الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Département d'Electronique

PROJET DE FIN D'ETUDES

Thème:

Manipulations graphiques sur PC. Application à la reconstitution tridimensionnelle d'objets à partir de leurs coupes 2D.

Proposé par:

Mme M. BEDDEK

M. R. SADOUN

Etudié par:

Samir BELKACEMI

Salim RENANE

Session: Octobre 1997

ملخص

هذه المذكرة تتطرق إلى تصميم و إنجاز برنامج يسمح بتحقيق و تحريك أجسام ثلاثية الأبعاد مع أخذ الإنارة بعين الإعتبار.

كما يعطينا إمكانية إعادة بناء الأجسام من قطوعها و الإبشار على تحريكها.
كل هذه البرامج أنجزت تحت نظام النوافذ (ويندوز) .

Résumé

Cette thèse traite de la conception et de la réalisation d'un logiciel permettant la création et la manipulation de formes tridimensionnelles avec simulation d'éclairage.

Il offre aussi la possibilité de faire la reconstitution d'objets à partir de leurs coupes et de procéder à leur manipulation.

Ce logiciel a été réalisé sous l'environnement Windows.

Abstract:

This thesis treats about the conception and the realization of a software which permits the creation and the manipulation of three-dimensionals objects with light simulation.

It also gives the possibility to do the reconstitution of the objects from their sections and to proceed to their manipulation.

This software was realised under Windows system.

SOMMAIRE

INTRODUCTION GENERALE

INTRODUCTION GENERALE

Aujourd'hui, plus aucun domaine n'échappe à l'informatique. Bien que cette discipline soit une des plus jeunes, elle a réussi à s'imposer et à s'infiltrer partout. Ainsi cinéma, bureautique, armement, médecine, dépendent beaucoup de l'outil informatique.

Mais s'il est un domaine qui a profité des progrès spectaculaires qu'a connue et que connaît encore l'informatique, c'est bien le traitement d'images. Ainsi, les vieux micro-ordinateurs qui n'affichaient que deux couleurs, et le plus souvent que du texte, sont maintenant des objets de musées. Eu égard aux développements prodigieux des microprocesseurs et des périphériques associés, les procédés de traitement des images se sont vite développés.

Ces développements ont surtout permis la naissance d'une branche : la synthèse d'image, encore appelée infographie *(Computer Graphics)*. On pouvait ainsi non pas travailler sur des images existantes, mais carrément les créer; ce qui a ouvert des voies encore inexplorées, comme la réalisation de trucage plus performants pour le cinéma.

Comme nous l'avons dit précédemment, le domaine biomédical a lui aussi profité de tous ces développements, car l'introduction des différentes techniques de traitement d'images a permis de faciliter les diagnostics en rendant les méthodes de dépistage plus sûres et plus précises.

Ces techniques sont par exemple associées à des méthodes d'acquisition comme l'échographie où on utilise les procédés de filtrage afin d'éliminer le bruit et de rendre les images plus nettes.

Ces techniques font donc appel aux différents procédés de traitement d'images 2D. Certes, cela permet un dépistage plus aisé de certaines maladies graves, mais on n'a toujours qu'une représentation bidimensionnelle et partielle (généralement des coupes), de l'organe étudié.

Il fallait donc penser à de nouveaux procédés de visualisation. Ainsi, n'aimerait-on pas allez demain chez son médecin, sachant que celui-ci est capable d'effectuer une acquisition de l'image de l'organe malade, puis d'en faire une reconstitution en 3 dimensions, qu'il pourrait observer sous tous les angles afin de déceler une éventuelle anomalie de façon plus rapide.

De tels systèmes existent déjà mais ne connaissent pas une grande diffusion, et sont le plus souvent utilisés en chirurgie. Cependant, les progrès réalisés dans le domaine des calculateurs, et du matériel informatique en général accélérera sans doute leur diffusion; cela permettrait d'offrir des possibilités inconnues jusqu'à ce jour.

Le travail que nous avons fait consiste en la réalisation d'un logiciel capable de générer des formes 3D avec prise en compte des surfaces cachées et du modèle d'éclairage.

Nous avons aussi implémenté un autre module capable de restituer des formes 3D à partir de coupes 2D d'un objet. Cette technique peut être préconisée en médecine pour la reconstitution tridimensionnelle des organes.

Cette implémentation s'est faite sur un PC, en utilisant sa carte VGA comme périphérique d'affichage.

Nous allons donc aborder au préalable, l'aspect matériel au chapitre II. Nous étudirons donc en détail la carte graphique afin de mieux comprendre le déroulement interne des opérations d'affichage.

Au chapitre III, nous développerons les différentes étapes de la génération d'images de synthèse 3D et ferons le choix des méthodes à adopter.

Au chapitre IV, nous nous intéresserons à la partie consacrée à la reconstitution des objets 3D à partir de leurs coupes. Nous verrons ainsi les différents procédés de traitement d'images et conclure sur les méthodes à utiliser.

Au chapitre V nous exposerons le travail effectué. Nous commencerons par présenter le logiciel en expliquant les différentes fonctions qu'il propose. On s'intéressera par la suite à l'implémentation des parties essentielles.

Enfin, nous finirons par tirer une conclusion sur tout ce travail et tracerons les différentes perspectives que nous jugeons envisageables.

CHAPITRE II

SYSTEME GRAPHIQUE

SYSTEME GRAPHIQUE

2.1 Introduction

Nous ne pouvons pas parler de graphisme sur micro-ordinateurs, sans évoquer les cartes vidéo. Ce sont elles, associées à un écran, qui se chargent de l'affichage et rendent ainsi possible la visualisation de données, graphes ou images par l'utilisateur.

Il existe plusieurs standards, mais de nos temps, les plus utilisés sur les PC sont sans doute, les standards VGA et Super VGA.

Le standard Super VGA désigne les cartes permettant l'affichage de 16 millions de couleurs en même temps, et permettent d'avoir donc une qualité d'affichage en "vraies couleurs".

Nous allons surtout nous intéresser dans ce qui suit aux cartes VGA. Elles sont moins performantes que les cartes Super VGA puisqu'elles permettent au maximum l'affichage de 256 couleurs avec une résolution de 320*200. Cependant, elles sont très populaires auprès des programmeurs, car elles offrent d'immenses possibilités d'accès direct à l'électronique (programmation des registres internes) et la programmation de ces registres reste assez simple.
On va donc décrire la structure des données à l'intérieur de la RAM vidéo pour les différents modes graphiques (16 et 256 couleurs) ainsi que le rôle de certains registres, qui interviennent le plus souvent dans la programmation de ces cartes.

2.2 Composants de la carte VGA

Il est nécessaire pour mieux comprendre le fonctionnement de la carte VGA, de connaître les différents constituants de cette carte [TIS 95].

La carte VGA compte ainsi plusieurs blocs, destinés à un rôle précis. Ils se composent tous de plusieurs registres qu'on peut programmer.

2.2.1 Les registres généraux

C'est un groupe de registres qui ont pour tâche essentielle de nous informer sur le mode de fonctionnement.

2.2.2 Le contrôleur CRT

Sa tâche est de configurer l'écran en créant des signaux pour le moniteur. Les différents registres qu'il comporte sont destinés par exemple, à jouer un rôle dans le timing horizontal ou vertical.

2.2.3 Le séquenceur

Il a entre autres la tâche de diriger les accès à la RAM vidéo vers les différents plans de bits. Il s'occupe aussi de rafraîchir la RAM vidéo.

2.2.4 Le contrôleur d'attributs

Le contrôleur d'attributs a pour rôle de préparer les signaux de couleur pour l'écran. Il contrôle aussi les registres intervenant dans la génération des couleurs.

2.2.5 Le convertisseur Digital/Analogique DAC

C'est lui qui s'occupe de la conversion des codes couleurs numériques en signaux analogiques.

2.2.6 Le contrôleur graphique

Le contrôleur graphique est utilisé dans tous les accès en lecture et écriture à la RAM vidéo. Nous allons l'étudier en détail plus loin.

2.3 Synoptique de la carte VGA

D'après ce qui a été dit, nous pouvons donner un schéma synoptique simplifié de la carte VGA. Nous mettons en évidence les parties les plus importantes intervenant dans la lecture, l'écriture ou l'affichage (cf. figure 2.1).

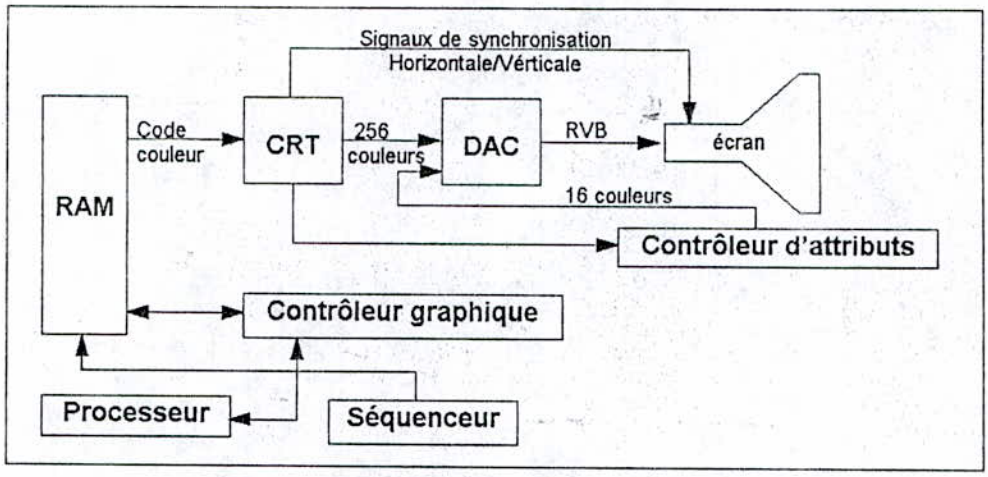


Fig. 2.1 : *Schéma synoptique de la carte VGA.*

2.4 Structure de la RAM vidéo

2.4.1 Segmentation de la RAM vidéo en plans de bits

La carte VGA possède une RAM vidéo de **256 Ko**, mais dans la zone d'adressage du PC, la carte VGA ne dispose que du segment mémoire **A** (à partir de **A000:0000** jusqu'à **A000:FFFF**).
Les constructeurs de cette carte la destinaient à une tâche inimaginable consistant à rendre adressable une RAM vidéo de 256 Ko à travers une zone de 64 Ko seulement.
La solution trouvée se présente sous la forme de quatre plans de bits répartis dans la RAM vidéo. La carte vidéo libère ainsi 64 Ko sur chaque plan de bits adressable à travers le segment mémoire **A**.

2.4.2 Fonction et signification des registres Latch

La carte VGA active quatre registres 8 bits entre l'unité centrale et la RAM vidéo appelés aussi registres

Latch. Chaque registre correspond à l'un des quatre plans de bits.

Si un programme exécute un accès en lecture sur un octet de la RAM vidéo, les quatre registres Latch sont chargés avec l'octet à partir de leur plan de bits.

Son adresse d'offset est indiquée lors de l'opération de lecture. Si cette adresse vaut par exemple 9, c'est le dixième octet (0 étant le premier) issu du premier plan de bits qui est chargé dans le premier registre Latch, le dixième octet du second plan de bits sera chargé dans le second registre Latch et ainsi de suite pour le troisième et quatrième registre Latch (cf. figure 2.2).

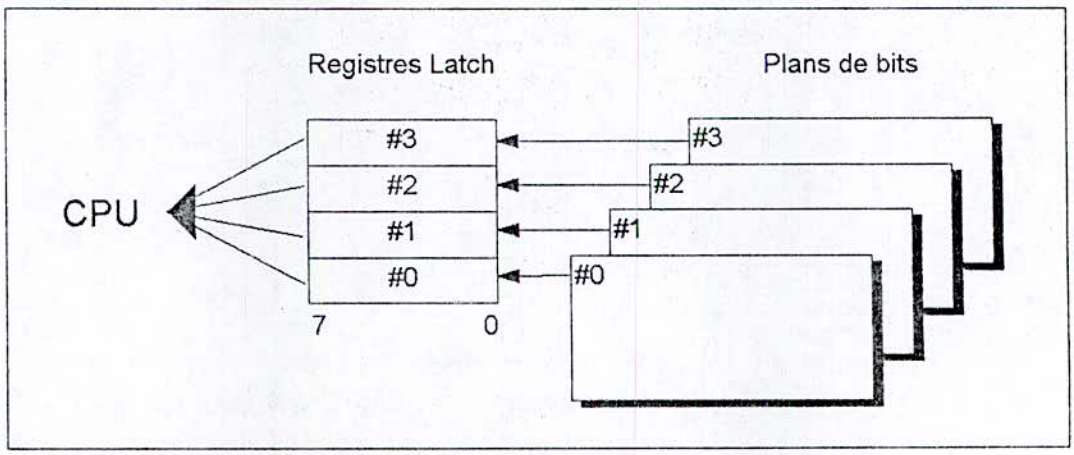


Fig. 2.2 : ***Les registres Latch en lecture.***

La même procédure s'applique lors d'un accès en écriture à la RAM vidéo. Ici, le contenu des quatre registres Latch est inscrit dans leur plan correspondant (cf. figure 2.3).

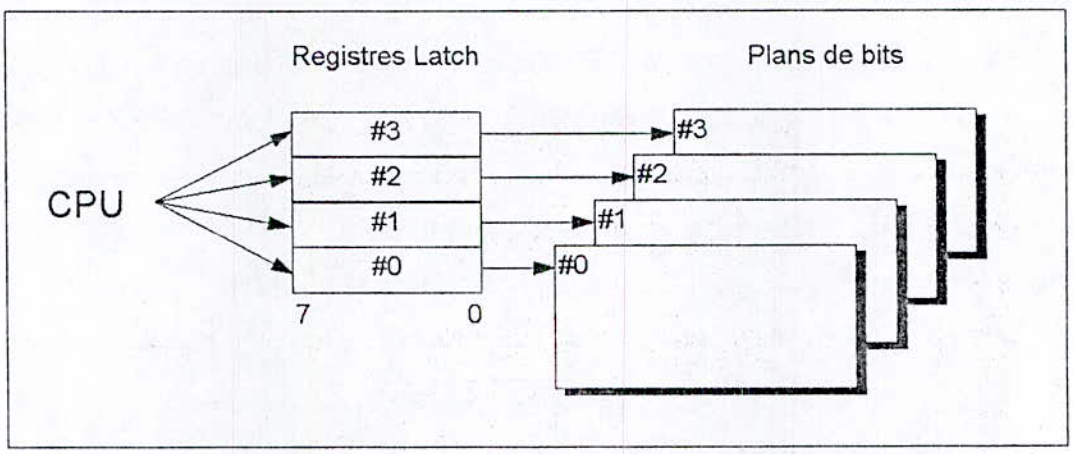


Fig. 2.3 : ***Les registres Latch en écriture.***

Mais lors d'un accès en lecture à la RAM vidéo, il faut renvoyer un octet à l'unité centrale. De même, un octet doit être transféré depuis l'unité centrale vers la RAM vidéo lors d'un accès en écriture, c'est là qu'intervient le rôle du contrôleur graphique.

2.5 Accès en lecture/écriture à la RAM vidéo

Nous allons parler des différents modes de lecture et d'écriture qu'offre la carte VGA. Mais il nous faudra d'abord détailler la fonction de certains registres appartenant au contrôleur graphique, qui joue un rôle important dans ces opérations.

2.5.1 Rôle du contrôleur graphique

Le contrôleur graphique est utilisé lors de tous les accès en lecture et en écriture à la RAM vidéo. Il contrôle les transferts effectués depuis l'unité centrale vers la RAM vidéo.
Il compte neufs registres, qui sont présentés au tableau 2.1 [TIS 95].

Registre	Signification	Défaut
00h	Set/Reset	00h
01h	Enable Set/Reset	00h
02h	Color compare	00h
03h	Function Select	00h
04h	Read Map Select	00h
05h	Graphics Mode	00h
06h	Miscellaneous	divers
07h	Color Don't Care	0Fh
08h	Bit Mask	FFh

Tableau 2.1 : ***Les différents registres du contrôleur graphique.***

Ils déterminent la provenance, la destination et le type de tous les accès à la RAM vidéo.
L'adressage de ces registres se fait à travers deux registres. Un registre de données qui se trouve à l'adresse 3CFh, et un registre d'index qui lui se trouve à l'adresse 3CEh. Le registre d'index sert à spécifier le numéro du registre à adresser, et le registre de données contient la donnée écrite dans ce registre ou lue à partir de ce dernier.

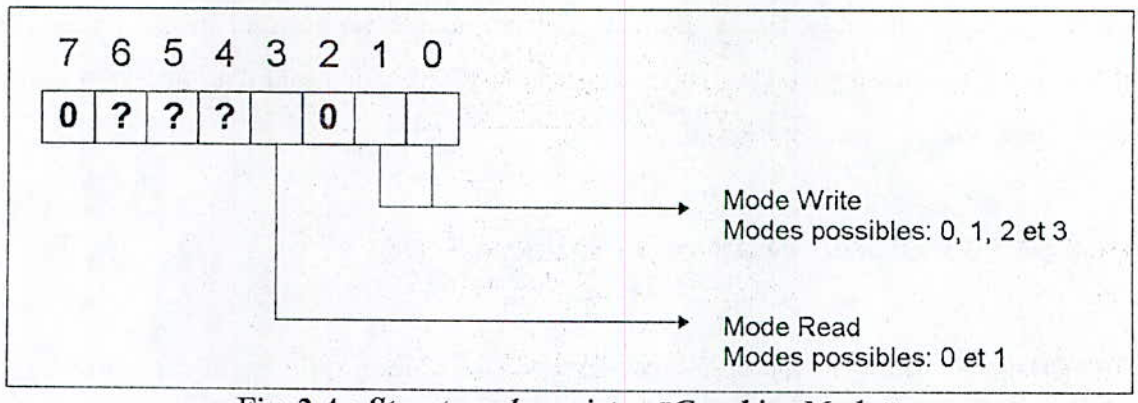


Fig. 2.4 : *Structure du registre "Graphics Mode".*

Les accès à la RAM vidéo sont réglés par le contenu du registre **Graphics Mode** (n°5). Son contenu définit les modes de lecture et d'écriture. Sa structure est donnée par la figure 2.4.

2.5.2 Accès en lecture

2.5.2.1 Mode Read 0

Le mode **Read 0** permet à un programme de lire des octets provenant d'un plan de bits déterminé. Cela est par exemple intéressant lorsqu'il s'agit de sauvegarder une partie de la RAM vidéo. Sous ce mode, lors d'un accès en écriture à la RAM vidéo, l'octet appelé pour chacun des quatre plans de bits est tout d'abord chargé dans les quatre registres Latch correspondants.

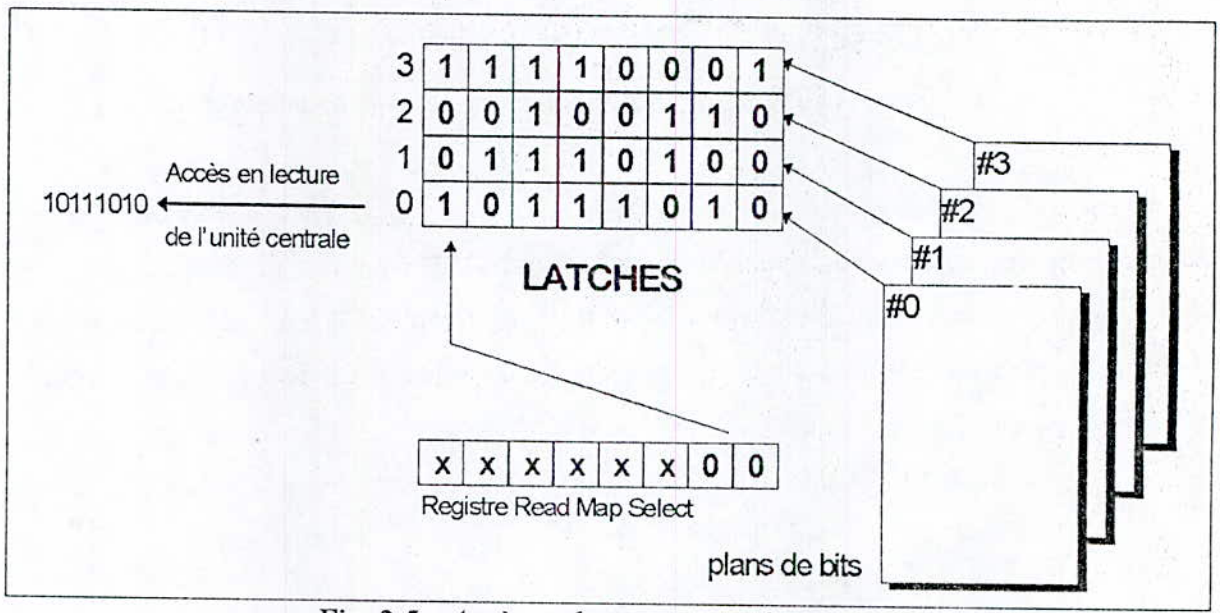


Fig. 2.5 : *Accès en lecture en mode Read 0.*

Comme le montre la figure 2.5, le contenu du registre Latch dont le numéro est spécifié dans les deux bits inférieurs du registre **Read Map Select** (registres n°4 du contrôleur graphique) est ensuite transmis à l'unité centrale.

2.5.2.2 Mode Read 1

Ici, il ne s'agit pas d'envoyer tel quel le contenu d'un registre Latch à l'unité centrale, mais on va lui appliquer toutes sortes de combinaisons logiques avec différents registres.
La tâche de ce mode consiste à vérifier si les bits des quatre registres Latch occupent une valeur déterminée. Ceci peut être utilisé pour la recherche des points dotés d'un code couleur spécifiques.

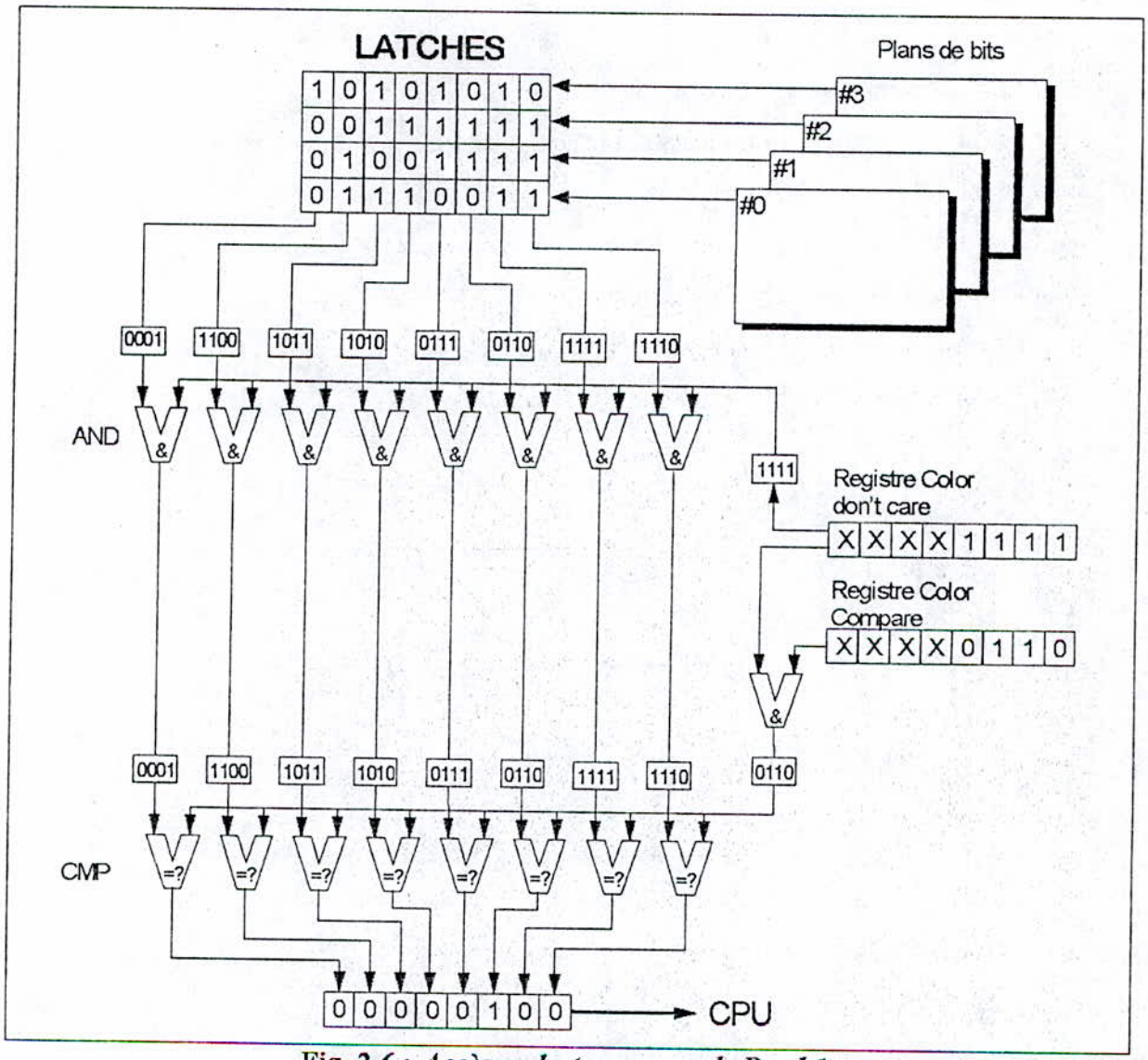


Fig. 2.6 : ***Accès en lecture en mode Read 1.***

Une fois les quatre registres Latch chargés, huit groupes composés chacun de quatre bits sont constitués. Chaque groupe contient les quatre bits qui occupent une position identique à l'intérieur des quatre registres Latch. Chaque groupe est composé ensuite avec la valeur fournie par le registre **Color Compare**. L'octet issu de la comparaison est transmis à l'unité centrale.

Dans cet octet, les bits mis à 1 sont ceux dont le groupe correspondant a obtenu la valeur fournie par le registre **Color Compare**. On est arrivé donc à identifier le groupe qui correspond à la valeur transmise par ce registre (voir figure 2.6).

2.5.3 Accès en écriture

2.5.3.1 Mode Write 0

Dans ce mode de nombreuses combinaisons liées au contenu des registres se créent lors de l'accès à la RAM vidéo. En fait le contenu du registre **Bit Mask** (registre n°8 du contrôleur graphique) décide si le contenu d'un bit des quatre registres Latch doit être transmis tel quel dans les quatre plans de bits, ou s'il nécessite quelques traitements. Par exemple si un bit contient **0** dans le registre **Bit Mask**, le bit correspondant dans les quatre registres Latch est transmis tel quel dans les quatre plans de bits.

Si le bit contient la valeur **1**, une combinaison se crée dans ce cas, sa structure est définie par le contenu du registre **Function Select** comme le montre la figure 2.7.

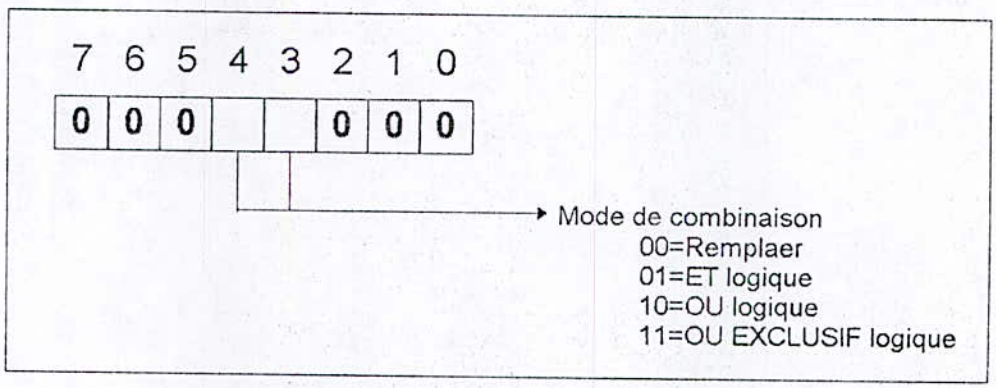


Fig. 2.7 : ***Structure du registre " Function Select ".***

Le contenu du registre **Enable Set/Reset** décide quel devra être le partenaire de ces bits dans l'opération logique. Si les quatre bits inférieurs contiennent la valeur **1111**, l'opération logique se fera avec le contenu des quatre bits inférieurs du registre **Set/Reset**. Chacun de ces bits sera combiné avec un Latch

par l'opération définie par le registre **Function Select**, mais si elle contient la valeur **0000** elle se fera avec le contenu des quatre bits inférieurs de l'octet de l'unité centrale (cf. figure **2.8**).

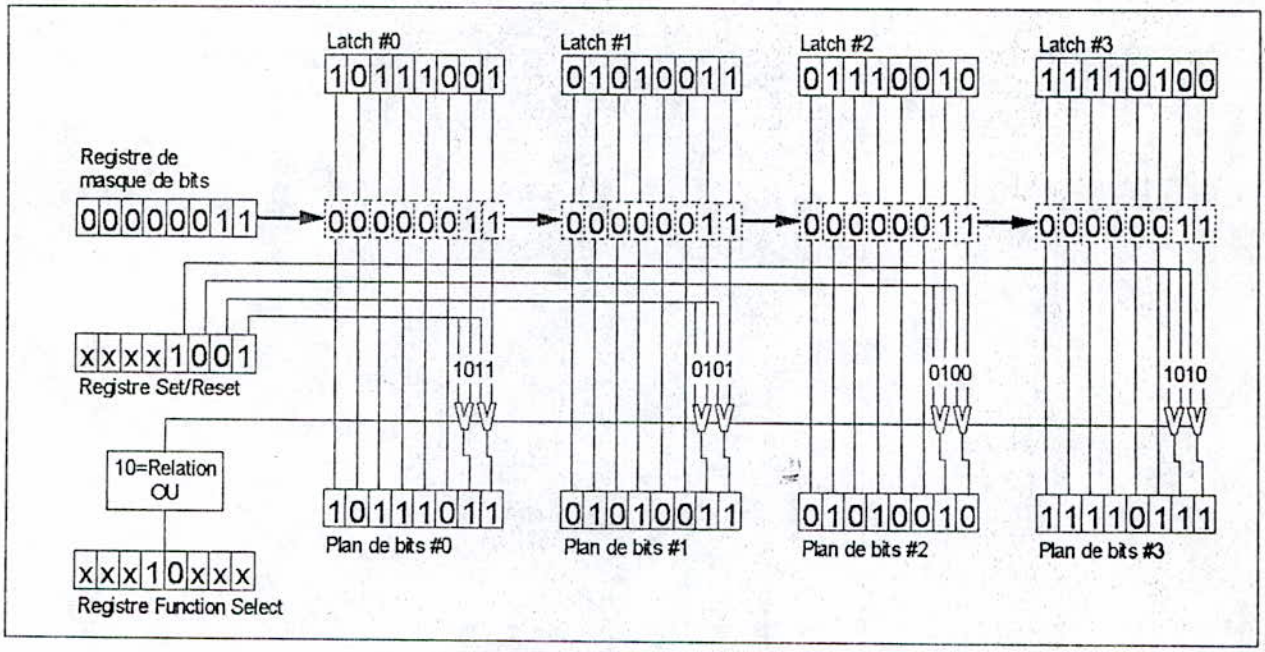


Fig. 2.8 : *Accès en écriture en mode Write 0.*

2.5.3.2 Mode Write 1

Le contenu des registres et celui de l'octet transmis par l'unité centrale ne joue aucun rôle ici, car le contenu des quatre registres Latch est écrit tel quel à l'adresse d'offset indiquée dans les quatre plans de bits.

Cela peut être intéressant, par exemple, lorsqu'il s'agit de copier les codes couleur de huit points contigus dans huit autres points.

Dans ce cas, l'octet contenant les huit points peut d'abord être lu avec un mode **Read** et être ainsi chargé dans les différents registres Latch. L'étape suivante consistera alors à effectuer un accès en écriture à un autre octet de la RAM vidéo.

2.5.3.3 Mode Write 2

C'est la même chose que pour le mode **Write 0** seulement dans ce cas quel que soit le contenu du registre **Enable Set/Reset** les bits inférieurs de l'octet de l'unité centrale sont systématiquement combinés avec les registres Latch (figure **2.9**).Ce mode convient donc à fixer la couleur d'un point isolé.

