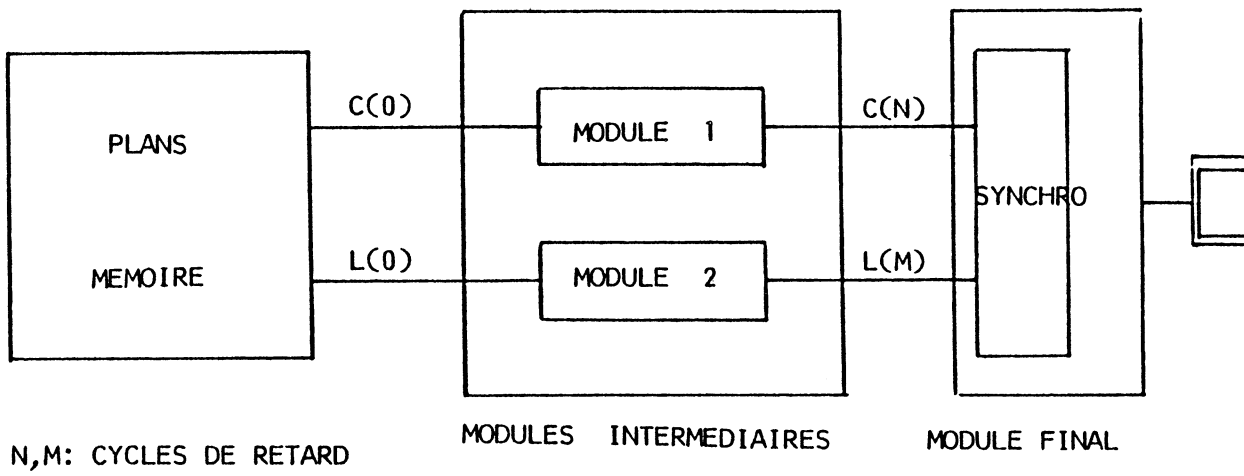


Figure IV.2. Synchronisation au niveau du module final.

I.I.I.) synchronisation distribuée : Cette solution est à l'origine des contraintes d'ordonnancement, car d'une part elle fixe la position fonctionnelle des modules et d'autre part elle empêche le réordonnancement matériel de la partie du post-processus. Cette solution est présentée dans la figure IV.3.

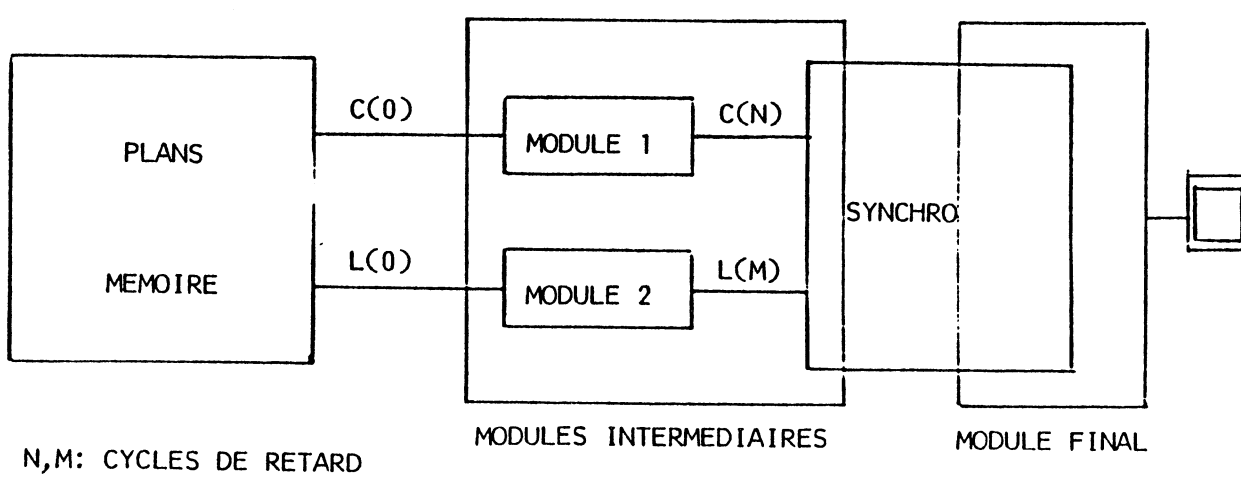


Figure IV.3. Synchronisation distribuée entre les unités.

Des essais ont montrés qu'il était plus facile et plus simple de réaliser, la synchronisation au niveau des modules intermédiaires. Cela permet de rendre indépendants les modules intermédiaires du module final, dans la mesure où les performances des uns n'empêchent l'autre de fonctionner.

Les informations de luminance et chrominance sont indépendantes et peuvent donc être mises en forme séquentielle (série) ou en parallèle. Cependant, les contraintes de vitesse, imposées par le rafraîchissement du moniteur vidéo, conduisent à une mise en forme parallèle, ainsi qu'à l'utilisation de la technique de pipe-line.

Chaque module met un temps fixe pour traiter l'information. Compte-tenu de l'architecture pipe-line du système, ce temps de traitement est divisé en un nombre bien précis du cycles de pipe-line.

La synchronisation entre modules se fait en égalisant les temps de traitement sur la référence du temps de traitement le plus lent, c'est-à-dire que le nombre de cycles pipe-line du module le plus lent est pris comme base de référence. Cela permet un calcul simple des retards (fonction du nombre de modules intermédiaires).

Cela veut dire aussi, que l'ordonnancement des processus de visualisation ne tiendra pas compte de la façon dont les modules intermédiaires sont construits, mais seulement de leurs fonctionalités.

Ainsi le module table de correspondance possède un retard de traitement similaire à celui du module de textures.

3. Réalisation des modules intermédiaires.

3.1. Le module "Table de Correspondance".

Ce type de module en tant qu'unité banalisée peut réaliser le traitement pour la chrominance ou pour la luminance, en changeant uniquement le contenu de sa mémoire.

Par exemple, à partir de l'identification (I) d'un objet issu d'un module plan mémoire, on peut choisir sa couleur à travers une table de couleurs. De la même façon, les paramètres d'un modèle spécifique d'éclairage (correspondant au matériau de l'objet) peuvent résider dans une autre table de correspondance.

L'interface d'entrée des modules table de correspondance doit alors permettre de choisir au minimum l'un quelconque des deux bus banalisés d'entrée, par exemple bus1 et bus2 (c.f. II.2.3.2.).

Par contre, l'interface de sortie doit aiguiller soit la luminance, soit la chrominance. Par conséquent, une table de correspondance doit sortir sur deux bus banalisés, par exemple bus3 et bus4. En outre, dans le cas d'utilisation d'un de ces modules en fonction de table de couleurs, sans module d'éclairage, les sorties doivent être connectées directement en bus vidéo.

Par ailleurs, l'utilisation de 24 bits de sortie en couleurs, nécessite le couplage de deux tables de correspondance. L'interface de sortie tiendra compte de ce fait et sortira aussi sur la partie de poids fort (12bits) ou de poids faible (12 bits) du bus vidéo. La figure IV.4. ci-dessous nous montre le schéma d'entrées-sorties du module.

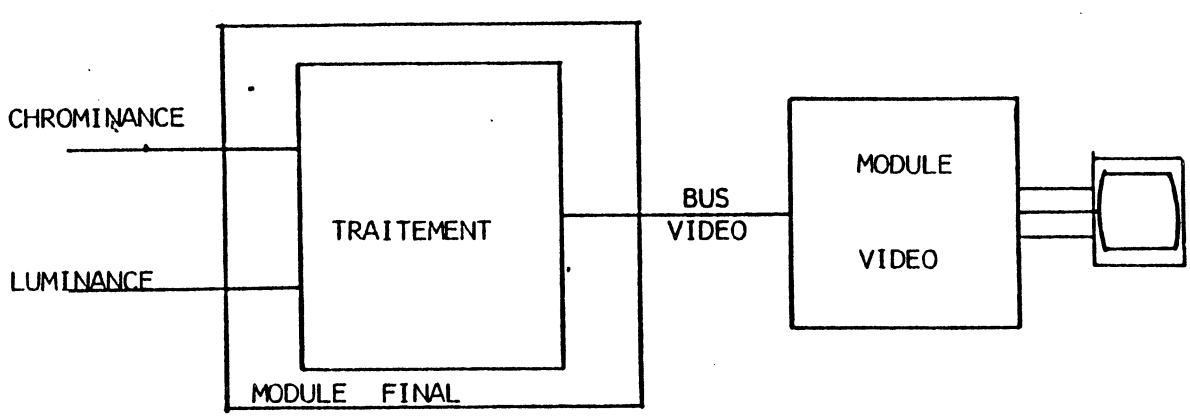


Figure IV.4. Interface entrée/sortie du module table de correspondance.

3.1.1. Techniques d'implémentation du module.

Il s'agit ici de réaliser le module banalisé en forme de mémoire vectorielle.En effet, le module doit pouvoir contenir une information de correspondance exprimée en 12 bits (selon convention des bus banalisés).

Cette information est fonction de l'opérateur assigné au module et joue un rôle bien précis dans le processus de visualisation. Le processeur graphique est chargé de calculer et de remplir cette mémoire.

Par ailleurs, les informations d'entrée sont exprimées en 12 bits (du fait de la largeur des bus banalisés). Elles demandent alors 4K éléments de correspondance.

Il faut remarquer que ce module émule des mémoires tampons synchrones (c.f. II.2.1), par conséquent les informations sont exploitées au rythme de balayage vidéo.

Les caractéristiques principales du module sont :

- 4K éléments de mémorisation exprimés sur 12 bits, et répartis dans une mémoire vectorielle.
- 2 bus banalisés d'entrée : bus1, bus2.
- 2 bus banalisés de sortie : bus3, bus4.
- le bus vidéo de sortie, exprimé sur 24 bits ou sur 12 bits (cela dépend de la résolution de couleur choisie par l'utilisateur).
- de mécanismes de lecture/écriture.

Nous considérons deux techniques différentes pour implémenter ce module, dépendant de l'utilisation de mémoires à haute intégration ou dépendant de mémoires à haute vitesse de réponse.

Les mémoires à haute intégration sont généralement des mémoires à

réponse lente, par contre celles à haute vitesse ont une intégration faible. Par la suite nous évaluerons de ces deux techniques.

3.1.1.1. Technique d'échelonnage de mémoires lentes.

Cette technique nécessite l'utilisation de mémoires RAMs à haute intégration. Ces mémoires sont généralement de type dynamique et nécessitent des cycles supplémentaires pour leur rafraîchissement.

Leur grand inconvénient est la faible vitesse de réponse, de l'ordre de 280 nsec pour des mémoires de 16 K x 1 ou 64 K x 1 et même pour celles de 256 k x 1 bits.

Par contre, leur grand avantage est le coût des boîtiers qui n'est pas du tout élevé (il y a un rapport de 4 entre le prix des mémoires dynamiques et celui des mémoires statiques).

Dans le cas des modules du post-processus, la vitesse de rafraîchissement du moniteur vidéo impose des cycles de pipe-line de l'ordre de 75 nsec.

Nous utilisons certaines astuces pour combler la faible vitesse de réponse des boîtiers :

- l'utilisation de techniques de transformation série-parallèle en entrée, l'accés à plusieurs mémoires en parallèle et finalement d'une transformation parallèle-série en sortie. Cette technique est utilisée assez couramment dans les systèmes graphiques à balayage trame [Whi84].

- L'utilisation de la même technique d'accès en alternance au 4 colonnes, employée auparavant dans la réalisation des plans mémoire [c.f. II. 2.2.4.].

Le grand (et plus important) inconvénient des deux techniques mentionnées ci-dessus est l'impossibilité d'accéder à des emplacements

adjacents dans la table de correspondance. En effet, l'accés à un emplacement adjacent dans le même boîtier de mémoire doit attendre jusqu'au prochain accés permis au même boitîer. Typiquement, les cycles d'accès en parallèle sont de l'ordre de 4 (de 75 nsec chacun). Cela signifie que la fréquence d'accès au même boîtier ne doit pas être supérieure à quatre cycles.

L'accés aléatoire au module est donc très pénalisé.

La technique proposée pour combler cette difficulté est de quadrupler l'information (et par conséquent le nombre de tables de mémoire), de telle façon que l'accès soit cyclique en continu.

Le prix des mémoires RAMs dynamique permettent cette solution. Si on utilise des mémoires dynamiques de 2 K x 8 bits, la multiplication par quatre des informations nécessite 16 boîtiers.

Ainsi, vu de l'extérieur, l'accès aux mémoires prend 4 cycles de pipe-line mais avec l'avantage d'accès aléatoires à la table de correspondance. (figure IV. 5).

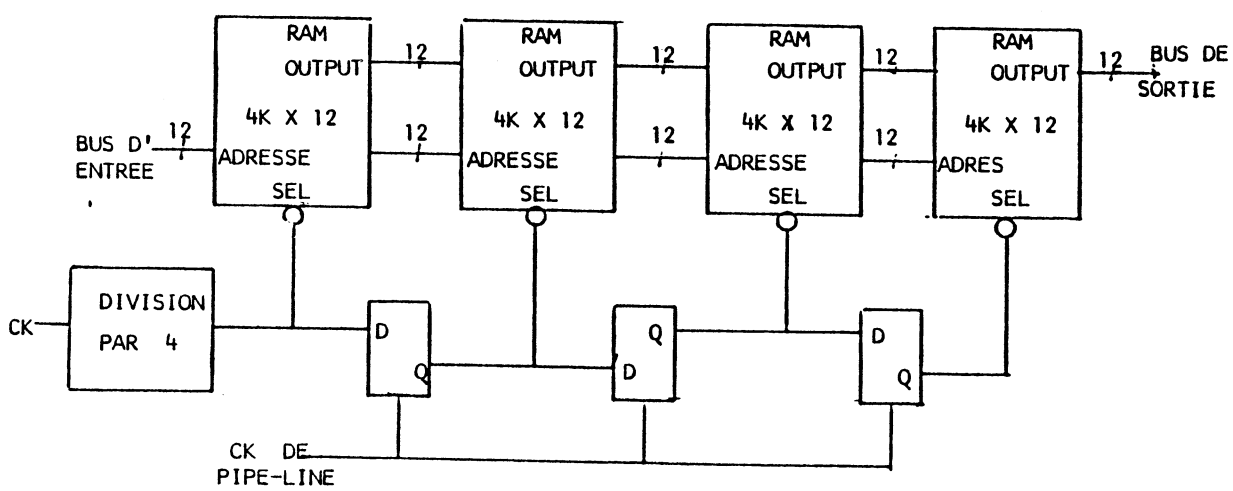


Figure IV. 5. Technique de multiplication des mémoires.

3.1.1.2. Technique à mémoires rapides.

Dans cette technique, l'utilisation de mémoires RAM statiques s'impose.

Le grand avantage de ces mémoires est le temps de réponse qui peut descendre jusqu'à 30 nsec en technologie MOS et sans nécessité de rafraîchissement.