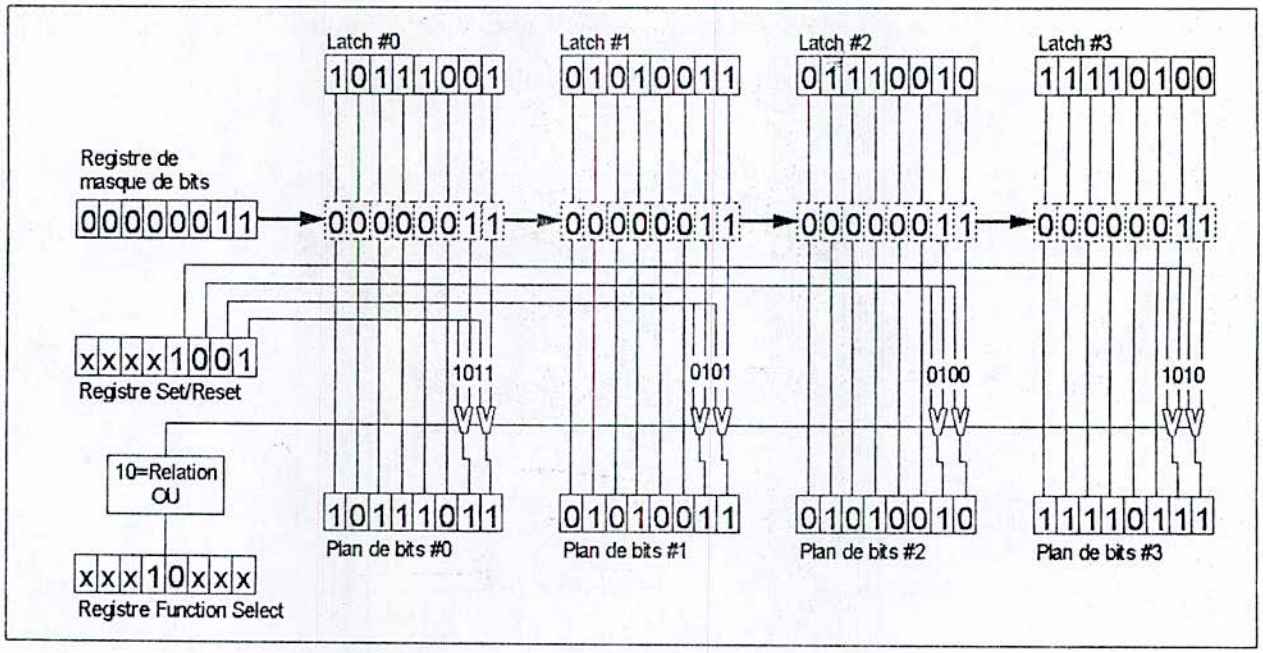


Fig. 2.9 : *Accès en écriture en mode Write 2.*

La sélection des différents plans de bits se fait grâce au registre **Map Mask**, c'est le registre n°2 du séquenceur. La figure 2.10 nous montre son schéma.

Il permet de verrouiller les différents plans de bits pour qu'ils soient protégés contre des accès en écriture et en lecture cela est utile lorsqu'il s'agit uniquement de manipuler le contenu d'un plan de bits déterminé.

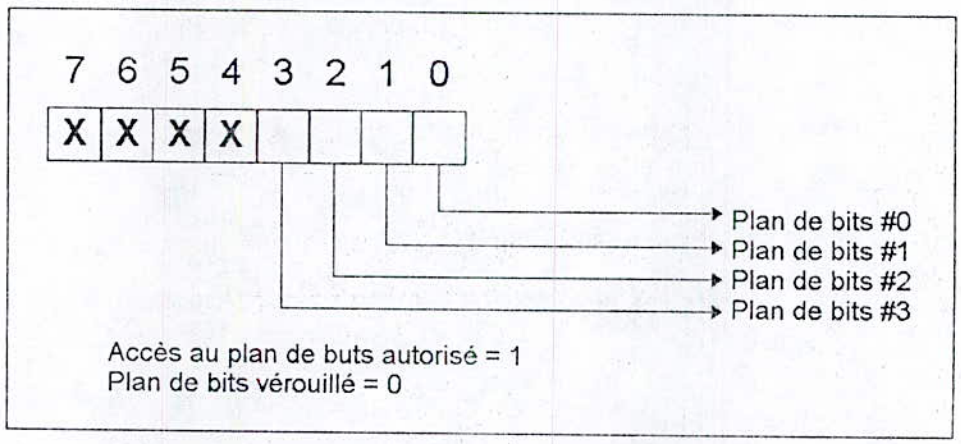


Fig. 2.10 : *Structure du registre " Map Mask ".*

2.6 Les modes graphiques 16 couleurs de la carte VGA

Les différents modes graphiques 16 couleurs de la carte VGA sont donnés au tableau 2.2 [TIS 95].

Mode	Résolution	Mémoire	Pages	Octets restants
0Dh	320*200	32000	8	6144
0Eh	640*200	64000	4	6144
10h	640*350	112000	2	38144
12h	640*380	153600	1	108544

Tableau 2.2 : *Les modes graphiques 16 couleurs de la carte VGA.*

On remarque que quand la résolution est peu importante, la place mémoire nécessitée par une page d'écran est inférieure à la taille totale. Cela permet de stocker en même temps plusieurs pages d'écran dans la RAM vidéo. Ceci est particulièrement appréciable dans certaines applications comme l'animation.

2.6.1 Structure de la RAM vidéo en mode 16 couleurs

Dans chaque octet de la RAM vidéo, l'information de couleur correspond à huit points situés côte à côte. Autrement dit, chacun des huit points représente huit bits distincts. Cela ne suffit pas pour les besoins d'un affichage en 16 couleurs qui nécessite quatre bits par pixel. Pour obtenir ce résultat, on rassemble à chaque fois quatre octets consécutifs issus des quatre plans de bits (voir figure 2.11).

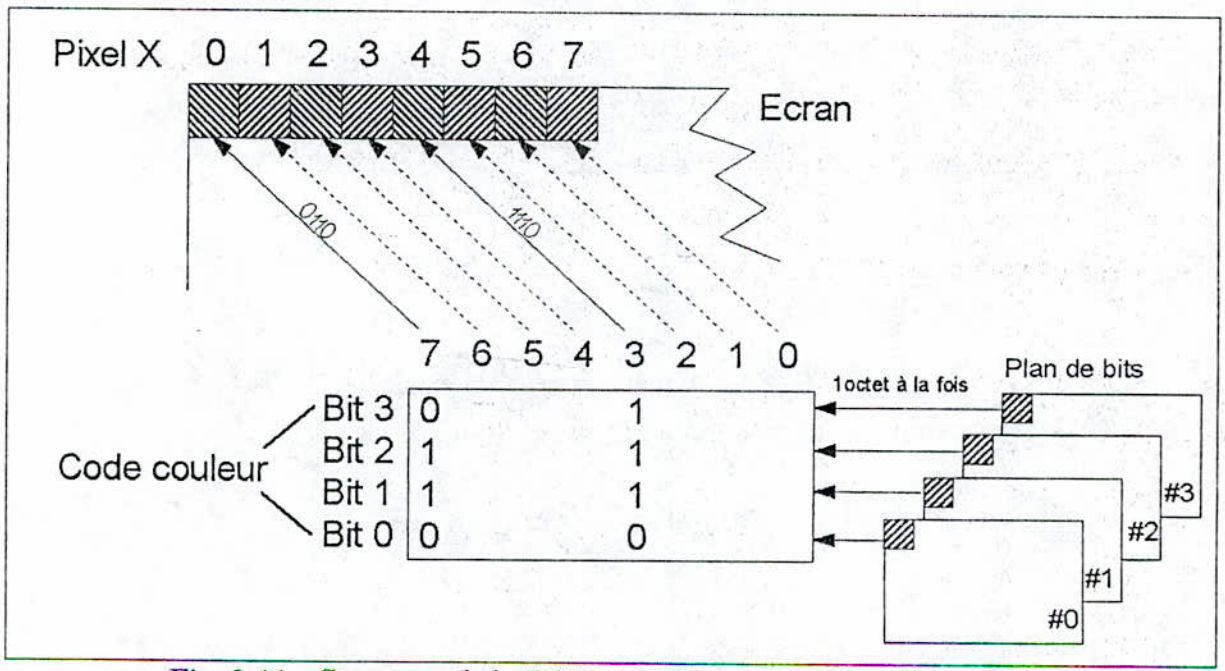


Fig. 2.11 : *Structure de la RAM vidéo dans les modes 16 couleurs.*

L'adresse d'offset d'un point de coordonnées x, y sur l'écran est calculée à l'aide de la formule suivante:

$$Offset = y * (résolution\ horizontale / 8) + [x / 8] \quad (2.1)$$

Où [ ] représente la partie entière.

L'adresse d'offset ainsi calculée contient les informations pour huit points consécutifs. Il faut donc isoler un bit dans les quatre octets de plans de bits pour atteindre le code couleur du point souhaité. Le numéro de ce bit résulte de la formule suivante:

$$Bit = 7 - (x \bmod 8) \quad (2.2)$$

Où **mod** représente le reste de la division par 8.

2.7 Les modes graphiques 256 couleurs

Comme nous l'avons dit précédemment, la carte VGA offre la possibilité de travailler en mode **256** couleurs. Mais contrairement au mode **16** couleurs, nous ne pouvons avoir qu'une seule résolution qui est **320*200**.

Dans ce mode, chaque pixel occupe un octet dans la RAM vidéo; ce qui correspond à un code couleur entre **0** et **255**. La carte VGA considère ce code couleur comme un index dans la table de couleurs **DAC** d'où est extraite la couleur finale du point concerné.

2.7.1 Le mode 256 couleurs classique

C'est le mode par défaut. L'adresse d'offset par rapport au début de la RAM vidéo d'un point de coordonnés x, y sur l'écran et calculée par la formule suivante :

$$Offset = y * 320 + x \quad (2.3)$$

Nous remarquons donc que l'adressage est linéaire.

Grâce à la structure d'organisation peu complexe de la RAM vidéo (voir figure **2.12**) les trois quarts de la RAM vidéo restent en effet inutilisés pour une page écran en **320*200** points, il nous faut **320*200**, soit **64000** octets en tout.

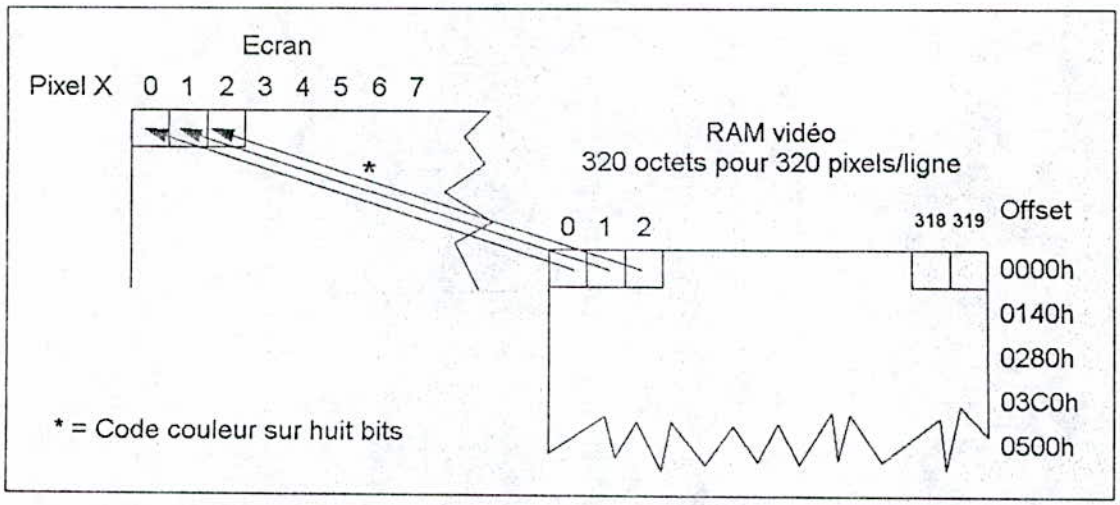


Fig. 2.12 : ***Structure de la RAM vidéo en mode 256 couleurs.***

2.7.2 Le mode X

C'est le nom qu'on donne au mode **256** couleurs avec quatre pages graphiques. Mais avant d'en parler, il faut connaître la manière dont sont stockées les données dans la RAM vidéo.

Il faut donc savoir que l'adressage linéaire de la RAM vidéo en mode **256** couleurs n'est qu'une illusion. En réalité, les informations de couleurs codées sur un octet sont reparties sur les quatre plans de bits. Le numéro du plan de bit correspond aux deux bits de poids faible de l'octet transmis. Ainsi, si ces deux bits sont égalés à **01b**, l'octet sera dirigé vers le plan n°1.

Cette répartition est dûe au mode *chaine 4* [TIS 95]. Mais il faut savoir que deux octets qui se suivent dans un même plan de bits ne sont pas adjacents (physiquement), mais sont séparés par trois octets.

Ceci fait qu'une image dont la taille est **64000** octets et qui occuperait **16000** octets dans chaque plan de bits, en occupe **64000**; soit, quatre fois plus comme le montre la figure **2.13**.

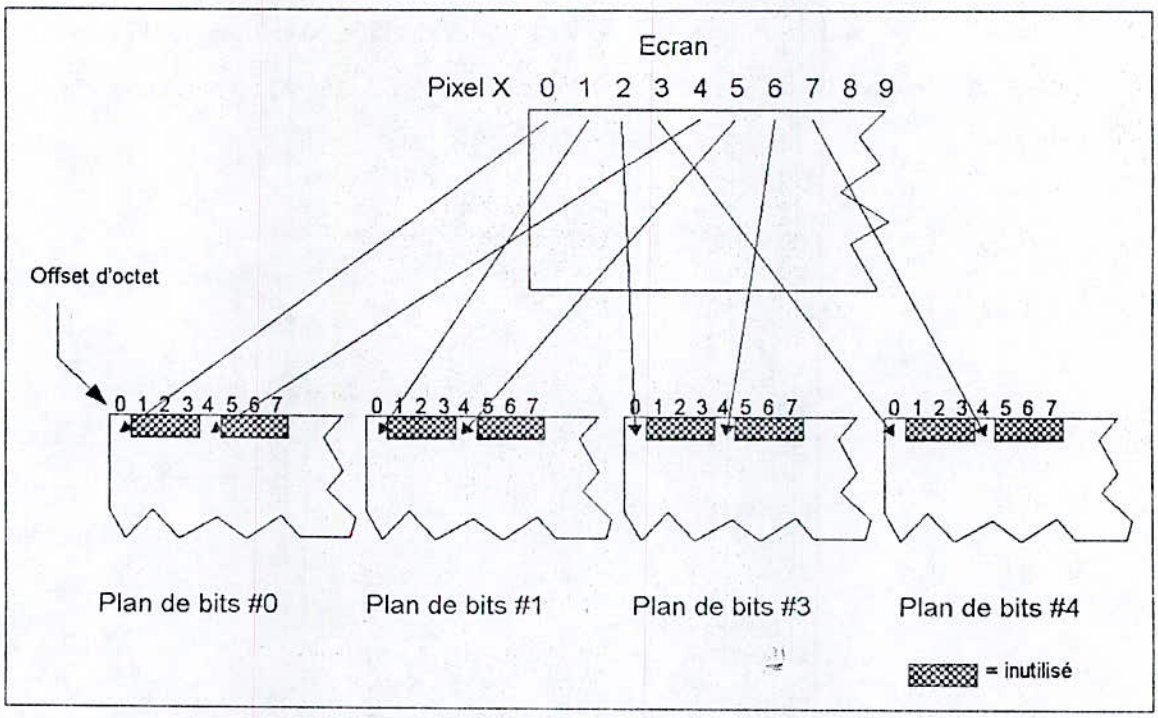


Fig. 2.13 : *Organisation interne des données en mode 256 couleurs classique.*

Il serait intéressant de supprimer cet espace (cf. figure 2.14), et de profiter des quatre pages d'écran que nous autorise les 256 Ko de la RAM vidéo, ce qui permet par exemple d'avoir des animations fluides.

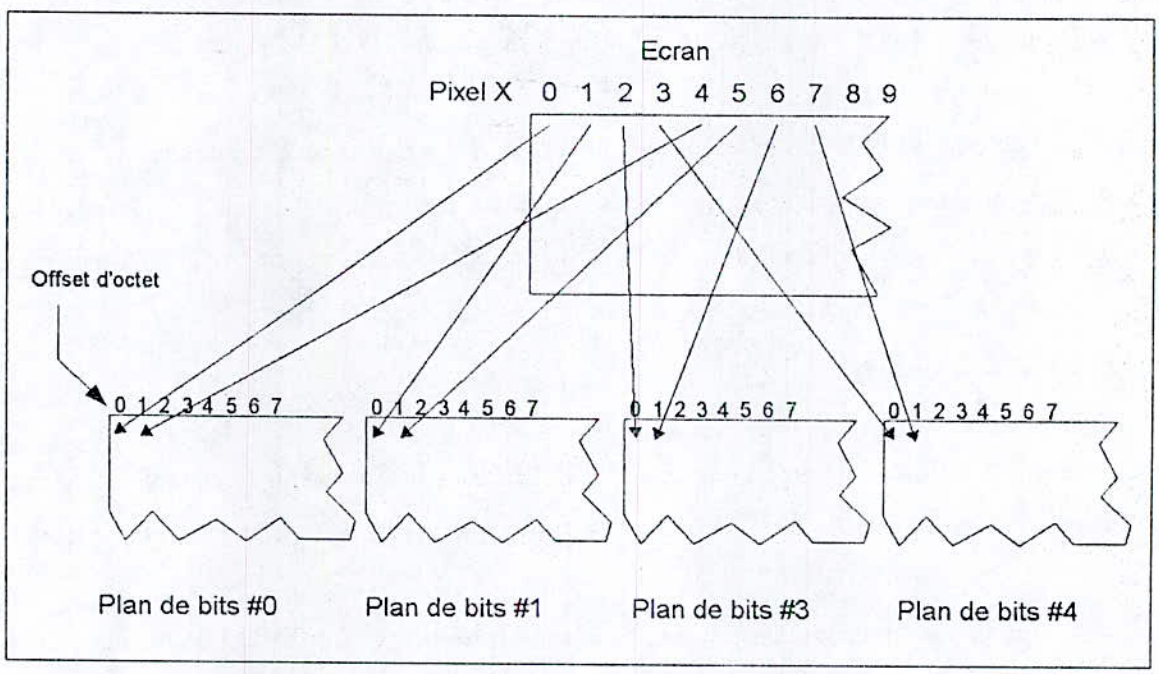


Fig. 2.14 : *Organisation interne des données en mode X.*

Pour cela, il faut renoncer au mode *chaine 4*, mais cela nous oblige à nous occuper nous même de la répartition des octets dans les différents plans de bits.
Il faut aussi programmer le **contrôleur CRT** en conséquence, et activer un mode d'adressage *Byte* au lieu du mode *Double Word* [TIS 95] [BER 94], pour éviter l'espace entre les octets de couleur.

2.8 Sélection des couleurs

2.8.1 Le mode 16 couleurs

La carte VGA dispose d'une table de couleurs DAC, constituée de **256** registres, où sont stockées les couleurs. Les codes couleurs transférés vers la RAM vidéo sont des indices dans cette table [TIS 95].
Dans la table de registres DAC, chaque couleur est codée sur **18** bits, soit **6** bits pour chacune des composantes rouge, vert et bleu. Ceci nous donne un éventail de **262144** couleurs, mais elles ne sont pas toutes affichées simultanément.
Ainsi, en mode 16 couleurs, le contrôleur d'attributs joue un grand rôle, car pour déterminer les adresses des **16** couleurs sélectionnées, on utilise **16** registres de palette et le registre **Color Select** tous appartenant au contrôleur d'attributs. Ces adresses sont conditionnée par l'état du bit **7** du registre **Mode Control** (registre n°17 du contrôleur d'attributs).
Si ce bit est à **0**, les adresses sont formées par le contenu des bits **0→5** des **16** registres de palette et des bits **2** et **3** du registre **Color Select.**
Si ce bit est à **1**, les différentes adresses sont formées par les bits **0→3** des registres de palette et les bits **0→3** du registre **Color Select.**

2.8.2 Le mode 256 couleurs

Ici, plus besoin des registres de palette, mais les codes couleurs indexent directement la table de couleurs DAC et la programmation des registres de palette n'a aucune influence sur l'affichage.

Remarque
Nous avons parlés en tout début de ce chapitre des cartes Super VGA et de leur capacités à afficher 16 millions de couleurs. Il faut dire que dans ce cas, la couleur n'est plus codée sur un seul octet, mais

chaque code couleur occupe trois octets. Dans ce cas, chaque composante rouge, vert, bleu est codée sur 8 bits, déterminant ainsi la proportion de chaque composante.

2.9 Le BIOS vidéo

Le BIOS *(Basic Input/Output System)* est un ensemble de fonctions gravées en ROM. L'emploi de ces fonctions se fait grâce à des interruptions, dites interruption BIOS.

Chaque interruption correspond à un groupe de fonctions qui exécutent des tâches de même genre (gestion des fichiers, gestion de l'impression...).

On s'intéresse surtout à l'interruption **10h**, appelée interruption vidéo. Les fonctions qu'elle regroupe offrent toutes sortes de services concernant les sorties à l'écran.

Le tableau **2.3** présente ces différentes fonctions [TIS 95].

N°	Tâche
00h	Régler le mode vidéo
01h	Définir la taille du curseur
02h	Régler la position du curseur
03h	Lire la position du curseur
04h	Lire la position du crayon optique
05h	Sélectionner la page écran actuelle
06h	Faire défiler l'écran vers le haut
07h	Faire défiler l'écran vers le bas
08h	Lire le caractère et l'attribut
09h	Ecrire le caractère et l'attribut
0Ah	Ecrire le caractère à la position du curseur
0Bh	Définir la palette de couleur pour le mode graphique
0Ch	Lire un point écran en mode graphique
0Dh	Ecrire un point écran en mode graphique
0Eh	Ecrire le texte
0Fh	Déterminer le mode vidéo en cours
10h	Définition de la couleur
11h	Accès au générateur de caractères
12h	Régler/Lire la configuration vidéo
13h	Ecrire une chaine de caractères
14h	Réservé
15h	Réservé
16h	Réservé
17h	Réservé
18h	Réservé
19h	Réservé
1Ah	Informations vidéo
1Bh	Lire les informations sur le BIOS vidéo
1Ch	Sauvegarder/ Restaurer l'état de la carte vidéo

Tableau 2.3 : *Les fonctions du BIOS vidéo.*

2.10 Conclusion

La compréhension de la structure interne des cartes VGA est essentielle pour le développement de certaines applications. Elle est même nécessaire pour leur optimisation, en recourant à la programmation directe du matériel à travers le langage machine.

Mais, il faut savoir que le grand nombre de constructeurs de cartes graphiques fait que chacun ajoute son petit "plus" au produit qu'il fabrique, comme des résolutions plus grandes en 256 couleurs, ou bien un plus grand choix de couleurs. Ceci a pour conséquence de rendre plus difficile l'utilisation optimale de ces cartes, ces nouvelles fonctions ne faisant pas partie du standard VGA.

En conclusion nous dirons que pour notre application, et en utilisant la carte VGA, nous privilégierons les modes 256 couleurs car ils offrent plus de confort et permettent la construction d'images de meilleure qualité.

CHAPITRE III

ELABORATION DES IMAGES 3D

ELABORATION DES IMAGES 3D

3.1 Introduction

L'un des domaines qui a le plus marqué cette décennie, est sans doute la synthèse d'images tridimensionnelles. Les performances sans cesse croissantes des ordinateurs n'y sont pas étrangères.

Le matériel réservé, il y a encore quelques années aux seuls professionnels et à quelques privilégiés (cartes graphiques qui permettent d'afficher 16 millions de couleur simultanément...); est maintenant accessible à tous et à des prix raisonnables, contribuant ainsi à la démocratisation d'un domaine si fascinant et qui a tant fait rêver: la création d'images 3D.

Mais nous devons voir au-delà de cet aspect attirant et fascinant, et remonter la chaîne de fabrication de ces images, pour connaître les différentes étapes de leur conception. Pour cela, nous devons connaître les modèles utilisés et les algorithmes exploités pour chaque étape, afin de distinguer lesquels sont les mieux adaptés à notre travail et à nos moyens, en choisissant par exemple ceux qui ne demandent pas de très long temps de calcul.

En général, la création d'images 3D suppose 2 grandes étapes:

Le modelage "Modelling" : C'est la première étape, et elle consiste à décrire les formes et les objets d'une image, utilisant pour cela plusieurs représentations, associées à un système de coordonnées.

Nous pouvons ainsi faire appel à une représentation discrète soit volumique (voxel) ou surfaciques (polyédrique), ou bien à une représentation analytique par courbes mathématiques.

Dans l'étape de modelage, nous pouvons inclure les différentes transformations géométriques (rotation, translation) qui vont nous permettre de "placer" nos objets autorisant ainsi la création d'images très complexes.

Le Rendu "Rendering": En second vient l'étape du rendu, qui permet à partir des données d'une image (forme des objets, caractéristiques physiques, et de l'éclairement) de calculer l'image perçue par l'observateur (ou plutôt par une caméra virtuelle qu'on simule). Cette étape englobe plusieurs étapes intermédiaires. On commence d'abord par éliminer les surfaces cachées, ensuite on applique les modèles d'éclairement pour simuler les sources lumineuses. Mais il ne faut pas perdre de vue que les dispositifs de visualisation sont bidimensionnels (écran à balayage), ce qui nous oblige à faire une transformation 3D→2D pour passer de l'espace tridimensionnel vers un espace bidimensionnel. Cette transformation porte le nom de projection.

Dans toutes ces étapes de création et selon le modèle utilisé pour la simulation de chaque partie, les temps de calcul peuvent varier d'une façon spectaculaire, et il est évident que le prix à payer pour avoir des images de synthèse hyperréalistes c'est le temps de calcul souvent trop excessif. Le problème reste posé même de nos jours, et les animations en temps réel avec une qualité d'image supérieur ne sont possible que sur des stations graphiques.

Cependant, une brèche a été ouverte et les domaines qui utilisent l'infographie sont de plus en plus nombreux allant de la CAO (Conception Assistée par Ordinateur) jusqu'à la médecine passant par la publicité et le cinéma.

3.2 Généralités

3.2.1 Systèmes de coordonnées

Il est possible de situer un point P de différentes manières dans l'espace.

On peut ainsi lui associer ses coordonnées cartésiennes $P(x,y,z)$ (cf. figure **3.1.a**), ou bien ses coordonnées sphériques $P(r, \theta, \varphi)$ (cf. figure **3.1.b**).

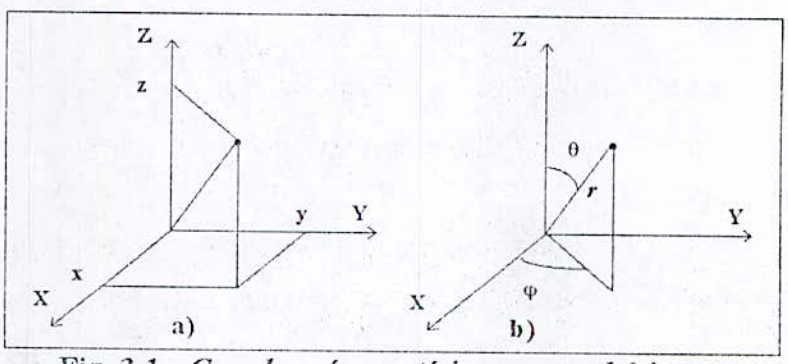


Fig. 3.1 : *Coordonnées cartésiennes et sphériques.*

Ayant les coordonnées sphériques d'un point P, il est possible de passer aux coordonnées cartésiennes par la relation suivante:

$$\begin{cases} x = r\sin\theta\ \cos\varphi \\ y = r\sin\theta\ \sin\varphi \\ z = r\cos\theta \end{cases} \qquad (3.1)$$

3.2.2 Transformations géométriques de base [SCH 87b][BRE 88]

Nous allons parler des transformations géométriques de base que sont la rotation, la translation, et le changement d'échelle et ceci dans un espace tridimensionnel.

3.2.2.1Translation

On considère le point $P(x,y,z)$, si nous lui appliquons une translation $Trans(T_x,T_y,T_z)$ nous obtiendrons le point $P`(x`,y`,z`)$ tel que:

$$\begin{cases} x' = x + T_x \\ y' = y + T_y \\ z' = z + T_z \end{cases} \qquad (3.2)$$

En écriture vectorielle on aura $\vec{P'} = \vec{P} + \vec{Trans}$ où $\vec{Trans}$ représente le vecteur de translation.
La translation inverse est obtenue simplement en changeant le signe des composants de *Trans*, ce qui nous donne $Trans^{-1}(-T_x,-T_y,-T_z)$.

La figure **3.2.a** représente la translation d'un triangle dans un plan, et la figure **3.2.b** la translation inverse.

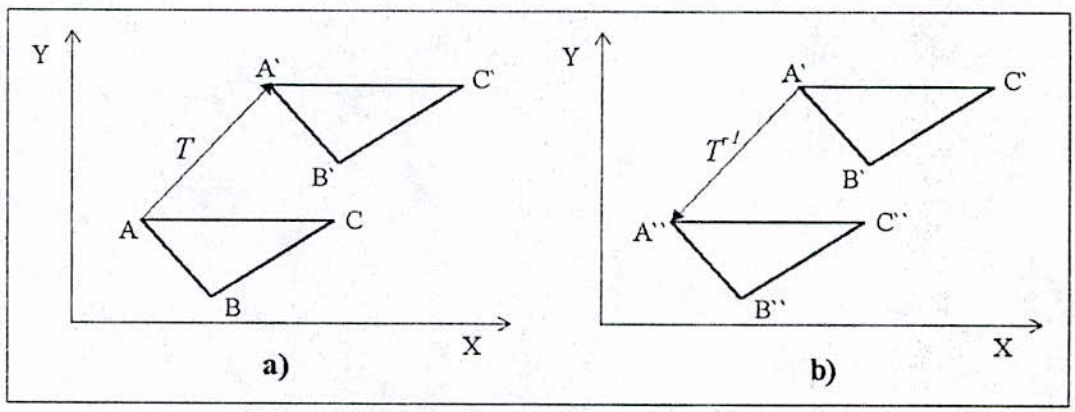


Fig. 3.2 : *Translation et translation inverse.*

3.2.2.2 Changement d'échelle

Pour un changement d'échelle il est nécessaire de préciser son centre et son facteur d'échelle. Il est possible d'affecter pour chaque axe son propre facteur d'échelle, mais en pratique et pour ne pas déformer les objets on choisit le même facteur pour les trois axes. Lorsque le facteur d'échelle est plus petit que un nous aurons une réduction, par contre lorsqu'il est supérieur à un c'est un agrandissement. La figure 3.3 montre le cas où le facteur d'échelle supérieur à 1.

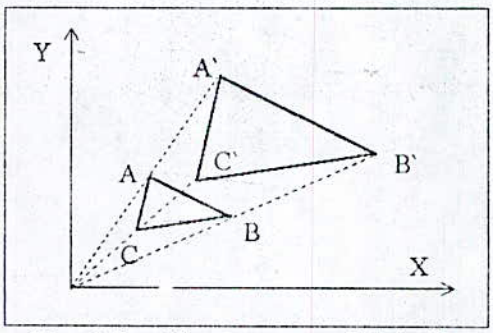


Fig. 3.3 : *Changement d'échelle.*

Ainsi si on applique un changement d'échelle dont le centre est à l'origine à un point $P(x,y,z)$, nous obtiendrons le point $P`(x`,y`,z`)$ avec:

$$\begin{cases} x' = S_x \cdot x \\ y' = S_y \cdot y \\ z' = S_z \cdot z \end{cases} \tag{3.3}$$

Ce qui en écriture matricielle nous donne:

$$\vec{P'} = S \cdot \vec{P} \quad \text{avec} \quad S = \begin{pmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{pmatrix} \tag{3.4}$$

3.2.2.3 Rotation

Contrairement au deux transformations précédentes, la rotation dépend de l'axe par rapport auquel elle

est faite. Ainsi nous pouvons distinguer trois rotations, chacune prise par rapport à un axe du repère comme le montre la figure 3.4.

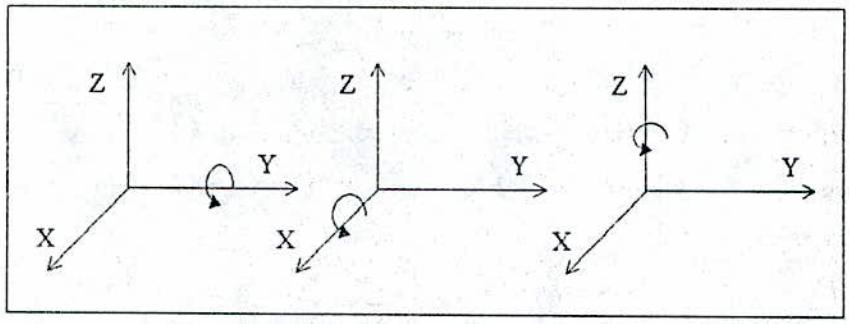


Fig. 3.4 : *Différentes rotations.*

On donne les matrices caractérisant ces rotations en prenant comme angle de rotation θ.

$$Rot_{\theta/X} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \tag{3.5}$$

$$Rot_{\theta/Y} = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \tag{3.6}$$

$$Rot_{\theta/Z} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.7}$$

Donc si on a un point $P(x,y,z)$ auquel on aurait appliqué une rotation d'angle θ suivant l'axe X, le résultat sera un point $P`(x`,y`,z`)$ tel que:

$$\begin{cases} x' = x \\ y' = y \cdot \cos\theta - z \cdot \sin\theta \\ z' = y \cdot \sin\theta + z \cdot \cos\theta \end{cases} \tag{3.8}$$

3.2.3 Les coordonnées homogènes

Nous avons remarqué dans ce qui précède que les différentes transformations ne donnaient pas lieu au même genre de calcul. Ainsi dans la translation on additionnait les coordonnées du point considéré avec les composantes du vecteur de translation; par contre pour les différentes rotations et pour la mise à l'échelle nous multiplions par la matrice de transformation ces mêmes coordonnées.
Or pour la manipulation des objets tridimensionnels, nous avons souvent besoin d'enchaîner ces différentes transformations.

Nous aurons ainsi besoin de translater l'objet puis de lui appliquer une rotation, ou une mise à l'échelle. Mais comme ces transformations ne font pas appel aux mêmes opérations, il va être difficile de les chaîner, et nous aurions aimé représenter toutes les transformations par des produits matriciels seulement, ainsi une série de transformations différentes donneraient lieu à une série de produits matriciels, d'où l'introduction des coordonnées homogènes.

Interprétation

Cela consiste à représenter un vecteur appartenant à un espace à n dimensions dans un espace à n+1 dimensions, en ajoutant une coordonnée supplémentaire w appelée *facteur d'échelle*.
Donc un point $P(x,y,z)$ pris dans l'espace cartésien sera exprimé en coordonnées homogènes par $P(X,Y,Z,w)$. Le passage des coordonnées homogènes aux coordonnées cartésiennes se fera en divisant ces dernières par le facteur d'échelle w.

$$\begin{cases} x = \dfrac{X}{w} \\ y = \dfrac{Y}{w} \\ z = \dfrac{Z}{w} \end{cases} \qquad (3.9)$$

3.2.4 Transformation et coordonnées homogènes

Les différentes transformations sont représentées par des matrices 4*4 en coordonnées homogènes. Elles sont données par [BRE 88]:

Translation de T_x, T_y, T_z suivant les axes X, Y, Z

$$Trans = \begin{pmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.10}$$

Facteur d'échelle *Sx*, *Sy*, *Sz* suivant les axes X, Y, Z

$$S = \begin{pmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.11}$$

Les différentes rotations d'angle θ suivant les trois axes sont données par:

L'axe X

$$Rot_{\theta/X} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.12}$$

L'axe Y

$$Rot_{\theta/Y} = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.13}$$

L'axe Z

$$Rot_{\theta/Z} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.14}$$

3.3 Représentation tridimensionnelle des objets

C'est la première étape du processus de réalisation d'images de synthèses tridimensionnelles. Plusieurs modèles sont proposés, selon l'application souhaitée et les moyens utilisés. Nous pouvons ainsi trouver des modélisations surfaciques, ou bien volumiques.

3.3.1 Modélisations surfaciques

3.3.1.1 Modèle polyédrique

Cette méthode suppose qu'on peut représenter n'importe quelle surface, aussi irrégulière soit elle par un ensemble fini de facettes.

Les sommets composant ces facettes sont choisis de façon à appartenir à l'enveloppe de l'objet à modéliser. Cela permet de former un maillage comme le montre la figure 3.5.

Il est clair que la fidélité de la représentation est directement liée au nombre de points considérés. Car plus on prend de points, plus on a de facette et plus l'approximation est meilleure.

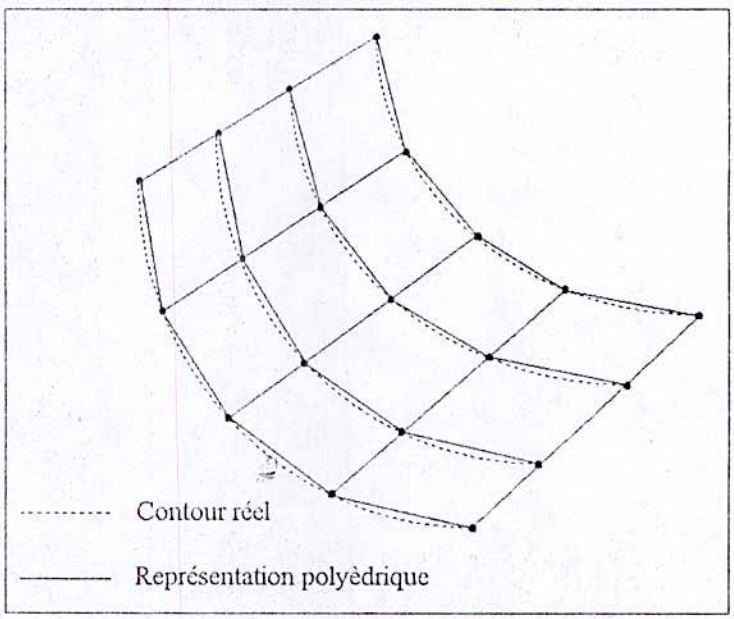


Fig. 3.5 : *Représentation polyédrique des surfaces.*

3.3.1.2 Modélisation par courbes mathématiques

Le modèle qu'on vient de voir est sans doute le plus utilisé, car le calcul des parties cachées est plus simple, comparés à d'autres modèles. Et pour la réalisation d'animation, la prise en compte des collisions est plus aisé.

Cependant ce modèle n'est pas très réaliste du point de vue des éclairages, car il donne un aspect facetté aux objets.

Pour résoudre ce problème, on peut avoir recours à un lissage (ce qu'on verra plus tard), ou bien adopter un autre modèle qui consiste à modéliser les surface par des courbes mathématiques, on utilise pour cela un certain nombre de points appartenant à l'objet, puis on fait une interpolation pour calculer l'équation de la courbe qui passe par ces points.

3.3.2 Modélisation volumique ou par "voxel"

Elle consiste à déterminer les parties de l'espace tridimensionnel occupé par l'objet, et de procéder ensuite à un remplissage de ces parties en utilisant des cubes élémentaires de taille identiques (voxel).

Elle est utilisée en imagerie médicale lorsque les images sont obtenues par des techniques d'acquisition de tranches planes [GOU 94] [BAS 91].

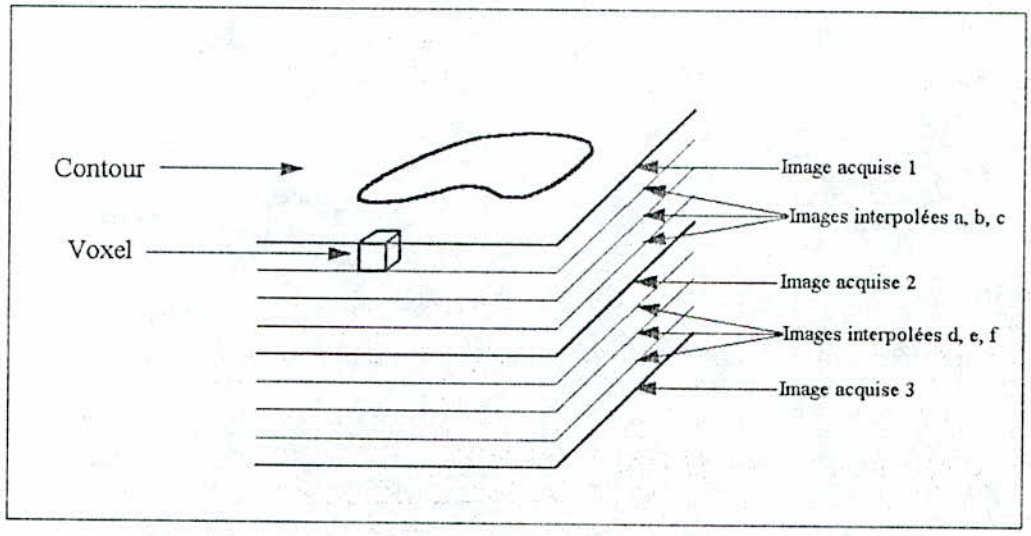


Fig. 3.6 : *Représentation par voxels.*

Ainsi, une fois récupérées, les coupes sont traitées (détection de contours), puis ces contours sont remplis avec les voxels. Pour avoir une bonne définition de l'objet ainsi reconstitué il faut que la distance entre deux coupes soit de l'ordre du voxel. Mais on peut procéder à une interpolation entre les coupes d'acquisition. On fait ainsi l'acquisition des images **1, 2, 3**, puis on interpole les images **a, b, c, d, e, f,** comme le montre la figure 3.6.

3.4 Les transformations de projection

3.4.1 Introduction

Pour pouvoir visualiser des objets tridimensionnels sur des supports bidimensionnels par exemple sur un écran, il est nécessaire de leur faire subir une transformation. Cette opération est réalisée par la projection. Elle nous permet donc de passer d'un espace à trois dimensions vers un autre à deux dimensions, qui est appelé plan de projection.
Un point P est projeté sur un point P` en multipliant ses coordonnées par la matrice de projection.
Nous pouvons distinguer deux grands types de projection:

- Les projections parallèles;
- La projection perspective.

Les projections parallèles regroupent les projections axonométriques et les projections obliques [GOU 94].
La projection perspective peut être à un, deux ou trois points de fuites.

Nous ne parlerons pas des projections parallèles car elles ne correspondent pas à la manière dont nous percevons les choses, par contre nous nous intéresserons plus à la projection perspective.

3.4.2 La projection perspective [SCH 87b]

3.4.2.1 La projection perspective à un point de fuites

C'est un modèle de projection qui reproduit le mécanisme de vision chez l'être humain.
Car si nous observons la figure 3.7, qui représente une projection parallèle, on voit que bien que l'objet #2 soit plus éloigné du plan de projection que l'objet #1, leur projection parallèle sur le plan XY est la même.

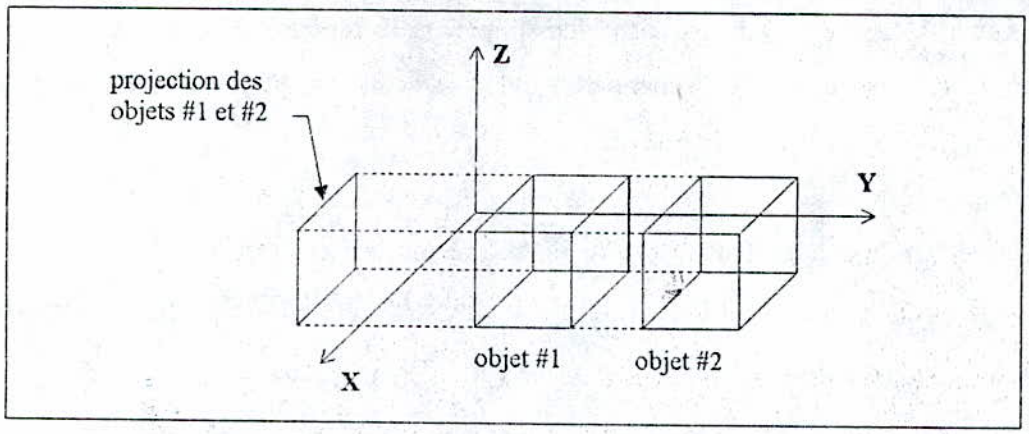


Fig. 3.7 : *Inconvénient de la projection parallèle.*

Ainsi la notion de distance par rapport au plan de projection n'existe pas. En projection perspective, on considère que tous les rayons de lumière convergent vers un centre de projection comme pour l'oeil humain. Ceci fait qu'un objet plus proche du centre de projection parait plus grand qu'un autre plus éloigné, ce qui fait ressortir la sensation de profondeur.

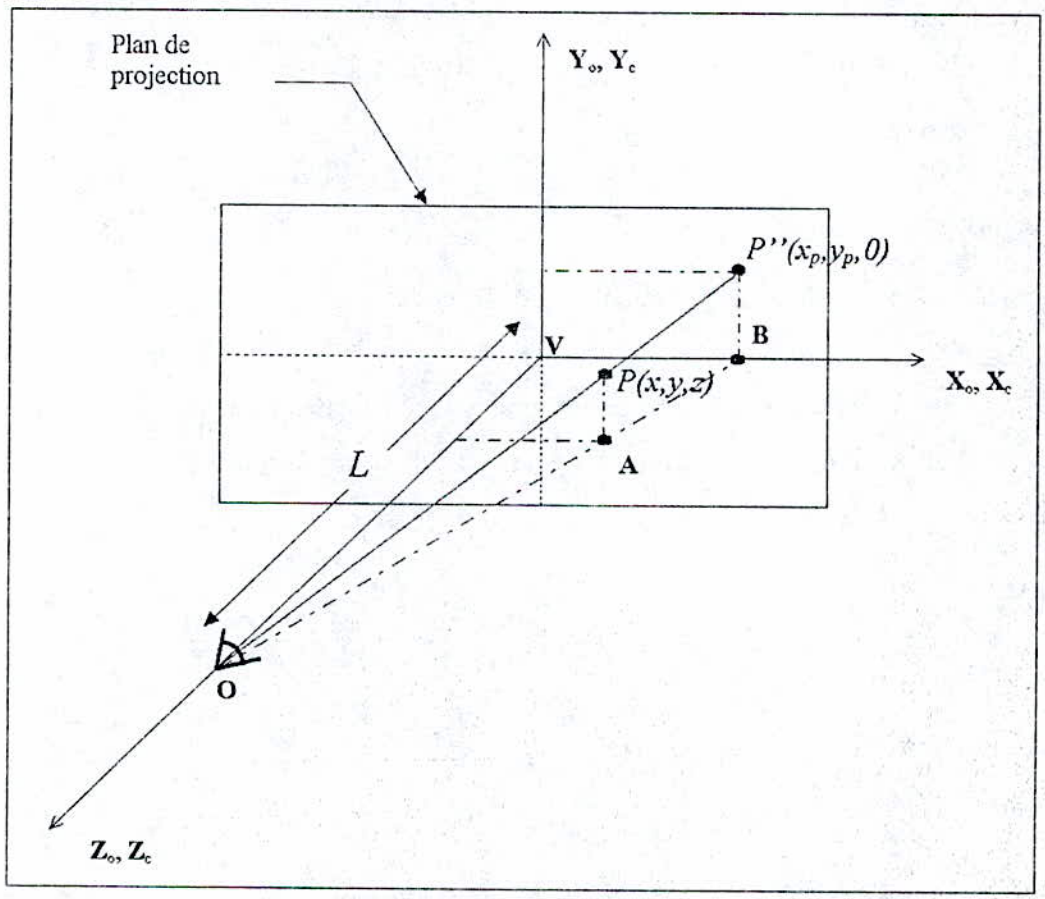


Fig. 3.8 : *Projection perspective à un point de fuite.*

Nous allons tout d'abord traiter le cas le plus simple où le repère de l'objet $\Re_o$ et le repère de l'oeil humain ou de la camera $\Re_c$ sont superposés, et où le centre de projection se trouve sur l'axe Z_c du repère $\Re_c$, le plan de projection étant le plan X_cY_c.

Soit un point $P(x,y,z)$ appartenant au repère $\Re_o$, sa projection perspective sur le plan X_cY_c et ayant pour centre de projection le point **O** nous donne le point $P''(x_p,y_p,0)$, comme le montre la figure **3.8.**
Si nous observons bien les deux triangles **OAP** et **OBP''** nous remarquons que:

$$\frac{x_p}{x}=\frac{y_p}{y}=\frac{L}{L-z} \quad \Rightarrow \quad x_p=\frac{x}{L-z}=\frac{x}{1-z/L},\ y_p=\frac{y}{1-z/L}, \tag{3.15}$$

L'équation (3.15) décrit la transformation perspective, qui peut être écrite matriciellement et en coordonnées homogènes de la manière suivante:

$$T_{pers}=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & -\frac{1}{L} & 1\end{pmatrix} \tag{3.16}$$

L représente la distance entre le centre de projection O et le point vise V.

Mais cette matrice correspond à une transformation perspective et non à une projection perspective. Pour avoir la projection il faut encore multiplier le résultat par la matrice de projection orthographique d'axe Z donnée par [SCH 87b]:

$$Orth_{/z}=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1\end{pmatrix} \tag{3.17}$$

Ainsi la projection perspective sera égale à:

$$\boxed{P_{pers} = Orth_{/z}.T_{pers}} \tag{3.18}$$

L'écriture matricielle nous donne:

$$P_{pers} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{L} & 1 \end{pmatrix} \tag{3.19}$$

Nous pouvons donc vérifier que si nous appliquons cette matrice à un point $P(x,y,z,1)$, on obtient un autre point $P'(x',y',z',w')$ tel que:

$$P' = P_{pers}.P \Rightarrow \begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{L} & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = \begin{pmatrix} x \\ y \\ 0 \\ 1-\frac{z}{L} \end{pmatrix} \tag{3.20}$$

Pour retrouver les coordonnées réelles, il faut diviser le résultat par w' :

$$\boxed{\begin{aligned} x_p &= \frac{x'}{w'} = \frac{x}{1-\frac{z}{L}} \\ y_p &= \frac{y'}{w'} = \frac{y}{1-\frac{z}{L}} \end{aligned}} \tag{3.21}$$

Cette projection est dite à un point de fuite principal. Il se trouve sur l'axe Z. Un point de fuite associé à un axe étant le point où convergent la projection des droites parallèles à cet axe.

Si nous prenons un cube dans le repère $\Re_o$ (figure 3.9.a), et si nous faisons sa projection perspective sur le plan X_cY_c, nous remarquons que les droites parallèles à la direction OV convergent vers le point de fuite comme le montre la figure 3.9.b.

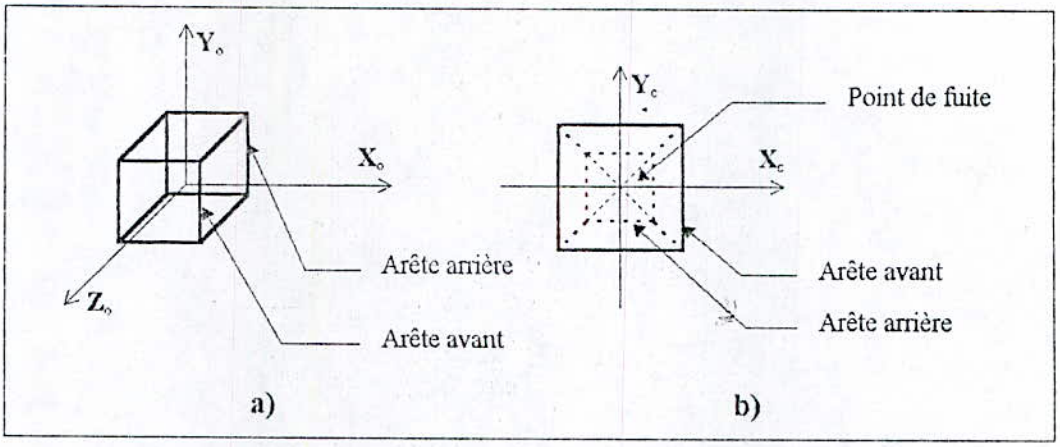


Fig. 3.9 : *Point de fuite dans une projection perspective.*

3.4.2.2 Projection perspective générale

Dans la réalité nous pouvons avoir des projections à deux points de fuite (figure **3.10.a**) ou même à trois points de fuites principaux (figure **3.10.b**). Il faut cependant souligner le fait que ce ne sont pas des types distincts de projections perspectives mais reflètent seulement le fait que la direction de projection soit parallèle à deux faces (un point de fuite), à une seule face (deux point de fuites), ou à aucune face principale de l'objet (trois points de fuites).

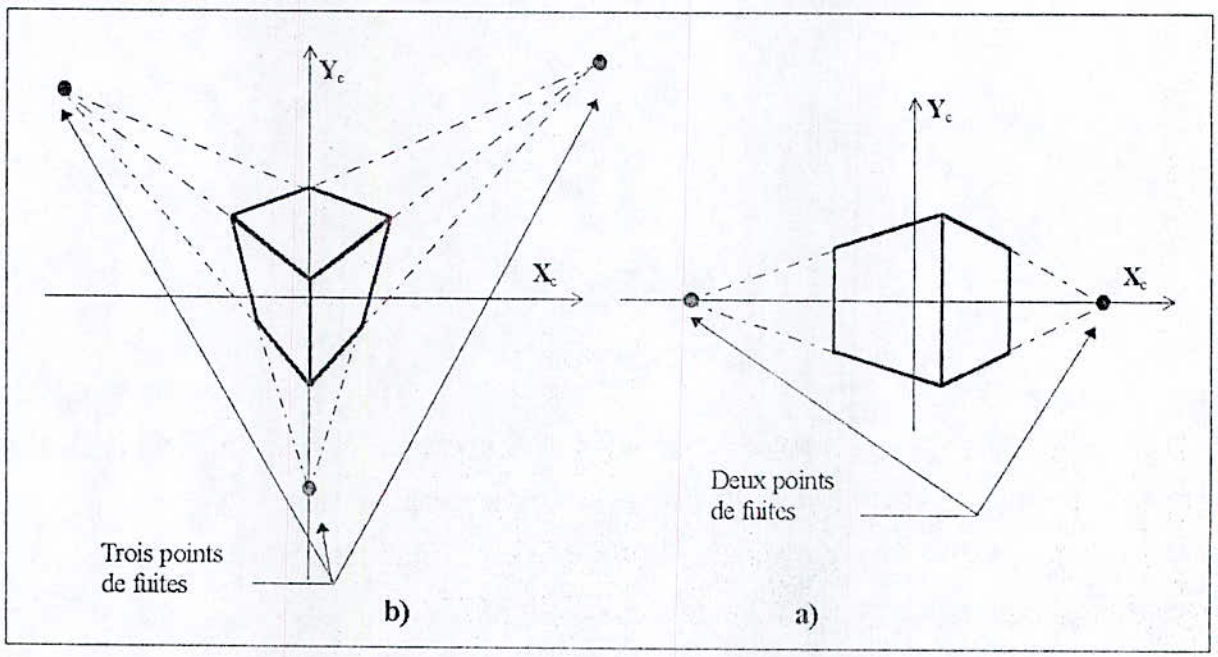


Fig. 3.10 : *Projection à deux et trois points de fuites.*

Dans ce qui suit, nous allons étudier le cas général où les deux repères $\Re_o$ et $\Re_c$ ne sont plus superposés.

Les points **O** et V sont définis par leurs coordonnées $O(x_o, y_o, z_o)$ et $V(x_v, y_v, z_v)$ dans le repère $\Re_o$ (figure **3.11**) (i.e. observateur positionné en (x_o, y_o, z_o) et regarde vers (x_v, y_v, z_v)).

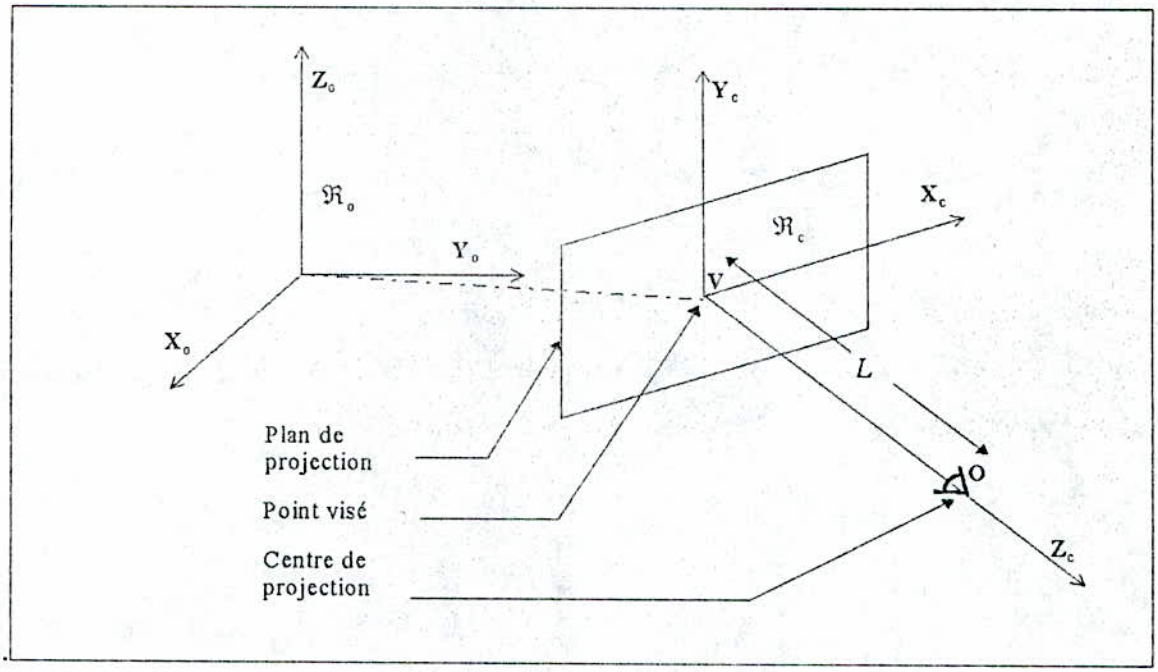


Fig. 3.11 : *Positionnement des repères.*

Comme on connaît les coordonnées de l'objet dans le repère $\Re_o$, il faut trouver la transformation liant les deux repères $\Re_c$ et $\Re_o$ afin de calculer les coordonnées de cet objet dans le repère $\Re_c$. Ceci revient à trouver la matrice T telle que $\Re_o = T\,\Re_c$. Cependant, il est plus facile de calculer la matrice T^{-1}. Pour cela on ramène tout d'abord le repère $\Re_c$ par translation $Trans(-x_v, -y_v, -z_v)$ pour le positionner à l'origine du repère $\Re_o$.

La distance entre le point **O** (oeil) et le point V (le point visé) est donné par:

$$L = \sqrt{(x_v - x_o)^2 + (y_v - y_o)^2 + (z_v - z_o)^2} \qquad (3.22)$$

Après avoir ramené le repère $\Re_c$ vers l'origine du repère $\Re_o$, le point V aura pour coordonnées V(0, 0, 0) et le point O aura pour coordonnées O (x_L, y_L, z_L) telles que:

$$\begin{cases} x_L = x_o - x_v \\ y_L = y_o - y_v \\ z_L = z_o - z_v \end{cases} \tag{3.23}$$

La projection de la distance OV qui est égale à L, sur le plan X_oY_o donnera:

$$L_{proj} = \sqrt{x_L^2 + y_L^2} = \sqrt{(x_o - x_v)^2 + (y_o - y_v)^2} \tag{3.24}$$

Il faut souligner que nous prenons le cas où les axes Z_c et Y_c se trouvent dans un plan perpendiculaire par rapport au plan X_oY_o. De ce fait, l'axe X_c sera toujours parallèle au plan X_oY_o. Nous considérons en plus que l'axe Y_c est dirigé vers le haut, comme on peut le voir sur la figure **3.13.a.**

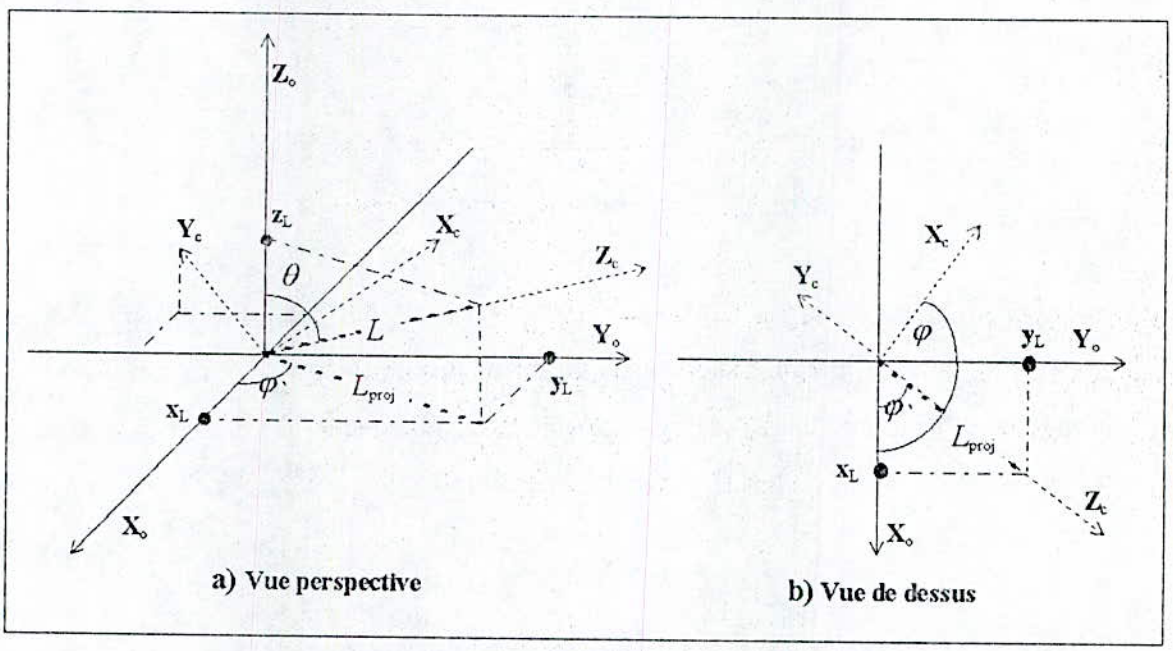


Fig. 3.12 : ***Orientation relative des repères.***

Le repère $\Re_c$ qu'on a translaté peut être considéré comme résultant de la rotation du repère $\Re_o$ d'un angle θ suivant l'axe X_o suivie d'une rotation d'un angle φ suivant l'axe Z_o.

D'après les figures **3.13.a** et **3.13.b** on a:

$$\sin\theta = \frac{L_{proj}}{L}, \quad \cos\theta = \frac{z_L}{L}, \quad \text{et} \quad \sin\varphi' = \frac{y_L}{L_{proj}}, \quad \cos\varphi' = \frac{x_L}{L_{proj}} \tag{3.25}$$

Comme on a: $\varphi = \pi/2 + \varphi'$ alors $\begin{cases} \sin\varphi = \cos\varphi' \\ \cos\varphi = -\sin\varphi' \end{cases} \Rightarrow \begin{cases} \cos\varphi' = \sin\varphi \\ \sin\varphi' = -\cos\varphi \end{cases}$ (3.26)

Ainsi on déduit

$$\sin\theta = \frac{L_{proj}}{L} \qquad \cos\theta = \frac{z_L}{L}$$
$$\sin\varphi = \frac{x_L}{L_{proj}} \qquad \cos\varphi = \frac{-y_L}{L_{proj}} \tag{3.27}$$

Sachant que les différentes rotations et translation sont données par [SCH 87b]:

$$Rot_{\theta/x_o} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.28}$$

$$Rot_{\varphi/z_o} = \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 & 0 \\ \sin\varphi & \cos\varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.29}$$

$$Trans(x_v, y_v, z_v) = \begin{pmatrix} 1 & 0 & 0 & x_v \\ 0 & 1 & 0 & y_v \\ 0 & 0 & 1 & z_v \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.30}$$

On aura: $$T^{-1} = Trans(x_v, y_v, z_v).Rot_{\varphi/z_o}.Rot_{\theta/x_o} \tag{3.31}$$

Cependant, il ne faut pas oublier que nous cherchons la transformation T qui est égale à:

$$T = Rot^{-1}{}_{\theta/x_o}.Rot^{-1}{}_{\varphi/z_o}.Trans^{-1}(x_v, y_v, z_v) \tag{3.32}$$

Comme: $Rot^{-1}_{\theta/x_o} = Rot_{-\theta/x_o}$, $Rot^{-1}_{\varphi/z_o} = Rot_{-\varphi/z_o}$

et $Trans^{-1}(x_v, y_v, z_v) = Trans(-x_v, -y_v, z_v)$ (3.33)

On déduit que:

$$T = \begin{pmatrix} \cos\varphi & \sin\varphi & 0 & -(x_v\cos\varphi + y_v\sin\varphi) \\ -\cos\theta\sin\varphi & \cos\varphi\cos\varphi & \sin\theta & -(-x_v\cos\theta\sin\varphi + y_v\cos\theta\cos\varphi + z_v\sin\theta) \\ \sin\theta\sin\varphi & -\sin\theta\cos\varphi & \cos\varphi & -(x_v\sin\theta\sin\varphi - y_v\sin\theta\cos\varphi + z_v\cos\theta) \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.34)$$

Pour ne pas alourdir inutilement les formules on pose:

$$\begin{cases} P_x = -(x_v\cos\varphi + y_v\sin\varphi) \\ P_y = -(-x_v\cos\theta\sin\varphi + y_v\cos\theta\cos\varphi + z_v\sin\theta) \\ P_z = -(x_v\sin\theta\sin\varphi - y_v\sin\theta\cos\varphi + z_v\cos\theta) \end{cases} \quad (3.35)$$

Cette matrice nous donne donc les coordonnées de l'objet dans le repère $\Re_c$ de l'observateur connaissant les coordonnées de ce dernier dans le repère $\Re_o$.