Par contre, l'inconvénient est sa faible densité d'intégration. Au début de cette étude, les boîtiers de mémoire pouvaient atteindre une intégration de 1 K x 1 bit avec un bon rapport de performances/prix.

Un autre inconvénient est la puissance consommée, de l'ordre de 1 Wt. par boîtier.

Toutefois, le développement de la technologie de circuits intégrés et des techniques d'implantation de mémoires d'état solide, notamment en technologie MOS, ont accru considérablement cette intégration, tout en conservant un rapport performances/prix assez raisonnable.A l'heure actuelle, les mémoires RAM statiques de 4 K x 1 bits sont très courantes et les mémoires 16 K x 1 bits le deviennent de plus en plus.

Tous ces facteurs nous permettent d'intégrer l'accès aux différents boîtiers de mémoire en un seul cycle de pipe-line de 75 nsec, sans aucune circuiterie additionnelle. (voir figure IV.6).

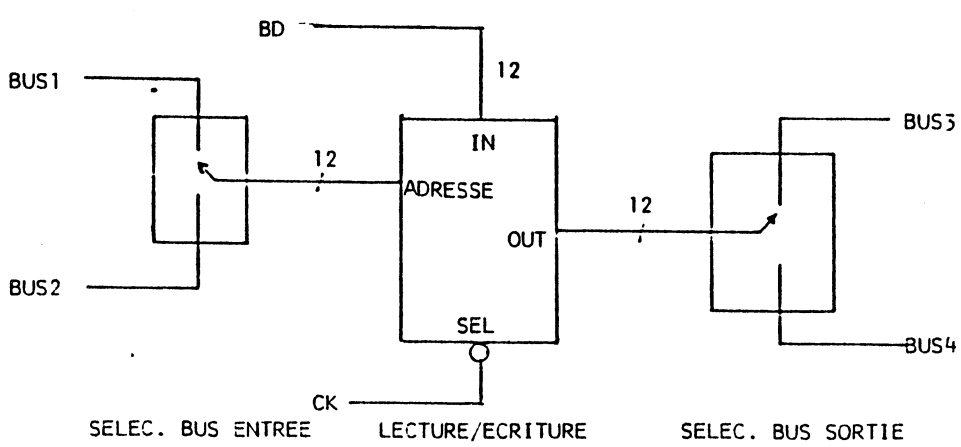


Figure IV.6. Technique de mémoires rapides.

3.1.2. Réalisation du module.

Vu la simplicité de la technique à mémoires rapides, c'est la solution qui a été adoptée.

3.1.2.1. Modes de fonctionnement.

Les modules du post-processus sont censés s'adapter à la puissance de l'architecture banalisée. Cela nécessite une capacité de réordonnancement des modules en fonction de l'application.

En effet, les applications qu'on trouve sur la synthèse d'images sont très variées : de la simple image calculée complètement par ordinateur jusqu'à l'image calculée en temps réel. Par exemple une maison construite pour l'architecture et une image pour un simulateur de vol.

Cependant, si on veut qu'un système soit souple du point de vue fonctionnel, le système doit être capable d'accepter plusieurs applications. Et si en plus il est performant, il doit accepter et combiner différentes applications en même temps.

L'architecture banaiisée par sa conception peut accepter différentes applications et les combiner en temps réel. Cela veut dire que différents processus de visualisation peuvent se dérouler en même temps et que le système suit ses changements au niveau du post-processus. Ainsi une voiture synthétisée par le système peut se déplacer dans un décor prise par une caméra T.V.

Les modules du post-processus ont le pouvoir de combiner des processus de visualisation en temps réel. Cela nécessite que les modules peuvent travailler sur plusieurs modes conmutables eux aussi en temps réel. Cette contrainte est satisfaite grâce à l'implémentation du bus de profondeur (c.f. II.2.4), qui indique à tous les modules du post-processus la nature du pixel à traiter.

Chacun des modules possède donc des dispositifs de lecture et

d'interprétation du bus de profondeur. L'interprétation est faite à l'aide d'une mémoire RAM reprogrammable dite "mémoire de priorité" qui exprime pour chaque pixel le traitement à effectuer dans le module. La progammation dépendra donc de l'application.

Dans le cas particulier des modules intermédiaires, ceux-ci doivent fonctionner en plusieurs modes, lesquels nous allons détailler ci-dessou:

-i) le mode "calcul" : C'est le mode normal de fonctionnement, pour lequel l'information d'entrée est traitée et transformée suivant la fonction du module.

- I.I.) Le mode "valeur par défaut" : Ce mode requiert l'utilisation d'une valeur de référence bien précise qui soit indépendante des valeurs d'entrée.

- I.I.I) Le mode "transparent" : Ce mode assigne la même valeur d'entrée à la sortie sans aucun traitement. Ce mode est nécessaire pour deux raisons :

1ère) la combinaison du processus de visualisation peut provoquer le mélange des images (stockées au niveau des plans mémoire), qui ont différents niveau de traitement et qui ont (ou n'ont pas) besoin du traitement fait par un module précis.

2ème) pour synchroniser les informations qui ont différents chemins de parcours, notamment dans le cas des combinaisons mentionnées ci-dessus.

Par exemple, une voiture synthètisée qui se promène dans une payssage prise par caméra. Le système suis en transparence le payssage tandisque la voiture suis le calcul.

IIII) Le mode "d'entrée/sortie" : Ce mode choisit le bus banalisé d'entrée et le bus banalisé de sortie dans l'ordonnancement des modules. En plus il résoud le conflit des sorties simultanées de plusieurs modules sur un même bus banalisé grâce à l'utilisation des registres trois-état et collecteur ouvert ("three states" et "open colector").

iiiii) Le mode "bus vidéo" : Le mode est utilisé pour valider l'information de sortie sur le bus vidéo. Il est notamment utilisé avec le bus d'entrée/sortie mentionné auparavant.

Après avoir défini les modes de fonctionnement des modules intermédiaires (et par conséquent le module table de correspondance), il est nécessaire d'assigner à la mémoire de priorité les sorties de configuration aux différents modes.

Il y a 5 lignes de sortie dans la mémoire de priorité :

1 ligne	choix du bus entrée	bus1/bus2
2 lignes	choix du mode :	00 : calcul
		01 : transparent
		10 : valeur par défaut
1 ligne	choix du bus sortie :	bus3 ou bus vidéo poids Fort
1 ligne	choix du bus sortie :	bus4 ou bus vidéo poids Faible

3.1.2.2. Réalisation matérielle.

Tout module de l'architecture banalisée est vu par le processeur graphique comme un ensemble de registres dans l'espace d'adressage de mémoire du microprocesseur. Cette solution à l'avantage de permettre la communication permanente entre le processeur graphique et les modules.

Les accès externes aux modules sont gérés par le module de communication et par la partie contrôle propre à chaque module. Cette partie contrôle est identique dans tous les modules et a été étudiée dans le chapitre II (c.f. II.2.2.2.).

Le module de communication fournit les données nécessaires pour les accès externes [c.f.II.2.2.2.]. Ces informations sont :

- un bus d'adresse externe :	BAX : 12 bits
	BAY : 12 bits
- un bus de données :	BD : 12 bits

- 8 signaux de fonctions : F0.. F7 1 bit chacun
- 8 registres programmables : R0... R7, 12 bits chacun
- 1 signal d'accès externe EXT 1 bit

Les modules intermédiaires table de correspondance et calcul de textures ont été implémentés simultanement. Un étude a montré que le nombre de cycles de pipe-line nécessaire pour le bon traitement de l'information est 3. Dans le cas du module qui nous intéresse, la répartition des fonctions est la suivante :

- 1er cycle : sélection du bus banalisé d'entrée.
- 2ème cycle : sélection et exécution du mode de fonctionnement du module : calcul, transparence, valeur par défaut.
- 3ème cycle : sélection du bus de sortie.

Le troisième cycle sert egalement pour sélectionner la partie du bus vidéo correspondant au poids fort ou au poids faible des 24 bits de sortie.

Le schéma de réalisation est montré par la figure de l'annexe D.

Dans le premier cycle, le bus de profondeur de 4 bits arrive à la mémoire de priorité de 16 x 5 bits. Le premier bit de sortie de cette mémoire sert à sélectionner le bus d'entrée à l'aide d'un multiplexeur rapide, cela est fait en tenant compte des retards possibles dans les lignes de bus banalisés bus1, bus2 qui parcourent le fond du panier. Il faut remarquer que les accès externes en écriture ou en lecture au module sont prioritaires, par conséquent la ligne EXT est toujours prise en compte, d'abord.

Dans le deuxième cycle les deux bits suivants de la mémoire de priorité sont décodés pour exécuter la fonction assignée au module : calcul, transparent ou valeur par défaut. Le mode calcul est réalisé en lisant la table de mémoire de 4K x 12 bits à partir de l'adresse fournie en entrée. Ces mémoires étaient composées dans un premier temps de 12 boîtiers de 4K x 1 bit, après elle ont été réduites à 3 boîtiers de 4K x 4 bits de 35 nsec de temps d'accès. Cette réduction a permis d'intégrer deux tables

de correspondance dans une même carte. L'écriture des mémoires est faite à l'aide de la ligne EXT, du bus de données (BD) et d'une fonction provenant de la partie contrôle du module (FO).

Dans le troisième cycle, la sélection du bus de sortie se réalise avec les deux derniers bits de la mémoire de priorité, on peut ainsi sélectionner soit le bus 3, soit le bus 4, soit aucun des deux, soit les deux. Cette selection depende de l'application énvisagée.

3.2. Le module de "Calcul de Textures".

Ce module est spécifique du point de vue fonctionnel. En effet, il vient se placer dans le chemin de l'information de la chrominance pour pouvoir assigner pour chaque pixel sa couleur correspondante dans un modèle de texture. Cette modèle de texture est un matrice de points coloriés et l'ensemble represente une la forme d'une texture.

Il vient remplacer un module table de correspondance pour améliorer les performances du système : un objet représenté par sa texture naturelle à l'aide d'un module spécifique est plus réaliste que le même objet avec une couleur uniforme (table de correspondance).

La représentation des objets à l'aide de textures est donc un moyen d'augmenter le réalisme des objets. Il est d'autant plus saisissant que les textures utilisées sont proches du réel.

Cependant, dans une scène donnée, il peut y avoir plusieurs objets, chacun possédant des aspects complètement différents, ce qui entraîne l'utilisation de plusieurs modèles de textures dans la même application.

La façon d'éliminer cette contrainte est l'utilisation d'une mémoire de textures. Cette mémoire contiendra les modèles de simulation de plusieurs d'entre elles. Les avantages de cette solution sont variés :

i) les textures peuvent être les plus réalistes possibles (par exemple une image réelle).

ii) un seul modèle peut servir à plusieurs objets.

iii) le changement de texture d'un objet est réalisé instantanement en changeant uniquement le pointeur d'entrée à la texture souhaitée.

Grâce à sa souplesse, c'est cette solution qui a été adoptée dans l'architecture banalisée.

Le grand problème posé par l'utilisation de textures est leur pavage sur les surfaces 3D des objets composant la scène.

Les techniques pour réaliser le pavage sont déterminantes pour l'implémentation du module.

Dans le paragraphe, nous avons étudié quelques techniques pour réaliser ce pavage. Nous nous attacherons maintenant à la réalisation propement dite du module .

3.2.1. techniques de pavage de textures.

A l'heure actuelle, aucun système autre que le système Hélios, (à notre connaissance) ne prend en compte le pavage des textures par matériel. Nous considérons qu'il n'y a pas de problèmes d'anamorphisme, c'est-à-dire qu'un point du modèle de la texture correspond à un pixel de l'écran, ce qui permet de simplifier notre problème (figure V.9.)

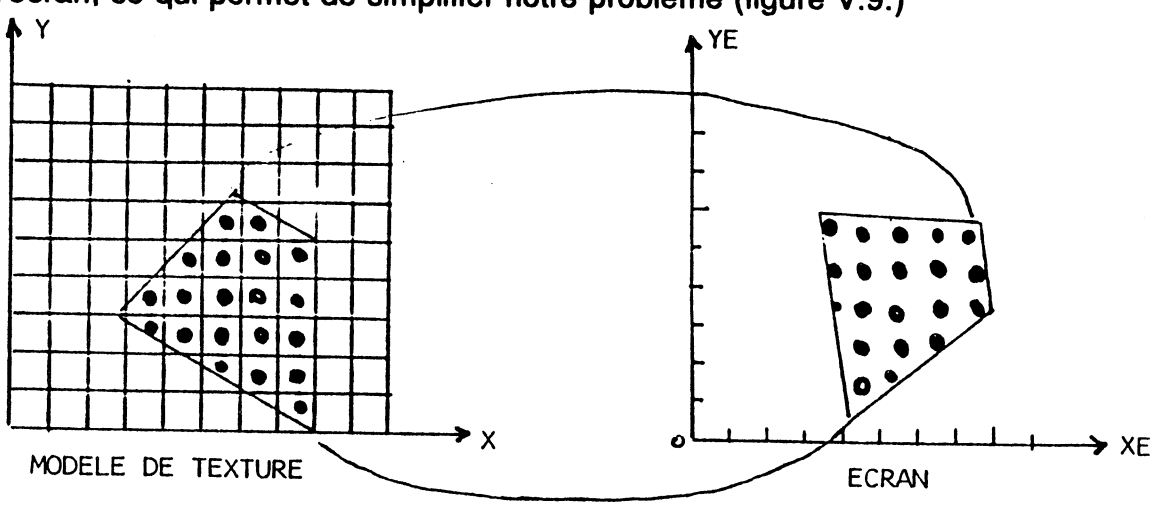


Figure IV.7. Pavage de textures sans anamorphisme.

3.2.1.1. Pavage par projection cavalière.

Cette technique prend la notion de parcours du balayage cavalier, c'est-à-dire la transformatione de vecteurs ou lignes droites au format de trame. Or, notre cas est un peu différent car nous voulons lire le contenu des points dans un modèle de textures .

On considère le pavage comme le parcours de vecteurs ayant la même direction (pente) que l'un des bords de la surface, obtenue par projection sur le plan de l'écran de la surface en trois dimensions. (figure IV.8).

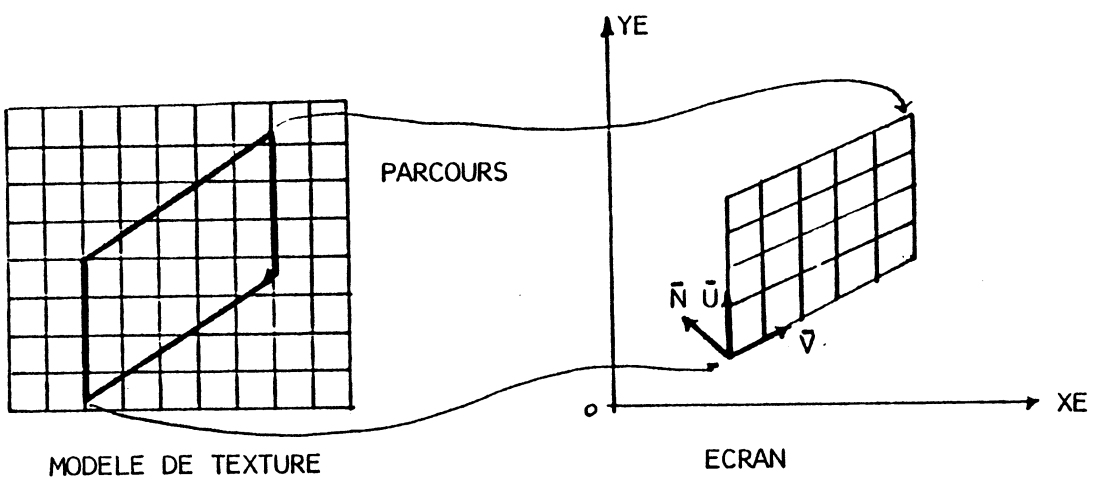


Figure IV.8. Pavage par projection cavalier.

On peut utiliser alors des algorithmes classiques de génération de droites en système à balayage trame pour la lecture des pixels, par exemple celui de Bresenham [Bre 65].

A partir des données géométriques de la face : son repère 3D (u,v) voir figure IV.8), on calcule la pente "m" du segment à parcourir dans le modèle de la texture.

Par ailleurs, des problèmes inhérents à la génération de lignes droites se posent, tel que des pentes supérieure à 45° [Fov 82], et qui doivent être

prises en compte au moment de l'implémentation. En effet, le passage de la pente "m" en 45° signifie un changement du singe de "m", qui répercute en l'incrémentation du pas en x ou y.

Mais les grands problèmes qui se présentent sont :

- d'une part la détermination du début de parcours pour la lecture dans la mémoire du modèle. En effet, le calcul du pavage doit se faire au rythme du balayage trame et par conséquent le premier pixel trouvé appartenant à la face étudiée ne correspond pas forcément au début du repère du modèle de la texture.

- d'autre part, des discontinuités trouvées dans des faces génèrent des discontinuités de parcours dans le modèle de la texture qui sont difficiles à évaluer (figure IV.9.)

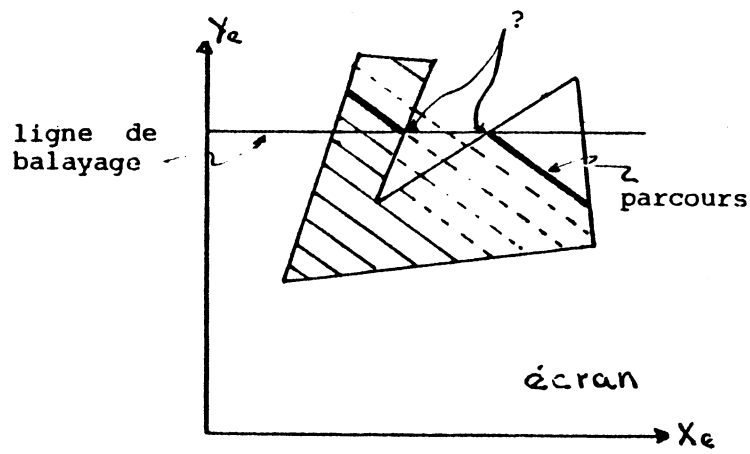


Figure IV.9 Discontinuités sur les faces.

Finalement, un autre problème inhérent au parcours de lignes est celui de la distance parcourue. En effet, un segment vertical ou horizontal est plus petit qu'un segment ayant une certaine pente (le rapport est $\sqrt{2}$ plus grand dans le pire des cas) comme on voit sur la ligne IV.10.

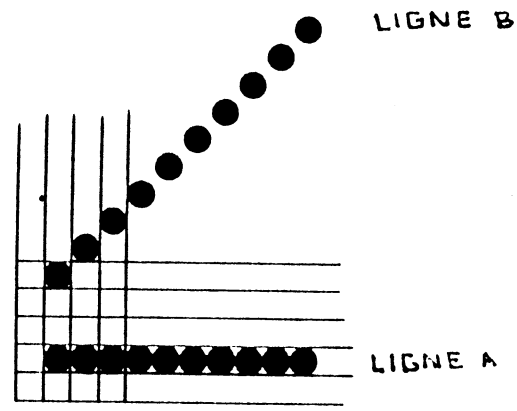


Figure IV.10 Problème d'allongement d'intensité des pixels.

Cette déformation est nettement perceptible et gêne parfois àl'utilisateur dans la compréhension de l'image.

L'utilisation d'algorithmes de génération de lignes pose aussi le problème des accès aux mémoires qui contiennent le modèle de la texture. En effet, l'incrémentation en x ou en y toujours d'un pas nous impose une cadence d'accès à chaque cycle de pipe-line aux emplacements de mémoire adjacents.Le problème est similaire au cas de tables de correspondance et par conséquent les solutions aussi.
(c.f. IV.3.1.1.).

3.2.1.2. Pavage par utilisation d'une matrice de projection

Cette solution a été envisagée et réalisée par [Fer 81]. Elle consiste à obtenir la projection de la face polygonale 3D (son repère u,v) dans le repère de l'écran à l'aide d'une matrice "M" dite matrice de projection.

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

d'où on peut arriver aux coordonnées d'écran xe, ye :

$$\begin{vmatrix} Xe \\ Ye \end{vmatrix} = M \begin{vmatrix} xf \\ Yf \end{vmatrix}$$

(Xf, Yf) coordonnées de la face 3D.

La figure IV. 11. nous montre cette démarche.

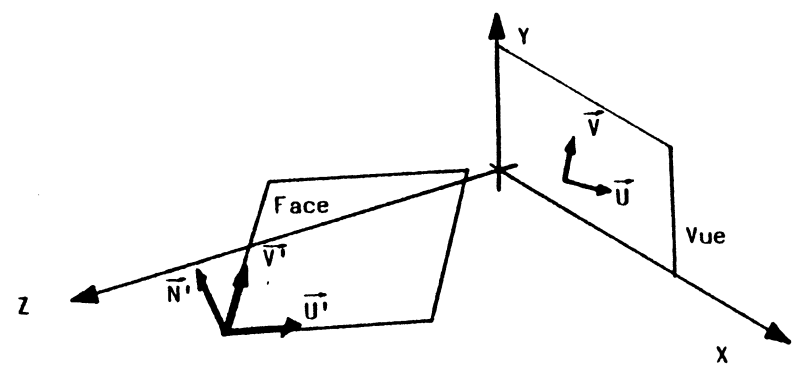


Figure IV. 11. Projection du repère face-écran.

Cependant, le fait de calculer le pavage au rythme du balayage de trame nécessite la détermination du point dans la face (Xf, Yf) à partir des coordonnées de balayage d'écran (Xe, Ye) :

$$\begin{vmatrix} Xf \\ Yf \end{vmatrix} = M\text{-}1 \begin{vmatrix} Xe \\ Ye \end{vmatrix}$$

Cette matrice inverse M-1 est calculée en avance pour chacune des faces et stockée dans des tables internes pour être exploitée au rythme de balayage vidéo.

Le pavage consiste à calculer, à partir des repères des faces (Xf, Yf), les coordonnées d'entrée à une mémoire contenant le modèle de la texture

assignée à la face. Ce pavage nécessite deux facteurs supplémentaires :

- la taille de la texture : taille
- le repère de début du coté de la texture dans la mémoire : (Xd, Yd), car cette mémoire contient plusieurs textures.

Finalement les coordonnées d'entrée dans la mémoire (Xtex, Ytex) sont calculées de la façon suivante :

Xtex = Xd * taille + Xf modulo taille
Ytex = Yd * taille + Yf modulo taille

Les avantages de cette méthode sont variés :

i) le calcul du pixel est fait en temps réel.
ii) le changement des coordonnées du repère de début du carré de la texture (xd, yd) origine un changement de texture instantanément.

Par contre divers inconvénients apparaissent :

i) très encombrant au niveau matériel, notamment lors des multiplications cablés de la matrice inverse M-1 et les coordonnées écran.
ii) difficile à mettre en oeuvre.
iii) gestion de la mémoire de textures assez délicate. En effet, ce système utilise des modèles de texture de taille variable [Sar 82], ce qui nécessite un algorithme d'optimisation de mémoire lors de la combinaison de textures à tailles différentes.

Si on veut appliquer cette technique au cas de l'architecture banalisée plusieurs difficultés apparaissent :

i) La texture pavée sur une face defile sur celle-ci au moment de translater le plan mémoire où elle se trouve . Pour le résoudre, on a besoin d'une table additionnelle contenant les repères de balayage des

plans mémoire (table de déplacements), qui vont s'ajouter à l'adresse de balayage écran : Xe, Ye. (figure V.12).

ii) Vue la taille de la mémoire de textures : 512x512 points, [Fer 81], l'utilisation des mémoires à haute intégration mais à faible vitesse de réponse (280 nsec), pose de problèmes lors de lectures à points adjacents. Une solution est l'utilisation d'une mémoire de textures plus petite et de boîtiers mémoire à plus faible intégration (mais plus rapides en réponse).

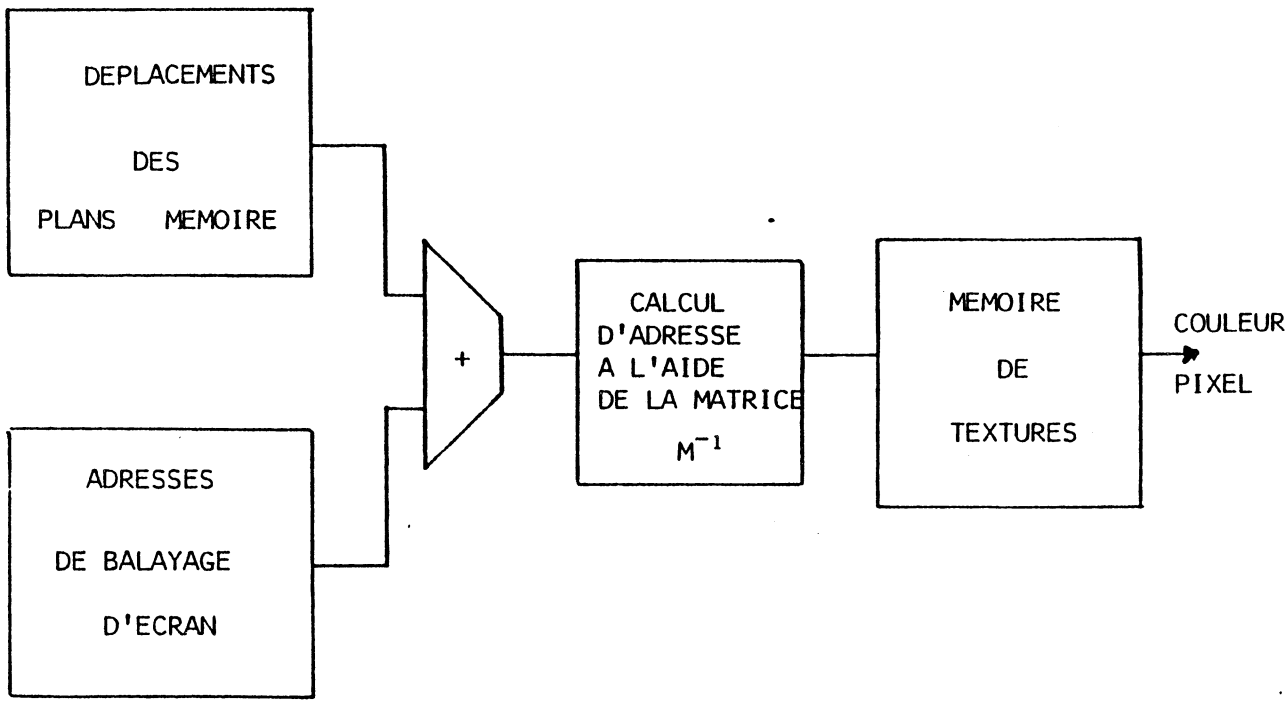


Figure IV.12 Technique de pavage par une matrice.

3.2.1.3. Pavage par logiciel.

Cette méthode utilise la puissance du processeur graphique pour calculer en temps différé des informations qui seront exploitées, par la suite, en temps réel.

Le logiciel prend en compte :

- le déplacement des plans mémoire.

- le calcul des matrices M-1
- le calcul de la projection et par conséquent le calcul des coordonnées (Xf, Yf) projetées sur la face.

Cependant pour exploiter toutes les informations en temps réel, il doit y avoir des mémoires synchronées et des mécanismes de lecture au rythme du balayage vidéo. Ces informations nous permettent de calculer l'adresse dans la mémoire de textures pour trouver la couleur du point correspondant.

Dans l'architecture banalisée on peut utiliser un plan mémoire additionnel pour mémoriser les projections de chacune des faces qui composent la scène.

Le calcul de la couleur d'un pixel nécessite dans ce cas :

- l'identification de la texture à paver
- la projection des faces issues d'un plan mémoire.

Le module "calcul de texture" est chargé de calculer en temps réel :

- La taille de la texture à paver
- Le début du repére carré de la texture dans la mémoire : (Xd, Yd).
- Le pavage de la texture sur la ou les faces, à partir des deux informations ci-dessus.

Le processeur graphique sera donc chargé du calcul et du remplissage des informations de projection dans un module plan mémoire additionnel. Des variations dans la structure de la scène nécessiteront la ré-ecriture des projections dans ce plan mémoire.

Les principaux avantages de cette méthode sont :

- Simplicité de mise en oeuvre.
- Calcul en temps réel de la couleur des pixels.
- Possibilité de changer la texture en temps réel.

Et les inconvénients :

- Utilisation d'un module plan mémoire additionnel.
- Prétraitement de la chrominance (calcul des projectiones) au niveau logiciel qui est assez lourd pour le processeur graphique.

La figure ci-dessous nous montre cette solution:

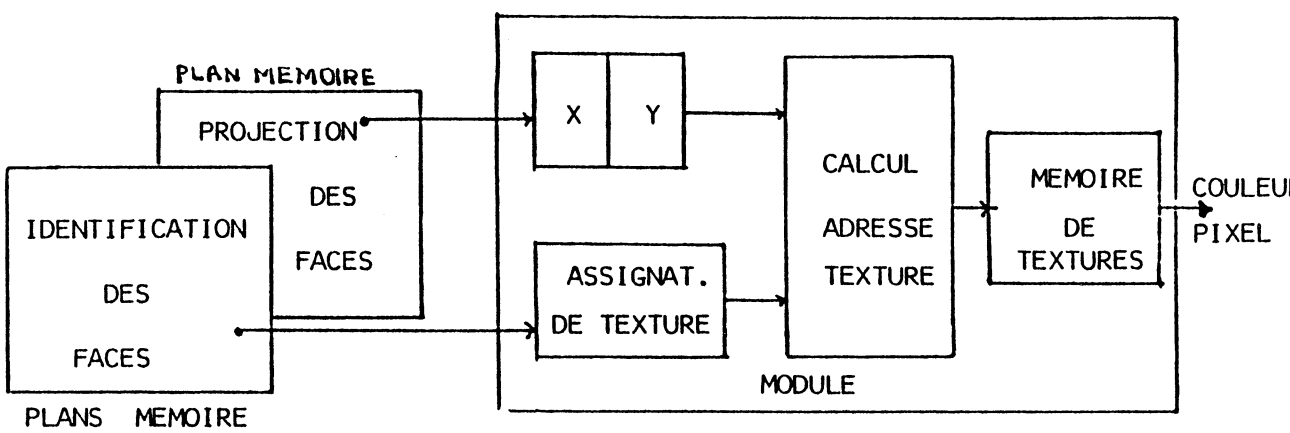


Figure IV. 13. Pavage par logiciel.