Ainsi la matrice de la transformation perspective dans le cas général sera:

$$T_{pers/général} = T_{pers}.T \quad (3.36)$$

Avec:

$$T_{pers/general} = \begin{pmatrix} \cos\varphi & \sin\varphi & 0 & P_x \\ -\cos\theta\sin\varphi & \cos\theta\cos\varphi & \sin\theta & P_y \\ \sin\theta\sin\varphi & -\sin\theta\cos\varphi & \cos\theta & P_z \\ \dfrac{-\sin\theta\sin\varphi}{L} & \dfrac{\sin\theta\cos\varphi}{L} & \dfrac{-\cos\varphi}{L} & 1-\dfrac{P_z}{L} \end{pmatrix} \quad (3.37)$$

De même que pour le cas d'une projection à un point de fuite, il faut multiplier le résultat par la matrice $Orth_{/z}$ donnée plus haut pour avoir la projection perspective.

$$\boxed{P_{pers} = Orth_{/z}.T_{pers}.T} \tag{3.38}$$

La projection perspective d'un point $P(x,y,z)$ sur le plan X_cY_c, donnera le point $P''(x_p,y_p,0)$, tel que:

$$\boxed{\begin{aligned} x_p &= \frac{x\cos\varphi + y\sin\varphi + P_x}{w} \\ y_p &= \frac{-x\cos\theta\sin\varphi + y\cos\theta\cos\varphi + z\sin\theta + P_y}{w} \\ w &= -\frac{x\sin\theta\sin\varphi}{L} + \frac{y\sin\theta\cos\varphi}{L} - \frac{z\cos\theta}{L} + 1 - \frac{P_z}{L} \end{aligned}} \tag{3.39}$$

3.5 Elimination des parties cachées

3.5.1 Introduction

La suppression des parties cachées d'un objet dans une scène a reçu un grand nombre de solutions qui ont évoluées et évoluent encore pour s'adapter au matériel disponible.
Ces techniques étaient plutôt orientées vers la suppression des arêtes cachées lorsque les écrans étaient du type oscilloscope, et qu'ils ne permettaient que le tracer de ligne.
Puis le passage aux écrans à balayage à fait basculer le problème, et on s'est plus intéressé à la suppression des faces cachées, car ce genre d'écrans autorisait le remplissage.

Comme la génération d'images de synthèses réalistes ne peut se faire sans donner aux objets leur aspect de volume plein, l'élimination des parties cachées devient nécessaire.

La représentation par des squelettes nous informe beaucoup plus sur la forme globale de ces objets, mais ces représentations sont loin de se rapprocher de la réalité, et donne même parfois lieu à une ambiguïté sur la position de l'objet. Ainsi un simple cube dessiné avec toutes ses arêtes peut être perçu de plusieurs manières, comme le montre la figure 3.13.

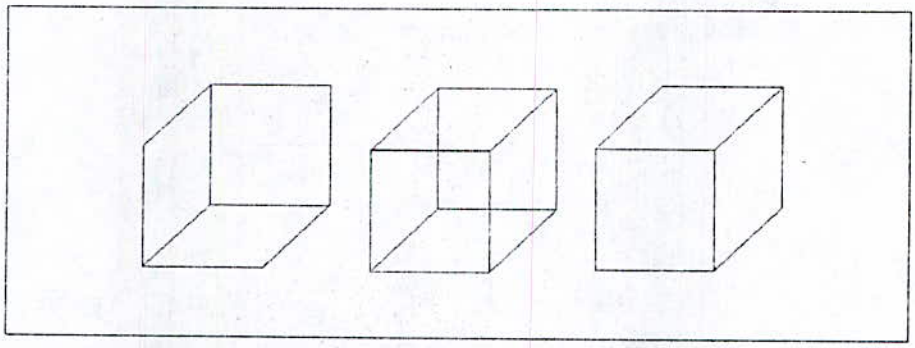


Fig. 3.13 : *Différentes interprétations du modèle filaire.*

Et une manière de rendre cette perception encore plus réaliste, c'est d'éliminer les faces cachées d'un objet dans le cas où celui-ci est opaque, et être ainsi en accord avec la réalité.

Il faut remarquer que les faces d'un objet peuvent être cachées soit par l'objet lui même, soit par un autre objet plus proche de l'observateur. Et selon le cas, les algorithmes d'élimination des parties cachées sont différents.

Cependant, il faut remarquer que l'élimination des parties cachées peut conduire à minimiser le nombre d'informations transmises vers l'observateur, de sort qu'il est nécessaire de changer d'angle de vue pour voir les parties initialement cachées. Vu que les parties visibles et cachées dépendent de l'angle de vue et donc de la position de l'observateur, l'élimination de ces dernières, doit se faire après avoir ramené l'objet vers le repère de l'observateur.

Nous allons nous intéresser dans ce qui suit à quelques algorithmes de suppression des faces cachées.

3.5.2 Modélisation des volumes à facettes

Un volume peut être représenté par:

- ses sommets;
- ses arêtes;
- ses faces.

Toutefois, une simple liste de sommets n'est pas en mesure de représenter correctement l'objet, car ne sachant pas comment ces sommets sont reliés entre eux.

Aussi, les côtés reliant deux sommets permettent de façonner un squelette filiforme de l'objet considéré mais ne donnent aucune information sur les faces, et ne peut donc être utilisée pour l'élimination des parties cachées.

Nous pouvons directement définir les faces par une liste de sommets les composant, en veillant à fermer la boucle et en parcourant les sommets de chaque facette dans le même sens préalablement choisi comme sur la figure **3.14.**

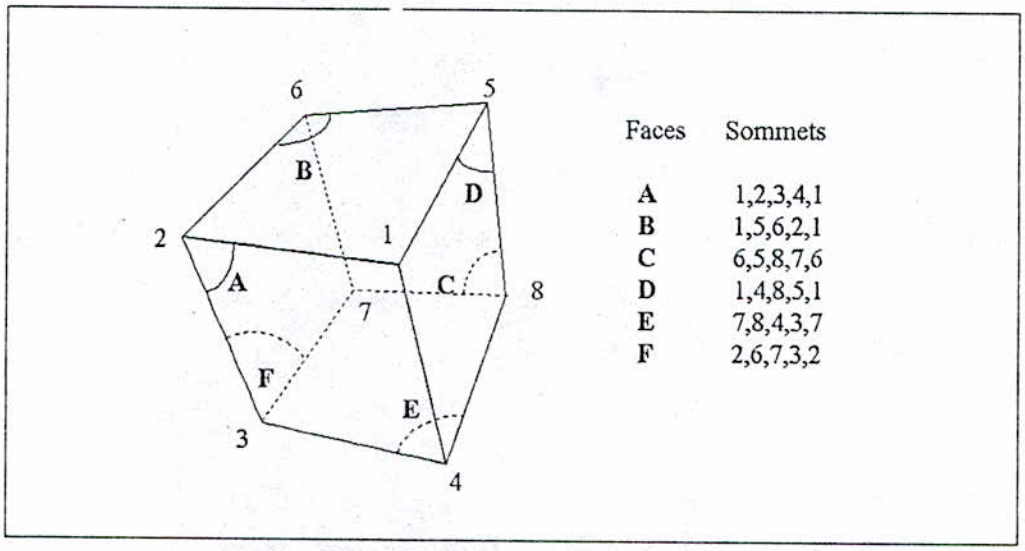


Fig. 3.14 : ***Représentation des objets par des facettes.***

En définissant ainsi les faces par leurs sommets et non par leurs coordonnées on économise de l'espace mémoire car les sommets sont des entiers et les coordonnées des réels.

De cette manière on calcule les coordonnées de chaque sommet qu'une seule fois.

Cette représentation est d'autant plus intéressante qu'elle permet par exemple de calculer la normale de chaque face pour effectuer un test de visibilité.

3.5.3 Elimination des faces arrières

Les algorithmes d'élimination des parties cachées sont réputés pour leur besoin en temps de calcul. Ce besoin s'accroît avec la complexité de la scène. Donc s'il était possible dès le départ de supprimer centaines faces du traitement cela ne ferait que diminuer le temps nécessaire à cette opération.

Pour cela on commence d'abord par éliminer les faces arrières, ce qui se fait en comparant leur vecteur normal avec la direction de vision.

3.5.3.1 Test de visibilité

C'est la méthode la plus simple pour l'élimination des parties cachées. Elle ne nécessite pas de grands temps de calcul, c'est pour cela d'ailleurs qu'on l'utilise en premier pour éliminer les faces arrières.
Pour savoir si une face est visible ou non, il suffit de calculer le produit scalaire entre la normale de cette face et le vecteur reliant l'observateur à un point de la face considérée. En général on prend un des sommets de cette face.

Nous utiliserons le fait que nous représentons les faces par des sommets pour calculer les vecteurs normaux.

Comme le montre la figure 3.15, le parcours des sommets d'une face se fait dans le sens trigonométrique et donc le produit vectoriel de deux arêtes consécutives prises dans ce sens nous donne le vecteur normal, qui est sortant [SCH 87b].

Autrement dit:

$$\vec{N} = \vec{s_2 s_1} \wedge \vec{s_3 s_2} \tag{3.40}$$

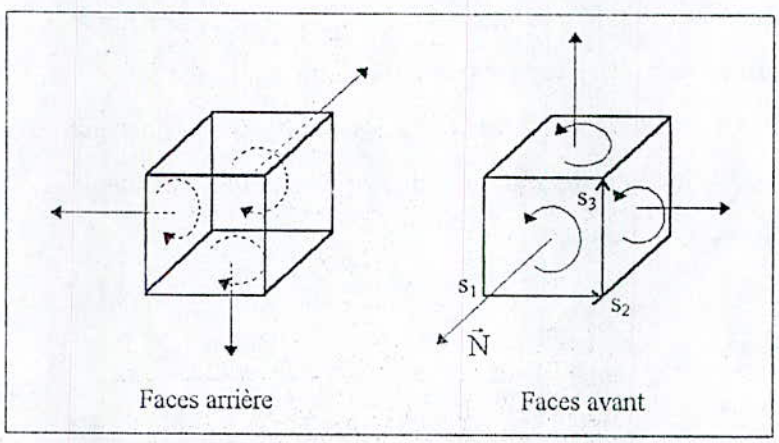


Fig. 3.15 : *Orientation des normales.*

Pour la face considérée dans la figure 3.15, si $s_1(x_1,y_1,z_1)$, $s_2(x_2,y_2,z_2)$, $s_3(x_3,y_3,z_3)$ représentent les coordonnées des différents sommets par rapport au repère de l'observateur, le vecteur normal s'écrira:

$$\vec{N}=\begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix}=\vec{s_2s_1}\wedge\vec{s_3s_2}=\begin{pmatrix} x_2-x_1 \\ y_2-y_1 \\ z_2-z_1 \end{pmatrix}\wedge\begin{pmatrix} x_3-x_2 \\ y_3-y_2 \\ z_3-z_2 \end{pmatrix} \tag{3.41}$$

$$\Rightarrow\begin{cases} n_x=(z_3-z_2)(y_2-y_1)-(y_3-y_2)(z_2-z_1) \\ n_y=(x_3-x_2)(z_2-z_1)-(z_3-z_2)(x_2-x_1) \\ n_z=(y_3-y_2)(x_2-x_1)-(x_3-x_2)(y_2-y_1) \end{cases} \tag{3.42}$$

Le vecteur parallèle à la direction de vision a pour composantes $\vec{S_1O}=\begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix}$ (On prend le sommet S_1 comme point appartenant à la face) avec :

$$\begin{cases} V_x=x_o-x_1 \\ V_y=y_o-y_1 \\ V_z=z_o-z_1 \end{cases} \tag{3.43}$$

Le produit scalaire entre cette normale et le vecteur $\vec{S_1O}$ est donné par:

$$\vec{S_1O}\cdot\vec{N}=\begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix}\cdot\begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix}=V_x\cdot n_x+V_y\cdot n_y+V_z\cdot n_z \tag{3.44}$$

Mais ce produit scalaire peut aussi s'écrire sous la forme $\vec{S_1O}\cdot\vec{N}=\left|\vec{S_1O}\right|\cdot\left|\vec{N}\right|\cdot\cos\theta$ où θ représente l'angle enter ces deux vecteurs, comme le montre la figure 3.16.

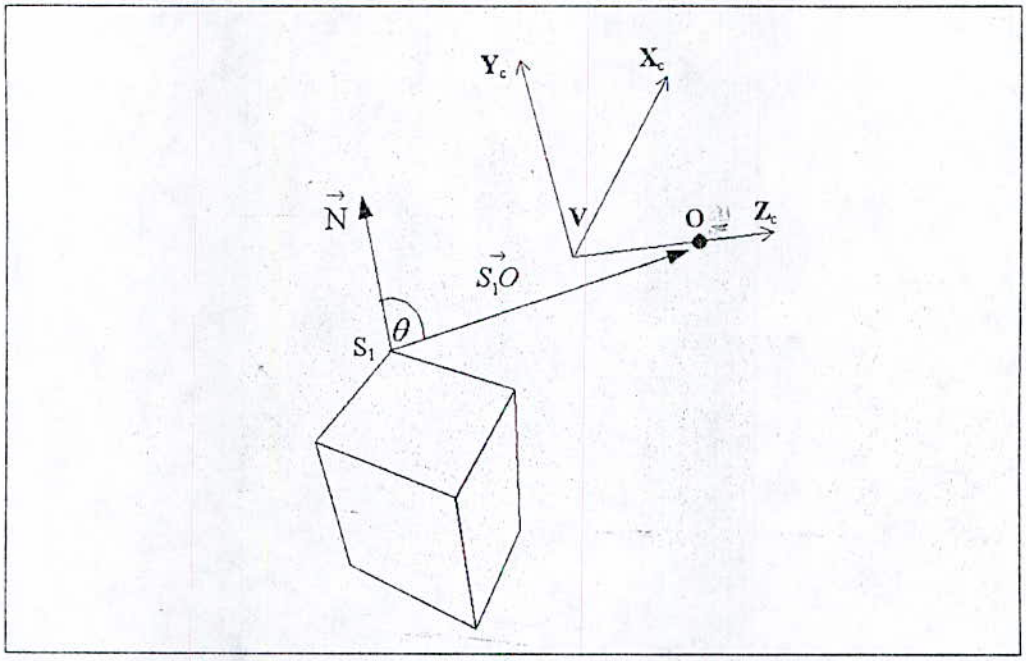


Fig. 3.16 : ***Position relative de la normale par rapport à la direction de vision.***

De la figure 3.16 on déduit que si la face est visible alors le produit scalaire est positif car l'angle θ est inférieur à $\pi/2$ ($cos\theta > 0$). Et quand la face est cachée le produit scalaire est négatif car l'angle θ est supérieur à $\pi/2$ ($cos\theta < 0$).

On conclue que pour savoir si une facette est visible, il suffit de calculer le signe du produit scalaire de sa normale avec la direction de vision.

3.5.3.2 Inconvénients du test de visibilité

Cette méthode est idéale pour résoudre le problème des surfaces cachées pour des objets convexes mais elle n'est plus suffisante pour des objets concaves ou pour des scènes comportant plusieurs objets.
On dit d'un objet qu'il est convexe lorsqu'on peut se déplacer d'un sommet vers un autre de l'objet sans en sortir.
Les figures 3.17.a, 3.17.b, 3.17.c, montrent trois cas qu'on peut rencontrer et démontre les limites de cette méthode.

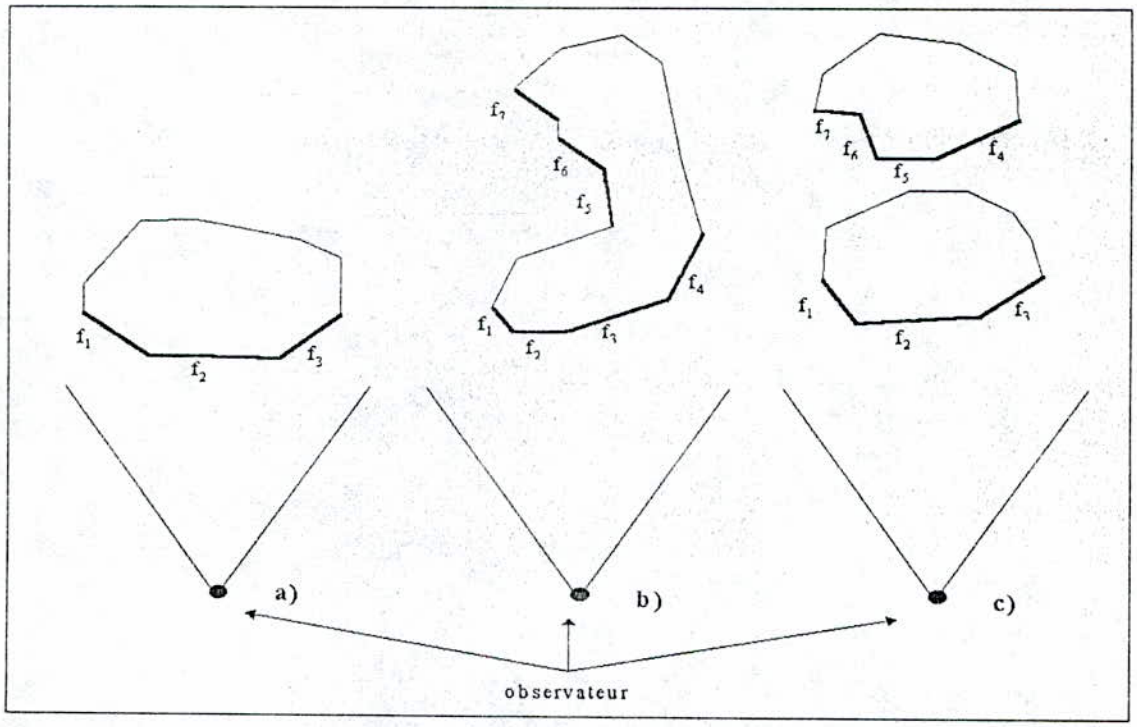


Fig. 3.17 : ***Inconvénients du test de visibilité.***

a) L'objet est convexe et ne pose donc aucun problème, car les faces f_1, f_2, f_3 sont réellement visibles.

b) L'objet est concave et toutes les faces supposées visibles ne le sont pas réellement, car bien que les faces f_5, f_6, f_7 soient effectivement dirigées vers l'observateur elles sont cachées par les faces avant f_1, f_2, f_3, f_4 et ne sont donc pas visibles.

c) Ici, nous sommes en présence de deux objets; de ce fait les faces f_1, f_2, f_3 de l'objet en avant cachent les faces f_4, f_5, f_6, f_7, même si elles sont orientées vers l'observateur.

3.5.4 Méthode du tri selon l'éloignement

Cette méthode se base sur tri des facettes selon l'éloignement et porte aussi le nom d'algorithme du peintre [SCH 87b], car elle découle de la manière qu'utilisent les peintres pour réaliser leurs peintures. Ils commencent donc par peindre l'arrière plan puis mettent en place les divers objets par plans successifs, en superposant les couches de peinture. On peut donc affecter aux facettes un numéro d'apparition à l'affichage en s'arrangeant à ce que les faces les plus éloignées apparaissent en premier et les plus proches en dernier recouvrant ainsi les autres.

Le problème de cette méthode c'est le choix du critère de tri. Que choisir? La coordonnée z maximale de chaque facette, ou sa coordonnée minimale.

Si par exemple nous optons pour la coordonnée z maximale il y a un risque d'ambiguïté.

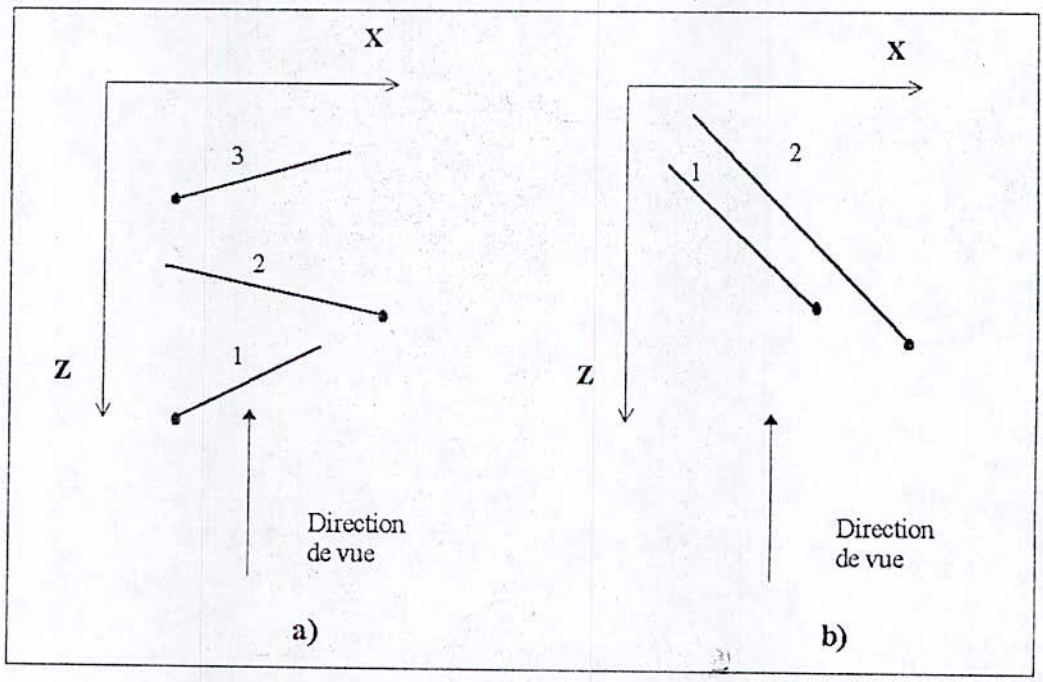


Fig. 3.18 : *Inconvénients de la méthode du tri.*

Dans la figure 3.18.a le tri se fait sans aucun problème, mais dans la figure 3.18.b nous remarquons que même si z_{max} de la facette 2 est supérieur à z_{max} de la facette 1, cette dernière cache partiellement la première, il y a donc une ambiguïté.

On suppose que les N facettes sont rangées dans un ordre quelconque dans un tableau t(i) i=1...N. L'algorithme consiste à comparer les profondeurs des facettes et à éventuellement les inverser dans le tableau t(i).

L'algorithme décrivant ce test se présente ainsi:

```
pour la facette courante t(j), j=1 à N-1
        pour toutes les facettes en cours de test (i) telles que j<i<N
        si z de t(i)> z de t(j)
        inverser les facettes t(i) et t(j)
    fin
fin
```

3.5.5 La méthode du *Z-buffer* [GOU 94]

Son principe est de colorier le pixel avec la couleur de la facette située le plus prés de l'oeil, en mettant l'observateur à l'infini, et en travaillant dans l'espace image au lieu de l'espace objet (cf. figure **3.19**).

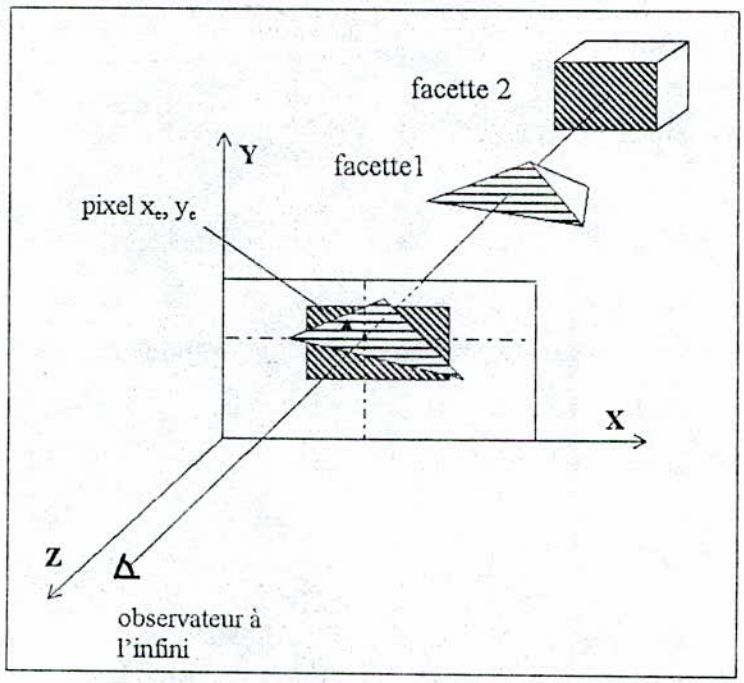


Fig. 3.19 : *Méthode du Z-buffer.*

Un écran matriciel rafraîchi comporte une mémoire où sont stockées les couleurs relatives à chaque pixel. En ajoutant une mémoire supplémentaire, appelée tampon de profondeur "*Z-buffer*" qui associe à chaque pixel sa coordonnée z on peut imaginer une méthode simple d'élimination des parties cachées.

Pour chaque facette et après une conversion ponctuelle (passage aux pixels), on calcule la coordonnée z pour chaque point à tracer. Si elle est plus grande que celle du point précédemment affiché à la même position le pixel peut être tracé et sa cordonnée z mise à jour.

Nous pouvons donc écrire l'algorithme suivant:

initialiser la couleur (x_e,y_e) à la couleur du fond pour tous les pixels
initialiser *Z-buffer*(x_e, y_e) à une valeur d'éloignement maximal
  pour chaque facette
  calculer sa projection

```
pour chaque pixel(xe,ye) situé dans la facette
calculer Ze en fonction des profondeurs aux sommets de la facette
    si Ze >Z-buffer(xe,ye) alors
    Z-buffer(xe,ye)=Ze
    couleur(xe,ye)=couleur de la facette
    fin
fin
```

3.5.5.1 Intérêt et inconvénient

Sa simplicité lui permet de traiter des scènes complexes assez rapidement, mais cette rapidité est obtenue au détriment de l'encombrement de l'espace mémoire, car il ne faut pas oublier que pour avoir une bonne précision il faut au moins 20 bits pour représenter la profondeur. Ainsi pour une image ayant une résolution de 800*600 , il faudra réserver une espace mémoire de 1,2 Mo, en plus de l'espace nécessaire pour stocker l'image. Mais ce problème tend à disparaître, vu qu'une telle taille mémoire est dérisoire de nos jours.

3.6 Modèles d'éclairage [GOU 94][BRE 88]

3.6.1 Introduction

L'obtention d'images de synthèses réalistes dont la qualité est proche des images photographiques, est difficilement réalisable sans prendre en compte les effets d'ombres et de lumière, et leur incidence sur les objets composants une scène.

Pour un meilleur rendu, il faut tenir compte des caractéristiques physiques de ces objets (forme, transparence, texture, indice de réfraction...); afin de calculer leur contribution à l'illumination de la scène (réflexion, réfraction...).

Cependant, s'il fallait prendre en compte les lois exactes de la propagation de la lumière on ne s'en sortirait pas à cause des temps de calcul excessifs que cela demanderait.

3.6.2 Modèles d'éclairement

Pour éviter des temps de calcul trop excessifs des modèles d'éclairement plus ou moins complexes ont été mis en oeuvre, parmi eux un modèle qui suggère que la lumière réfléchie par un point appartenant à une facette est constituée de trois composantes:

$$I=I_a+I_d+I_s \tag{3.45}$$

Où I_a est dû à la lumière ambiante, I_d à la réflexion diffuse et I_s à la réflexion spéculaire.

3.6.2.1 La lumière ambiante

On suppose que la scène est plongée dans une lumière ambiante provenant de tous les côtés avec la même intensité I_A. Si un objet possède un coefficient de réflexion de la lumière ambiante K_a alors l'intensité que réfléchi chaque point est :

$$I_a=K_a.I_A \text{ avec } 0<K_a<1 \tag{3.46}$$

La quantité de lumière perçue par l'observateur ne dépend pas de sa position.

3.6.2.2 La réflexion diffuse

Elle caractérise les objets mat (non brillants) lesquels absorbent la lumière puis la renvoient dans tous les sens (diffusion). L'intensité réfléchie dépend de l'angle d'incidence du faisceau lumineux. Ainsi plus l'angle formé entre la normale de la surface $\vec{N}$ et le rayon incident $\vec{L}$ (cf. figure **3.20**) est petit plus l'intensité réfléchie sera plus grande.

On déduit que:

$$I_d=K_d.I_p.\ \vec{N}.\vec{L} = I_p.K_d.\cos\theta \quad \text{si } |\vec{L}|=|\vec{N}|=1 \tag{3.47}$$

Là aussi l'intensité perçue par l'observateur ne dépend pas de sa position, et il faut souligner que ce modèle est relatif aux sources ponctuelles directionnelles.

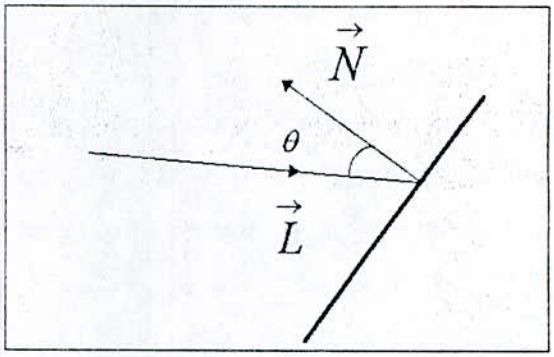


Fig. 3.20 : ***La réflexion diffuse.***

3.6.2.3 La réflexion spéculaire

Ce modèle permet de simuler la réflexion d'une source de lumière sur une surface brillante.
L'intensité réfléchie est donnée par:

$$I_s = I_p . w(\theta, \lambda).\cos^n\alpha \qquad (3.48)$$

Où α est l'angle entre le rayon réfléchi et la direction de vision; $w(\theta, \lambda)$ est le coefficient de réflexion spéculaire, il dépend du matériau, de l'angle d'incidence et de la longueur d'onde. Enfin l'exposant n dépend du matériau et croît avec la brillance de l'objet.
Contrairement au deux modèles précédents, l'intensité de lumière perçue par l'observateur dépend directement de sa position (cf. figure 3.21).

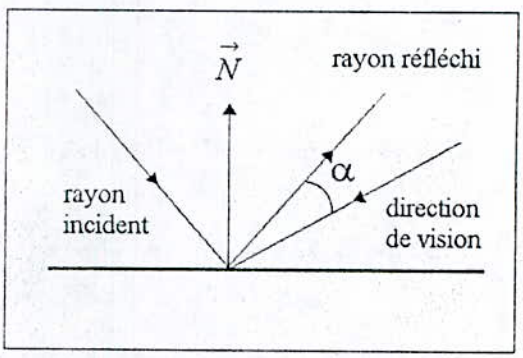


Fig. 3.21 : ***La réflexion spéculaire.***

3.6.3.Modèles du rendu

Si nous utilisons les relations précédentes telles qu'elles pour calculer l'illumination d'un point, nous aurons des objets avec un aspect facetté.
Deux méthodes permettent d'éliminer cet aspect gênant et de donner un aspect modelé aux surfaces, ils procèdent tous les deux à des interpolations linéaires.
Ces deux méthodes sont celles de **Gouraud** et de **Phong** [SCH 87b] [GOU 94].

3.6.3.1 Le modèle de *Gouraud*

Il se base sur un algorithme qui réalise une interpolation linéaire des intensités aux sommets.
L'intensité de chaque sommet est obtenue en calculant la moyenne des normales des facettes contenant ce sommet (figure 3.22), ensuite grâce à cette normale calculer l'intensité en ce sommet.
Ensuite on fait un balayage de l'image, et pour chaque facette, et pour chaque point x_e, y_e appartenant à cette facette calculer l'intensité de départ et d'arrivée et faire une interpolation linéaire entre elles, comme sur la figure 3.23.
D'après les intensités I_0, I_1, I_2 des sommets, on peut interpoler l'intensité du point P en calculant d'abord pour chaque ligne de balayage les intensités I_A, et I_B entre lesquelles on fera l'interpolation.

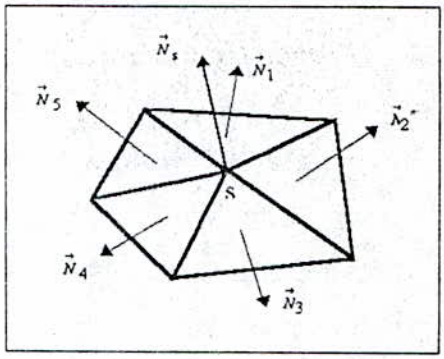


Fig. 3.22 : *La normale d'un sommet.*

On aura donc:

$$I_A = \frac{I_0(y_A - y_1) + I_1(y_0 - y_A)}{y_0 - y_A}$$

$$I_B = \frac{I_0(y_B - y_2) + I_2(y_0 - y_B)}{y_0 - y_2}$$

$$I_p = \frac{I_A(x_B - x_p) + I_B(x_p - x_A)}{x_B - x_A}$$

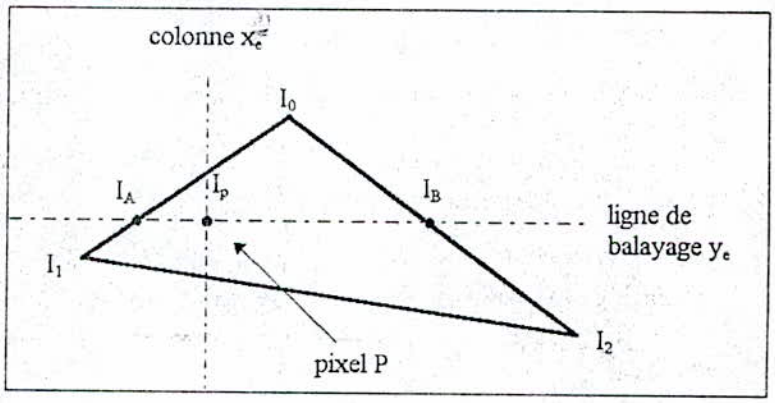


Fig. 3.23 : *Calcul de l'intensité en un point.*

L'algorithme peut donc s'écrire de cette façon:

<u>pour</u> chaque facette

<u>pour</u> chaque sommet s commun à plusieurs facettes

calculer la moyenne des normales

$$\vec{N}_S = \frac{1}{m}\sum_{I=1}^{m}\vec{N}_I$$

calculer l'intensité I_p émise par le sommet s au moyen du modèle d'éclairement $I_p = I_a + I_d + I_s$

fin

pour chaque pixel situé à l'intérieur de la facette

réaliser une interpolation linéaire des intensités aux trois sommets 0, 1, 2, pour en déduire l'intensité I_p du pixel P de coordonnées x_e, y_e.

fin

fin

3.6.3.2 Le modèle de *Phong*

Contrairement au modèle de **Gouraud** le modèle de **Phong** fait une interpolation sur les vecteurs normaux en chaque point, comme sur la figure **3.24**. Le temps de calcul est plus grand car l'interpolation se fait sur des vecteurs à trois composantes, mais la qualité de l'image est meilleure car le calcul de la réflexion spéculaire est plus précis.

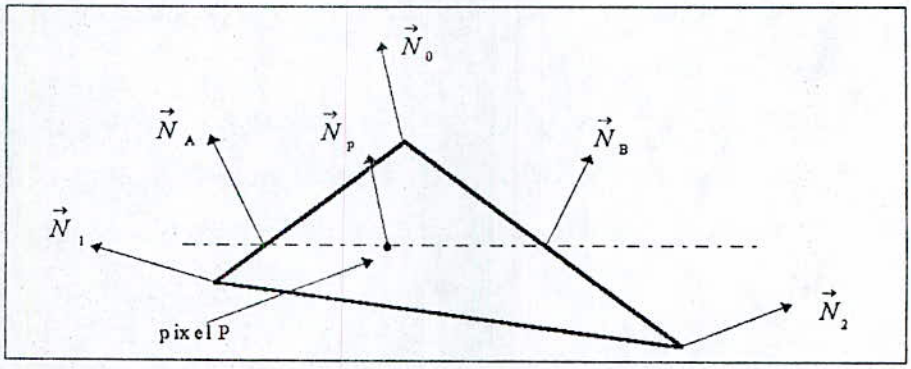


Fig. 3.24 : *Calcul de l'intensité en un point.*

L'algorithme de Phong se présente ainsi:

pour chaque facette

pour chaque sommet **s** commun à plusieurs, facettes calculer la moyenne des normales

$$\vec{N}_S = \frac{1}{m}\sum_{i=1}^{m}\vec{N}_i$$

fin

pour chaque pixel situé à l'intérieur de la facette

réaliser une interpolation linéaire des normales aux points 0, 1, 2, pour en déduire la normale $\vec{N}_S$ au pixel P de coordonnées x_e, y_e

calculer l'intensité I_p émise par le point P au moyen du modèle d'éclairement $I_p=I_a+I_d+I_s$

fin

fin

3.7 Conclusion

La réalisation d'images de synthèse tridimensionnelles nécessite donc la connaissance de certains principes et la compréhension du processus de vision chez l'être humain, comme la manière dont sont projetées les images a l'intérieur de l'oeil, ou bien la manière dont se réfléchie la lumière pour enfin être perçue. Ceci doit se faire afin de pouvoir reproduire ce processus très complexe aussi fidèlement que possible.

Dans ce chapitre nous avons vu que les méthodes proposées sont nombreuses, que ce soit pour la modélisation ou bien pour l'obtention de l'image finale. Notre travail est donc de tirer des conclusions sur les méthodes à utiliser, et de choisir les méthodes qu'on pense les mieux adapter à nos moyens. Nous privilégierons donc les méthodes ne nécessitant pas des temps de calcul énormes. Pour la modélisation nous avons opté pour la représentation volumique, qui est très adaptées à la deuxième partie de notre travail, à savoir la reconstitution d'objet 3D a partir de coupes 2D, nous procéderons ainsi par remplissage des contours. Pour la projection nous utiliserons la projection perspective afin de garantir une vision plus réelle des objets.
Et pour satisfaire à notre première exigence, qui est la minimisation des temps de calcul, nous emploierons la méthode du peintre pour l'élimination des parties cachées, et le modèle de la réflexion diffuse pour l'éclairement.

CHAPITRE IV

RECONSTITUTION D'OBJETS 3D A PARTIR DE COUPES 2D

RECONSTITUTION D'OBJETS 3D A PARTIR DE COUPES 2D

4.1 Introduction

Dans ce chapitre nous allons exposer les étapes qui nous conduisent, à partir des coupes d'un objet, à faire sa reconstitution en 3D.
Nous pourrons ainsi manipuler cet objet reconstitué, en lui faisant par exemple subir des rotations afin de le voir sous tous les angles.

Cette technique permet ainsi la visualisation d'objets tridimensionnels qu'on ne pourrait pas voir dans la réalité comme les organes humains. Il est sûr que dans ce cas on ne fait pas une vraie coupe de l'organe, mais on utilise plutôt une sonde à ultrasons.

On a donc la possibilité d'observer l'organe malade sans avoir à faire une dissection sur le patient, rendant ainsi les soins plus confortables pour celui-ci.

Mais pour pouvoir localiser les tissus malades, la reconstitution tridimensionnelle doit s'accompagner d'une reconnaissance des tissus, pour pouvoir différencier entre les partie seines et les parties malades d'un même organe.

Les étapes nécessaires pour la reconstruction d'un objet 3D à partir de ses coupes 2D sont :

1) Acquisition des différentes images de coupes de l'objet.
2) Prétraitement de ces images en utilisant les techniques de traitement numérique d'image (filtrage, etc.), afin de faciliter l'étape de détection de contour.
3) Détection des contours des images 2D. Ces contours représentants la frontière de l'objet 3D.
4) Echantillonnage des contours.
5) Reconstruction de l'objet en faisant un remplissage des différents contours avec des voxels.

6) Visualisation 3D de l'objet:

4.2 Acquisition des images 2D

La phase d'acquisition des images 2D se déroule de la manière suivante:

- Sélection d'une série d'images de coupes régulièrement réparties sur tout l'objet.
- Définition d'un point origine d'un repère par exemple le centre de l'objet.
- Numérisation des différentes images par un scanner et enregistrement sur disque sous un format graphique (BMP, TIFF, GIF, PCX, . . .).

Il est préférable d'utiliser les formats d'image compressés comme le format **GIF** qui utilise la compression **LZW [BAS 91]**, afin que l'occupation mémoire soit la plus faible possible.

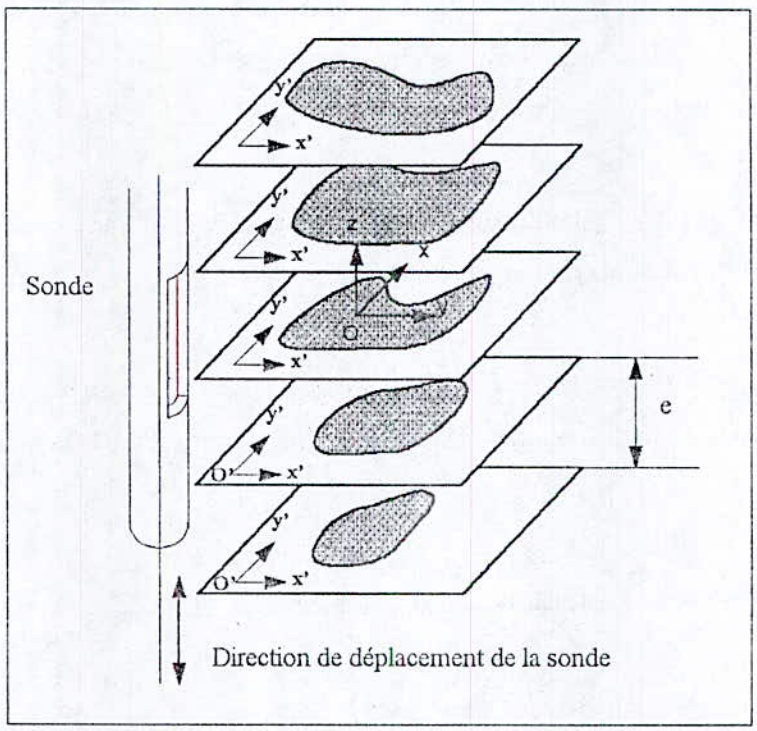


Fig. 4.1 : ***Découpage transverse avec une sonde échographique.***

La figure 4.1 montre l'acquisition des coupes transverses, en utilisant une sonde ultrasons.

4.3 Prétraitement des images [COC 95]

L'étape de prétraitement a pour objet de faciliter la détection de contour en renforçant la ressemblance entre pixels appartenant à une même région, ou en accentuant la dissemblance entre pixels appartenant à des régions différentes.

Quelques méthodes de pretraitement sont maintenant présentées. Elles concernent :

- La modification d'histogramme.
- La réduction du bruit (le filtrage).
- Le rehaussement du contraste.

4.3.1 La modification d'histogramme

Le principe est d'appliquer une transformation ponctuelle à l'intensité de chaque pixel constituant l'image. Il est clair que toutes ces transformées doivent avoir des fonctions de transfert croissantes pour préserver l'ordre initial. Dès lors, les formes des régions ne sont pas affectées.

4.3.1.1 Expansion de dynamique

Le but est d'adapter l'échelle de gris du système d'acquisition à notre système visuel (via un dispositif qui est apte à restituer l'image dans la gamme du système visuel humain). On fait une expansion du domaine $[a_{min}, a_{max}]$ du système d'acquisition au domaine $[b_{min}, b_{max}]$ de notre système de vision et ce par une loi affine:

$$b=\alpha.a+\beta \qquad (4.1)$$

Avec

$$\alpha = \frac{b_{max} - b_{min}}{a_{max} - a_{min}}$$

$$\beta = \frac{b_{min}a_{max} - a_{min}b_{max}}{a_{max} - a_{min}}$$

La figure 4.2 montre la courbe caractérisant l'expansion de dynamique.

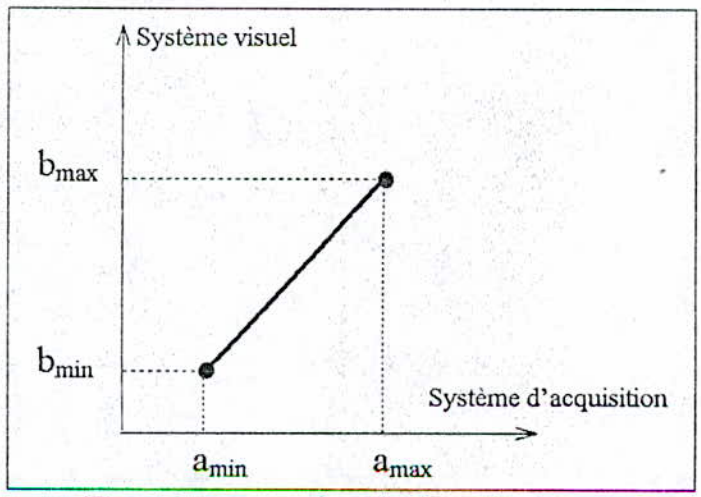


Fig. 4.2 : *Expansion de dynamique.*

4.3.1.2 Egalisation d'histogramme

Le but est de rendre le plus plat possible l'histogramme d'une image et ce pour renforcer le contraste sur des détails de l'image qui sont masqués par des variations de plus grande amplitude à grande échelle.
L'image peut être considérée comme un ensemble de réalisations d'une variable aléatoire. Sa densité de probabilité est son histogramme et sa fonction de répartition est son histogramme cumulé. On cherche une transformation T telle que:

$$T(A)=B \tag{4.2}$$

Avec B : variable aléatoire suivant la loi uniforme sur [bmin,bmax].
Il est clair comme il a été précédemment cité que T doit être strictement croissant, de plus on impose que T soit continu dérivable pour éviter que B ait un histogramme discontinu.
Si g est la densité de probabilité de B, alors:

$$g(b)=\begin{cases}\dfrac{1}{T'(A)}f(a)=\dfrac{1}{b_{max}-b_{min}} & \text{avec } b_{min}<b_{max}. \\ 0 & ailleurs\end{cases} \tag{4.3}$$

D'où

$$T'(a)=f(a) * (bmax-bmin), \tag{4.4}$$

Soit par intégration et en posant $T(a_{min})=b_{min}$, on trouve:

$$T(a)=(b_{max}-b_{min}).F(a)+b_{min} \tag{4.5}$$

Formule qui nous donnera l'égalisation désirée, comme le montre la figure 4.3.

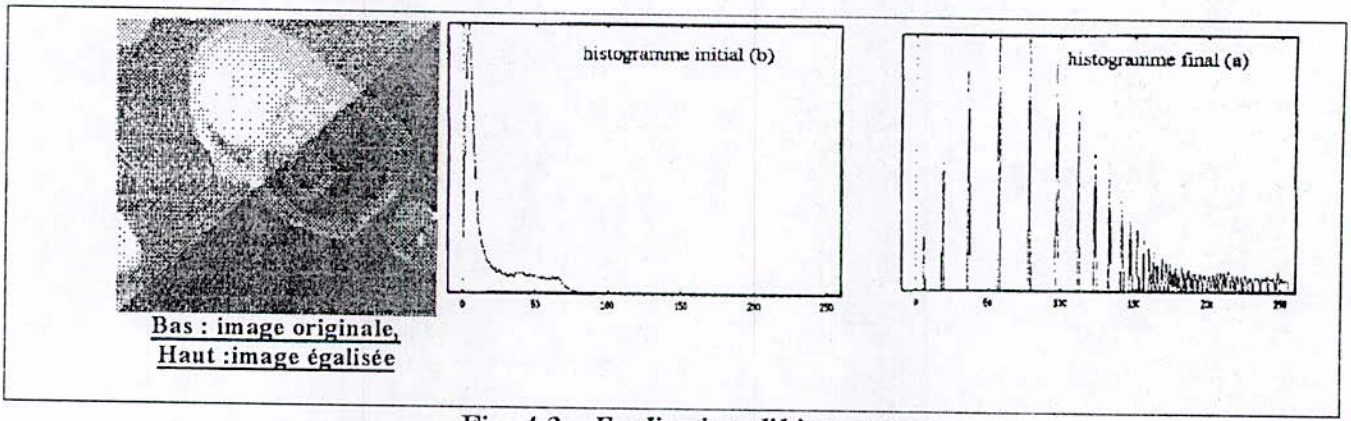


Fig. 4.3 : *Egalisation d'histogramme.*

4.3.1.3 Spécification d'histogramme

Le but est de rendre la distribution d'intensité de l'image voisine d'une distribution spécifiée à l'avance (par exemple d'une autre image).

Soit A l'image d'origine, R une image de référence et B l'image A transformée. Soit T la transformation B=T(A).

Soient f_A, f_B, f_R et F_A, F_B, F_R respectivement les densités de probabilités (histogrammes) et les fonctions de répartition (histogrammes cumulés) des variables aléatoires A, B et R.

Nous imposerons la condition $f_R \neq 0$ sur $[r_{min}, r_{max}]$ et ceci pour que F_R soit bijective.

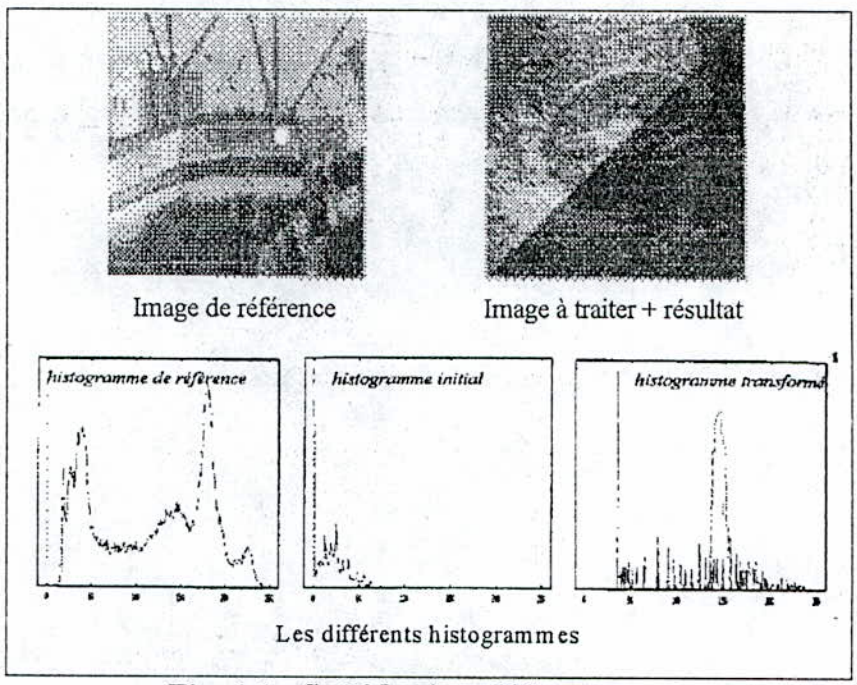


Fig. 4.4 : *Spécification d'histogramme.*

Les histogrammes f_R et f_B sont égaux équivaut à dire que:

$$F_B(b)=F_R(b)=F_B(T(a))=F_A(a) \quad \text{d'où} \quad T(a)=F_R^{-1}(F_A(a)) \tag{4.6}$$

Dans la figure 4.4 nous avons spécifié comme histogramme de référence celui de l'image de gauche et nous avons essayé de lui rapprocher celui de l'image de droite.

4.3.2 Réduction du bruit

Rappelons que filtrer signifie convoluer une image *x(i,j)* avec une fonction *h(i,j)* qui s'appelle réponse impulsionnelle du filtre.

Dans le cas discret, et si on considère que le domaine de *x* est [-N/2, N/2] [-N/2, N/2] et le domaine de *h* est [-K/2, +K/2] [-K/2, +K/2], avec K ≤ N, l'image filtrée est donnée par [RAD 93]:

$$x_f(i,j) = (h^* x)(i,j) = \sum_{i'=-\frac{K}{2}}^{\frac{K}{2}} \sum_{j'=-\frac{K}{2}}^{\frac{K}{2}} h(i-i', j-j')x(i', j') \tag{4.7}$$

On notera que le filtrage consiste simplement à remplacer chaque niveau de gris par une combinaison linéaire des niveaux de gris des points voisins, les coefficients de cette combinaison linéaire sont définis par réponse impulsionnelle du filtre.

4.3.2.1 Les filtres linéaires stationnaires

Des filtres linéaires stationnaires peuvent ainsi être utilisés tels:

Le filtre moyenneur

Sa réponse impulsionnelle est:

$$h_1(x,y) = \begin{cases} \frac{1}{t^2} & pour \quad -\frac{t}{2} < (x,y) < \frac{t}{2} \\ 0 & ailleurs \end{cases} \tag{4.8}$$

Si on a à l'entrée de ce système un bruit blanc B de variance N_0^2, nous obtenons à la sortie un bruit blanc N=B*h de variance

$$var_1((N) = N_0/t^2 \tag{4.9}$$

Le filtre gaussien

Sa réponse impulsionnelle est donnée par:

$$h_2(x,y)=\frac{1}{2\pi\sigma^2}\exp(-(\frac{x^2+y^2}{2\sigma^2})) \tag{4.10}$$

La variance du bruit filtré est:

$$\mathrm{var}_2(\mathrm{N})=\frac{N_0^2}{4\pi\sigma^2} \tag{4.11}$$

Le filtre exponentiel

Sa réponse impulsionnelle est donnée par

$$h_3(x,y)=\frac{\beta^2}{4}\exp(-\beta|x|+|y|) \tag{4.12}$$

La variance du bruit de sortie vaut:

$$\mathrm{var}_3(N)=\frac{\beta^2}{16}N_0^2 \tag{4.13}$$

Quel filtre choisir ?

Les capacités d'un filtre sont évaluées par deux facteurs essentiels qui sont:

- La variance du bruit à la sortie,
- La capacité du filtre à ne pas modifier l'image "utile". En particulier si le traitement à faire à posteriori est la détection de contour. Il est nécessaire de préserver les transitions.

Pour ce faire, quand il s'agit de filtres dont on connaît l'expression de la réponse impulsionnelle, on peut envisager de calculer la variance de manière analytique. On peut également évaluer la largeur de la transition en appliquant au système une excitation du type échelon vertical d'expression :

$$U(x,y)=\begin{cases}0, & x<0\\ 1, & x\geq 0\end{cases} \tag{4.14}$$

et on évaluera la largeur de la sortie au point de transition.

A titre d'exemple, pour les réponses impulsionnelles précédentes h_1, h_2 et h_3 et pour notre signal test U, on obtient respectivement les sorties D_1, D_2, D_3.

Si la réponse impulsionnelle est sous la forme de données, alors le meilleur moyen d'évaluer ce filtre est de calculer la variance de sortie ainsi que la largeur de transition par simulation.

$$D_1(x,y)=\begin{cases} 0, & x \le -\frac{t}{2} \\ \frac{x}{t}+\frac{1}{2}, & -\frac{t}{2} < x < \frac{t}{2} \\ 1, & x \ge \frac{t}{2} \end{cases} \tag{4.15}$$

$$D_2(x,y)=erf(\frac{x}{\sigma})^* \tag{4.16}$$

$$D_3(x,y)=\begin{cases} \frac{1}{2}\exp(\beta x), & x \le 0 \\ \frac{1}{2}\exp(-\beta x), & x > 0 \end{cases} \tag{4.17}$$

Avec:

$$erf(\frac{x}{\sigma})=\int_0^x \frac{1}{\sqrt{2\pi}}\exp(-\frac{x^2}{2})dx \tag{4.18}$$

Notons qu'il est nécessaire de faire un compromis entre la raideur des transitions et la variance du bruit à la sortie du filtre. En effet, pour avoir une faible variance à la sortie, il faut avoir β et σ aussi petits que possible ce qui implique de larges zones de transition, donnant un effet de flou, comme le montre la figure 4.5.

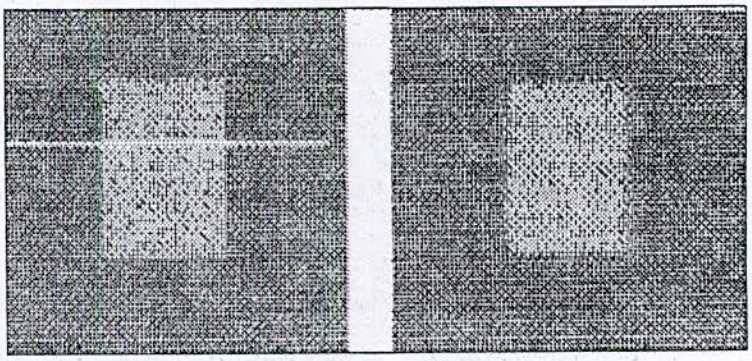


Fig. 4.5 : ***Effet du filtrage.***

Les méthodes de filtrage linéaires n'ayant pas pu résoudre tous les problèmes de restauration engendrant eux même des problèmes (tels, les élargissements des zones de transition), cela a donné naissance aux filtres non linéaires. Ce sont des filtres qui ont la caractéristique d'apporter des modifications irréversibles

sur les images, puisqu' ils ne sont pas basés sur des transformations bijectives. D'autres types de filtres existent tels que les filtres adaptatifs. En voici une brève description.

4.3.2.2 Les filtres adaptatifs

Nous avons vu que les différents filtres précédents ne sont pas sans effet sur l'image "utile". Ceci est dû au fait qu'ils appliquent "bêtement" une expression mathématique.
Si on munissait ces filtres d'un mécanisme leur permettant de s'auto-modifier, on obtiendrait un filtre adaptatif c'est à dire, qui s'accommode aux différentes zones de l'image. Le filtre adaptatif comporte donc un bloc propre de filtrage et un autre bloc de décision relatif à une consigne fixée par l'utilisateur. Voici un exemple simple d'un tel filtre.

Le moyennage adaptatif

Soit une image A à filtrer et F une fenêtre d'analyse incluse dans l'image.
L'image résultante est donnée par une expression du type:

$$C(i,j) = \frac{\sum_{A(k,l)\in F} W(A(k,l) - A(i,j)) \times A(k,l)}{\sum_{A(k,l)\in F} W(A(k,l) - A(i,j))} \qquad (4.19)$$

où

$w(x) = \begin{cases} 1, & |x| < t \\ 0, & sinon \end{cases}$ avec t, consigne fixée par l'utilisateur.

Le principe de fonctionnement de ce filtre est simple.
Dans une zone, limitée par la fenêtre à faible perturbation (la variation maximale des niveaux de gris <t) (cf. figure 4.6), cette expression n'est rien d'autre qu'une moyenne statistique. Par contre lorsque la fenêtre inclut une zone de transition, l'opération ne tient compte dans le calcul que des pixels respectant la loi de la consigne $w(x)$. Nous constatons que ce filtre est intéressant vu qu'il garde les fortes transitions intactes.

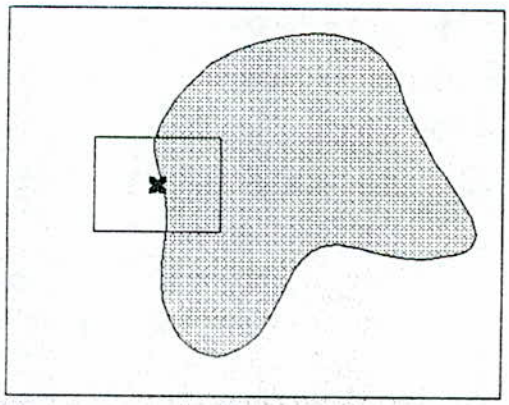


Fig. 4.6 : ***Principe du filtre adaptatif.***

4.3.3 Le rehaussement de contraste

Le but du rehaussement de contraste est de diminuer l'étendue d'une transition qui était initialement large (image peu contrastée). Ceci doit se faire bien entendu sans affecter l'intensité moyenne des régions situées de part et d'autre de la transition, comme on peut le voir sur la figure 4.7.

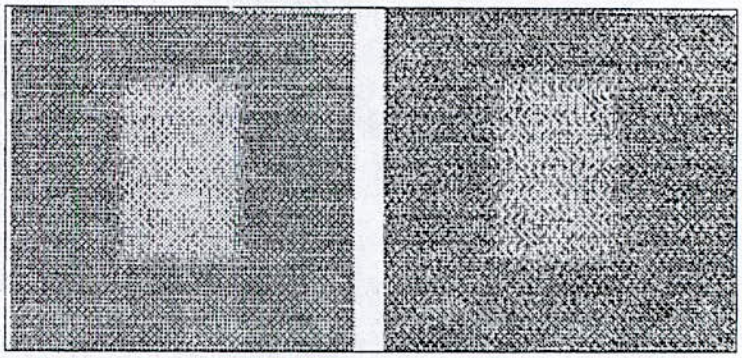


Fig. 4.7 : ***Image initiale et image rehaussée.***

4.3.3.1 Méthode du Laplacien

L'image rehaussée C s'obtient en soustrayant à l'image initiale A une certaine proportion de son Laplacien ΔA :

$$C(x,y)=A(x,y)-\lambda\,\Delta A \qquad (4.20)$$

Le choix du Laplacien est justifié comme suit:

Soit une transition quelconque. On peut, par un changement d'échelle et une translation appropriée tomber sur un cas de figure similaire à la figure ci-contre. On peut alors approximer A à un arctg(x), comme le montre la figure 4.8.
On juge la transition à travers A'(0) (=1 pour l'arctg).
Dans notre cas

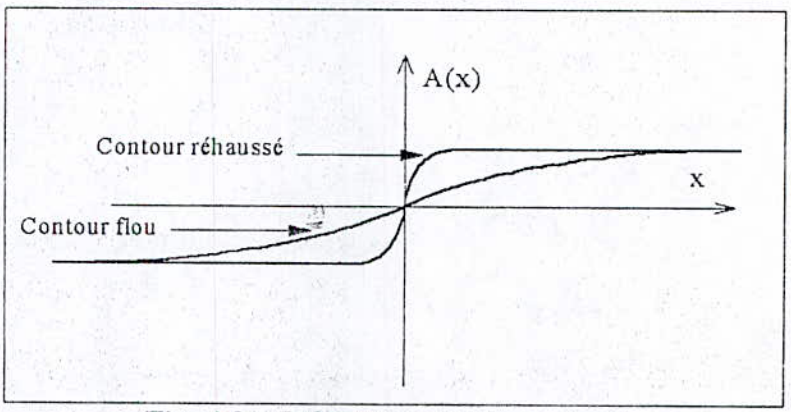


Fig. 4.8 : ***Rehaussement de contraste.***

$$C(x)=A(x)-\lambda A''(x) \text{ avec } \lambda>0 \qquad (4.21)$$

En dérivant cette expression et sachant que A'''(0)= -2, on trouve:

$$C'(0)=A'(0)+2l>A'(0) \qquad (4.22)$$

$$C'(0)=A'(0)+2l>A'(0) \tag{4.22}$$

On ressort avec une pente plus grande; c'est bien ce qu'on voulait.

La discrétisation du Laplacien permet d'obtenir une matrice qui sera le noyau de convolution pour le traitement. On obtient:

$$L=\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \tag{4.23}$$

On peut définir un Laplacien qui tient compte des dérivées diagonales. On obtient un masque de la forme

$$L_D=\begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{4.24}$$

4.4 Détection de contours par les méthodes dérivatives [COC 95]

Ces méthodes sont basées sur le changement de l'intensité. Tout changement de la valeur d'un signal peut être repéré par sa dérivée: C'est pour cela qu'on utilise la dérivation pour rechercher les contours. L'inconvénient majeur de ces méthodes est qu'elles sont très sensibles au bruit d'où la justification de l'utilisation d'un prétraitement rigoureux. Une solution à ce problème consiste à imposer un seuil aux dérivées

Le principe de la dérivée (comme le Laplacien) peut se résumer à une convolution avec un noyau (matrice) adéquat. Plusieurs noyaux existent.

4.4.1 Opérateurs de Roberts

Il est défini par les matrices de convolution suivantes:

$$F=\begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \qquad G=\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \tag{4.25}$$

Ce qui nous donne: $A_j=A*F$ et $A_i=A*G$ (4.26)

L'image finale sera de forme:

$$C(i,j)=|\nabla(i,j)| = \sqrt{A_j^2(i,j)+A_i^2(i,j)} \tag{4.27}$$

La figure 4.9 indique la direction préférentielle du gradient.