3. 2. 2. Réalisation du module.

Le choix de la technique de réalisation du module de calcul de texture prend en compte :

i) La facilité d'implémentaion.
ii) Les performances requises.
iii) L'utilisation optimale des facilités offertes par l'architecture banalisée.

Le pavage par projection cavalier peut atteindre les performances requises par le module, au prix d'une implémentation assez compliquée pour résoudre les problèmes inhérents à la génération de lignes droites.

Le pavage cablé, à l'aide de la matrice inverse M-1, est très performant du point de vue fonctionnel, mais très encombrant du point de vue

matériel.

La solution par projection logicielle est simple à mettre en oeuvre, peut atteindre les performances requises, mais ce qui est le plus important, elle exploite les performances de l'architecture : la puissance de calcul du processeur graphique et du remplissage de mémoires de trame en temps réel (cf. II. 2. 2. 4.). En plus, le système GETRIS est perfectible, ce qui signifie que l'addition des nouveaux processeurs de calcul (tels que microprocesseurs en tranche), viendra améliorer les performances de cette méthode de pavage.

Compte tenu des facteurs mentionnés ci-dessus, la méthode de pavage par logiciel a été choisie pour être implantée.

Un autre facteur à considérer est le fait d'avoir dans une même scène des objets qui nécessitent des textures mais aussi des objets qui ont des couleurs uniformes.

La mise en oeuvre de la mémoire de textures impliquerait l'itilisation d'un modèle de textures pour chaque face y compris les faces uniformes. On leur attribuerait inutilement un modéle de texture (car le modèle contiendrait la même couleur).

La solution pour éviter ce gaspillage de mémoire est d'intégrer dans la même mémoire de textures une table de couleurs qui associe à tous les points de la face uniforme un seul emplacement dans cette table. La différenciation de la nature de la face (uniforme ou avec texture)peut se faire au moment de l'affectation de la texture à l'aide de l'identification de la face.

En ce qui concerne de la mémoire de textures, l'expérience a montré que des textures matricielles 16x16 points était le minimum suffisant pour avoir des bonnes définitions de textures. On a donc choisi d'avoir 64 textures de 16x16 points. Chaque point est exprimé en 12 bits (4 bits pour chacune des couleurs primaires rouge,vert, bleu), (figure IV. 14).

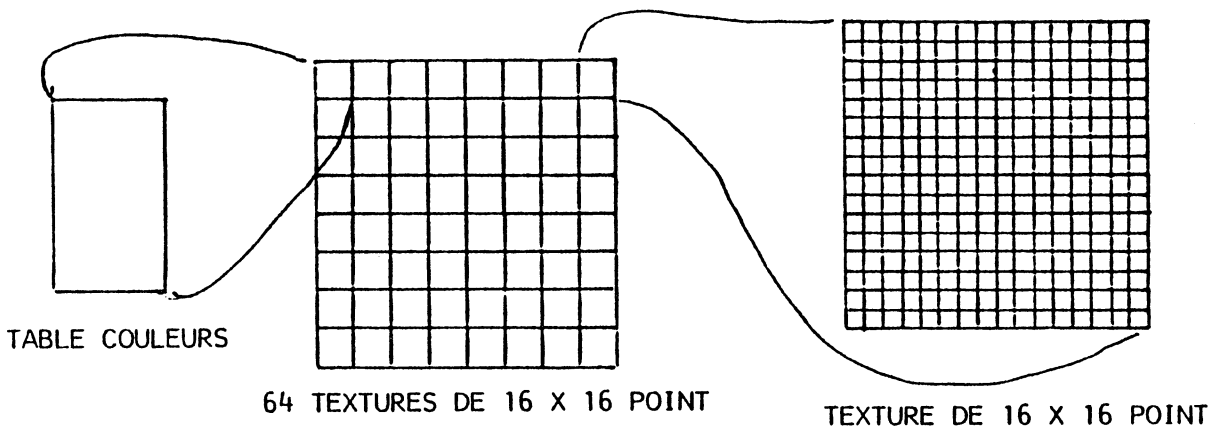


Figure IV. 14. Organisation de la mémoire de textures.

Cette organisation est très souple car elle nous permet d'associer plusieurs modèles de 16x16 pour obtenir des tailles plus importantes, par exemple 32x32 ou 64x64 points. Des combinaisons de différentes tailles sont possibles simultanement en mémoire. La figure IV.15 nous montre l'exemple d'une configuratuion.

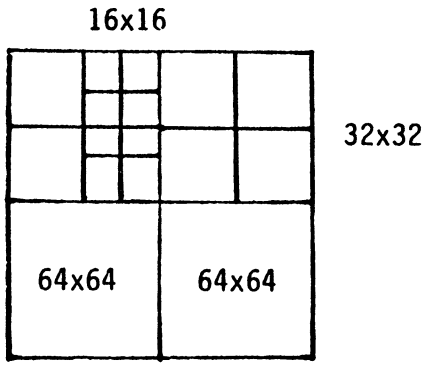


Figure IV.15. Exemple de configuration de la mémore de textures.

Par ailleurs, le carré supérieur gauche de 16x16 est réservé à la table des couleurs uniformes, ce qui nous permet de définir 256 couleurs uniformes distinctes. Cette solution ne empêche pas l'utilisation de plusieurs carrés comme table de couleurs.

3. 2. 2. 1. Modes de fonctionnement.

Ce module en tant que module intermédiaire doit suivre les mêmes modes de fonctionnement que ceux du module table de correspondance, à savoir :

i) Le mode "calcul" : A partir des données d'entrée (identification et projection de la face), le module calcule la couleur correspondante dans la mémoire de textures.

ii) Le mode "valeurs par défaut" : Il s'agit de déterminer une texture ou couleur de référence précise et indépendante des informations d'entrée. (par exemple, lors de la définition d'une texture de fond).

iii) Le mode " transparent" : Pour le cas de pixels d'entrée déjà coloriés. On retrouve en sortie la valeur de l'entrée.

iiii) Le mode "entrée/sortie" : Pour choisir les bus banalisé d'entrée et de sortie.

iiii) Le mode "bus vidéo" : Pour valider l'information de sortie directement sur le bus vidéo lorsque le module d'éclairage n'est pas représenté dans la scéne finale.

De la même manière que dans le cas du module de correspondance, la mémoire de priorité prend en compte ses facteurs et décode le bus de profondeur convenablement :

1 ligne	choix du bus d'entrée : BUS1/BUS2
2 lignes	choix du mode 00 : calcul
	01 : transparent
	10 : valeur par défaut

1 ligne	choix du Bus sortie : BUS3 ou POIDS FORTS BUS VIDEO
1 ligne	choix du Bus sortie : BUS4 ou POIDS FAIBLEs BUS VIDEO

Il faut remarquer que les informations d'entrée sont de deux types : l'identification de la face et la projection 3D de la même face. Dans le premier cas , on travaille soit sur le BUS1, soit sur le BUS2. Par contre la projection doit être associée à un Bus différent de ceux d'entrée (BUS1/BUS2) ou de sortie (BUS3/BUS4). Dans le chapitre II, nous avons défini les bus banalisés qui parcourent le fond du panier et leur nombre (cf. II. 2. 3. 2.). Nous en avons 5 des Bus banalisés. Nous assignons donc le cinquième bus à l'information d'entrée de la projection : BUS5.

3. 2. 2. 2. Réalisation matérielle.

Il s'agit ici de conserver la même cadence de calcul que celle du module table de correspondance. Par conséquent les trois cycles de pipe-line sont divisés de la façon suivante :

1er cycle : Sélection du bus d'entrée et prise en compte d'accés externes. Le cinquième bus banalisé BUS5 est utilisé directement comme bus de projection.

2° cycle : Sélection du mode de fonctionnement du module calcul, transparent, ou valeur par défaut.
Calcul de l'adresse de de lecture dans la mémoire de textures. Discrimination entre textures lisses ou coloriés.

3° cycle : Lecture de la mémoire de textures et sélection du bus de sortie.

La figure de l'annexe E nous montre le schéma de ce module.

La mémoire de textures de 64 modèles de 16x16 points nécessite une taille de mémoire de 16K x 12 bits. La contrainte de vitesse imposée par le troisième cycle du fonctionnement du module oblige à utiliser des

mémoires rapides. Nous utilisons 12 boitiers RAM statiques de 16 K x 1 bit avec un temps d'accés de 45 nsec maximum.

En ce qui concerne la table de correspondance de textures nous utilisons des mémoires RAM statiques de 4 K x bits à temps d'accés de 35 nsec. Six bits sont nécessaires pour accéder aux 64 modèles de textures.

Nous utilisons aussi un bit additionnel pour différencier l'utilisation des textures de l'utilisation de la table de couleurs. Dans ce dernier cas les six bits d'accés plus deux sont utilisés pour accéder à cette table.

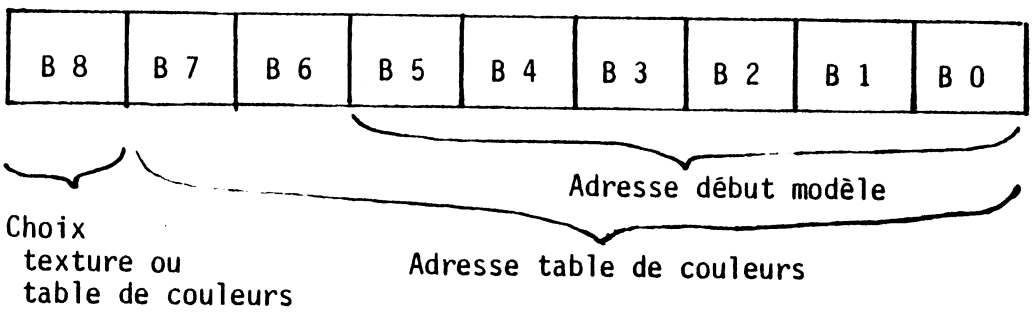


Figure IV.16. Format de la mémoire de sélection.

Dans le premier cycle de calcul, le choix du bus d'entrée est réalisé à l'aide du premier bit de sortie de la mémoire de priorité. Les accés externes sont toujours pris en compte en priorité (à travers la ligne EXT).

Dans le deuxième cycle, le décodage du mode : calcul, transparent ou par défaut est fait avec les deux bits suivants de la sortie de la mémoire de priorité. En même temps l'assignation de la texture ainsi que le calcul de l'adresse de lecture dans la mémoire de textures, sont calculés à l'aide des informations d'entrée : l'identification de la face et de la projection de la face.

Dans le troisième cycle, l'exécution du mode décodé auparavant est réalisé : lecture dans la mémoire de textures ou passage en transparence de l'entrée. En même temps on choisit le Bus de sortie à l'aide des deux derniers bits de la mémoire de priorité.

CHAPITRE V. REALISATION DU MODULE FINAL DE SYNTHESE.

1. Présentation.

La phase finale de la synthèse d'une image est la simulation de l'éclairage de la scène.
Nous nous plaçons dans le cas où l'éclairage est la combinaison de :

i) La lumière ambiante.
ii) La lumière provenant d'une source lumineuse, placée hors du champ de vision de l'observateur.

C'est à dire les facteurs d'illumination ou luminance. (c.f. II. 2. 3. 1). Par ailleurs, il faut arriver à simuler l'éclairage en tenant compte des différents facteurs relatifs aux objets :

i) Leur texture.
ii) Leurs types de matériaux.

Ces facteurs sont relatifs à l'aspect des objets et font donc partie de l'information de chrominance (c.f. II. 2. 3. 1).

Ces deux informations : chrominance et luminance, sont traitées en parallèle et de manière synchrone par les modules intermédiaires (c.f. IV. 2), sans qu'il ait besoin de mécanismes de synchronisation au niveau du module final. En effet, dans la conception des modules intermédiaires nous avons décidé d'égaliser le temps de traitement dans tous les modules. Ceci nous permet de considérer le retard des traitements intermédiaires comme fonction, non pas du temps d'exécution dans chaque module, mais uniquement du nombre de modules utilisés pour ce traitement.

Dans le cas du module final, la contrainte de vitesse de réponse est aussi criciale que dans les modules intermédiaires.En effet, le débit du nombre de pixels/sec vers le moniteur est toujours de 13,5 M (75 nsec par pixel), par conséquent la technique pipe-line est toujours utilisée.

Cependant le cas de ce module est un peu différent, car le temps de traitement est indépendant de tout autre module ainsi que le nombre de cycles de pipe-line utilisés.

Dans ce chapitre nous effectuons l'étude du module d'éclairage, chargé du calcul final de la couleur des pixels. Tout d'abord nous présenterons le modèle mathématique qui simule l'éclairage puis la réalisation proprement dite.

2. Le module d'éclairage.

Le module d'éclairage est sensé émuler la couleur finale à partir des données issues des modules intermédiaires. Par conséquent, il reçoit :

i) Les informations de chrominance (propres à la scène) : aspect des objets (couleur, textures).

ii) Les informations de luminance (propres à l'éclairage de la scène) : outre l'intensité et la couleur de la lumière ambiante et de la source lumineuse, ainsi que la position de cette dernière, le type de materiau.

A partir de ces informations, le module constitue l'image finale à afficher sur le moniteur vidéo.

L'interface matérielle à l'extérieur est constituée de deux bus banalisés en entrée (correspondants aux informations précitées), et du bus vidéo en sortie. (Fig. V. 1).