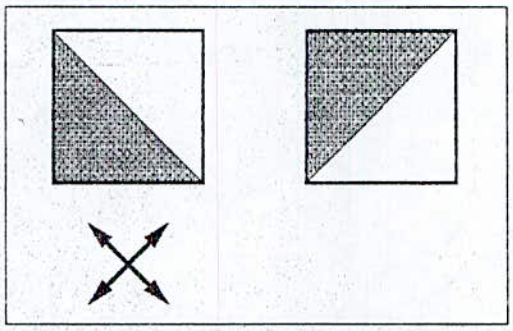


Fig. 4.9 : ***Direction préférentielle de l'opérateur de Roberts.***

4.4.2 Opérateurs de Prewitt et Sobel

Les matrices de convolutions sont les suivantes:

$$h_j = \begin{pmatrix} 1 & 0 & -1 \\ c & 0 & -c \\ 1 & 0 & -1 \end{pmatrix} \quad \text{et} \quad h_i = \begin{pmatrix} 1 & c & 1 \\ 0 & 0 & 0 \\ -1 & -c & -1 \end{pmatrix} \tag{4.28}$$

Il faut créer les matrices $A_j=A*h_j$ et $A_i=A*h_i$ et on considère l'approximation du module de son gradient:

$$C(i,j)=|\nabla(i,j)| = \sqrt{A_j^2(i,j)+A_i^2(i,j)} \tag{4.29}$$

L'image C représentera les contours recherchés.
Le noyau de Prewitt correspond à c=1 tandis que celui de Sobel correspond à c=2.
La figure 4.10 indique la direction préférentielle du gradient.

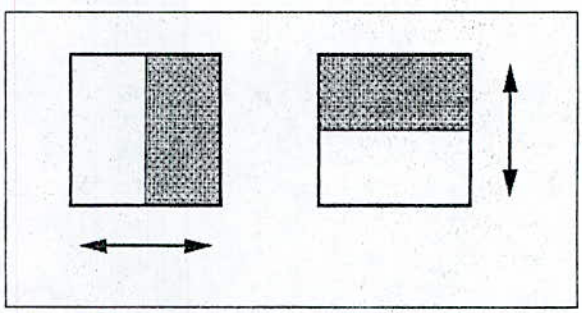


Fig. 4.10 : ***Direction préférentielle de l'opérateur de Prewitt.***

4.4.3 Opérateur de gradients directionnels de Kirsch

Cet opérateur est plus sophistiqué car il permet de chercher le contour dans huit directions. C'est donc un opérateur à huit masques obtenus par la rotation d'un masque initial h_0.:

$$h_0 = \begin{pmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{pmatrix} \qquad (4.30)$$

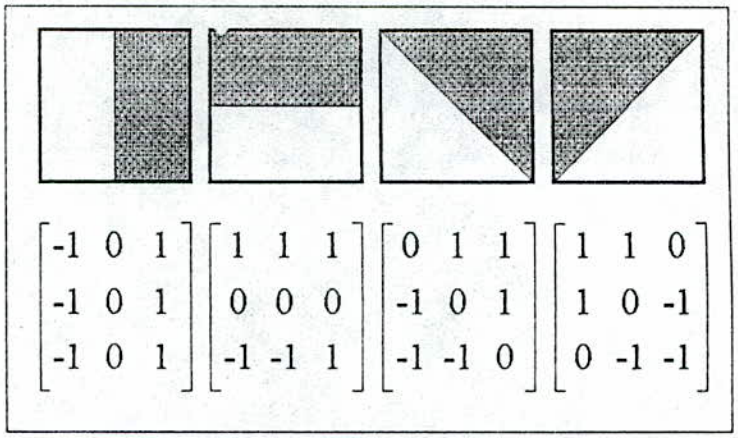


Fig. 4.11 : ***Opérateur de Kirsch à 4 directions.***

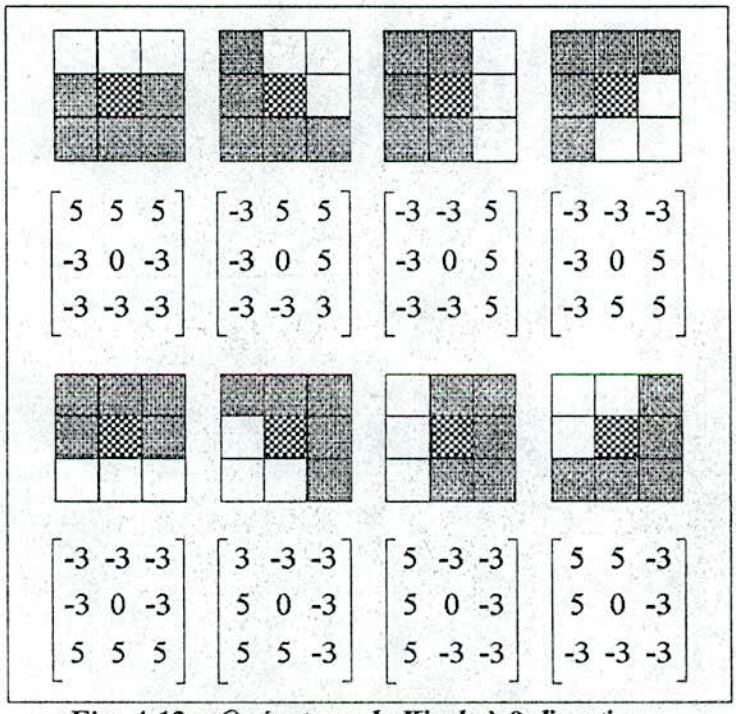


Fig. 4.12 : ***Opérateur de Kirsh à 8 directions.***

Les figues 4.11 et 4.12 montrent respectivement l'opérateur de Kirsch à 4 et 8 directions, issues de la

rotation d'un nouveau Prewitt.

Le gradient final est donné par:

$$C(i,j)=\max_k\{|h_k * A|; k = 0,7\} \tag{4.31}$$

4.4.4 Opérateur MDIF

Les matrices à utiliser sont issues de la discrétisation d'un filtre moyenneur de noyau m et d'un dérivateur utilisant les masques directionnels de Prewitt h_0 et h_2.

Le filtre moyenneur est donné par la matrice:

$$m=\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{4.32}$$

et les dérivateurs (suivant lignes et colonnes respectivement)

$$h_0=\begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} \quad \text{et} \quad h_2=\begin{pmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{pmatrix} \tag{4.33}$$

Le masque à utiliser étant la convolution de ces deux catégories de matrices, on obtient:

$$m_j=m*h_2=\begin{pmatrix} 0 & 1 & 0 & -1 & 0 \\ 1 & 2 & 0 & -2 & -1 \\ 1 & 3 & 0 & -3 & -1 \\ 1 & 2 & 0 & -2 & -1 \\ 0 & 1 & 0 & -1 & 0 \end{pmatrix} \quad \text{et} \quad m_i=m*h_0=\begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 2 & 3 & 2 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & -2 & -3 & -2 & -1 \\ 0 & -1 & -1 & -1 & 0 \end{pmatrix} \tag{4.34}$$

Puis nous déterminons les matrices $A_j=A*m_j$ et $A_i=A*m_i$. L'image résultante sera issue du calcul du module comme pour le cas de Prewitt.

4.4.5 Utilisation de la dérivée seconde

Il s'agit de rechercher le passage à zéro du Laplacien de l'image, qui va correspondre à un maximum de la dérivée première.

4.5 Echantillonnage des contours

A ce stade, l'image considérée comporte un contour fermé, sans "trous" représenté par un trait à un niveau de gris constant.

Le système informatique doit reconnaître ce contour et le stocker en mémoire. Cette étape pourrait être réalisée en mémorisant les coordonnées de tous les pixels appartenant au contour. Mais cette solution implique un volume de données à stocker et à traiter beaucoup trop important. Une solution consiste à ne conserver que quelques informations représentatives du contour.

Il existe un certain nombre de méthodes permettant d'approcher une courbe par un nombre réduit de primitives (points, segments de droite, courbes de degré supérieur).

On citera notamment les méthodes d'approximation polygonales et les méthodes d'approximation par des courbes splines [SCH 87b], [DUR 90].

Ces techniques sont en général très performantes, car une approximation précise d'une courbe peut être obtenue à partir d'un nombre réduit de données. Mais leur mise en oeuvre s'avère souvent délicate, aussi nous leur avons préféré une méthode plus simple, qui est l'échantillonnage des contours [BAS 91].

4.5.1 Echantillonnage radial

Notre but est de faire un remplissage des contours avec des voxels. Pour diminuer le nombre de voxels nécessaire pour chaque contour, on procède à un échantillonnage radial effectué à partir d'un point central au contour (cf. figure 4.13).

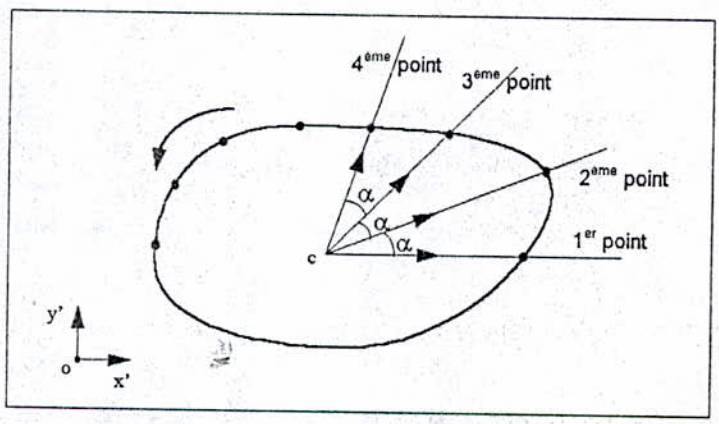


Fig. 4.13 : *Principe de l'échantillonnage radial.*

Le premier point appartenant au contour est défini par l'intersection du contour avec le rayon partant du point central C et de direction X'. Les points suivants sont donnés par l'intersection du contour avec les rayons faisant un angle α, 2α, 3α, ... $2\pi-\alpha$ avec l'horizontal.

La définition du point central du contour est donnée par un point approchée qui est obtenu simplement en mémorisant lors du tracé du contour, les coordonnées maximales et minimales selon l'axe horizontal X' et verticale Y' des points qui lui appartiennent.

Les coordonnées $C_{x'}$ et $C_{y'}$ du point central sont données par [BAS 91]:

$$\begin{cases} C_{x'} = (x'_{min} + x'_{max})/2 \\ C_{y'} = (y'_{min} + y'_{max})/2 \end{cases} \tag{4.35}$$

A l'issue de l'échantillonnage, un contour est défini par une suite de N points dont on connaît les coordonnées dans le plan de l'image.

A ce stade, les points des contours échantillonnés sont définis dans un repère (O',X',Y') associé à chaque plan de coupe. Les équations de changement de repère doivent tenir compte du point de référence qui est choisi au centre de l'objet.

Ce point particulier, de coordonnées $(V_{x'}, V_{y'})$ dans le repère associé à une image, définit l'origine O du repère 3D (O,X,Y,Z), (cf. figure 4.1).

Ensuite, pour chaque coupe, les coordonnées des points $M(x,y,z)$ des contours échantillonnés sont transcrites du repère (O',X',Y') vers le repère (O,X,Y,Z) à l'aide des équations suivantes:

$$M\begin{cases} x = x' - V_{x'} \\ y = y' - V_{y'} \\ z = n \cdot e \end{cases} \tag{4.36}$$

e: La distance entre deux coupes successives.

n: Numéro de coupe à partir de la coupe initiale.

Si la coupe est du côté Z > 0, N est positif. Sinon il est négatif.

4.5.2 Problème de l'échantillonnage radial

La recherche des pixels appartenant au contour le long du rayon d'échantillonnage nécessite quelques

précautions pour éviter des erreurs d'échantillonnage, sinon on risque de ne pas détecter le contour.
La figure 4.14 illustre un cas particulier pouvant conduire à ne pas détecter le contour.

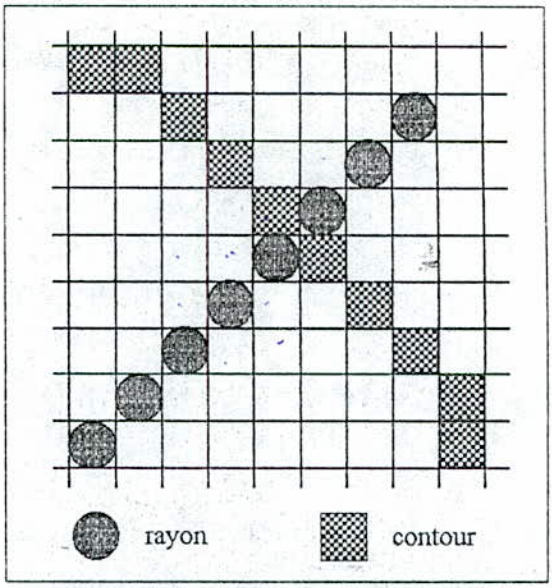


Fig. 4.14 : ***Problème de l'échantillonnage radial.***

Pour éviter ce genre de problème, le rayon est construit à partir du point central et pour chaque pixel *P(i,j)* lui appartient, on teste si *P(i,j)* est un pixel du contour, si la réponse est négative, le test est effectué avec deux des 4 voisins du point *P(i,j)*. Le choix de ces deux points dépend de la direction du rayon : Les deux cas considérés sont illustrés dans la figure 4.15.

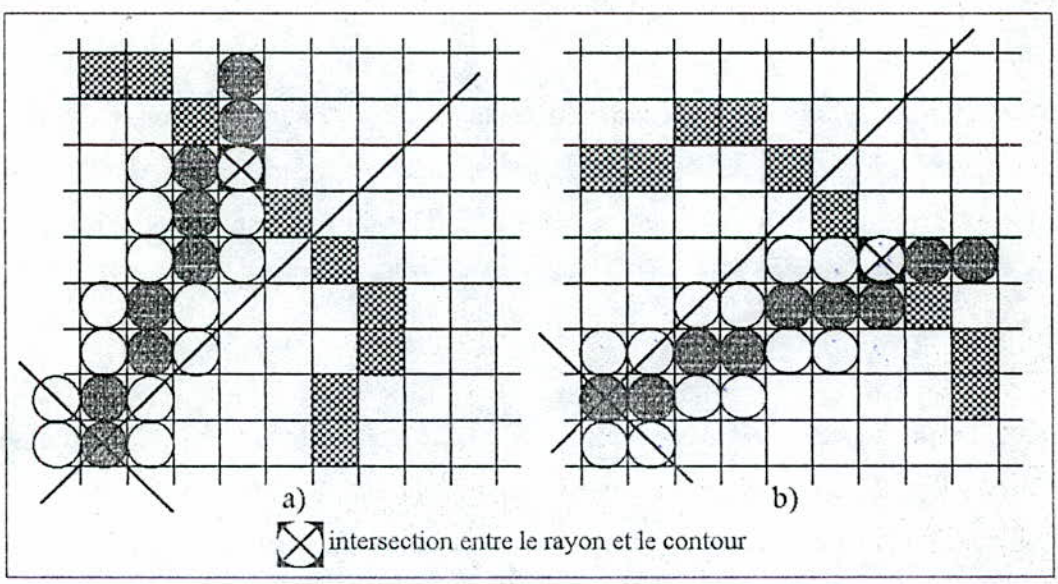


Fig. 4.15 : ***Principe de la détection de contour.***

a) Dans la figure 4.15.a la direction du rayon est comprise dans les intervalles angulaires $\left[\frac{\pi}{4},\frac{3\pi}{4}\right]$ et $\left[\frac{5\pi}{4},\frac{7\pi}{4}\right]$. On teste si les deux voisins horizontaux de chaque point du rayon appartenant au contour.

Le point marqué ⊠ est dans ce cas considéré comme l'intersection du rayon et du contour.

b) Dans la figure 4.15.b la direction du rayon est comprise dans les intervalles $\left[0,\frac{\pi}{4}\right]$, $\left[\frac{3\pi}{4},\frac{5\pi}{4}\right]$ et $\left[\frac{7\pi}{4},2\pi\right]$. On teste alors si les deux voisins verticaux de chaque point du rayon appartiennent au contour.

En fait l'idéal serait de prendre les deux directions à la fois. Mais cette méthode se base sur la probabilité de détection du contour qui est plus grande suivant l'horizontale ou la verticale, selon la l'intervalle angulaire choisi. Cette méthode est applicable seulement pour des contours convexes ce qui est le cas de la plus part des objets 3D simple.

4.6 Visualisation de l'objet

Après avoir obtenu les contours et avoir procédé à leur échantillonnage, on peut faire le remplissage de ces contours en utilisant la représentation volumique.

Comme le montre la figure 4.16, on commence par placer le premier voxel en un point du contour échantillonné, puis on lui applique des rotations et translations pour l'amener vers le prochain point du contour. On choisira le sens trigonométrique, pour le sens du parcours du contour. La translation est donnée par la distance séparant ces deux points, et la rotation par l'angle formé par les droites tangentes en ces deux points.

La figure 4.16 montre aussi l'un des problèmes de cette méthode, à savoir la continuité des contours formés par les voxels. Ainsi pour avoir une bonne reconstitution des contours et éviter les discontinuités, il faut que le pas d'échantillonnage ne soit pas trop grand, et les arêtes du voxel aussi petites que possible.

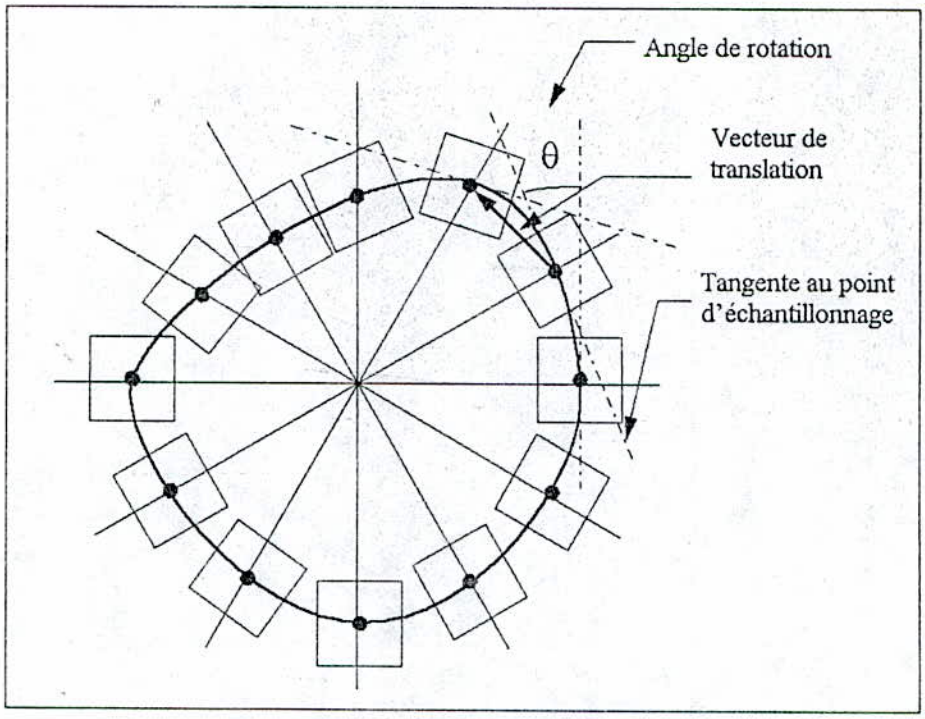


Fig. 4.16 : *Remplissage des contours.*

4.7 Conclusion

Nous avons vu dans ce chapitre les différents traitements qu'on peut appliquer aux images acquises par scanner pour pouvoir obtenir un contour, afin de faire une reconstitution tridimensionnelle.

Ces traitements ne sont pas tous nécessaires car il n'y a pas de méthode unique pour le traitement de toutes les images, mais il faut procéder le plus souvent au cas par cas.

Nous avons constaté que pour la partie filtrage le filtre adaptatif donnait les meilleurs résultats, mais malheureusement il demande beaucoup de temps de calcul. Et vu le nombre important d'images à traiter; nous avons optes pour un filtre moyenneur, qui après plusieurs essaies s'est avéré satisfaisant.

CHAPITRE V

IMPLEMENTATION DU LOGICIEL

IMPLEMENTATION DU LOGICIEL

5.1 Introduction

On présente dans ce chapitre le travail effectué, qui se traduit par la réalisation d'un logiciel destiné à la génération de formes tridimensionnelles avec effet d'éclairage.
Il se divise en deux grandes parties selon la nature des formes à générer.

La première est destinée à la génération et à la manipulation de formes simples telles que les cylindres et les sphères.

La deuxième partie étant destinée à la manipulation des formes 3D obtenues en faisant une reconstitution d'objets quelconques à partir de leurs coupes.
Ce logiciel a été développé pour fonctionner sous l'environnement **Windows (3.1 ou 95).**

Ce choix s'est fait pour de multiples raisons, parmi elles:

1) La possibilité de réaliser un logiciel interactif. Ceci est dû à la facilité de créer des interfaces conviviales, (menus déroulants, barres de boutons...).

2) La possibilité d'allouer des espaces mémoires suffisant pour pouvoir stocker les données relatives à l'objet, comme les coordonnées des sommets.

3) L'utilisation de puissantes fonctions graphiques, exploitées à travers l'interface graphique de Windows (GDI: Graphic Device Interface) [PET 91].

5.2 Outils de développement

La création de notre application s'est faite essentiellement en langage C++. Pour cela nous avons donc utilisé la suite de développement BORLAND C++ version 4.5 [LEB 95], qui nous a permis dans un premier temps d'implémenter les différents algorithmes et procédures de calcul.
La programmation a été facilitée par l'utilisation des OWL *"ObjectWindows Library "*, qui consistent en des bibliothèques de classes facilitant la programmation sous **Windows**.

La création de l'interface utilisateur s'est faite avec **Ressource Workshop**, fourni avec la suite BORLAND. L'interactivité qu'il offre dans la création des menus, boutons et autres boîtes de dialogues, nous a beaucoup facilité la tâche.

Nous avons aussi utilisé **Turbo Profiler** dans un souci d'optimisation, car on pouvait avoir connaissance des temps d'exécution de chaque fonction. Il nous ainsi possible de déceler celles dont l'exécution prend trop de temps afin de les reprogrammer.

En dehors des outils BORLAND, nous avons utilisé MATLAB 4.2 [MAT 93], afin de faire le traitement des images pour la reconstitution des objets à partir de coupes. Il nous a notamment servi pour faire la détection de contours.

5.3 Présentation du logiciel

Le logiciel comporte un menu principal offrant à l'utilisateur la possibilité de choisir différentes options, ainsi qu'une barre de boutons, (cf. figure 5.1).

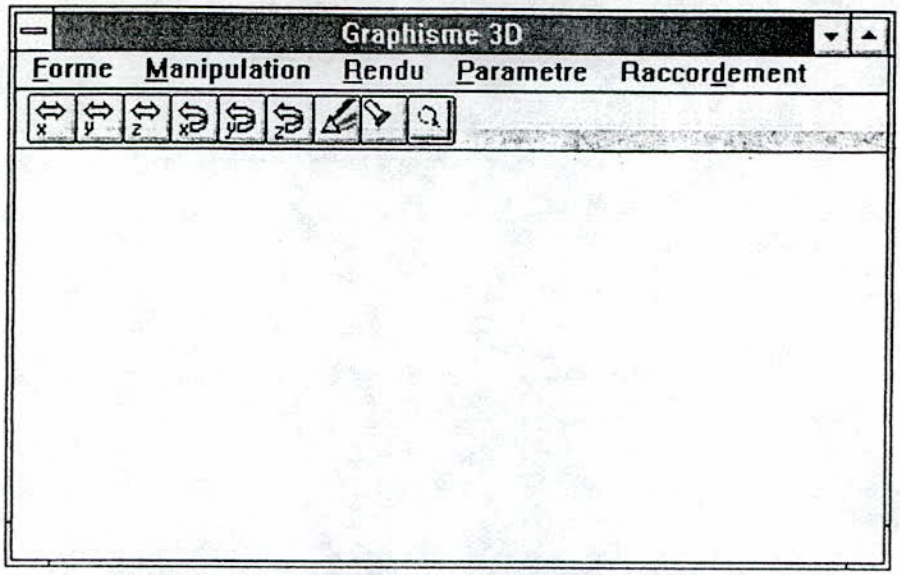


Fig. 5.1 : *Le menu principal et la barre des boutons.*

La barre des boutons, reprend les fonctions les plus importantes, comme le montre la figure **5.2**.

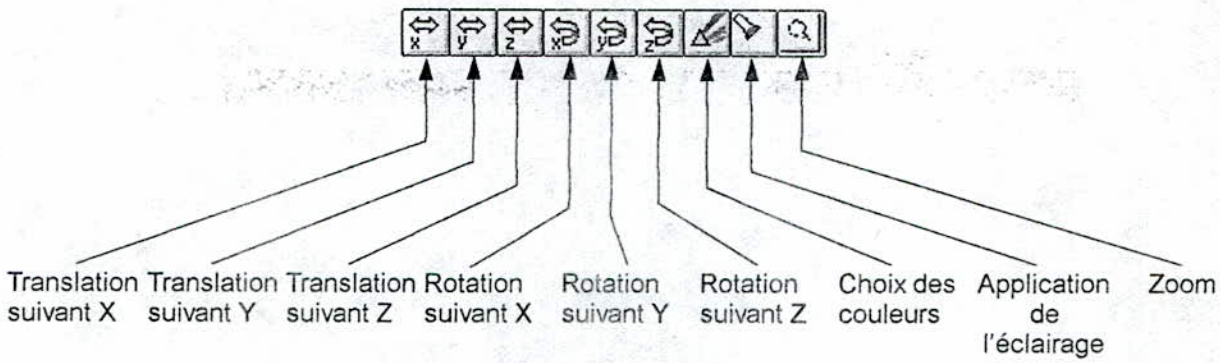


Fig. 5.2 : *Les fonctions de la barre des boutons.*

5.4 Génération de formes standards

Nous allons présenter le menu permettant la visualisation et la manipulation des objet standards.

5.4.1 Forme

L'activation de ce menu, permet d'accéder à une liste d'articles, qui nous donnent la possibilité de choisir entre plusieurs formes de bases, comme le montre la figure **5.3**.

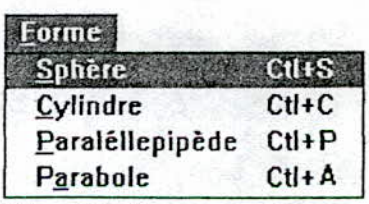


Fig. 5.3 : *Le menu Forme.*

On peut par exemple générer des cylindres (cf. figure **5.4**), ou des parallélépipèdes (cf. figure **5.5**).

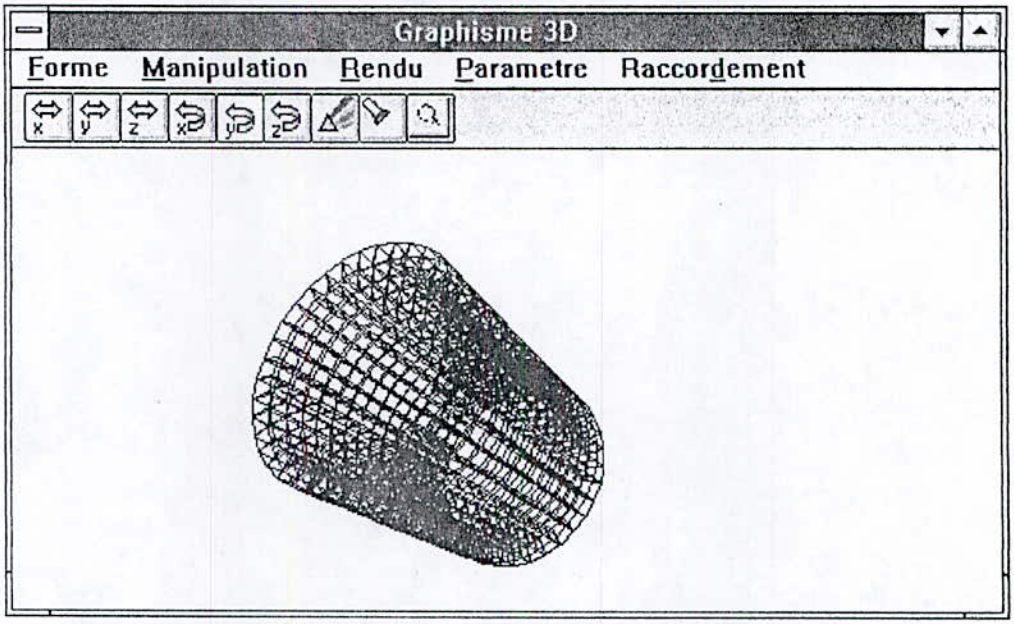


Fig. 5.4 : *Construction d'un cylindre.*

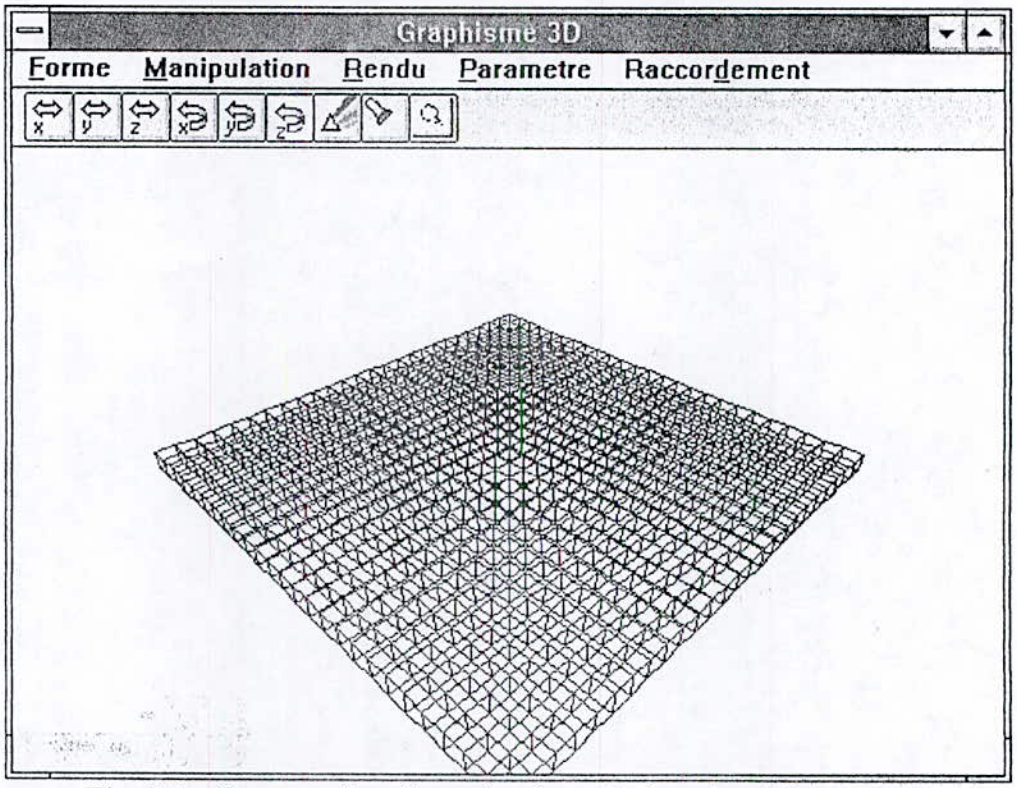


Fig. 5.5 : *Construction d'un plan à partir de cubes élémentaires.*

5.4.2 Manipulation

On a la possibilité à travers ce menu et les articles qu'il propose, de faire subir à l'objet construit précédemment différentes manipulations. On peut ainsi le translater ou lui faire subir des rotations suivant les trois axes, ou bien faire un zoom sur l'objet. La figure **5.6** montre ces options, regroupées dans le menu *Manipulation.*

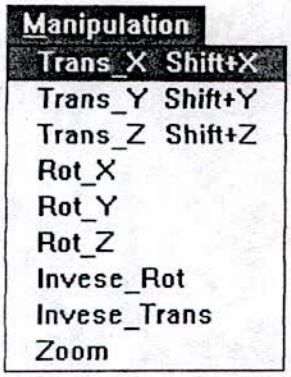


Fig. 5.6 : ***Le menu Manipulation.***

5.4.3 Rendu

On trouve des articles correspondant aux différentes méthodes de rendu (cf. figure **5.7**)

Fig. 5.7 : ***Le menu Rendu.***

Fil de fer: Cet article permet de dessiner l'objet avec une représentation fil de fer. Il faut souligner que c'est la représentation par défaut. La figure **5.8** montre la représentation d'une cylindre en fil de fer avec la couleur verte.