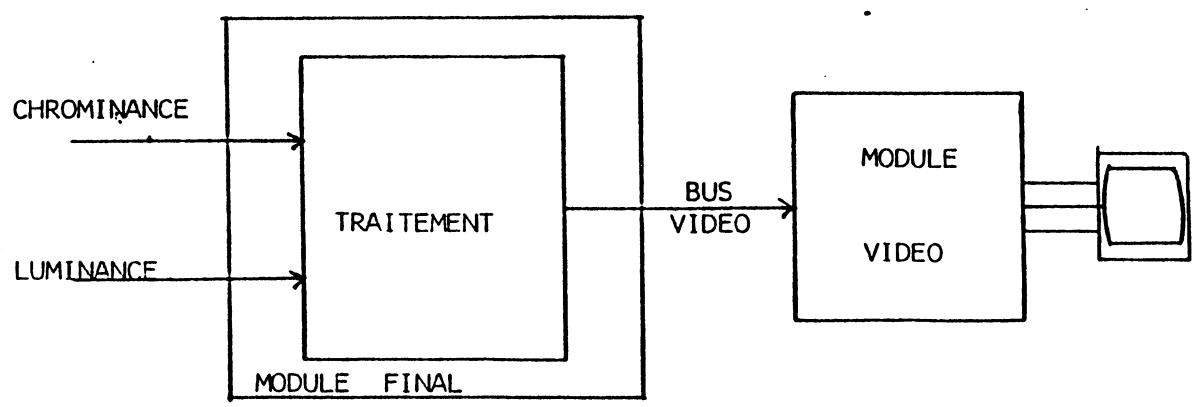


Figure V. 1. L'interface entrée/sortie du module final.

Du point de vue fonctionnel, le module d'éclairage est un module spécifique. Il se place toujours à la fin du processus de visualisation (en cas d'utilisation du calcul d'éclairage de toute la scène), par conséquent les interfaces à l'extérieur sont aussi spécifiques. Nous pouvons alors assigner un bus précis pour chaque information d'entrée (par exemple le bus BUS3 pour la chrominance et le bus BUS4 pour celle de la luminance).

Cette solution demande, des modules intermédiaires, le bon choix de ces interfaces de sorties. En effet, tout module intermédiaire qui traite l'information de chrominance doit sortir sur le BUS3 tandis que pour la luminance la sortie est le BUS4.

Le modèle mathématique choisi est basé sur celui de Phong [c.f. III.1.1.4.2.]:

$$Coul = [(A + Rd) * T + Rs] * S \qquad (2)$$

où

Coul = couleur finale
A = lumière ambiante
T =couleur ou aspect des objets
S =Intensité et couleur de la source lumineuse
Rd = coefficient de réflexion diffuse
Rs =coefficient de réflexion spéculaire

Il a été choisi par sa simplicité mais qui relationne tous les facteurs d'éclairage.

2. 1. Techniques d'implémentation du Module.

Le modèle mathématique choisi a déjà été testé [MaF 81], et donne des résultats acceptables.

Il est possible possible de diviser les techniques de réalisation possibles en trois catégories : celles pour lesquelles le matériel est prépondérant, celles qui privilégient le logiciel et celles qui combinent le matériel et le logiciel dans des proportions comparables.

2. 1. 1. Technique matérielle.

Le calcul du modèle mathématique d'éclairage nécessite tout d'abord le calcul des informations géométriques des faces par rapport à la source lumineuse, autrement dit le calcul des angles "l" et "m" (voir figure III.). Ce calcul en dépendant donc de chaque face. Dans le cas des faces gauches, un calcul point par point s'avère nécessaire.

Les autres calculs du modèle (additions et multiplications) ne posent pas de problèmes de conception bien qu'il s'agisse de calculs lourds et encombrants.
La grande difficulté est donc le calcul des angles "l" et "m" en temps réel pour suivre la cadence de pipe-line vers le moniteur vidéo. Une solution a été implantée par [Fev 81]. L'utilisation des coordonnées sphériques permet d'exprimer les directions par rapport au repère de l'écran (figure V. 2).

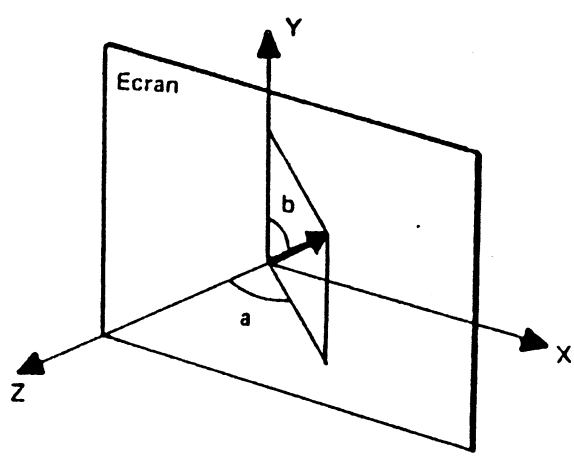


Figure V. 2. Représentation d'une direction en coordonnées sphériques.

La procédure à suivre pour calculer les angles "l" et "m" est la suivante :
Premièrement : Exprimer l'angle correspondant à l'aide des fonctions sinusoïdales : sin, cos et des vecteurs unitaires L, M, N. Le produit scalaire des vecteurs unitaires donne le cosinus de l'angle entre ces

deux vecteurs. Soit j‹ = (aj, bj) et k‹ = (ak, bk) deux vecteurs unitaires de coordonnées cartésiennes :

$$xj = \sin(aj)\sin(bj) \qquad xk = \sin(ak)\sin(bk)$$
$$yj = \cos(bj) \qquad yk = \cos(bk)$$
$$zj = \cos(aj)\sin(bj \qquad zk = \cos(ak)\sin(bk)$$

Le produit scalaire nous donne :

$$\cos(d) = \sin(bj)\sin(bk)\,[\,\sin(aj)\sin(ak) + \cos(aj)\cos(ak)] + \cos(bj)\cos(bk)$$

d = angle entre les deux vecteurs j et k.

Deuxièmement : Faire apparaître les sommes et les différences des angles pour simplifier le calcul, notre expression devient :

$$\cos(d) = 1/2\,[\cos(bj-bk) - \cos(bj+bk)]\cos(aj-ak) + \cos(bj+bk) + \cos(bj-bk)$$

Troisièmement : Effectuer des transformations pour faire apparaître des termes positifs seulement :

Soit $h(x) = \cos(x) + 1$

alors :

$$h(d) = 1/2\,[h(bj-bk) - h(bj+bk)]\,h(aj-ak) + h(bj+bk)$$

A partir de cette fonction h(d), on peut trouver les coefficients de réflexion diffuse et spéculaire :

$$Rd = f(h(l)) = f \circ h(l)$$
$$Rs = f(h(m)) = f \circ h(m)$$

Les fonctions résultantes f o h(l), f o h(m) sont calculées et stockées dans une petite table ROM pour être lue à la vitesse vidéo et suivre tout d'abord le calcul des coefficientxs Rd,Rs et finalement les transformations nécessaires en fonction de la direction de la source lumineuse L, la normale à la face N, la lumière ambiante A et la couleur de la face traitée.

D'après l'expression (2), le schéma du calcul final est le suivant (figure V. 3).

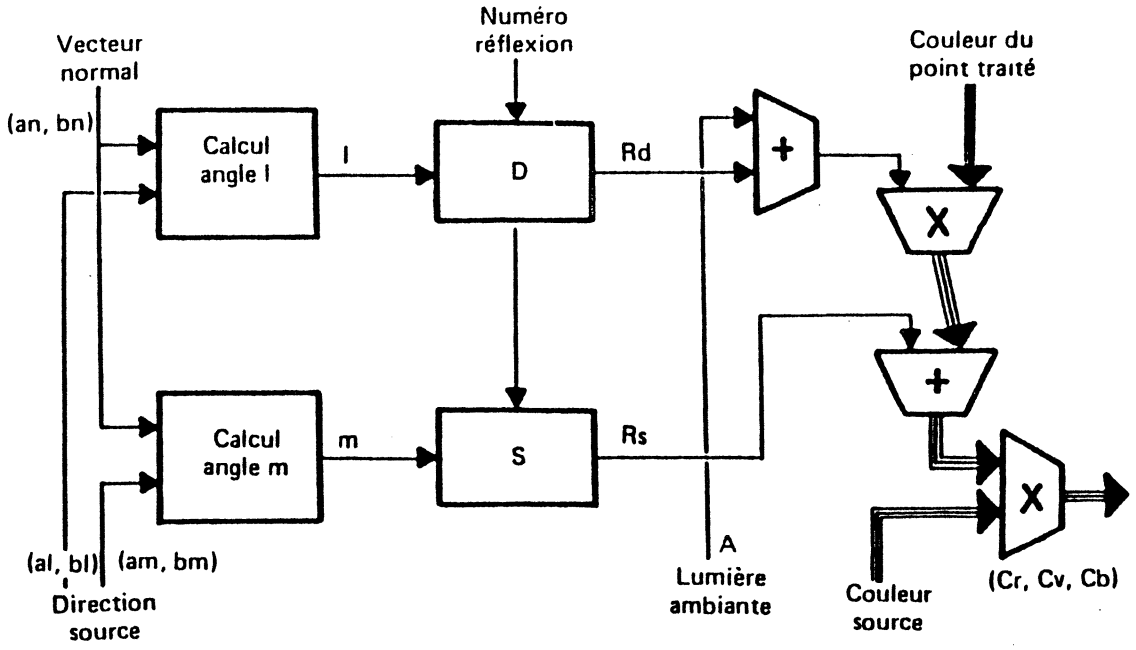


Figure V. 3. Calcul du modèle d'éclairage par matériel.

Le grand avantage de cette méthode est le haut niveau d'interactivité proposé à l'utilisateur. En effet, tout changement de direction, couleur de la source lumineuse et intensité de la couleur ambiante est prise en compte en temps réel.

Cependant le volume de matériel demandé pour son implémentation est très importante, ce qui induit pour le système un encombrement et une mise en oeuvre très lourdes.

Il reste à se questionner sur le besoin de développer des outils de cette performance étant donné les limitations dues au temps de réponse de l'opérateur humain (quelques dixièmes de seconde).

2. 1. 2. Technique logicielle.

Cette technique utilise des tables de transformations en sortie, pour le calcul des coefficient de réflexion. Il est intéressant de remarquer que ceci est un exemple typique d'utilisation de ce type de tables de correspondance après la mémoire de trame [Bas 85], [War 83].

Le calcul est donc laissé au logiciel qui est responsable de la gestion des tables.

Néanmoins, une solution similaire est le calcul direct de l'information de l'éclairage dans l'image. Cette information est stockée dans la mémoire de trame du système.

Pour le cas de l'architecture banalisée, ces implémentations peuvent se faire directement :

- dans les plans mémoire, pour le cas de l'image complètement calculée c'est le processeur graphique qui est complètement chargé de cette tâche.
- dans une table de correspondance pour le cas de tables de transformation, on utilise un module en sortie. En plus, lorsqu'il faut disposer d'une définition assez importante en sortie (24 bits de définition couleur), l'utilisation de deux tables accolées est possible.

Le schéma d'utilisation d'une table de sortie est représentée par la figure V. 4 ci-dessous.

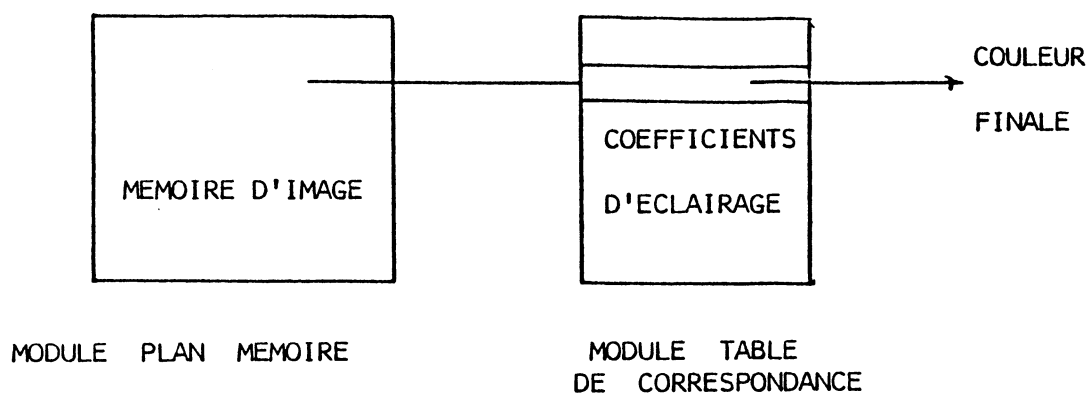


Figure V. 4. Utilisation d'une table pour les coefficients d'éclairage.

Ces techniques présentent divers avantages : une grande simplicité,

conséquence de leur implémentation directe. D'autre part, le fait que le calcul soit réalisé par le logiciel nous permet d'implémenter tout type de modèle.

Cependant, le niveau d'interactivité est assez bas. En effet, le changement d'un paramètre dans le modèle mathématique entraîne le ré-calcul de toute la table avec l'inconvénient des multiplications qui sont à l'origine des temps de calcul assez importants. Un autre inconvénient de ces techniques est la gestion de plus en plus lourde laissée au processeur graphique.

Dans le cas d'utilisation des textures dans une application bien précise, le module spécifique de calcul de textures ne peut pas être utilisé et il est necessaire d'effecture le calcul complet de l'image par logiciel pour être enregistré dans un plan mémoire.

2. 1. 3. Technique mixte.

La caractéristique la plus importante de cette technique est l'utilisation des performances de calcul du processeur graphique.

En effet, une partie des calculs peut être laissé aux soins du processeur graphique et la partie restante au matériel de telle façon à obtenir des performances d'intéractivité raisonnables.

Les calculs faits par le processeur graphique doivent être exploités au rythme de la vitesse vidéo. Il est donc nécessaire d'utiliser des opérateurs de mémoire tampon synchrones (cf. II. 2. 1.). Ces opérateurs sont des tables de mémoires programmées par le processeur graphique et exploitées judicieusement par le module final de calcul.

Le premier pas vers le mélange des techniques logicielles/matérielles dans l'implémentation du modèle mathématique d'éclairage est le calcul des angles "l" et "m". En effet, dans l'expression (1) la première étape est le calcul des angles pour obtenir les coefficients de réflexion. Ces angles peuvent être calculés par le processeur graphique et stockés dans une table de correspondance. De cette façon, les variations de position de la

source lumineuse sont réalisées en concordance avec la puissance du microprocesseur du processeur graphique.

Par ailleurs, pour maintenir une bonne intéractivité, il est nécessaire de conserver une partie importante de calcul effectués par le matériel. En outre l'interactivité n'est pas la seule raison pour conserver la partie matérielle. L'utilisation du module de calcul de textures l'impose également.

En effet, le pavage de la texture sur une face génère différentes couleurs. Ces couleurs nécessitent un calcul d'éclairage différent de celui de ces mêmes couleurs sur d'autres faces (si les faces ont des normales différentes), et ce calcul doit être effectué pour chaque point de l'image.

Il faut remarquer que le niveau d'interactivité dépend aussi de l'opérateur humain. Tant que les variations d'illumination sont plus rapides de quelque dixième de seconde, l'opérateur ne les percevra pas.

En ce qui concerne la synchronisation des calculs, le calcul des angles "l" et "m" en tant qu'information de "luminance" peut être fait en parallèle avec celui des textures. C'est d'autant plus évident que le calcul des angles est émulé par un module intermédiaire banalisé. La figure V. 5 ci-dessous nous montre cette technique de réalisation.

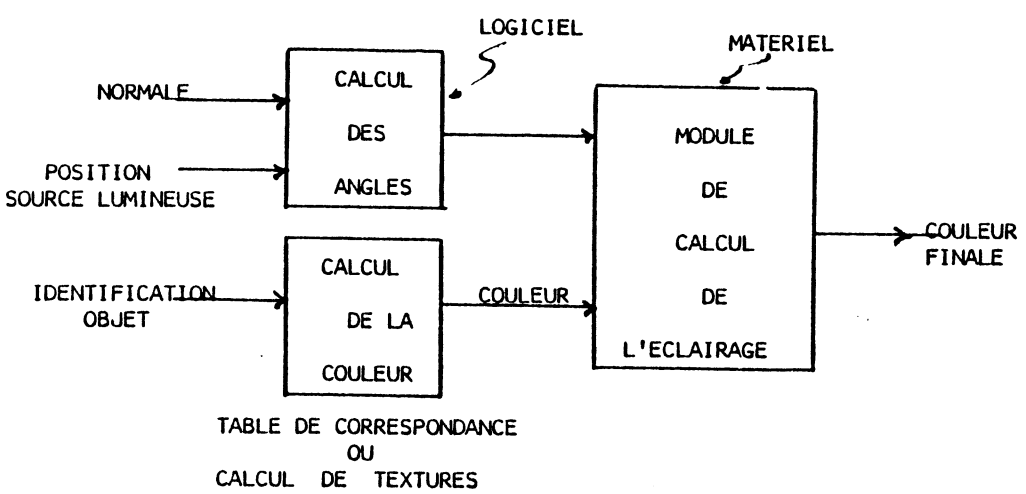


Figure V. 5. Technique mixte de réalisation du module d'éclairage.

Les avantages de cette technique sont divers : en même temps que l'implémentation par matériel est moins lourde, on profite des performances du processeur graphique pour le calcul des angles. Cette combinaison assure un bon niveau d'interactivité. En plus, l'utilisation des aspects différents tels que textures ou faces lisses est prise en compte par le module sans aucune difficulté.

En ce qui concerne les inconvénients on peut citer, l'utilisation d'un module table de correspondance additionnel mais avec une simplification de l'architecture matérielle du module d'éclairage. La gestion du système d'architecture banalisée sera aussi plus lourde, car le nombre de modules sera plus importante.

2. 2. Réalisation du module d'éclairage.

Malgré la nécessité d'une bonne interactivité, le temps de réponse du module doit être à l'échelle humaine : quelques dixième de seconde.
Les techniques "matérielles" ont des temps de réponse très courts (de l'ordre de la microseconde), inutiles à l'opérateur humain. En plus, elles sont coûteuses en électronique, encombrantes et difficiles à mettre en oeuvre.

Les techniques "logicielles" ont des temps de réponse trop longs (de l'ordre de 10 secondes). Ce qui est au-delà du minimum requis pour une interactivité satisfaisante.

Par conséquent, les techniques "mixtes" paraissent les plus adaptées aux objectifs coût-perfôrmance souhaités. Nous avons donc choisi d'implémenter cette technique dans le cadre de l'architecture banalisée.

2. 2. 1. Modes de fonctionnement.

La première étape vers la réalisation du module est la détermination de ses modes de fonctionnement pour assurer la compatibilité avec la philosophie du système d'architecture banalisée.

I) **Mode "calcul"** : C'est le mode de fonctionnement normal, pour lequel à partir des données d'entrée :

chrominance-luminance, on calcule la valeur de la couleur finale. Cette couleur s'exprime sur 24 bits pour avoir toutes les nuances de couleur possibles.

ii) **Mode "transparent"** : Ce mode s'applique aux pixels qui suivent des processus différent de visualisation et pour lesquels la couleur finale est figée et calculée auparavant (dans un module intermédiaire ou dans un plan mémoire). Cependant, ils ont besoin d'un retard pour conserver le synchronisme par rapport au reste de l'information.

iii) **Mode "entrée/sortie"** : Ce mode définit les bus banalisés d'entrée, et de sortie. En ce qui concerne les entrées, le bus BUS3 est assigné à la chrominance et le bus BUS4 à la luminance. Cette rigidité est justifiable car les modules intermédiaires ont les moyens d'aiguiller les informations vers les différents bus banalisés BUS3 ou BUS4 (l'ordonnancement des modules est toujours valable). D'autre part, l'information en sortie doit être assignée au bus vidéo directement. En effet, la sortie du module d'éclairage est déjà la couleur finale à afficher sur le moniteur vidéo. Cette sortie doit pouvoir s'exprimer sur 12 ou 24 bits dépendants de l'application.

iiii) **Le mode " modèle d'éclairage"** : Ce mode est particulier au module. En effet, on sait que pour chaque matériau constituant un objet il existe un modèle d'éclairage distinct. Or, dans une image synthétique, il existe différents objets composant la scène. Pour le cas de l'éclairage, chacun des objets doit avoir son propre modèle pour obtenir un réalisme plus saisissant. Ainsi, de la même manière qu'on a une mémoire avec différentes textures distinctes, on doit avoir aussi une mémoire ou une table des différents modèles d'éclairage.

En ce qui concerne le décodage du bus de profondeur, le module nécessite uniquement 3 lignes en sortie, à savoir :

1 ligne choix mode : calcul/transparent.
1 ligne choix bus vidéo poids fort.
1 ligne choix bus vidéo poids faible.

Il faut remarquer que le choix des bus banalisés d'entrée ne nécessite aucune ligne de sélection car ces bus sont fixés d'avance.

2. 2. 2. Réalisation matérielle.

Le premier pas dans la réalisation de ce module est le choix du modèle d'éclairage. En effet, à partir des informations chrominance ou luminance arrivées au module, on doit choisir le modèle parmi ceux stockés qui s'adapte le mieux au matériau manipulé. L'idéal est d'avoir une table qui contient tous les modèles, mais on voit facilement l'impossibilité de cette solution : trop de mémoire, plus de gestion, etc. Nous avons donc choisi de limiter la table à 4 modèles simultanés. L'utilisateur pourra accéder à ces tables pour enregistrer ses propres modèles.

- Si on fait le choix à partir des informations de chrominance, nous pouvons diviser les 12 bits en deux parties : 2 bits pour le choix des modèles, 10 bits pour la couleur de l'objet. Cette solution nous diminue la définition du nombre de couleurs en entrée, et complique la gestion des tables de couleurs.
- Si on fait le choix à partir des informations de luminance, nous sommes confrontés au même problème que dans le cas précédent.

Une façon plus élégante de réaliser ce choix est d'utiliser l'identification même de la face à traiter comme entrée dans une table interne pour sélectionner un modèle parmi les 4 existants.

Cependant cette solution soulève le problème de synchronisme entre les différentes entrées. D'une part l'identification des faces est une information du domaine des plans mémoires, tandis que la chrominance et la luminance sont le domaine des modules intermédiaires. (figure V.5)

Ce problème est résolu d'une façon directe en égalisant le nombre de cycles de pipe-line nécessaires au choix du modèle d'éclairage avec ceux des modules intermédiaires.

Si on reprend la notation de la chrominance "C" et de la luminance "L" vue en [II. 2. 3.1] ainsi que les cycles de retard au niveau de chaque module de traitement comme paramètres, nous pouvons suivre le cheminement de l'information issue des plans mémoires. De la même façon les cycles de pipe-line de traitement jusqu'au module d'éclairage sont mis en évidence. La figure V. 6 nous montre cette démarche.

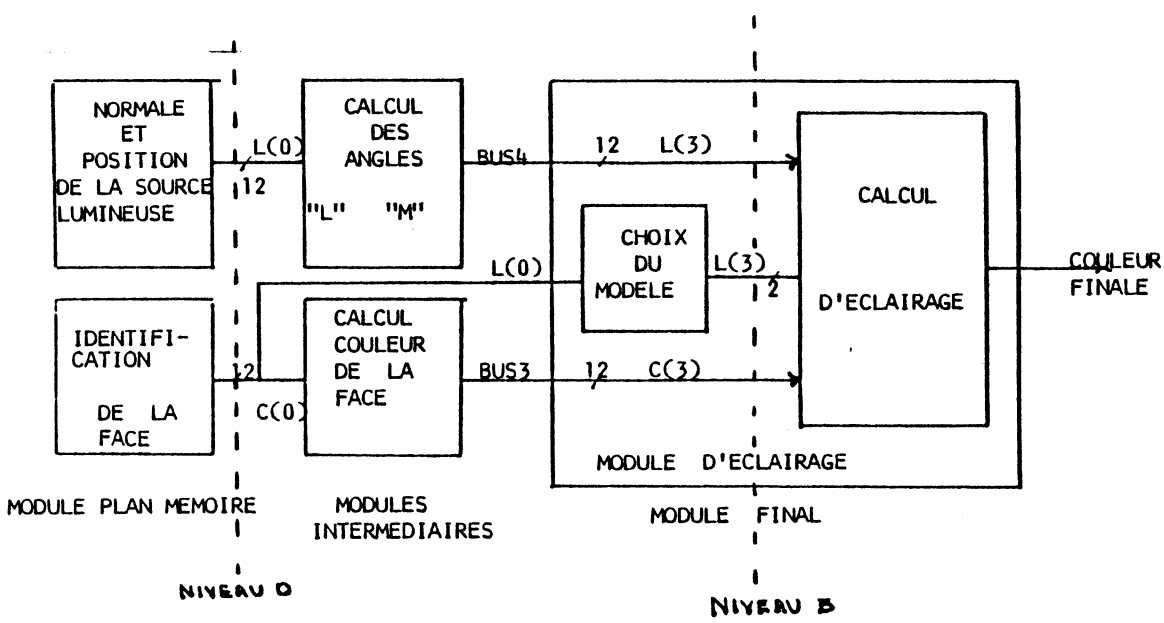


Figure V. 6. Cheminement de l'information vers le module final.

Des contraintes d'ordre industriel nous amènent cependant à considérer plusieurs simplifications d'ordre matériel dans le modèle mathématique choisi. En effet un produit industriel nécessite, (en plus de son bon fonctionnement), la minimation des coûts et de l'encombrement et donc du nombre réduit de composants électroniques.

Ce facteur est primordial pour deux raisons :

i) La consommation d'énergie et la dissipation de la chaleur dans l'ensemble du système sont considérablement réduites. Cela évite que le système ne tombe pas

fréquemment en panne.

ii) Le système est moins encombrant et par conséquent il est plus facile à manipuler et à entretenir.

Nous rappelons le modèle mathématique d'éclairage :

$$Coul = ((A + Rd * T + Rs) * S \qquad (3)$$

L'équation (3) nous conduit à utiliser deux multiplications. Or, pour le cas des 3 couleurs primaires : rouge, vert, bleu, l'expression (3) devient :

$$\begin{aligned} Cr &= ((Ar + Rd) * Tr + Rs) * Sr \\ Cv &= ((Av + Rd) * Tv + Rs) * Sv \qquad (2) \\ Cb &= ((Ab + Rd) * Tb + Rs) * Sb \end{aligned}$$

Dans les équations ci-dessus tous les paramètres varient entre 0 et 1, par conséquent la couleur finale "Cou" varie entre 0 et 2. Nous considérons une dé-saturation de couleur (vers le blanc) pour limiter les variations entre 0 et 1 seulement :

$$\begin{aligned} Cr &= Max(Max(1, Ar+Rd) * Tr + Rs) * Sr \\ Cv &= Max(Max(1, Av+Rd) * Tv + Rs) * Sv \\ Cb &= Max(Max(1, Ab+Rd) * Tb + Rs) * Sb \end{aligned}$$

Les coefficients Rs et Rd sont exprimés sur 8 bits chacun. Cette définition est le résultat de la lecture de la table de 4 modèles. Par ailleurs, ces coefficients expriment la portion de lumière refletée par les surfaces. 256 niveaux différents nous paraissent assez raisonnables.

Par contre les paramètres S, T, A sont exprimés sur 12 bits : ce qui signifie que chaque composant primaire de couleur est exprimé sur 4 bits uniquement. Cette définition est due à la largeur des bus banalisés. 4096 couleurs peuvent se définir. Cette definition est largement suffisant pour des couleurs de base (non modulées).

Alors, il faudra disposer de 3 multiplicateurs à 12 bits (8 bits + 4 bits = 12 bits) et de 3 autres à 16 bits. Or les multiplicateurs sont très encombrants et en même temps ils sont chers.

Les simplifications du modèle mathématique porteront donc sur la réduction de ces multiplications.

Une première simplification a été essayée au niveau de la multiplication finale;

soit :

$$Z = Max (Max (1, A+Rd) * Tr + Rs,1)$$

l'expression (3) devient :

$$Coul = Z * S \qquad (4)$$

On voit d'après l'expression (4) que la couleur finale Coul est la modulation de Z par rapport à la valeur de la source lumineuse S.

Par ailleurs, Z est de la forme exponentielle e-x et la valeur de S reste constante (entre 0 et 1) pour toute une application.

La couleur Coul est donc une expression exponentielle e-x avec un facteur d'échelle S.

Deux possibilités de simplification ont été considérées : par décalage et par troncature de la fonction Z:

- **Simplification par décalage de Z.** Cette simplification consiste à approximer la valeur réelle de la couleur Coul par un décalage de la fonction Z proportionnel à la constante S. En effet, si on décale de (1-S) vers le bas la fonction Z, nous aurons comme résultat une fonction de Z qui coïncide au début avec celle de Z * S au même endroit. Comme on voit sur la figure V. 7

Cette simplification est correcte pour des angles autour de zéro. Par contre, pour des angles supérieurs à 90° elle devient aberrante. En effet, une source lumineuse devient toute sombre pour des angles supérieur à ce seuil.

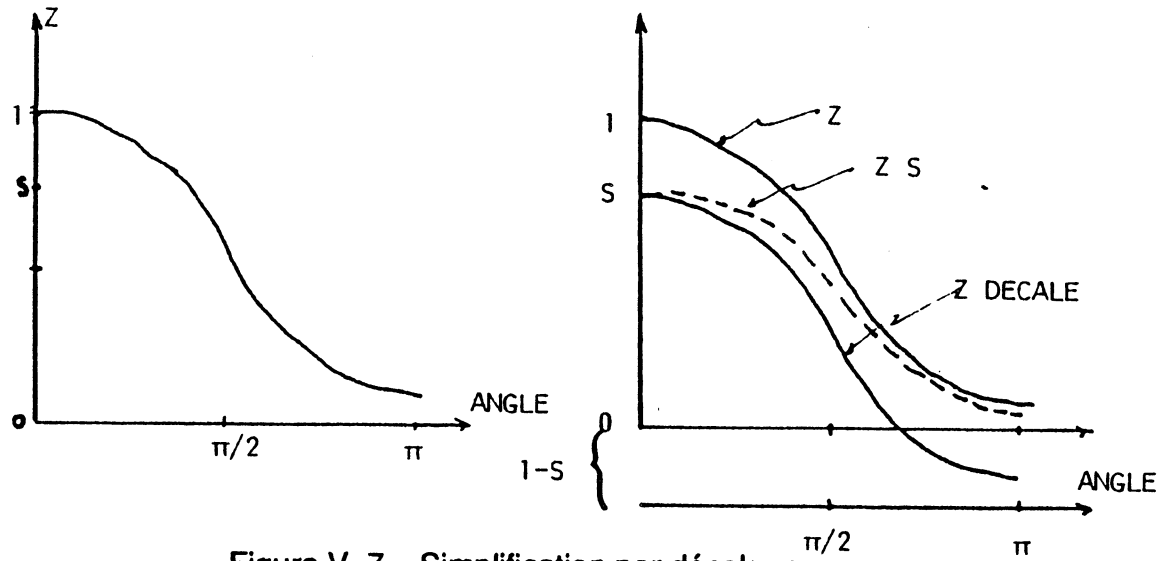


Figure V. 7. Simplification par décalage.