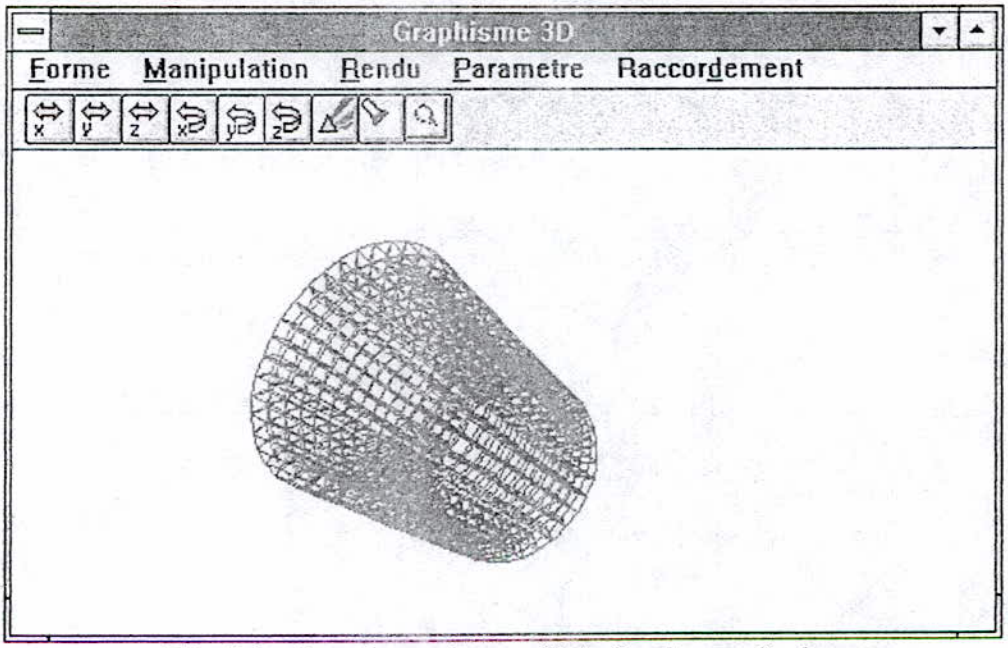


Fig. 5.8 : ***Représentation en fil de fer d'un cylindre.***

Surfaces cachées : On peut, en activant cet article, faire l'élimination des parties cachées (cf. figure **5.9** et figure **5.10**).

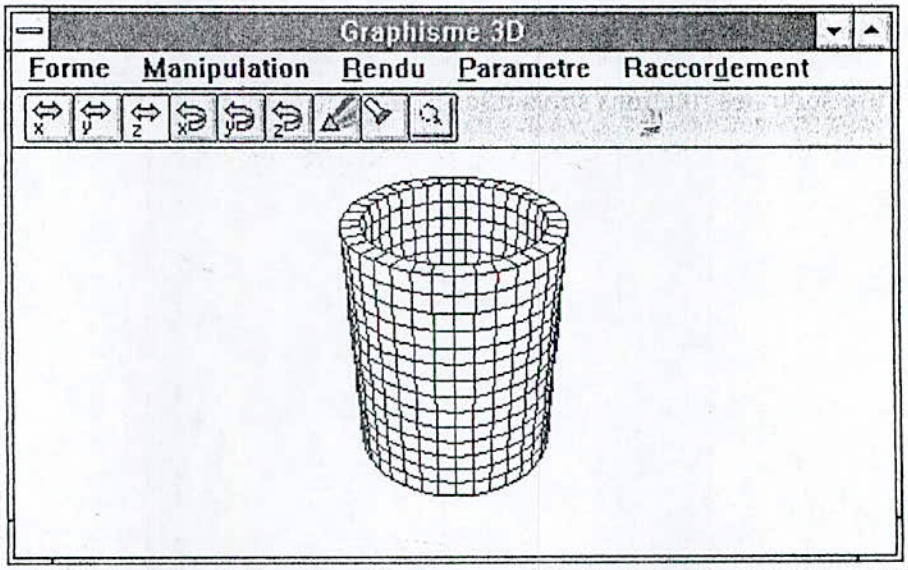


Fig. 5.9 : ***Représentation du cylindre avec élimination des parties cachées.***

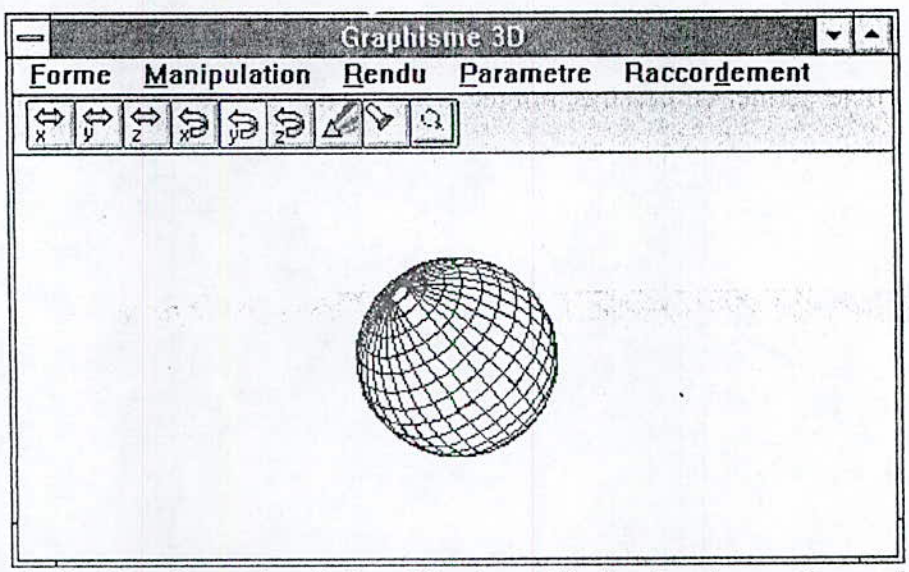


Fig. 5.10 : ***Représentation de la sphère avec élimination des parties cachées.***

Couleur: En sélectionnant cet article, une boîte de sélection de couleur apparaît, figure **5.11**.

La partie gauche permet de choisir une couleur parmi **48**.

On peut choisir une couleur personnalisée en activant le bouton "*définir les couleurs personnalisées*".

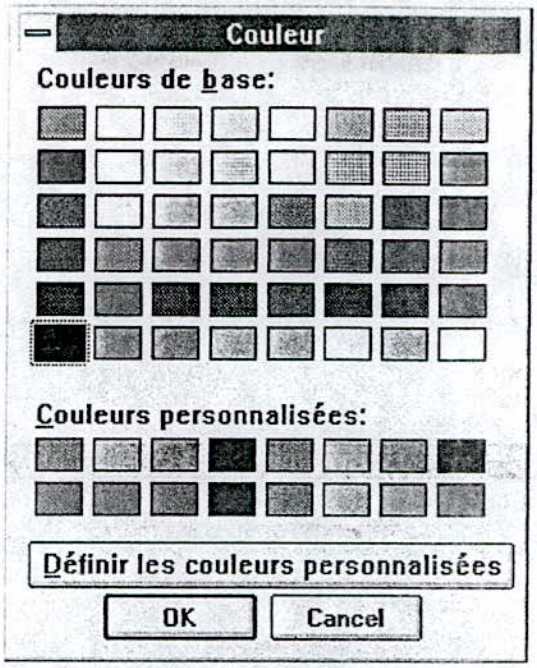


Fig. 5.11 : ***Boîte de sélection des couleurs.***

L'objet sera donc dessiné avec la couleur sélectionnée. La figure 5.12 montre une sphère avec prise en compte des parties cachées; la couleur de remplissage étant le vert.

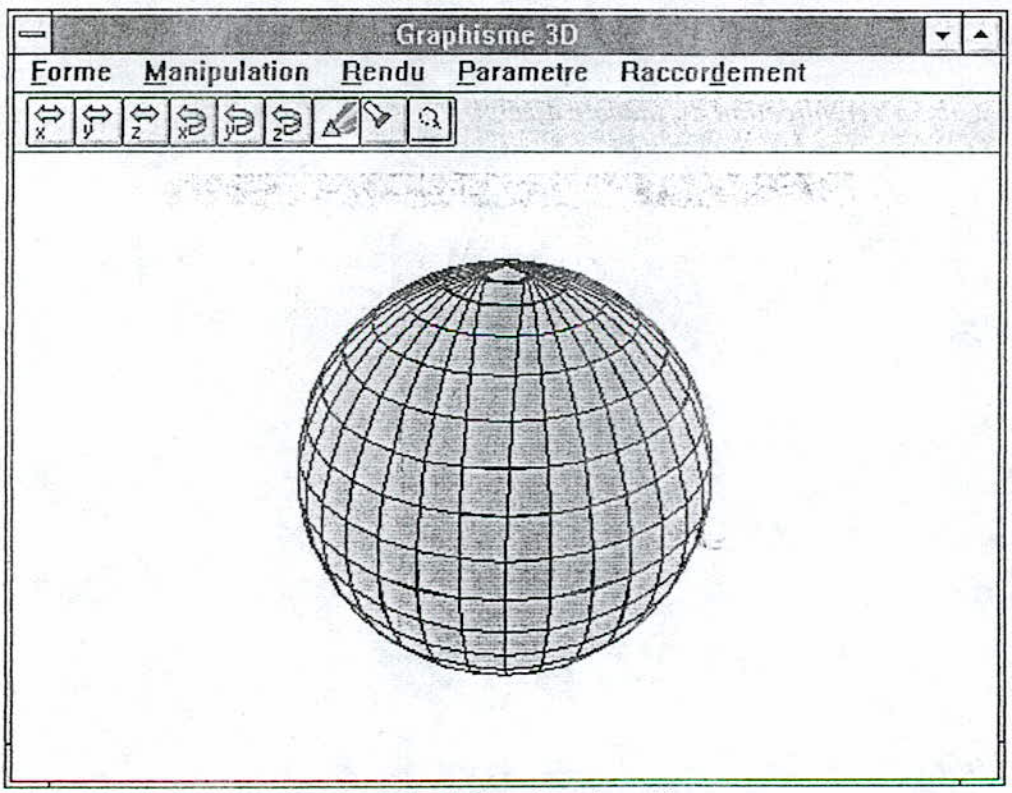


Fig. 5.12 : ***Coloration des objets.***

Eclairage: Cette option permet la mise en oeuvre éclairage, l'objet apparaît très illuminé dans la partie qui se trouve en face de la source de lumière et plus sombre sur les cotés, car nous avons placé la source de lumière au point d'observation, pour minimiser les temps de calcul.

La spécification de la couleur se fait grâce au sous menu ***Couleurs*** de l'article ***Eclairage***. On a donc accès à une boîte de dialogue qui permet de choisir une couleur parmi 8. Les figures **5.13** à **5.16** montrent la représentation de différents objets, en simulant un éclairage frontal, et en choisissant plusieurs couleurs pour les surfaces.

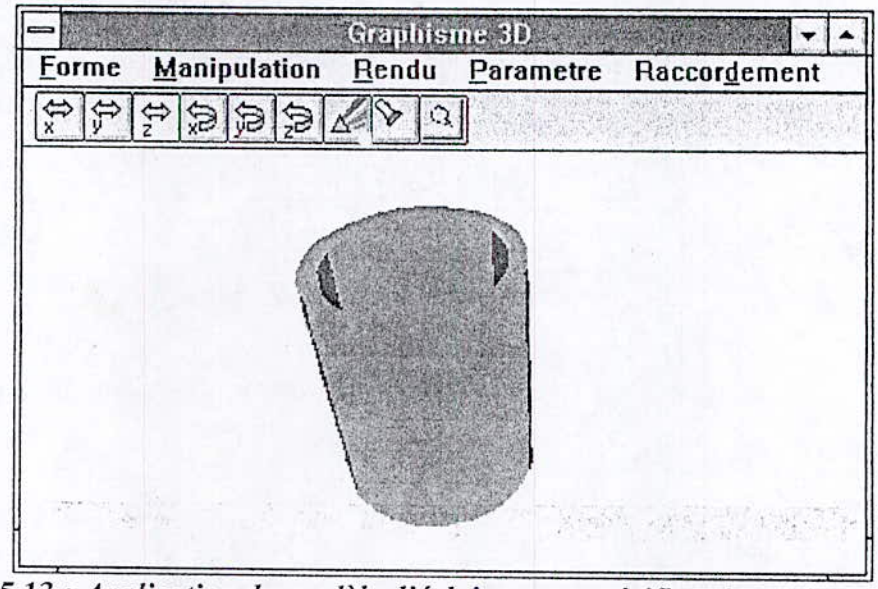


Fig. 5.13 : *Application du modèle d'éclairage en spécifiant la couleur jaune.*

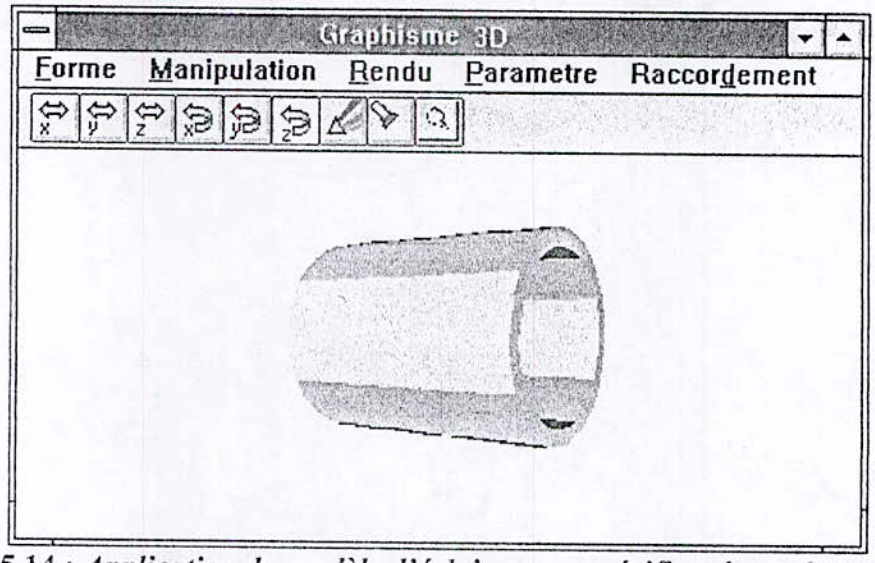


Fig. 5.14 : *Application du modèle d'éclairage en spécifiant la couleur grise.*

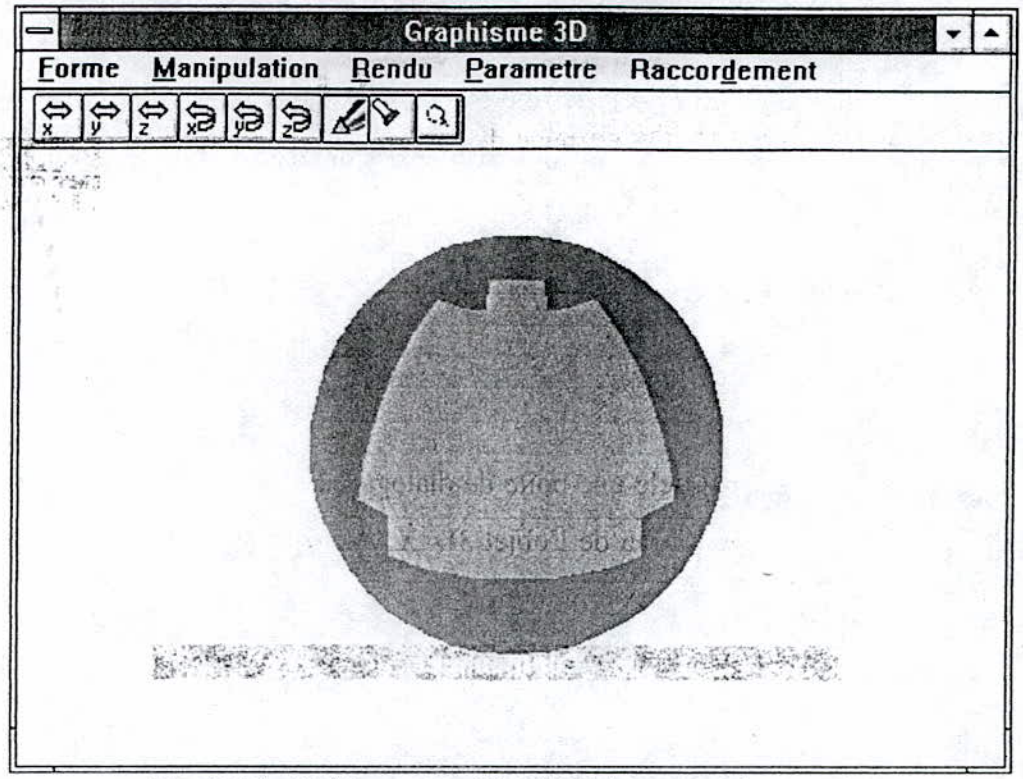


Fig. 5.15 : *Sphère avec effet d'éclairage, couleur verte.*

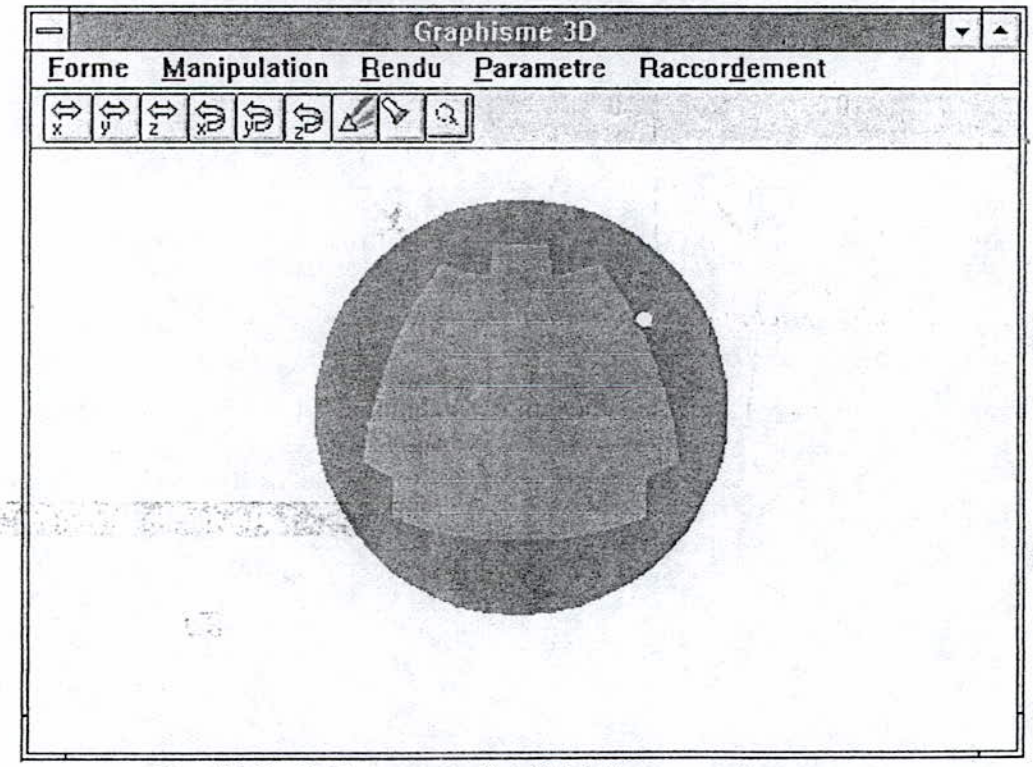


Fig. 5.16 : *Sphère avec effet d'éclairage, couleur mauve.*

5.4.4 Paramètre

Ce menu est donné par la figure **5.17**. On y trouve des options qui permettent de changer les paramètres de vision, ou bien les dimensions des objets.

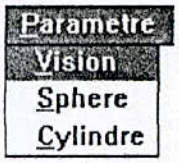


Fig. 5.17 : ***Le menu Paramètre.***

Vision: Lors de la sélection de cet article une boîte de dialogue apparaît (cf. figure **5.18**), qui permet de changer les paramètres de visualisation de l'objet **3D**. $\mathbf{X_o}$, $\mathbf{Y_o}$, $\mathbf{Z_o}$ représentent les coordonnées de l'observateur et $\mathbf{X_v}$, $\mathbf{Y_v}$, $\mathbf{Z_v}$ représentent les coordonnées du point visé.

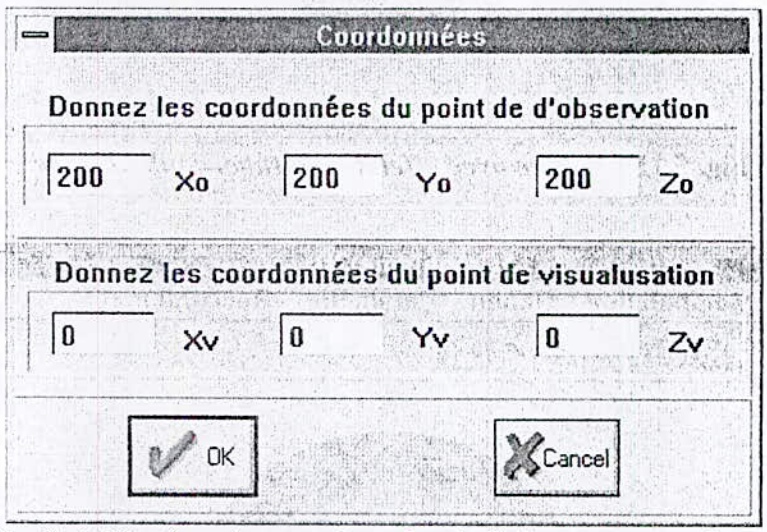


Fig. 5.18 : ***Spécification des coordonnées de l'observateur et du point visé.***

Sphère: Lorsqu'on sélectionne cet article, une boîte de dialogue apparaît, et on peur y spécifier le rayon de la sphère (cf. figure **5.19**).

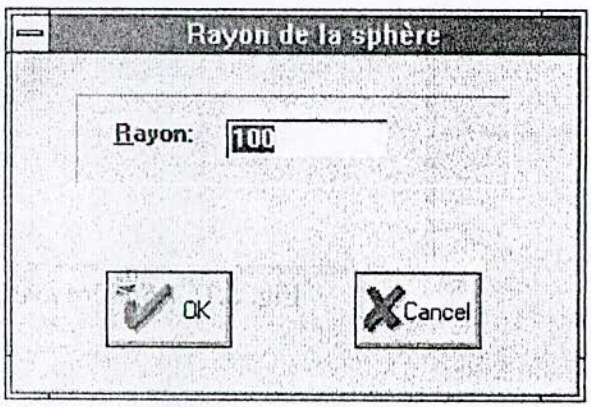


Fig. 5.19 : ***Boîte de saisie du rayon de la sphère.***

Cylindre: C'est à travers cette option qu'on fixe les paramètres du cylindre grâce à la boîte de dialogue de la figure **5.20**.

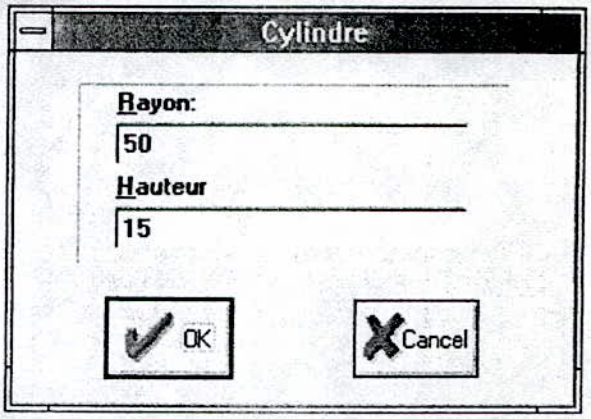


Fig. 5.20 : ***Boîte de saisie des paramètres du cylindre.***

5.5 Reconstitution des objets à partir de coupes

Pour la partie destinée à la génération d'objets tridimensionnels à partir des coupes il faut activer le menu ***raccordement***. Il faut au préalable stocker les contours des différentes coupes de l'objet, afin de pouvoir les reconstituer en utilisant la méthode des voxels.

Nous avons aussi fait une reconstitution surfacique afin de comparer les deux méthodes.

La figure **5.21** montre le menu raccordement et ses options.

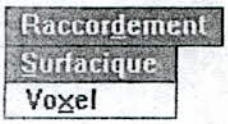
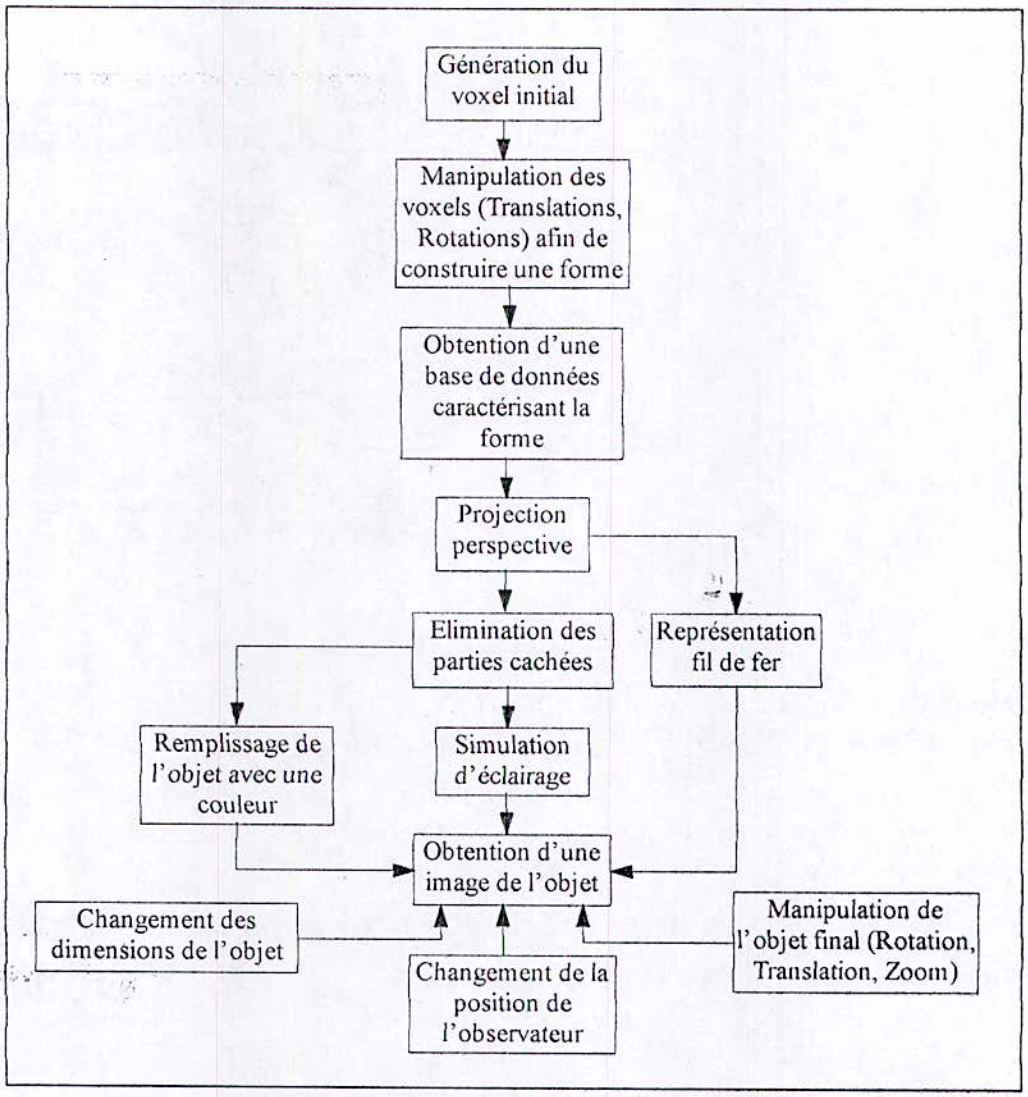


Fig. 5.21 : ***Le menu Raccordement.***

5.6 Implémentation de la partie 3D

Après avoir présenté notre logiciel, nous allons nous intéresser plus en détail aux procédures les plus importantes. Il est donc intéressant de donner un organigramme de fonctionnement qui illustre les différentes parties conduisant à la création d'une forme **3D**.



Organigramme de fonctionnement.

5.6.1 Le voxel initial

C'est la base de toute construction. Il est représenté par un cube centré par rapport au repère (O,X,Y,Z).

Ces sommets sont stockés dans un tableau SomV. Chacune de ses faces est représentée par une suite de sommets pris dans le sens trigonométrique et est stockée dans un tableau DefPoly.

5.6.2 Génération d'une forme 3D

La figure **5.22** montre le principe de la construction d'une forme 3D à partir de voxels.

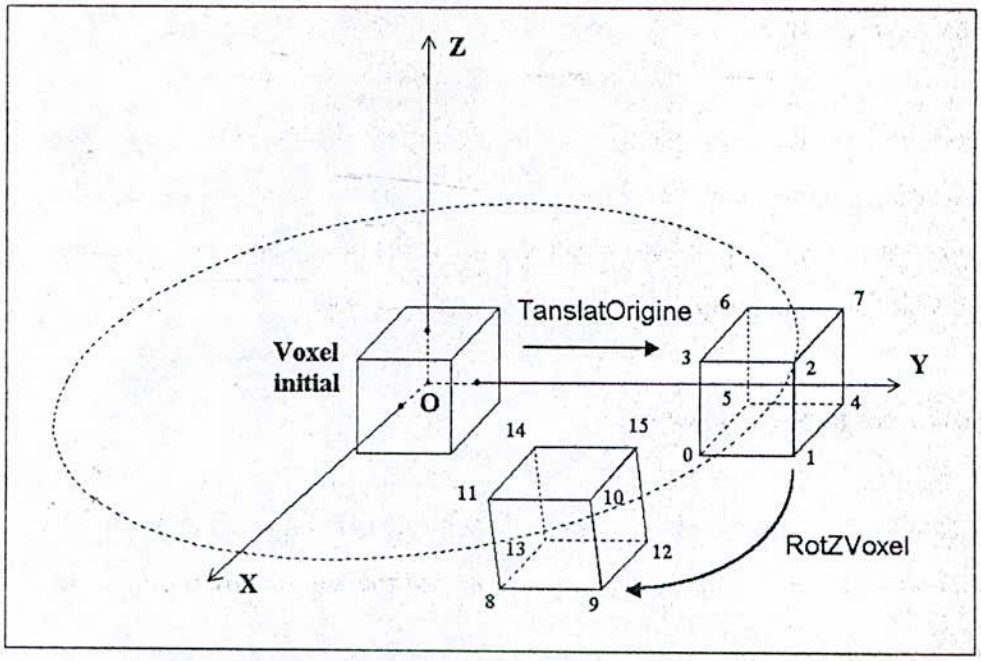


Fig. 5.22 : ***Manipulation des voxels pour la construction d'objets.***

Si par exemple on veut obtenir un cylindre, on doit placer le voxel initial sur la circonférence de ce cylindre. Pour cela, on doit lui appliquer une translation en lui appliquant la fonction TranslatOrigine. On procède ensuite à la construction de la première couronne en faisant la rotation du premier voxel par la fonction RotZVoxel. Si ce denier est formé des sommets 0→7, le voxel issu de sa rotation sera formé des sommets 8→15, de sort que le sommet 8 résulte de la rotation du sommet 0, le sommet 9 de la rotation du sommet 1, et le sommet i de la rotation du sommet i-8. Il faut souligner que ces numéros représentent des indices dans le tableau des sommets SomV.

Il faut ensuite définir les facettes par les listes de sommets correspondants. Si l'on observe la figure 5.22 on voit que la face formée des sommets 0, 1, 2, 3 engendre par sa rotation la face 8, 9, 10, 11. Il faut donc ajouter 8 aux numéros des sommets des six dernières faces du tableau DefPoly (les six faces du voxel dont on vient de faire la rotation), et ajouter les six nouvelles faces ainsi calculées à ce tableau.

Après avoir construit la première couronne on passe à la deuxième en translatant le voxel suivant l'axe Z par la fonction TranslatVoxel et refaire le même travail.

Nous obtenons ainsi deux tableaux, le premier SomV contenant les coordonnées des sommets de tous les voxels qui forment l'objet, et le second DefPoly contenant les numéros des sommets composant chaque face.