- Simplification par troncature de Z. Cette simplification écrête la fonction Z au-dessus de la valeur de la constante S, comme on voit sur la figure V. 8.

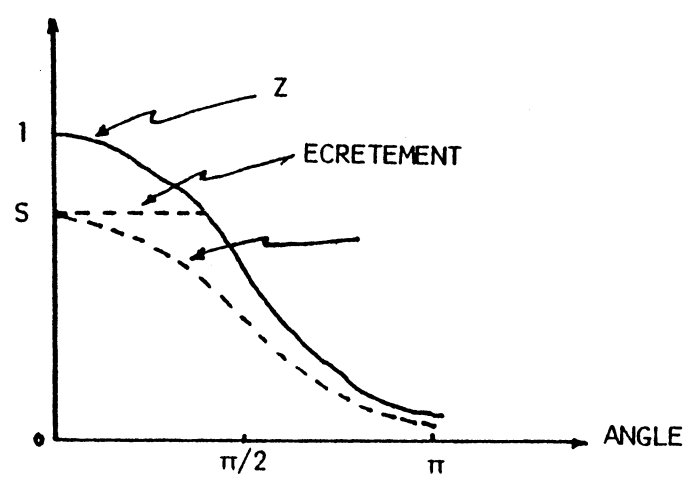


Figure V. 8. Simplification par troncature de Z.

Ces deux simplifications ont été simulées par logiciel et les résultats ont été assez pauvres au niveau du réalisme pour qu'une implémentation matérielle soit justifiable.

Il y a en effet des situations où le résultat visuel, est complètement

faussé, par exemple, pour une source lumineuse jaune (influence du rouge et du vert), si l'influence du vert est inferieur a celle du rouge, la couleur de la source devient rouge dans les conditions suivantes :

- pour des angles grands pour la méthode de décalage de la fonction Z.
- pour des angles petits pour la méthode de troncature de la fonction Z.

On est donc obligé de conserver les multiplications finales de la source lumineuse.

Une deuxième simplification a été essayée pour les multiplications de la couleur des textures avec les coefficients de réflexion.
A partir de l'expression (2), on peut trouver :

$$Coul = Max(Max(1, A+Rd) \times S \times T + Rs \times S, 1) \quad (5)$$

Les simplifications envisagées sont de type logiciel.
On sait qu'à partir des angles "l", "m" on trouve les coefficients de réflexion Rd et Rs à l'aide des tables de modèles. (Figure V. 9)

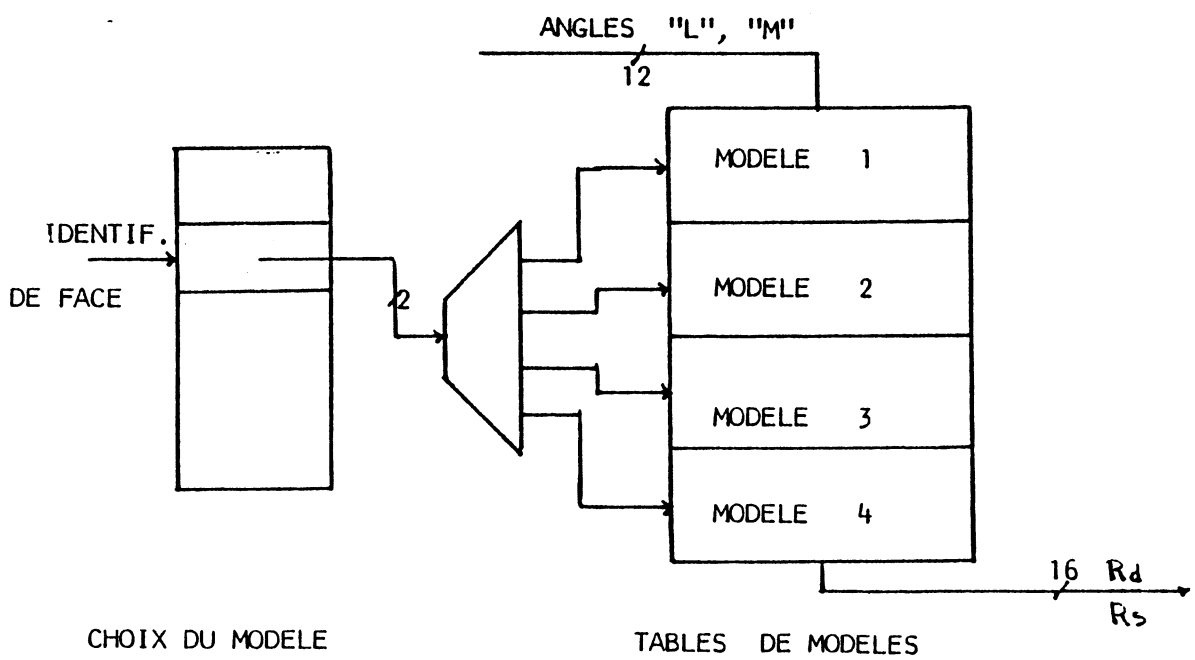


Figure V. 9. Les tables des modèles de réflexion.

Une première simplification est faite en intégrant aux tables de modèles

l'influence de la lumière ambiante A, ainsi l'expression (5) devient :

$$C = \text{Max}(D * S * T + Rs^* S, 1) \qquad (6)$$

où

$$D = \text{Max}(1, Ar + Rd)$$

De la même manière on peut associer les mêmes tables pour intégrer les multiplications de la source lumineuse. Cependant le cas est un peu différent, car pour chacune des couleurs primaires on a besoin d'une multiplication distincte.

La solution la plus simple est de tripler les tables des coefficients de réflexion diffuse et spéculaire et d'intégrer dans chaque table la multiplication de la couleur primaire correspondante.

Cette multiplication des tables ne pose pas de problèmes matériels vu l'intégration des mémoires RAMs statiques actuelles, qui sont plus économiques que les multiplieurs.

Il faut faire deux remarques :

- D'une part, les variations de la position (angle "l" et "m") (table de correspondance des angles), et de la couleur de la sourcelumineuse (Rd * S et Rs* S) (table de modèles) sont complètement prises en compte par le processeur graphique.
- D'autre part, l'influence de la couleur des objets (T) est prise en compte par le matériel à la vitesse vidéo, ce qui assure un bon niveau d'interactivité au niveau des couleurs d'objets.

Les avantages de cette solution sont divers : premièrement la simplicité de mise en oeuvre est obtenue grâce à la réduction du matériel dans l'implémentation. Par ailleurs la triplication de tables de réflexion permet l'implémentation des modèles indépendants pour chaque couleur primaire, ce qui aide à obtenir un niveau de réalisme plus saisissant, et à produire des effets spéciaux .

Mais il y a aussi des inconvénients. En effet, cette évolution du calcul vers

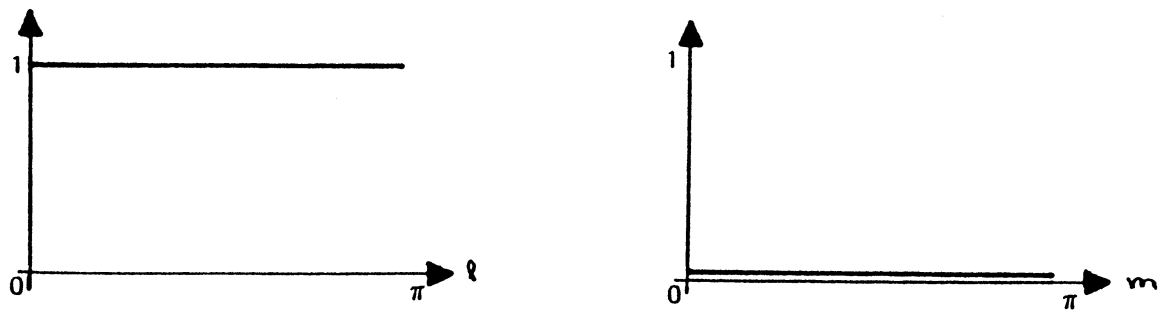
le logiciel nécessite une gestion assez lourde des tables et mémoires internes. Par ailleurs, le temps de réponse du système aux variations de lumière est laissé aux soins du processeur graphique. Des résultats actuels nous montrent un temps de réponse de l'ordre du dixième de seconde pour une scène de complexité moyenne (300 faces planes).

La figure de l'annexe F nous montre le schéma du module final.

Nous utilisons 6 cycles de pipe-line dont les 3 premiers nous servent à choisir les modèles mathématiques parmi les 4 disponibles. Ce choix est fait en synchronisme avec les modules intermédiaires.

Actuellement on utilise les modèles d'éclairage suivants :

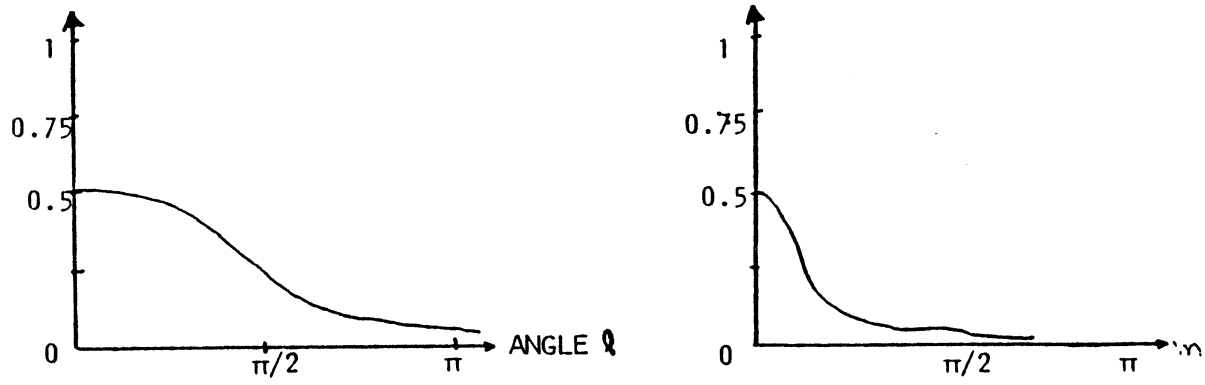
i) Neutre : Il est necessaire pour quelques objets, d'invalider le calcul d'éclairage. La couleur finale de l'objet reste toujours la même que celle en entrée. La simulation de ce modèle nécessite une contribution nulle du coefficient spéculaire comme on le voit sur la figure V.10. Cette modèle peut être realicé en utilisant le mode transparent du module. Mais, le choix de ce mode applique la transparence à toutes les faces d'un plan mémoire.



Réflexion diffuse Réflexion spéculaire

Figure V. 10. Modèle neutre.

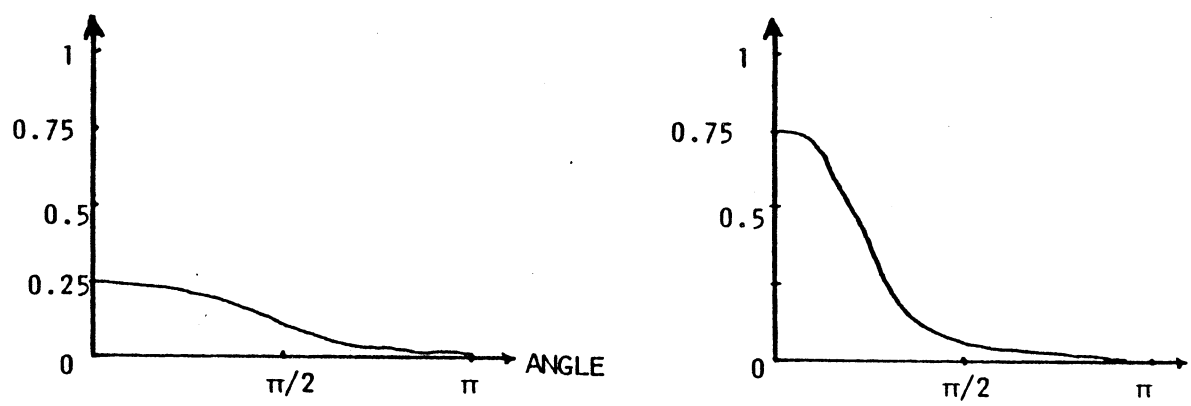
ii) Satiné : ce modèle considère la contribution des réflexions des matériaux : cd et cs égaux à la moitié de l'échelle maximale comme on voit sur la figure V.11.



Réflexion diffuse Réflexion spéculaire

Figure V. 11. Modèle satiné.

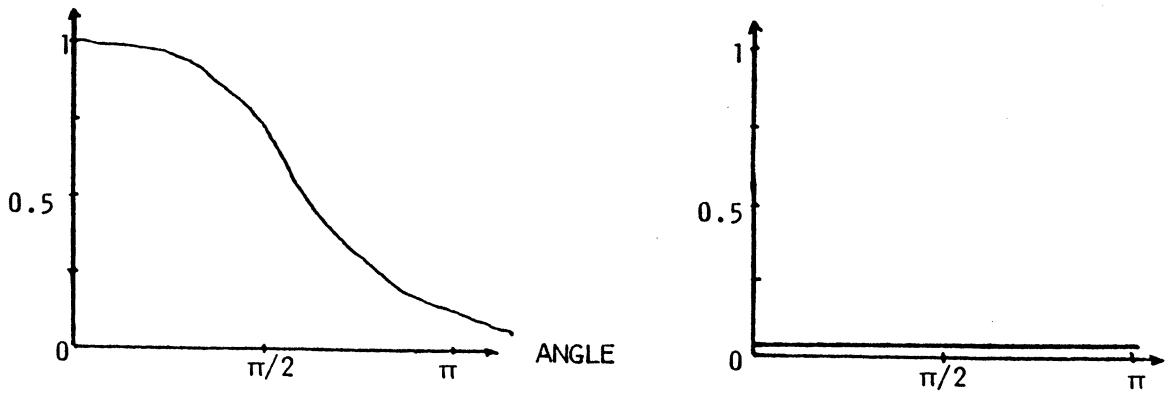
iii) Brillant : Pour considérer les reflets des objets métalliques. La contribution de la réflexion spéculaire est plus importante que celle de la réflexion diffuse : Rd < Rs. (figure V.12).