5.6.3 La mise en perspective

L'étape suivante est la projection perspective. On calcule tout d'abord les coordonnées des sommets par rapport à l'observateur. On obtient le tableau Pers, qui servira ensuite pour le calcul de la profondeur moyenne des facettes par rapport à l'observateur. On fait ensuite la projection perspective. Les coordonnées x, y des sommets projetés sont dans le tableau Proj.

5.6.4 Elimination des parties cachées

A ce stade les faces sont rangées dans un ordre arbitraire dans le tableau DefPoly. Comme nous l'avons déjà dit au chapitre III, nous allons employer l'algorithme du peintre pour régler le problème des surfaces cachées.

Mais tout d'abord il est préférable d'éliminer les faces arrières pour diminuer la taille du tableau à trier. Et vu que dans un cube on peut voir au maximum trois faces, le nombre de faces éliminées est d'au moins la moitié.

Les faces satisfaisant au critère de visibilité forment le tableau DefPolyV. La procédure de tri se fera sur ce tableau.

Après avoir calculé la profondeur moyenne de chaque facette, qui représente la moyenne des profondeurs des sommets composant la face et qu'on tire du tableau Pers, on commence à faire le tri selon z croissant. Lors de la première implémentation de cet algorithme, nous avons utilisé une procédure de tri classique, qui consiste à balayer le tableau dans le sens croissant et à tester chaque face avec celles de rang supérieur. Si sa profondeur moyenne est plus petite alors on procède à une permutation sinon on continue le test. Mais il s'avère que cette méthode prend beaucoup de temps surtout pour de grands tableaux, car le nombre de tests est égal à (N-1)! pour un tableau de N éléments.

On a donc préféré l'utilisation d'une autre méthode de tri basée sur la dichotomie appelée ***Quicksort***.

Nous avons remarqué grâce à **Turbo Profiler** que les temps de calcul sont passé de **12s** à **3s**.

Les facettes seront donc affichées dans l'ordre qu'elles occupent dans le tableau DefPolyV.

5.6.5 Simulation d'éclairage

C'est la dernière étape dans la rendu. Comme nous l'avons souligné dans la conclusion du chapitre III, nous utiliserons le modèle de la réflexion diffuse pour simuler l'éclairage. Nous supposons que la source de lumière est à l'infini, donc tous les rayons lumineux sont parallèles à une direction. Pour minimiser les

temps de calcul on choisit la direction $\vec{OV}$ (cf. figure **3.16**). Nous avons donc l'impression d'un éclairage frontal et l'objet paraît plus sombre sur les côtés où la lumière est plus rasante.
Nous devons tout d'abord charger une palette correspondant aux différents degrés de luminosité des facettes. On crée ainsi un dégrade de couleurs, où les couleurs les plus intenses correspondent aux plus grandes luminosités. Théoriquement il est possible de créer une palette de **256** couleurs (les **256** registres DAC, voir chapitre II). Mais en pratique le système se sert de **16** couleurs pour ses besoins d'affichage, ce qui nous laisse **240** couleurs à spécifier.
La palette est créée de couleur la plus sombre vers celle la plus lumineuse et le produit scalaire entre le vecteur $\vec{OV}$ et la normale de cette facette détermine un indice dans cette palette, qui associera donc une couleur à la facette.

5.6.6 Manipulation des objets

Une fois obtenu, l'objet peut être manipulé, par exemple en lui faisant subir des rotations ou des translations. En réalité on recalcule les coordonnées de tous les sommets du tableau SomV, puis on refait les opérations de projection, élimination des faces cachées et calcul d'éclairage.

5.7 Traitement des images 2D

L'acquisition des images servant à la reconstitution des objets 3D à partir de coupes s'est faite avec un scanner. Dans les applications professionnelles on procède rarement à des coupes réelles sur l'objet, mais on utilise plutôt des sondes ultrasons ou bien des capteurs de positions, avec lesquels on détecte l'enveloppe extérieure de l'objet. Vue la nécessité de faire des coupes, nous avons utilisé une aubergine afin de faciliter cette opération.
Nous avons donc obtenu **11** coupes d'environs **5** mm d'épaisseur, comme le montre la figure **5.23**.
Les images de ces coupes sont sauvegardées sur le support de stockage grâce au logiciel d'acquisition sous format **BMP**.

Il faut ensuite faire une détection de contour, mais comme la reconstitution ne se fait pas sous **MATLAB** il est nécessaire de stocker ces contours dans des fichiers de données exploitables par les fonctions de **C++**.

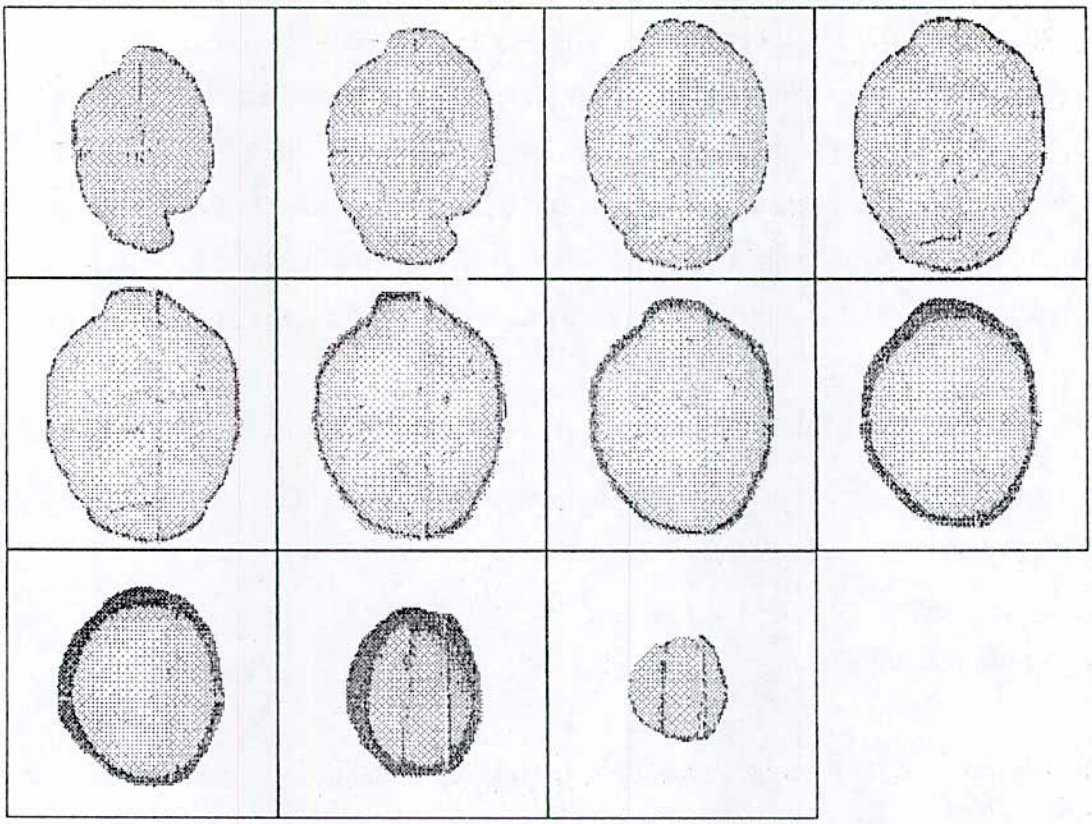


Fig. 5.23 : *Différentes coupes de l'objet.*

5.7.1 Lecture des images

MATLAB possède des fonction qui permettent la lecture de fichiers graphiques. Pour le format BMP nous avons donc utilisés la fonction **bmpread**, qui permet de transférer l'image dans une matrice. Les éléments de cette matrice représentent les codes couleur de chaque point de l'image.

5.7.2 La détection de contours

Notre but est de faire une détection de contours pour faire la reconstitution de l'objet. Nous avons exposé dans le chapitre IV les différents prétraitements que peuvent subir les images afin de permettre la détection de contours correspondant aux contours réels de l'objet. Mais sur la série d'images exposées plus haut nous remarquons qu'elles ont toutes un fond blanc (niveau de gris 255). Il est donc possible de détecter le contour extérieur de l'objet en faisant une simple binarisation de

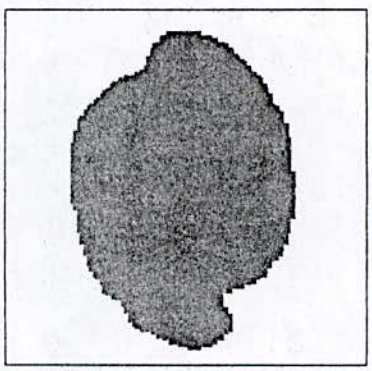


Fig. 5.24 : *La binarisation.*

l'image, en veillant à ce que les seuils comprennent tous les niveaux de gris présent dans l'image utile "aubergine". Nous obtenons donc une région noire représentant l'objet et une région blanche représentant le fond, comme sur la figure 5.24.

La détection de contours se fait donc très facilement, vu que les transitions sont nettes.

Cependant, un problème est apparu dés la 6ème coupe, créé par la bande sombre et qui fait que le contour de l'image ne correspond pas au contour réel de l'objet.

Ainsi, si pour les 5 premières coupes on ne se souciait pas du seuil inférieur lors de la binarisation, il est nécessaire cette fois de l'ajuster afin de ne pas englober cette bande sombre.

La figure 5.25 montre le résultat obtenu après avoir procédé à une détection de contour sur une des coupes.

Fig. 5.25 : ***Détection de contour.***

5.7.3 L'échantillonnage des contours

Arès avoir obtenu un contour il est nécessaire de faire un échantillonnage comme nous l'avons exposé au chapitre II, afin de minimiser les temps de calcul, lors de la reconstitution.

Sur la figure 5.25 nous remarquons que le contour est au niveau de gris 255, nous testerons donc les points possédant ce niveau de gris afin de savoir s'il y a un contour.

Les grandes lignes de la méthode ont été exposées au chapitre III, mais il serait intéressant de la détailler encore plus.

Nous présentons à cet effet l'organigramme de la procédure utilisée. Il se base sur la construction d'une ligne partant du centre de l'objet et ceci à des intervalles angulaires réguliers. Au début on procède à un test sur la pente de la droite à tracer. Si cette pente est comprise entre -1 et 1 alors la construction de la droite se fait en incrémentant l'abscisse i puis on calcule l'ordonnée j correspondante. Si la pente est en dehors de cet intervalle alors on incrémente l'ordonnée j puis on calcule l'abscisse i correspondante, ainsi les droites construites ne serons pas discontinues. De cette manière on est sûr d'avoir une intersection entre la droite d'échantillonnage et le contour, en ajoutant le test des pixels horizontaux ou verticaux selon le cas.

La figure 3.26 présente l'organigramme de l'échantillonnage des contours.

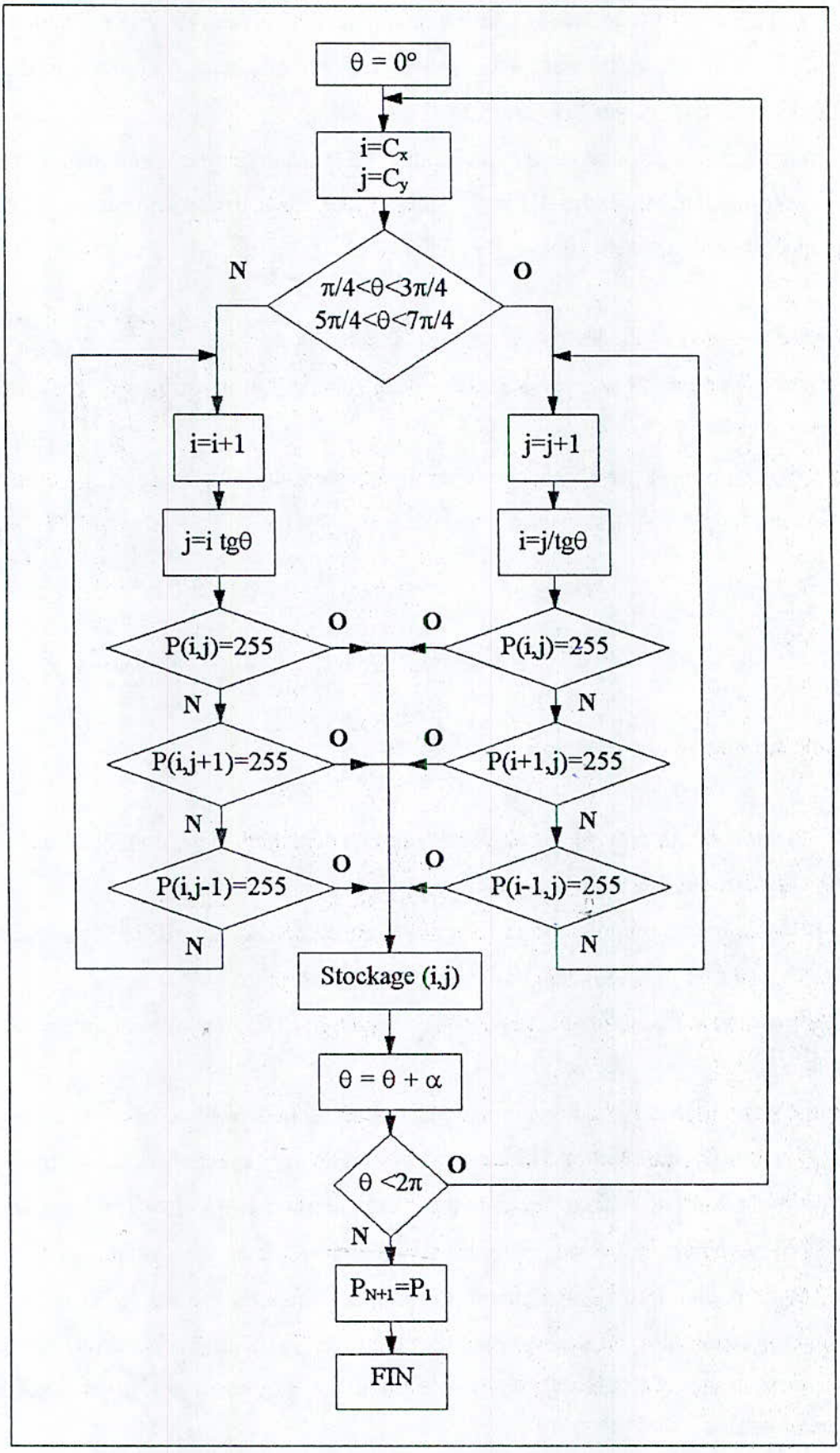


Fig. 3.26 : *Organigramme d'échantillonnage des contours.*

5.7.4 Transfert de données entre MATLAB et BORLAND C++

Après avoir obtenu les coordonnées x, y des points échantillonnées pour chacune des 11 images, nous devons calculer la composante z avec:

$$z=n.e \qquad (5.1)$$

Où **e** représente l'épaisseur d'une coupe qu'on prend égale à l'arête du voxel et **n=0** à **10**.
On sauvegarde ainsi ces coordonnées dans un fichier de données au format **MAT** de **MATLAB**.

Pour les exploiter sous **BORLAND**, on utilise la routine **readmat** fournie avec **MATLAB** et qui permet la lecture des fichiers au format **MAT**.

5.8 Reconstitution des objets

5.8.1 Méthode volumique

On fait donc la reconstitution de l'objet en remplissant les contours de ses coupes par des voxels.
Nous remarquons que la représentation n'est pas très fidèle et ceci est dû à la taille des voxels qui est assez importantes. Ainsi pour avoir une bonne résolution il faut que la taille des voxels soit la plus petite possible afin de restituer le contour.

Mais dans notre cas cela n'aurait servit à rien d'augmenter la résolution car les coupes sont épaisses et cela ferait que les différents contours reconstruits ne formeraient pas une surface continue mais on aura des vides entre les différents contours du fait que l'épaisseur du contour est plus grande que les dimensions du voxel.

Il est donc nécessaire pour avoir une bonne reconstitution de l'objet que les coupes soient les plus fines possible pour utiliser des voxels de petite taille.

Les figures **5.27** et **5.28** montrent la reconstitution d'une aubergine en utilisant le remplissage des contours avec des voxels.

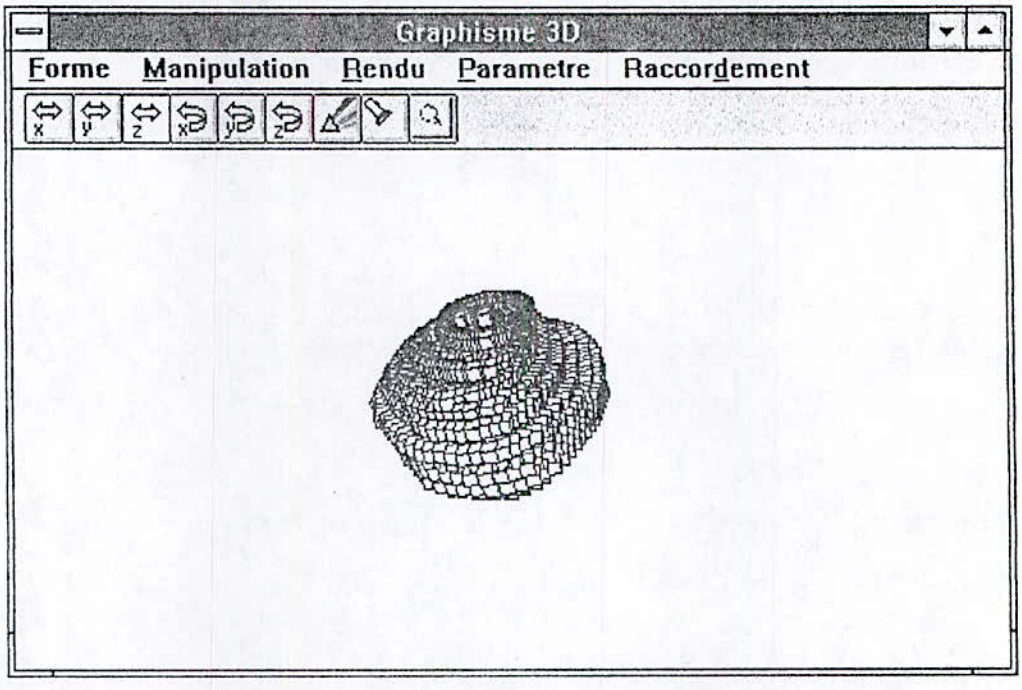


Fig. 5.27 : *Reconstitution par voxels avec élimination des parties ccachées.*

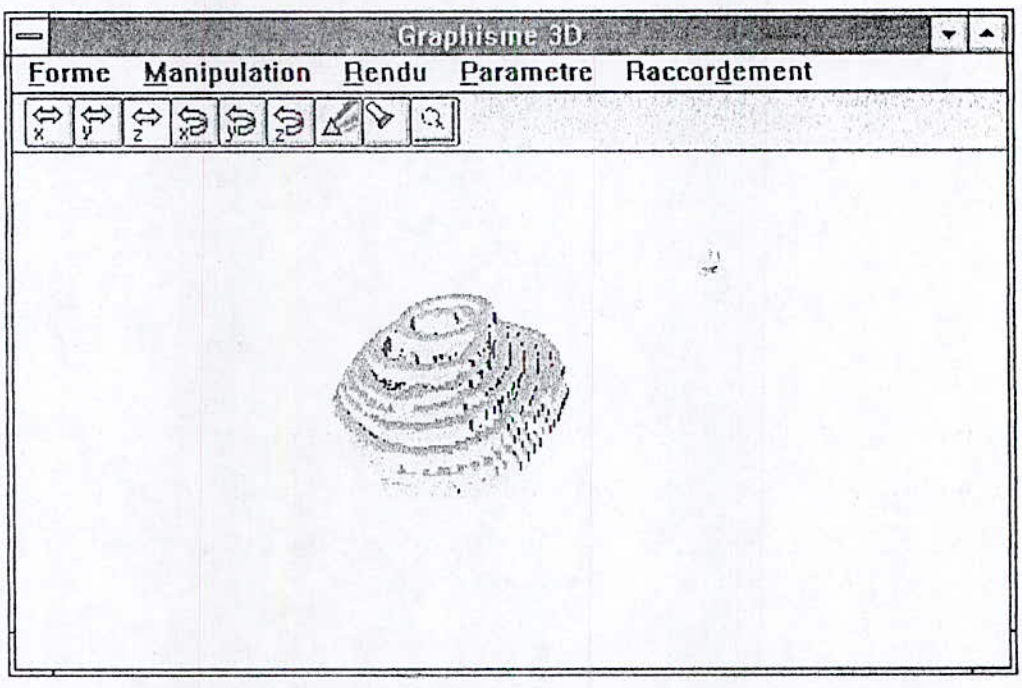


Fig. 5.28 : *Reconstitution avec des voxels avec effet d'éclairage.*

5.8.2 Méthode surfacique

Elle consiste à construire un maillage à partir des points des contours échantillonnés. Cette méthode donne lieu à une meilleure reconstitution que la méthode par voxel, mais elle est surtout utilisée pour reconstituer les surfaces extérieures des objets. Par contre la méthode volumique peut être préconisée pour l'obtention d'objets pleins.

Les figures **5.29**, **5.30** et **5.31** montrent les reconstitution de l'aubergine par la méthode surfacique.

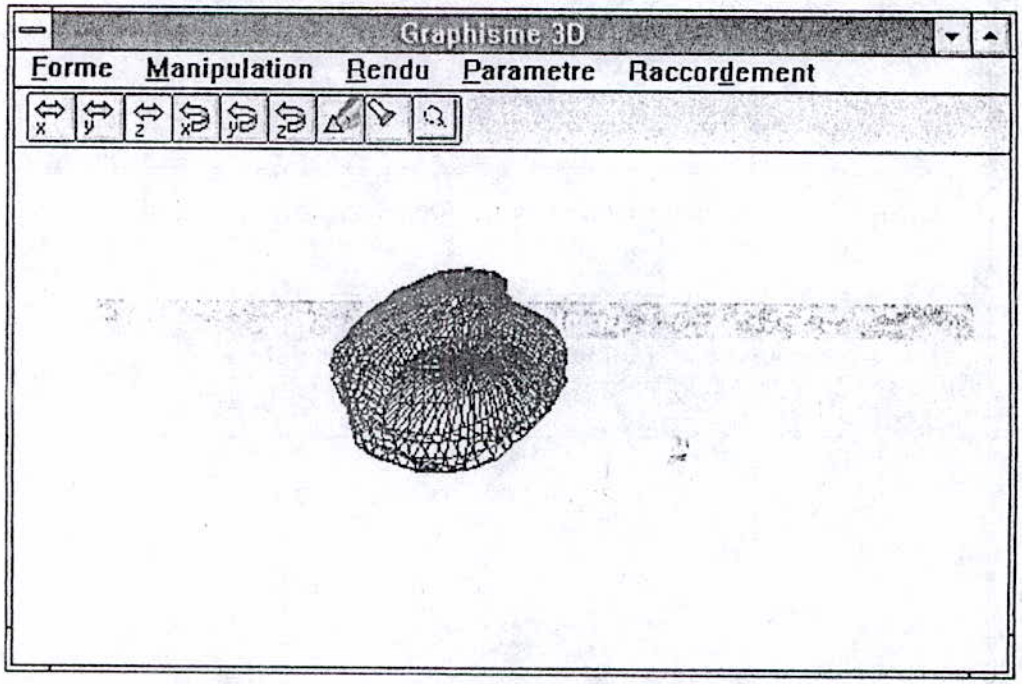


Fig. 5.29 : *Reconstitution surfacique en fil de fer.*

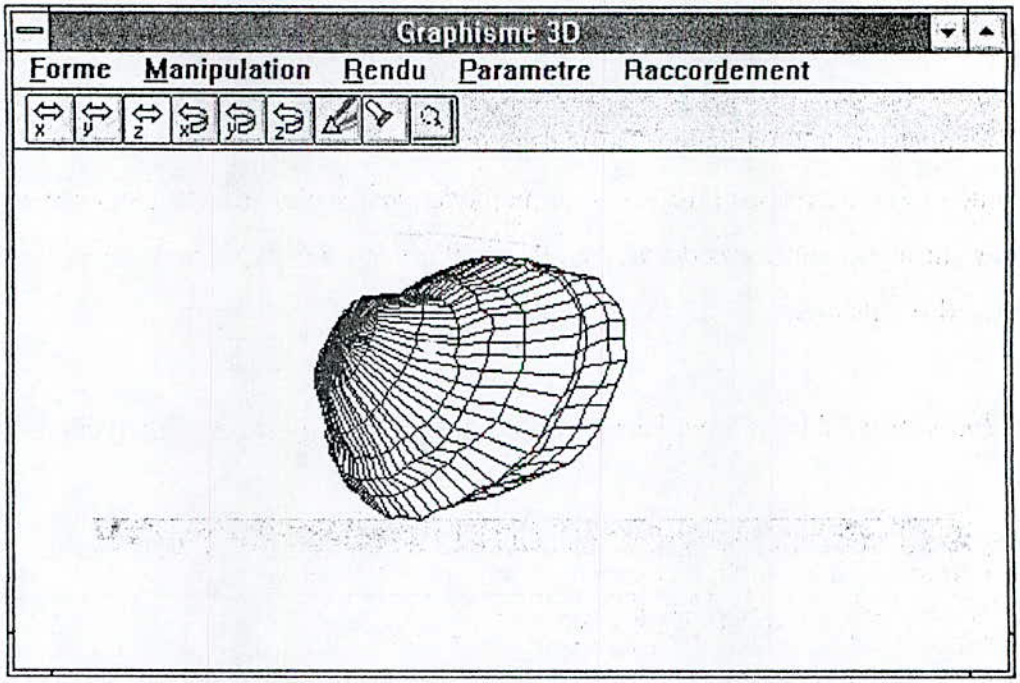


Fig. 5.30 : ***Reconstitution surfacique avec surfaecs cachées.***

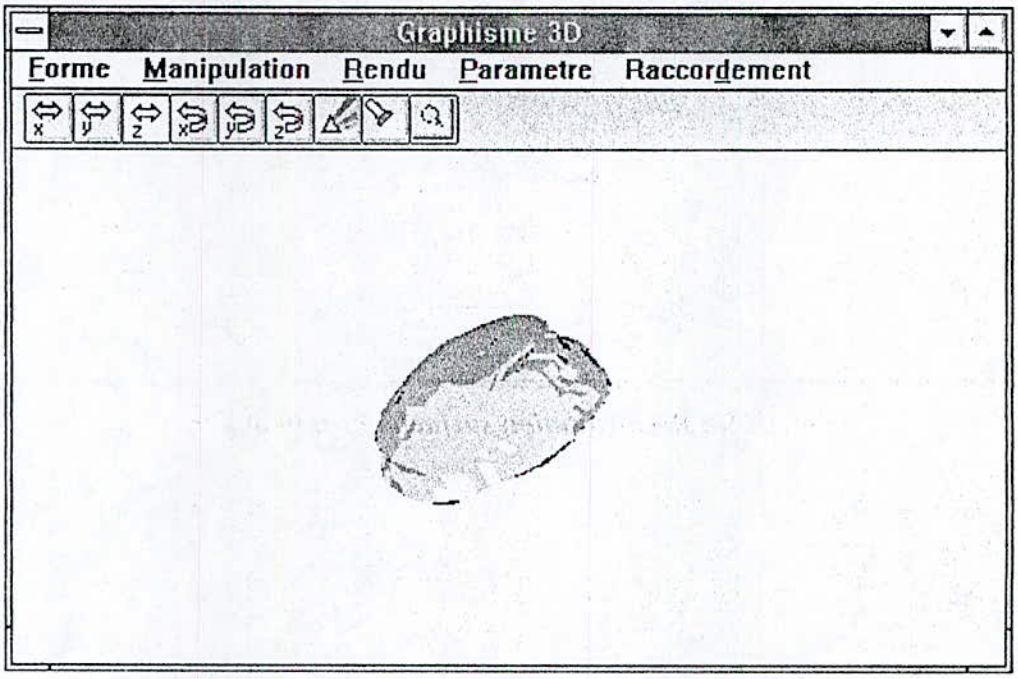


Fig. 5.31 : ***Reconstitution surfacique avec simulation d'éclairage.***

5.9 Conclusion

Nous avons vu que la reconstitution volumique que nous avons faite avait une mauvaise résolution à cause de l'épaisseur des coupes, qui était trop grande. Il est possible d'y remédier sans avoir à multiplier les coupes et cela en faisant une interpolation en deux coupes successives. Mais il est nécessaire dans ce cas de connaître l'équation de chaque contour afin de calculer les contours intermédiaires.

Dans le calcul des objets c'est surtout la suppression des surfaces cachées qui prend le plus de temps. Il est nécessaire pour des applications professionnelle d'avoir des circuits spécialisée dans l'élimination des surfaces cachées, comme pour les stations graphiques [GOU 94].

Nous avons aussi remarqué que la méthode surfacique prend moins de temps que la méthode volumique, car le nombre de surfaces à trier est beaucoup plus important dans la deuxième.

CONCLUSION GENERALE

CONCLUSION GENERALE

Nous avons voulus par notre présente étude déblayer le terrain, espérant ainsi que ce travail pourrait servir de base à un futur projet, ayant pour objet de s'approfondir encore plus dans ce domaine, en apportant des améliorations à ce qui a été déjà fait.

Et en matière d'améliorations nous pouvons déjà penser au lissage des objets, en appliquant par exemple, une des méthodes que nous avons exposées.

L'évolution peut aussi toucher la partie concernant le modelage, pour rendre possible le groupage des objets. Il serait ainsi possible de créer des formes complexes à partir de formes simples comme les cylindres, les sphères, les cônes...

On peut aussi s'intéresser à l'optimisation des routines d'affichage en les implémentant directement sur la carte graphique (accès direct à la RAM vidéo), chose qui nous a été impossible en mode protégé "windows".

Il est aussi possible d'exploiter la représentation par voxels via la méthode des éléments finis, pour simuler l'aspect dynamique des objets. Il serait donc possible de calculer la déformation d'un objet sous l'effet d'une force extérieure.

Dans notre travail, la partie dédiée à la reconstitution d'objet à partir de coupes n'était pas très au point en ce qui concerne la restitution des couleurs de l'objet. Il serait intéressant de pouvoir le faire, et ceci pour un plus grand réalisme. On pourrait par exemple procéder à la détection de la couleur du contour. Cette couleur est donc celle du voxel utilisé pour le remplissage.

Nous soulignerons enfin que la méthode d'acquisition des images limite la reconstitution aux objets dont on peut avoir les coupes. L'association de ce travail avec une autre méthode d'acquisition comme les capteurs de position ou des sondes ultrasons agrandirait sensiblement le domaine d'utilisation.

BIBLIOGRAPHIE

BIBLIOGRAPHIE

[BAS 91] BASSET O. Traitement d'images échographiques de la prostate. Application à la chirurgie assistée par ordinateur. Thèse de Doctorat. Institut National des Sciences Appliquées de Lyon, 1991.

[BER 94] BERTELSON B., RASCH M. PC interdit. Micro-Application, Paris, 1994.

[BON 91] BONIN P. Méthodes systématiques de conception et réalisation d'applications en vision par ordinateur. Thèse de Doctorat. Université Paris VII, 1991.

[BRE 88] BRET M. Images de synthèse : méthodes et algorithmes pour la réalisation d'images numériques. BORDAS, Paris, 1988.

[CHU 89] WEI-CHUNG L., SHIUH-YUNG C., CHIN-TU C. A new su rface interpolation technic for reconstructing 3D objects from serial cross-sections. Computer vision and image processing 48, 1989, p. 124-143.

[COC 95] COCQUEREZ J.-P., PHILIPP S. Analyse d'images: filtrage et segmentation. Masson, Paris, 1995.

[DEL 94] DELANNOY C. Apprendre à programmer en Turbo C. CHIHAB-EYROLLES, 1994.

[GOU 94] GOURRET J.-P. Modélisation d'images fixes et animées. Masson, Paris, 1994.

[HOR 93] HORAUD R. MONGA O. Vision par ordinateur, outils fondamentaux. Hemès, Paris, 1993.

[LEB 95] LEBLANC G. BORLAND C++ version 4. Programmation windows. EYROLLES, Paris, 1995.

[MAT 93] MATLAB user's guide. High performance numeric computation and visualisation software. The MATHWORKS inc, 1993.

[SCH 87a] SCHWEIZER P. Infographie, volume 1. Presses Polytechniques Romandes, Lausanne, 1987.

[SCH 87b] SCHWEIZER P. Infographie, volume 2. Presses Polytechniques Romandes, Lausanne, 1987.

[TIS 95] TISCHER M. PC Programmation système, 6ème édition. Micro-Application, Paris, 1995.