Réflexion diffuse Réflexion spéculaire

Figure V. 12. Modèle brillant.

iiii) Mat : Pour les matériaux qui n'ont aucun reflet de lumière et par conséquent la contribution de la réflexion spéculaire est nulle. (figure V. 13).



Réflexion diffuse Réflexion spéculaire.

Figure V.13 Modèle mat.

Dans le quatrième cycle, on accède aux tables des modèles de réflexion diffuse. Le résultat est l'influence de la couleur de la source lumineuse et du coefficient de réflexion diffuse.

Le cinquième cycle effectue la multiplication des résultats des tables des modèles par la couleur des objets. En même temps, on lit l'influence de la source lumineuse par rapport à la contribution du modèle spéculaire RS dans des tables correspondantes.

Finalement, le sixième cycle effectue l'addition des contributions des facteurs diffusés et spéculaire ainsi que le test de désaturation vers le blanc. Le choix des bus vidéo de sortie ainsi que l'éventuelle transparence de valeurs d'entrée est réalisée dans ce même cycle.

CHAPITRE VI. CONCLUSION ET PERSPECTIVES FUTURES.

Nous avons présenté dans ce travail une réalisation matérielle d'un système de synthèse d'images (choix d'algorithmes, choix d'implémentation) qui finalement a aboutit à une commercialisation.

Pendant tout ce travail on a mis en valeur deux aspects importants de l'architecture banalisée :

i) Sa modularité et sa facilité d'extension.
ii) La puissance du processeur graphique couplé convenablement à une logique câblée donne des résultats intéressants du point de vue de l'intéractivité du système.

En effet, à partir d'une structure de base, l'architecture banalisée évolue vers un système plus performant avec calcul de l'éclairage et de la texture des faces à l'aide des modules câblés.

La mise en oeuvre du module table de correspondance nous donne un module banalisé qui peut, avec un ordonnancement judicieux, servir à plusieurs tâches. Les exemples actuels sont ceux des table de couleurs sur 12 ou 24 bits et ceux des tables des angles ou de normales (pour le module d'éclairage).

La mise en oeuvre du module de textures nous a permis de valoriser les facilités d'extension du système. Ce module est constitué de la logique cablée associée à un plan mémoire convenablement rempli par logiciel.

La réalisation mixte (logicielle/matérielle)du module d'éclairage permet d'envisager des extensions notamment dans sa partie logicielle. En effet, l'architecture du module autorise l'utilisation des tables de modèles de réflexion de différentes manières. On peut ainsi réaliser les effets spéciaux suivants :

- simulation des 4 sources lumineuses (une pour chaque table du module).
- modélisation indépendante de la réflexion de chacune des couleurs primaires de la lumière incidente sur les différents objets de la scène.

En ce qui concerne les perspectives envisageables pour améliorer le système, on peut citer :

i) L'implémentation de l'algorithme de z-buffer par matériel. En effet , les premières évaluations de l'algorithme implémenté par logiciel nous montrent des performances moyennes qui pourraient être améliorées grâce au module matériel de lecture, de comparaison et d'écriture dans les plans mémoire.

ii) L'implémentation des processeurs spécialisés dans la génération de tâches graphiques, par exemple, l'utilisation d'un micro-processeur en tranches pour réaliser des fonction spécifiques.

iii) L'amélioration de l'aspect final de l'image grâce à un opérateur d'"antialiasing".

iiii) L'amélioration de la résolution de l'image avec l'aide, par exemple, d'un double tampon en entrée du module vidéo.

Toutefois, dans l'état actuel il est déjà possible d'adapter le système aux demandes spécifiques des utilisateurs avec un compromis "qualité, rapidité, coût " tout à fait prommeteur.

BIBLIOGRAPHIE

[Ase 84] J. Aseo
Partial pixel addressing increases effective display resolution.
Computer Desing June I. 1984

[Bas 81] D. H. Bass
Using the video Look up Table for Replectivity Calculations : Specific techniques and Graphics Results
Computer Graphics and Image Processing 17 - 1981.

[Bin 84] A.K. Bindra
Technologies vie for to dominance
Electronics May 1984.

[Biw 85] H. Bierman. D. M. Weber.
New Display Formats Give users better show for fewer bucks.
Electronics week, april 1985.

[Bln 76] J.F. Blinn. M. E. Newell
Texture and Reflection in Computer Generated Images
Communications of the ACM. October 1976.

[Bli 77] J.F.Blinn
Models of light reflection for computer Synthetized Pictures.
Computer Graphics SIGGRAPH-ACM, Vol. 11, N°2, 77

[Bon 83] P.E. Bonice
How to Evaluate Raster Scan CRT Monitors.
Computer Graphics World. 4 - 1983.

[Bov 80] P. Boulle.
Etude et réalisation d'algorithmes pour la visualisation de scènes composées de facettes planes.
Thèse Docteur-Ingénieur - Grenoble - Septembre 1980.

[Bre 65] Bresenham
System Algorithm for computer control of a digital Plotter.
IBM. System journal, Vol. 4 N° 1 1965.

[Bun 82] W. M. Bunker
Training in a Simulated World.
Computer Graphics World. 12 - 1982.

[Bur 83] D. Bursky
Silicon Support par video displays grows smarter
Electronic Desing. January 20 - 1983.

[Cat 78] E. Catmull
A Hidden - Surface algorithme with anti-Aliasing
Computer Graphics - Vol 12 N° 3 - ACM 1978.

[Chi 85] K. Chibane
Evaluation d'Architectures de la pré-synthèse d'images.
Thèse Docteur-Ingénieur. Grenoble. En préparation.

[ChM 83] P. Chu. W. Miller.
µp. architecture suits bi-mapped graphics
Electronic Desing. January 20 - 1983.

[Cla 76] J. H. Clark
Hierarchical model for visible surface algorithms
ACM Communications - Vol. 19, N° 10 - Octobre 1976.

[ClD 83] J. H. Clark, T. Davis
Work station unites real-time graphics with Unix, Ethernet.
Electronics - October 20 - 1983.

[CoT 81] R.L. Cook and K. E. Torrance.
A refectance Model for Computer Graphics
Computer Graphics-SIGGRAPH - ACM Vol 15 , N° 3 - 08-81.

[Cro 77] F. C. Crow
Shadows Algorithms for Computer Graphics.
Computer Graphics SIGGRAPH-ACM, Vol 11, N°. 2, 77.

[DSS 85] A. Daniel. A. Strupat. A. Sparti
Un jeu de circuits intégrés pour des systèmes graphiques
à haute performances.
Electronique Industrielle N°. 83. 15 - 02 - 1985.

[Eve 52] R. Everett
The Whirlwind I computer
joint AIEEE - IRE, Electr. Digit Computer, 1952.

[Fer 81] F. Nunes Ferreira
Conception et realisation d'un système interactif pour la synthèse d'images réalistes : Helios
Thèse Docteur-Ingénieur - Grenoble - Septembre 1981.

[Gau 85] A. Gaulène.
Terminaux Graphics couleur : Performances et tendances
Le nouvel Automatisme. Juin 1985.

[GeG 85] Ph. Genoud. J.F. Grabowiecki
A general purpose graphic environment : CLOVIS.
IASTED - 18-2O juin 1985 - Paris.

[Gra 80] R. Gray.
Bit-map architecture realizes raster display potencial
Computer Desing. July 1980.

[Gro 84] H. De Groot
Selection Criteria for Graphics Hardware.
Computer and Graphics Vol. 8 N° 3 - 1984.

[Gro 85] C. Gross
Des processeurs graphiques pour des images plus vivantes.
Electronique Industrielle N° 83 - 15 - 02 - 1985.

[Hui 84] J. Huisman
Pipe-line model Promotes Extensible Display System
Computer Desing - May 1984.

[Hub 84] R. J. Hubbold
Computer Graphics and displays
Computer Aided Desing Vol 16 - N° 3 - May 1984.

[Ike 84] T. Ikedo
High-speed Techniques for a 3D Color Graphics Terminal
IEEE Computer Graphics and Applications, May 1984.

[Maf 82] F. Martinez. F. Ferreira
Helios : terminal graphique interactive pour les images réalistes.
Nouvel Automatisme - Mai 1982.

[Man 81] D. Mansion
CAO : Technologie des postes de travail
Le Nouvel Automatisme - Octobre 1981.

[Man 84] Tom Manuel
Computer Graphics Special Report.
Electronics June 1984.

[Mar 82] F. Martinez
Vers une approche systématique de la synthèse d'image : aspects logiciels et matériels.
Thèse Doctorat D'Etat - Grenoble - Novembre 1982.

[Mar 84a] F. Martinez
Vers une architecture banalisée des synthétiseurs d'images
Premier Colloque Image, Biarritz - Mai 1984.

[Mar 84b] F. Martinez
La synthèse d'image
Editest - 1984.

[Mat 84] P. Matherat.
Vers un contrôleur d'écran graphique VLSI.
Technique et Sciences Informatiques, Vol. 3 - N° 2 -1984.

[MaM 84] C. Machover - W. Myers
Interactive Computer Graphics.
IEEE. Computer - October 1984.

[McC 84] J.J. McC Ormick
Present futur color display technologies for graphics.
Computer and Graphics Vol 8 - N° 3 - 1984.

[Mer 79] M. Meriaux
Etude et Réalisation d'un Terminal Graphique Couleur Tridimensionnel Fonctionnant par Taches.
Thèse de docteur-ingénieur, Lille, Janvier 1979.

[MKs 84] A. Mohsen - R. Kung. J. Schutz
C. MOS 256 K RAM with wideband out put stands by on microwatts
Electronics - June 1984.

[Mok 84] N. Mokhoff
Thirty-two bit micros power workstations
Computer Desing- June 1984.

[Mur 84] G. M. Murch
Physiological Principles for the Effective use of Color.
IEEE Computer Graphics and Applications, November 1984.

[Mye 84] W. Myers
Staking out the Graphics Display Pipe-Line
IEEE Computer Graphics and Applications july 1984.

[NeB 77] M. Newell - J. Blinn
The progression of realism in computer generated images
Proc. Annal conference A.C.M. 77
Seattle USA - October 1977.

[Nic 84] R. Nickel
The IRIS Workstation
IEEE Computer Graphics and Applications August 1984.

[OJP 84] D. Oakley - M.E. Jones. D. Parsons - G. Burke
Pixel Phasing smoothes out jagged lines.
Electronics. June 1984.

[PNG 83] R Pinkham - M. Novak - K. Guttag
Video RAM excels at fast graphics.
Electronic Design. August. 1983.

[RuW 80] S. M. Rubin - T. Whitted
A 3-D Dimensional Representation for fast Rendering of Complex Scenes
ACM Computer Graphics SIGGRAPH 1980

[Sar 82] M. T. Sarrazin
Analyse et implémentation du logiciel pilote d'un synthétiseur d'images
Mémoire d'Ingénieur CNAM - Grenoble 1982

[Sch 83] D. Schweitzer
Artificial Texturing : an Aid to Surface Visualisation
ACM Computer Graphics. Vol. 17, N°.3, July 1983.

[SSS 74] J. Sutherland - F. Sproull and R. Schumacker
A Characterization of ten hiddden-surface algorithms
ACM Computing Surveys, Vol 6 N° 1 ACM - march 1974.

[Sha 85] S. F. Shapiro.
SID' 85 Displays variety of information
Computer Desing April 1985.

[Sut 63] I. E. Sutherland
SKETCHPAD : A Man Machine Graphical Communication Systeme
AFIPS SJCC Conf. Proc. Vol 23. 1963.

[TaG 78] E. Tannas - W. F. Goede
Flat-panel displays : a critique
IEEE Spectrum - July 1978.

[Thl 83] N. T. Thanhauser, - C. Infante
Opening new windows on the world
Computer Graphics world 4/83.

[VeG 84] C. P. Verbeck, D. P. Greenberg
A comprehensive light-source description for Computer Graphics
IEEE, Computer Graphics and Application, July 1984

[War 69] J. E. Warnock
A hidden surface Algorithm for Computer Generated Half-Tone Pictures
TR N° 4-15 University of Utah 6-1984.

[War 83] D. R. Warn
Lighting Controls for Synthetic Images
ACM Computer Graphics Vol. 17, N°3, 1983.

[Wat 70] G. S. Watkins
A Real-Time Visible surface Algorithm
TR UTEC CSC - 70 - 107 - University Utah 6-70.

[WeA 77] K. Weiler and P. Atherton
Hidden surface Removal Using Polygon Area Sorting
Computer graphics Vol 11 - N° 3 - ACM 1977.

[Whi 80] J.T. Whitted
An Improved Illumination Model for Shaded Display
CACM, Vol. 23, N°. 6, 6 - 80.

[Whi 84] M. C. Whitton
Memory Resing for Raster Graphics Displays
IEEE Computer Graphics and Applications - March 1984.

[Zar 84] V.H. Zarate
L'architecture modulaire d'Hélios
Document Interne au laboratoire ARTEMIS. Février 1984.

A U T O R I S A T I O N de S O U T E N A N C E

VU les dispositions de l'article 3 de l'arrêté du 16 avril 1974

VU les rapports de présentation de Messieurs
. F. MARTINEZ, Professeur
. M. MERIAUX, Chargé de recherche

Monsieur ZARATE SILVA Victor Hugo

est autorisé à présenter une thèse en soutenance en vue de l'obtention du diplôme de DOCTEUR-INGENIEUR, spécialité "Informatique".

Fait à Grenoble, le 17 octobre 1985

D. BLOCH
Président
de l'Institut National Polytechnique
de Grenoble

Thèse de Docteur-Ingénieur

Nom de l'auteur: Victor Hugo ZARATE SILVA.

Etablissement: Institut National Polytéchnique de Grenoble.

Résumé:

Les systèmes de syntèse d'images conçus sur une base matérielle fixe posent des problèmes, parfois sans solution, lorsque l'on veut changer leur champ d'application (ou les faire évoluer). L'utilisation d'une architecture banalisée permet la reconfiguration du système en fonction des besoins spécifiques des diverses applications. Elle donne au système une modularité et une facilité d'extension autorisant des performances variées (bas de gamme jusqu'au temps réel).

L'objet de cette thèse est l'étude d'opérateurs de synthèse adaptés à une architecture banalisée. Une attention particulière a été accordée aux opérateurs améliorant l'aspect réaliste des objets synthétisés. Ce travail a abouti à la réalisation d'opérateurs cablés (calcul d'éclairage, apposition des textures) fonctionnant en temps réel. Cette réalisation confère au système une grande interactivité alliée à un haut degré de réalisme.

Mots clés :

Synthèse d'images - Système graphique interactif - Architecture banalisée Processeur graphique - Processus cablé - Textures - Eclairage